Time Window Assignment in Distribution Networks

Time Window Assignment in Distribution Networks

Tijdvenstertoekenning in distributienetwerken

Thesis

to obtain the degree of Doctor from the
Erasmus University Rotterdam
by command of the
Rector Magnificus

Prof.dr. R.C.M.E. Engels

and in accordance with the decision of the Doctorate Board.

The public defence shall be held on

Friday 15 November 2019 at 09:30 hours

by

Kevin Dalmeijer
born in Rotterdam, The Netherlands

Erasmus University Rotterdam

Doctoral committee

Promotor:	Prof.dr. A.P.M. Wagelmans
Other members:	Prof.dr. D. Huisman
	Prof.dr. G. Desaulniers
	Prof.dr. D. Vigo
Copromotor:	Dr. R. Spliet

Erasmus Research Institute of Management - ERIM
The joint research institute of the Rotterdam School of Management (RSM)
and the Erasmus School of Economics (ESE) at the Erasmus University Rotterdam
Internet: www.erim.eur.nl

ERIM Electronic Series Portal: repub.eur.nl

ERIM PhD Series in Research in Management, 486
ERIM reference number: EPS-2019-486-LIS
ISBN 978-90-5892-562-6

Cover image: Baran Sarigul
Cover design: PanArt, www.panart.nl

This publication (cover and interior) is printed by Tuijtel on recycled paper, BalanceSilk®.
The ink used is produced from renewable resources and alcohol free fountain solution.
Certifications for the paper and the printing production process: Recycle, EU Ecolabel, FSC®, ISO14001.
More info: www.tuijtel.com


Acknowledgements

Back in 2014, my motivation for starting a PhD was simple: it sounded like a lot of fun. Looking back, this turned out to be an excellent prediction. Over the last years, I have spent hours in front of whiteboards and computer screens to solve challenging problems with my colleagues, and I have spent hours in front of students to teach them about topics that I care about. I thoroughly enjoyed this, and I am very grateful that my PhD has prepared me for a career in which I can keep combining research and teaching. Many people deserve credit for making this possible and for making my time at the Erasmus University memorable. I would like to use this opportunity to thank them.

First, I have to thank my promotor, Albert Wagelmans. Without you, Albert, I would not be where I am today. You are the one who convinced me to study econometrics and operations research, the one who hired me as a PhD student, and the one who helped me to secure a postdoc position at Georgia Tech. I had a great time working with you and I really appreciate our conversations both about work and non-work.

During the past five years, I have worked most closely with my copromotor and daily supervisor, Remy Spliet. Remy, I think you are an amazing supervisor who truly cares about his students. You have not only taught me how to write papers, but you have also been a huge influence on my values and beliefs about how research should be done. I will never forget how you interrupted my presentation to point out that my results were better than yours, and you encouraged me to emphasize this fact instead of hiding it. You have given me a lot of freedom during my PhD, and I am grateful for that confidence you have put in me.

Before a thesis can be defended, there has to be a committee who is prepared to scrutinize its contents. I want to thank Prof.dr. Dennis Huisman, Prof.dr. Guy Desaulniers, and Prof.dr. Daniele Vigo for being part of my inner doctoral committee, and I would like to thank Prof.dr. Wout Dullaert, Prof.dr. Richard Eglese, and

Prof.dr. Joaquim Gromicho for completing the opposition. These six professors are some of the best experts in the field, and I truly admire their work. I am absolutely honored that you have taken time out of your busy schedules to serve on my doctoral committee.

The fact that Guy Desaulniers is part of the committee is extra special to me, because we had the opportunity to actually work together. Guy, I want to thank you for inviting me to Montréal and for mentoring me. You have taught me more about column generation than I ever thought there was to know. Next to doing research together, I really enjoyed playing *jeu de tarot* and spending time with you at conferences. I have experienced GERAD as a very inspiring and a very welcoming group, and I would like to thank everybody who was there for making me feel at home. My office mate Marilène and her partner Olivier went above and beyond to involve me in social activities and to teach me about québécoise culture. This meant a lot to me, and I want to thank you both for that.

In the Netherlands, I was fortunate to work in a group of exceptional people. We PhD students jokingly referred to our group as the *Awesome PhD's*, but I am completely serious when I say that my colleagues absolutely deserve this title. I want to thank you, Awesome PhD's, for actively supporting and helping each other out, and for being wonderful people in general. In this group, it does not matter whether you work on maritime logistics or on lot sizing problems: if you need help, your colleagues are there. This environment is something you create and should be very proud of. I have also really enjoyed all the events that we organized, ranging from the *PhriDay* problem solving meetings to the *PhD Game Nights*.

I want to give some special attention to the people that I have spent the most time with in the same office: Harwin, Thomas, Judith, and Rutger. Harwin was my first office mate, and he is somebody who truly promotes a positive atmosphere by leading by example. We could talk about anything, ranging from silly jokes to serious personal topics. After Harwin left, Thomas became my new office mate. Thomas is one of the most multi-talented people that I know. We did not only discuss research ideas, but Thomas would also teach me about classical music and quantum mechanics, to give some examples. I am honored that Thomas has agreed to be my paranymph, and I am proud to be his paranymph in return. I was never officially assigned to the same office as Judith or Rutger, but we had no problem in finding each other's desks. Judith is a person who is nice to be around, and I appreciate that you always made time to talk if I needed a break. Rutger and I started our PhD's at the same time, and he has been a great support from start to

finish.

As mentioned earlier, teaching has been an important part of my development as an academic. I want to thank Jan Brinkhuis and Christiaan Heij for inspiring me to teach, and Paul de Boer for giving me the first opportunity to do so as a student assistant. I also owe a debt of gratitude to Patrick Groenen. Patrick and I have taught non-linear optimization together for three years, and he has given me every opportunity to grow. Next to that, I would like to thank you for our long discussions on majorization and for all the professional advice you have given me over the years. After teaching with Patrick, I have taught together with Ilker Birbil. Ilker, it was a pleasure working with you, and your career advice has been very important to me. Bas van Goozen has motivated me to improve education by innovation, and I appreciate our conversations on what education could look like in the future. Finally, I want to thank all the student assistants that I had the joy of working with, and all the students that made me enjoy teaching.

Before and during my PhD, I have also been involved with S-ray Diagnostics, a company that started as a scientific enterprise of the Erasmus University. At S-ray Diagnostics, scientific research is directly applied in practice, and I learned a lot by being part of that effort. I want to thank Marco de Haas for allowing me to be part of the team, and I want to thank Daan, Patrick, Murat, Jeanine, Lisanne, and Ben for being great S-ray colleagues.

Next to the many people mentioned above, I have met many more incredible people at the Erasmus University. To prevent that the acknowledgements will be longer than the remainder of this thesis, I will not thank you one by one. But please know that you contributed to me being excited to go to work everyday, and I sincerely thank you for that.

Outside of university, I want to thank my family and friends who have supported me during my PhD. It feels appropriate to do so in Dutch.

Beste familie en vrienden, ik wil jullie van harte bedanken voor jullie steun de afgelopen jaren. Het betekent erg veel voor mij dat jullie je interesseren in waar ik me mee bezig houd en dat jullie me altijd hebben aangemoedigd. In het bijzonder wil ik mijn oom, tante, en mijn neven, Wouter, Yvonne, Sander en Joris bedanken dat jullie altijd voor me klaar staan. Ook wil ik mijn grootouders bedanken die helaas niet bij de verdediging aanwezig kunnen zijn, al weet ik zeker dat ze trots zouden zijn geweest. Ik mag me ook gelukkig prijzen met mijn fantastische vrienden Maurits, Wayne, Rafe en Romy. Ik wil jullie bedanken voor jullie steun en afleiding wanneer ik dat nodig had, en vooral dat jullie het al zo lang met mij volhouden: Maurits al

meer dan 20 jaar!

De drie belangrijkste personen heb ik voor het laatst bewaard. Mijn broer Tim is de beste broer die ik me kan wensen. Tim, ik ben onwijs trots op je, en ik vind het fantastisch hoe wij altijd voor elkaar klaar staan. Het is dan ook niet meer dan logisch dat je ook bij mijn verdediging als paranimf naast me staat. Ook mijn ouders ben ik enorm dankbaar. Pap en mam, jullie hebben me altijd aangemoedigd eruit te halen wat erin zit, en jullie hebben er alles aan gedaan om dat mogelijk te maken. Zonder de steun van jullie drieën had dit proefschrift niet bestaan.

Table of contents

Chapter 1

Introduction

In this thesis, we consider assigning time windows to customers in distribution networks. The following chapters are quite technical in the sense that they require familiarity with Operations Research (OR) to be completely understood. However, our work is motivated by very practical problems that most readers will recognize.

For this reason, I have chosen for a more informal introduction that can be read by anybody. I will explain the motivation behind the problems that we study, and I will discuss the key ideas that allow us to solve them. Part of this introduction is taken from my posts on the Last-Mile Logistics blog: Dalmeijer (2016) and Dalmeijer (2017).

1.1 Time window assignment in an uncertain world

It is 12.05h and 37 seconds when the doorbell rings. You already know who it is: it is the delivery man with the parcel you ordered. His arrival is the exact time he said he would be there.

While this would be perfectly normal in an ideal world, it is simply not possible in our world filled with uncertainties. For example, the delivery man may be late due to busy traffic. To deal with such uncertainties, companies assign time windows to customers, informing them between which times they may expect service. Unfortunately, these time windows are often too wide to be informative, for example from 9am to 5pm.

Such wide time windows allow the distributor to construct efficient routes, but at the expense of customers satisfaction. Nowadays, more and more companies realize

that the satisfaction of their customers is important (Kovacs et al., 2014). This motivated our work in Chapters 2 and 3, where we use mathematical models to assign time windows that balance operational costs and customer satisfaction. We assign smaller time windows, but we do this in such a way that the distributor can still construct relatively efficient routes.

Another way for distributors to deal with uncertainty is by updating the time window of delivery throughout the day. This can be done, for example, by sending text messages to the customers when the delivery man is behind schedule, or by providing tracking information online. Major logistics companies FedEx, UPS and DHL all provide such services.

We call these updates *dynamic time window adjustments*. Dynamic time window adjustments are common in practice, but have not yet been considered in the literature. In Chapter 4, we therefore study the following question: when should customers be contacted, and what should they be told?

In the above, the problems that we study were stated in terms of deliveries to individual customers. However, similar problems occur in different settings. For example, one can consider a setting in which deliveries are made from a central depot to retail stores or pickup points. The methods that we have developed may even be applied to completely different problems with a similar mathematical structure.

1.2 Problem 1: the TWAVRP

In Chapter 2 and Chapter 3, the Time Window Assignment Vehicle Routing Problem (TWAVRP, Spliet and Gabor (2015)) is considered. For this problem, we assume that the demand of the customers is the only uncertain quantity. In other words, it is known in advance which customers have to be served, but it is not known in advance how much vehicle space we require for each order. This is the case in retail networks, for example, where the customers are retail stores which place orders on a regular basis.

The TWAVRP is the problem of assigning time windows in this setting. We assign time windows of a given width (e.g., two hours) in such a way that the expected operational cost is minimized. Chapters 2 and 3 are dedicated to developing methods to solve the TWAVRP.

1.2.1 Key ideas of Chapter 2

In Chapter 2, we present our first method to solve the TWAVRP. At the time that this method was developed, it was able to solve the TWAVRP on average about 200 times faster than the best known method in the literature. Furthermore, we could assign time windows in larger distribution networks than previously possible.

These computational improvements are due to two key ideas. The first idea is to use a different mathematical model than the one introduced by Spliet and Gabor (2015). I will not discuss the differences here, but refer to Chapter 2 instead.

The second idea is to exploit an observation that is similar to the following. Consider customers A and B, such that it takes five hours to drive from A to B. Both customers are assigned a time window of two hours in width. We then observe the following: if A and B are served within their time windows, by the same vehicle, then the time windows of A and B do not overlap.

For example, customer A may be visited at 1pm within the time window [1pm, 3pm]. The vehicle then arrives at customer B at 6pm, which is within the time window [5pm, 7pm]. The two time windows in this example indeed do not overlap. I leave it to the reader to confirm that this is true for any choice of two-hour time windows.

Although seemingly insignificant, this observation gives us important information about the mathematical structure of the TWAVRP. In Chapter 2, we use this information in different ways to improve our solution method. Also in Chapter 3, we make use of this observation.

1.2.2 Key ideas of Chapter 3

In Chapter 3, we again consider the TWAVRP. Recall that in Chapter 2, we have found that the new mathematical model leads to significantly better performance than the model by Spliet and Gabor (2015). For technical reasons, this surprised me, and I expected that the model by Spliet and Gabor (2015) still had some hidden potential.

This prompted the study in Chapter 3, for which we reimplement the method by Spliet and Gabor (2015) and we identify exactly why the original performance was not as good as expected: symmetry. If we properly address symmetry, we can obtain the optimal solution to the TWAVRP by doing only 7.4% of the work, compared to when symmetry is ignored[1]. Our new method that addresses symmetry outperforms the

[1] measured in average number of nodes in the search tree

method in Chapter 2, and is competitive with the recently published decomposition algorithm by Subramanyam et al. (2018). Furthermore, we believe that the way in which we handle symmetry can also be applied to other problems.

I will now illustrate the issue of symmetry with an example problem. This problem differs from the TWAVRP, but the intuition is the same. There are five customers, A, B, C, D, and E, who all have to be assigned a time window in the morning, in the afternoon, or in the evening. Customers A, B, C, and E have to be visited by the first truck, which is represent by a solid line. Customers A, C, D, and E have to be visited by the second truck, which is represented by a dotted line.

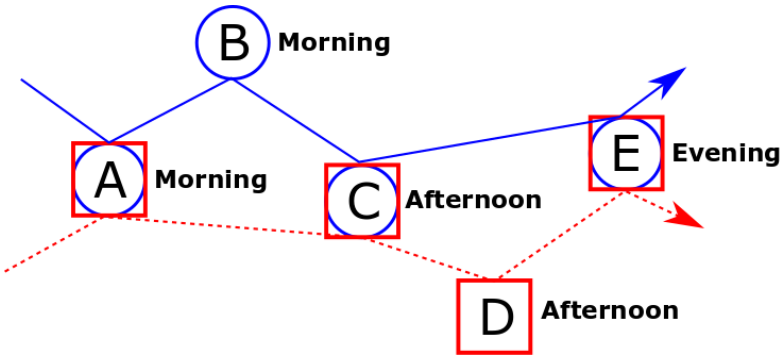


Figure 1.1: Optimal solution when A is visited in the morning.

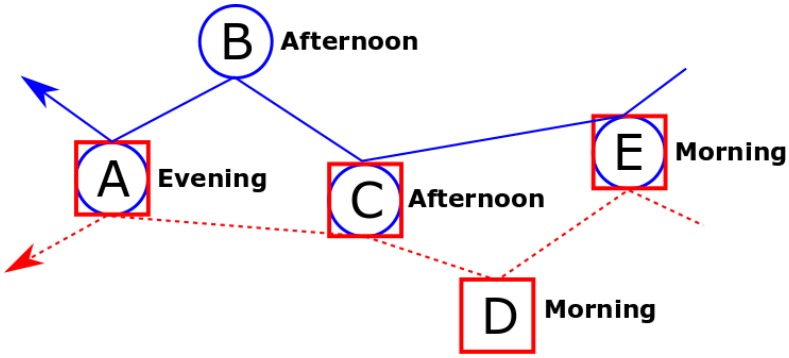


Figure 1.2: Optimal solution when A is *not* visited in the morning.

To determine if it is optimal to visit customer A in the morning, the problem can be split into two subproblems. For one subproblem, we enforce that customer A is visited in the morning, and for the other, we enforce that this is not the case. We then solve both subproblems. Based on which subproblem provides the best solution,

we can tell whether it is optimal to visit customer A in the morning or not.

The solutions to the subproblems are presented in Figures 1.1 and 1.2, respectively. In both solutions, the trucks travel the exact same roads, just in different directions. We call solutions like this *symmetric solutions*.

In hindsight, it would have been more efficient to compute only one of these symmetric solutions. The other solution can then be obtained by reversing the directions of the routes. It follows that we did unnecessary work because we did not take symmetry into account.

The key to avoid this additional work is to be careful about how problems are split into subproblems. In our example, splitting the problem based on the time window of customer A introduces symmetry. In Chapter 3, we discuss how additional work due to symmetry can be avoided.

1.3 Problem 2: the DTWAP

In Chapter 4, the Dynamic Time Window Adjustment Problem (DTWAP) is introduced. For this problem, we consider a single delivery man who delivers parcels to individual customers. The order in which the customers are visited is determined in advance by some routing software that we have no control over. What makes the problem interesting, is that we assume that the travel times are uncertain. For example, the delivery man can be late due to busy traffic.

In this setting, the DTWAP is the problem of answering the following question: when should customers be contacted, and what should they be told? For example, we can contact a customer if we expect to be at least 30 minutes late. The customer can then be told that the time window of delivery is postponed from [1pm, 2pm] to [1.30pm, 2.30pm].

1.3.1 Key ideas of Chapter 4

When the customers should be contacted, and what they should be told, depends on the preferences of the customers. Some customers may not care about time window adjustments, because they will be at home anyways. Other customers may be angry if the delivery man is only one minute late. These characteristics are captured in *dissatisfaction functions*.

In Chapter 4, we derive general properties of the DTWAP, and we present three different solution methods. We then describe which solution methods can be used, depending on the dissatisfaction functions. As such, this chapter serves as a guide

for those who want to solve the DTWAP in a specific setting. To demonstrate this use, we also discuss an example application.

In Chapter 4, we make use of different insights to analyze the DTWAP. I will highlight one of these insights, which is based on the dynamic programming principle (Bellman, 1966). This principle states that future actions should be based on where you are now, not on how you got there.

Applied to the DTWAP, this leads to the following observation. To determine optimal dynamic time window adjustments, we only need three pieces of information:

1. The current time.
2. The current location.
3. The current (possibly adjusted) time windows of the customers.

We can thus ignore our previous interactions with the customers, except for the most recent time windows that we have communicated. This seemingly insignificant fact is of major mathematical importance, and is used extensively in Chapter 4.

1.4 Clarification of contribution

The chapters of this thesis are self-contained papers that have resulted from my collaborations with different authors. The majority of the work in these chapters has been done independently under close supervision of my co-authors.

Chapter 2. For this chapter, I collaborated with my copromotor, dr. Remy Spliet, who also acted as my daily supervisor. Apart from minor differences, this chapter is a direct copy of Dalmeijer and Spliet (2018), which was published in *Computers & Operations Research*.

Chapter 3. This chapter is based on the work that I did together with prof.dr. Guy Desaulniers, while visiting Montréal in 2017. Chapter 3 is an improved version of our working paper Dalmeijer and Desaulniers (2018). This work is currently under review at *INFORMS Journal on Computing*.

Chapter 4. For this chapter, I worked together with my copromotor, dr. Remy Spliet, and my promotor, prof.dr. Albert P.M. Wagelmans. Chapter 4 is based on the working paper Dalmeijer et al. (2019), which is currently being prepared

for publication.

Finally, in Chapter 5, we conclude the main findings of this thesis.

Chapter 2

A branch-and-cut algorithm for the Time Window Assignment Vehicle Routing Problem

Abstract

This chapter presents a branch-and-cut algorithm for the Time Window Assignment Vehicle Routing Problem (TWAVRP), the problem of assigning time windows for delivery before demand volume becomes known. A novel set of valid inequalities, the precedence inequalities, is introduced and multiple separation heuristics are presented. In our numerical experiments the branch-and-cut algorithm is 3.8 times faster when separating precedence inequalities. Furthermore, in our experiments, the branch-and-cut algorithm is about 200 times faster than the best known algorithm in the literature. Finally, using our algorithm, instances of the TWAVRP are solved which are larger than the instances previously presented in the literature.

This chapter is based on Dalmeijer and Spliet (2018).

2.1 Introduction

The Time Window Assignment Vehicle Routing Problem (TWAVRP) is the problem of assigning time windows for delivery to clients in a distribution network when the demand volume of the clients is uncertain, such that the expected traveling costs are minimized. Each time window, which we refer to as an endogenous time window, has a fixed width, and has to be assigned within an exogenous time window. For example, a two hour endogenous time window is assigned during the opening hours of a shop. First introduced by Spliet and Gabor (2015), the TWAVRP is inspired by distribution networks of retailers.

In a retail network, the clients are retail stores which place orders on a regular basis. It is common that the time windows for delivery are fixed for a long period of time (e.g., a year). This is convenient for the retailers, as it allows them to ensure that enough personnel is available to process the delivery. Furthermore, it simplifies inventory control. Demand is uncertain and fluctuates per delivery. This results in orders of varying sizes, which become known shortly before the vehicles are dispatched.

To deal with demand uncertainty, the TWAVRP requires demand scenarios and their probability of occurrence to be known in advance. Note that if the number of scenarios is equal to one, the problem reduces to a Vehicle Routing Problem with Time Windows (VRPTW). Hence, the TWAVRP is NP-hard.

The TWAVRP differs from common Stochastic Vehicle Routing Problems, where client demand is revealed only on arrival at the client, and only smaller recourse actions are considered, e.g., to perform a return trip to the depot (Gendreau et al., 1996). For the TWAVRP, determining the recourse action requires solving a VRPTW, which is already NP-hard. To be able to deal with demand uncertainty, the TWAVRP assumes demand scenarios and their probabilities of occurrence to be known in advance.

An increasing number of companies focus on achieving consistent service with their deliveries (Kovacs et al., 2014). Also in the scientific literature, we see the same trend towards consistency considerations in routing, as can be seen in the recent survey by Kovacs et al. (2014). In this survey, three main pillars of consistency are described: arrival time, person-oriented and delivery consistency, and the TWAVRP is categorized within the first. Our study adds to the limited amount of research done so far on exact methods for solving routing problems with consistency considerations.

Among the routing problems with consistency considerations, the TWAVRP is in particular closely related to two specific models. Firstly, the TWAVRP is similar

to the Consistent Vehicle Routing Problem (ConVRP) introduced in Groër et al. (2009). The ConVRP does not only impose consistent arrival time but also requires the same driver to service the same client. Another closely related model is the Vehicle Routing Problem with self-imposed time windows, as introduced by Jabali et al. (2015). In their paper, the authors assume demand to be given while travel times are uncertain.

The TWAVRP is a generalization of both the Capacitated Vehicle Routing Problem (CVRP) and the VRPTW, and hence similar solution methods can be used. In a recent survey, Baldacci et al. (2012) mention that there are three formulations that have been most successful when used to solve the CVRP. Two of them make use of flow variables (Laporte et al. (1985), Baldacci et al. (2004)), while the third is a set partitioning formulation (Balinski and Quandt, 1964). For the VRPTW, a set partitioning formulation in a branch-price-and-cut algorithm is very successful (Desaulniers et al., 2008). To solve the TWAVRP, Spliet and Gabor (2015) also introduce a branch-price-and-cut algorithm based on a set partitioning formulation, which allows for instances with up to 25 clients and three demand scenarios to be solved to optimality within a one hour time limit. Similarly, Spliet and Desaulniers (2015) solve the DTWAVRP, the discrete time window variant of the TWAVRP.

In this chapter, we present a new flow formulation for the TWAVRP that is of polynomial size in the number of clients and scenarios. This formulation is based on the MTZ-inequalities in Miller et al. (1960) and the 2-commodity flow formulation in Baldacci et al. (2004). Based on this formulation, we construct a branch-and-cut algorithm that is faster than the algorithm in Spliet and Gabor (2015). This new algorithm does not only allow for obtaining solutions faster, but also allows for solving larger instances of the TWAVRP, making it applicable to larger networks than previously possible.

One of the factors that contributes to the success of the branch-and-cut algorithm, is the introduction of a novel class of valid inequalities specifically designed for the TWAVRP: the precedence inequalities. We identify pairs of routes in different scenarios that cannot be selected simultaneously for any feasible time window assignment. Because time windows are not fixed in advance, identifying such pairs is a main challenge, which we address. We subsequently create valid inequalities using these pairs, similar to the valid inequalities designed by Ascheuer et al. (2000), which disallow infeasible paths for the Asymmetric Traveling Salesman Problem with Time Windows (ATSPTW). We show that separating precedence inequalities is co-NP-hard

and present several separation heuristics to find violated inequalities.

The remainder of this chapter is structured as follows: in Section 2.2, we present the formulation that is used in our branch-and-cut algorithm. In Section 2.3 we present the branch-and-cut framework. Section 2.4 is dedicated to the precedence inequalities and heuristics for separating them. Our numerical experiments and their results are presented in Section 2.5. In the final section, we write our conclusions and present some directions for further research.

2.2 Problem definition

In this section, we first formally introduce the TWAVRP. Then, we present a mixed integer linear program to solve the problem.

Consider a set of n clients $V' = \{1, 2, ..., n\}$. Furthermore, location 0 represents the starting depot and location $n+1$ the ending depot. Let $V = V' \cup \{0, n+1\}$ represent the set of all locations. The directed graph G on vertex set V and with arc set A is used to represent our distribution network. Arc set A consists of all arcs leaving the starting depot, $(0, i)$ for $i \in V'$, all arcs entering the ending depot, $(i, n+1)$ for $i \in V'$, and all arcs between the clients in V'.

For each directed arc $(i, j) \in A$ a travel cost c_{ij} and a travel time τ_{ij} is given. The travel costs are assumed to be non-negative, $c_{ij} \geq 0$, and to adhere to the triangle inequality, $c_{ij} + c_{jk} \geq c_{ik}$. We assume the same properties for the travel times. Moreover, we require all travel times to be strictly positive, the reason for this is highlighted later.

Let u_i be the width of the time window that has to be assigned to client $i \in V'$. We refer to this time window as the *endogenous* time window of client i, as opposed to the *exogenous* time window of client i which defines the interval in which the endogenous time window should be chosen.

The exogenous time window of each client $i \in V'$ is fixed and given by the interval $[s_i, e_i]$, with $e_i - s_i \geq u_i$. Furthermore, the opening hours of the starting depot are given by $[s_0, e_0]$ and the opening hours of the ending depot are given by $[s_{n+1}, e_{n+1}]$.

We assume that we have access to an unlimited number of homogeneous vehicles, each with capacity Q. To model demand uncertainty, consider a finite set of possible demand scenarios Ω and corresponding probabilities p_ω such that $\sum_{\omega \in \Omega} p_\omega = 1$. The demand of client $i \in V'$ in scenario $\omega \in \Omega$ is given by $0 < d_i^\omega \leq Q$.

We now formally state the TWAVRP: find both 1) an assignment of the endoge-

nous time windows within the exogenous time windows and 2), for every demand scenario, a routing of the vehicles adhering to the capacity constraints, and consistent with the assigned endogenous time windows, such that the expected routing cost is minimized.

2.2.1 Mixed integer linear program

Next, we present a new mixed integer linear programming formulation for the TWAVRP, based on the MTZ-inequalities in Miller et al. (1960) and the 2-commodity flow formulation in Baldacci et al. (2004). First we introduce the decision variables. The time window decisions are given by the continuous variables y_i for $i \in V'$, which indicate the starting times of the endogenous time windows. That is, the endogenous time window of client i is given by $[y_i, y_i + u_i]$. As the endogenous time window has to be within the exogenous time window, we require $y_i \in [s_i, e_i - u_i]$.

The vehicle routes are determined by the binary flow variables x_{ij}^{ω} for $(i,j) \in A$, which are equal to one if a vehicle travels from i to j in scenario ω. The continuous variable t_i^{ω} indicates at what time client $i \in V'$ receives delivery in scenario $\omega \in \Omega$.

For notational convenience, we introduce the arc set $\bar{A}$. Let $\bar{A}$ be the set of arcs A to which the arcs $(i,0)$ and $(n+1,i)$ have been added for all $i \in V'$. To model capacity, we use a formulation similar to the one used in Baldacci et al. (2004), but applied to a directed graph. We introduce the flow variables z_{ij}^{ω} for all $(i,j) \in \bar{A}$, $\omega \in \Omega$. Its interpretation depends on the direction the arc is traversed, as given by the x-variables. If z_{ij}^{ω} follows this direction ($x_{ij} = 1$), it represents the total vehicle load when traveling from client i to client j. If it follows the opposite direction ($x_{ji} = 1$), z_{ij}^{ω} represents the leftover capacity on the vehicle when traveling from client j to client i. If the connection between i and j is not used ($x_{ij} = x_{ji} = 0$), z_{ij}^{ω} is zero.

We provide the following mixed integer linear programming formulation:

$$\min \quad \sum_{\omega \in \Omega} p_{\omega} \sum_{(i,j) \in A} c_{ij} x_{ij}^{\omega} \tag{2.1}$$

$$\text{s.t.} \sum_{j \in V' \cup \{n+1\}} x_{ij}^{\omega} = 1 \qquad \forall i \in V',\ \omega \in \Omega \tag{2.2}$$

$$\sum_{i \in V' \cup \{0\}} x_{ij}^{\omega} = 1 \qquad \forall j \in V',\ \omega \in \Omega \tag{2.3}$$

$$z_{ij}^{\omega} + z_{ji}^{\omega} = \left(x_{ij}^{\omega} + x_{ji}^{\omega}\right) Q \qquad \forall (i,j) \in \bar{A}, i < j, \omega \in \Omega \tag{2.4}$$

$$\sum_{j \in V} (z_{ji}^{\omega} - z_{ij}^{\omega}) = 2d_i^{\omega} \qquad \forall i \in V', \omega \in \Omega \tag{2.5}$$

$$\sum_{j \in V'} z_{0j}^{\omega} = \sum_{i \in V'} d_i^{\omega} \qquad \forall \omega \in \Omega \tag{2.6}$$

$$\sum_{j \in V'} z_{n+1,j}^{\omega} = \left(\sum_{j \in V'} x_{0j}^{\omega} \right) Q \qquad \forall \omega \in \Omega \tag{2.7}$$

$$\sum_{j \in V'} z_{j0}^{\omega} = \left(\sum_{j \in V'} x_{0j}^{\omega} \right) Q - \sum_{i \in V'} d_i^{\omega} \qquad \forall \omega \in \Omega \tag{2.8}$$

$$t_j^{\omega} - t_i^{\omega} \geq \tau_{ij} x_{ij}^{\omega} + (s_j - e_i)(1 - x_{ij}^{\omega}) \qquad \forall i \in V', j \in V', \omega \in \Omega \tag{2.9}$$

$$s_0 + \tau_{0j} \leq t_j^{\omega} \qquad \forall j \in V', \omega \in \Omega \tag{2.10}$$

$$t_i^{\omega} + \tau_{i,n+1} \leq e_{n+1} \qquad \forall i \in V', \omega \in \Omega \tag{2.11}$$

$$t_i^{\omega} \geq y_i \qquad \forall i \in V', \omega \in \Omega \tag{2.12}$$

$$t_i^{\omega} \leq y_i + u_i \qquad \forall i \in V', \omega \in \Omega \tag{2.13}$$

$$y_i \in [s_i, e_i - u_i] \qquad \forall i \in V' \tag{2.14}$$

$$x_{ij}^{\omega} \in \mathbb{B} \qquad \forall (i,j) \in A, \omega \in \Omega \tag{2.15}$$

$$z_{ij}^{\omega} \geq 0 \qquad \forall (i,j) \in \bar{A}, \omega \in \Omega. \tag{2.16}$$

The objective (2.1) is to minimize the expected traveling costs over all scenarios. Constraints (2.2) and (2.3) are the flow conservation constraints. Constraints (2.15) make sure all flows are integral.

Vehicle capacity is modeled by Constraints (2.4)-(2.8) and (2.16), which are based on the 2-commodity flow formulation in Baldacci et al. (2004). Constraints (2.4) relate opposing arcs: when either (i, j) or (j, i) is used, the corresponding z-variables sum to the vehicle capacity.

Constraints (2.5) can be seen as flow conservation constraints for the z-variables: before visiting the client, the load is d_i^{ω} units more than afterwards. After visiting, the empty space is d_i^{ω} units more than before. Hence, the total difference in both flows is equal to $2d_i^{\omega}$. The total vehicle load, capacity and excess capacity in the system are constrained by (2.6)-(2.8). Constraints (2.6) set the total vehicle load equal to the total demand of all clients, and Constraints (2.7) set the total capacity equal to the number of used vehicles multiplied by the vehicle capacity. The total excess capacity of all vehicles leaving the starting depot is set by Constraints (2.8). Finally, we enforce vehicle capacity to be respected by constraining all empty space on the vehicles to be non-negative, as is done in Constraints (2.16). For more details

on these constraints, see Baldacci et al. (2004).

Constraints (2.9)-(2.14) deal with the time windows. Constraints (2.9) are the MTZ-inequalities that model the time-of-service (Miller et al., 1960). If a vehicle travels from client i to client j in scenario $\omega \in \Omega$, then $x_{ij}^{\omega} = 1$ and hence $t_j^{\omega} - t_i^{\omega} \geq \tau_{ij}$, i.e., the time-of-service increases with at least the travel time from i to j. If no vehicle travels from i to j in scenario ω, then $x_{ij}^{\omega} = 0$ and the constraint reads $t_j^{\omega} - t_i^{\omega} \geq s_j - e_i$, which always holds as $t_j^{\omega} \geq s_j$ and $t_i^{\omega} \leq e_i$. Note that the MTZ-inequalities eliminate cycles, because we have assumed that $\tau_{ij} > 0$ for all $(i, j) \in A$.

Constraints (2.10) guarantee that vehicles can only leave the starting depot after it opens and Constraints (2.11) ensure that vehicles arrive at the ending depot before it closes. Constraints (2.12) and (2.13) enforce that each client is served within its endogenous time window, while Constraints (2.14) make sure these endogenous time windows are within the exogenous time windows.

2.2.2 Remarks

In the above formulation, we model capacity using constraints that are based on the 2-commodity flow formulation in Baldacci et al. (2004). The MTZ inequalities are used to model time-of-service. Other formulations for capacity and time-of-service have been considered as well, including several adaptations of models for the CVRP (Baldacci et al., 2004) and the ATSPTW (Ascheuer et al., 2001).

Preliminary testing with all our combinations of capacity constraints and time-of-service constraints in a branch-and-cut algorithm showed that the best performance is achieved by the combination of the 2-commodity flow formulation to model capacity and the MTZ-inequalities to model time-of-service. This may seem surprising, as in general the MTZ-inequalities typically do not contribute to strong bounds in the LP relaxation. For our instances, however, we have seen that this is compensated for by the relatively strong formulation for capacity.

The MTZ-inequalities require no additional variables and only $n(n+1)|\Omega|$ constraints. This allows for a branch-and-cut strategy in which we process the nodes of the search tree faster than in all other alternatives we considered. Using the 2-commodity flow formulation to model capacity yields a good trade-off between the number of variables and constraints, and the strength of the formulation.

2.3 Branch-and-cut algorithm

In this section, we present our branch-and-cut algorithm. First, we present valid inequalities from the literature, which we use to strengthen the LP-bound of (2.1)-(2.16). Next, we introduce our branching strategy. In Section 2.4 we separately introduce the novel precedence inequalities.

2.3.1 2-cycle elimination

The current mixed integer linear program ensures that no integer feasible solution contains cycles. Explicit cycle elimination, however, may still strengthen the LP-bound.

As there are only a quadratic number of 2-cycles in a given graph, we can eliminate all 2-cycles with a relatively small number of constraints. We do so by adding all of the following inequalities to the formulation:

$$x_{ij}^{\omega} + x_{ji}^{\omega} \leq 1 \ \forall i \in V', j \in V', \omega \in \Omega. \tag{2.17}$$

2.3.2 Rounded capacity inequalities

The capacity inequalities are well-known valid inequalities for the VRPTW, and thus may be applied directly to the TWAVRP per scenario. Let λ_S^{ω} be the minimum number of vehicles required to satisfy the demand of all clients in $S \subseteq V'$ in scenario ω. The capacity inequalities are given by

$$\sum_{(i,j) \in A | i \in S, j \in V \setminus S} x_{ij}^{\omega} \geq \lambda_S^{\omega} \ \forall S \subseteq V', |S| \geq 2, \omega \in \Omega. \tag{2.18}$$

That is, we require for each subset $S \subseteq V'$ that the total number of vehicles leaving S is sufficient to satisfy all demand in S.

Calculating λ_S^{ω} requires solving a bin-packing problem, which is NP-hard in general. Hence, as is commonly done, we only consider the weakened inequalities in which λ_S^{ω} is replaced by the bin-packing lower bound $\lceil \left(\sum_{i \in S} d_i^{\omega} \right) / Q \rceil$. These valid inequalities are known as the rounded capacity inequalities. Simply adding all rounded capacity inequalities is not efficient, but when used in a branch-and-cut algorithm, they can be very effective (Baldacci et al., 2012).

We add the following $|\Omega|$ inequalities to the formulation:

$$\sum_{i \in V'} x^{\omega}_{i,n+1} \geq \left\lceil \frac{\sum_{i \in V'} d^{\omega}_i}{Q} \right\rceil \ \forall \omega \in \Omega. \tag{2.19}$$

These inequalities put a lower bound on the number of vehicles that have to be used per scenario. Preliminary experiments suggest that adding these inequalities from the beginning and adding the other rounded capacity inequalities in a cutting plane fashion speeds up our branch-and-cut algorithm. To find violated rounded capacity inequalities, we use the separation algorithm from Lysgaard (2003). Details on this algorithm can be found in Lysgaard et al. (2004).

2.3.3 Branching strategy

In the formulation in Section 2.2.1, we require each x^{ω}_{ij} to be binary. However, we show that it is sufficient to require that all $x^{\omega}_{ij} \in [0,1]$, that x^{ω}_{ij} is binary for all arcs connected to one of the depots, and furthermore that $x^{\omega}_{ij} + x^{\omega}_{ji}$ is binary for all $i, j \in V'(i \neq j)$. We define the following integrality conditions:

$$x^{\omega}_{0j} \in \mathbb{B} \ \forall j \in V', \omega \in \Omega, \tag{2.20}$$

$$x^{\omega}_{i,n+1} \in \mathbb{B} \ \forall i \in V', \omega \in \Omega, \tag{2.21}$$

$$x^{\omega}_{ij} + x^{\omega}_{ji} \in \mathbb{B} \ \forall i < j, i \in V', j \in V', \omega \in \Omega, \tag{2.22}$$

$$x^{\omega}_{ij} \in [0,1] \ \forall (i,j) \in A, \omega \in \Omega. \tag{2.23}$$

Proposition 2.1. *All integer feasible solutions to* (2.1)-(2.14), (2.16), (2.17), (2.19), *and* (2.20)-(2.23) *satisfy*

$$x^{\omega}_{ij} \in \mathbb{B} \ \forall (i,j) \in A, \omega \in \Omega. \tag{2.15}$$

Proof. See Appendix 2.A. □

Proposition 2.1 suggests that for any $i, j \in V'$ instead of branching on whether a single directed arc is used, we may also branch on whether a connection between i and j (regardless of direction) is used. This decision can be made per connection, as $x^{\omega}_{ij} \in \mathbb{B}$ and $x^{\omega}_{ji} \in \mathbb{B}$ together imply $x^{\omega}_{ij} + x^{\omega}_{ji} \in \mathbb{B}$, $x^{\omega}_{ij} \in [0,1]$ and $x^{\omega}_{ji} \in [0,1]$. That is, Proposition 2.1 is still applicable if for some connections we branch on both x^{ω}_{ij} and x^{ω}_{ji} and for the other connections we branch on $x^{\omega}_{ij} + x^{\omega}_{ji}$.

Branching on x_{ij} means that we partition the feasible region in two parts. In the first part it is assumed that j is visited directly after i by the same vehicle. In the other part we assume j is not visited directly after i. Knowing whether j is visited directly after i has important implications for the time-of-service variables t_i^ω and t_j^ω. The value $(s_j - e_i)$ in the MTZ-inequalities is essentially a big-M, hence fractional values of the flow variables cause the time constraints to be very weak or inactive. When $x_{ij}^\omega = 1$, however, we have $t_j^\omega \geq t_i^\omega + \tau_{ij}$. Especially when τ_{ij} is big, this inequality has a big effect on the time-of-service variables.

If the value of τ_{ij} is close to zero, this argument no longer holds, whether $x_{ij} = 1$ or $x_{ji} = 1$ is not that important for the time-of-service variables. For the capacity constraints it is important to know whether i and j are visited by the same vehicle, but it is less important to know in which order this happens. Hence, it makes sense to branch on $x_{ij} + x_{ji}$ to split the feasible region in two parts: one part in which j is visited directly after i, or the other way around, and one part in which there is no direct connection between i and j.

To take these effects into account, we introduce the parameter $\rho \in [0, 1]$. Then, for the fraction ρ of all arcs with the shortest travel time, we branch on the connections. For the other arcs, we branch on x_{ij}^ω and x_{ji}^ω separately. In our implementation, we leave the choice for which arc or connection to branch on to the MIP solver CPLEX. Note that $\rho = 0$ corresponds to always branching on individual arcs. If $\rho = 1$, we always branch on connections. In our computational experiments, we vary ρ to find a good compromise between the number of variables and the strength of the LP relaxation.

2.4 Precedence inequalities

Through the assignment of time windows, the TWAVRP connects the VRPTWs corresponding to each of the scenarios. The valid inequalities discussed earlier apply separately to the VRPTWs in each scenario. In this section, we present a novel set of valid inequalities, the *precedence inequalities*, in which each inequality involves multiple scenarios. First, we make some important observations.

If we want to visit first i and later j, both within their respective time windows, then we have to leave i after y_i, and arrive at j before $y_j + u_j$. Let $\bar{\tau}_{ij}$ be the maximum time between these visits, for all $i, j \in V'$. That is, we define $\bar{\tau}_{ij} = (y_j + u_j) - y_i$ and similarly $\bar{\tau}_{ji} = (y_i + u_i) - y_j$. It follows that $\bar{\tau}_{ij} + \bar{\tau}_{ji} = u_i + u_j$.

Now consider a solution to the TWAVRP for which in scenario ω there is a route

visiting first i, and later j. Furthermore, assume there is a different scenario ω' with a route visiting first j, and later i. It follows that the time taken to get from i to j in one scenario, plus the time taken to get from j to i in another scenario, can be at most the sum of the widths of the time windows, $u_i + u_j$.

We formalize this in Observation 2.2. Let A_p be the set of arcs used by path p in G and let $\mathscr{P}_{ij}$ be the set of all elementary paths in G starting at $i \in V$ and ending at $j \in V$.

Observation 2.2. *For given vertices $i, j \in V'$ $(i \neq j)$, for any integer feasible solution to the TWAVRP in which both path $p \in \mathscr{P}_{ij}$ is used in scenario $\omega \in \Omega$ and path $q \in \mathscr{P}_{ji}$ is used in scenario $\omega' \in \Omega$ the following holds:*

$$\sum_{(k,l)\in A_p} \tau_{kl} + \sum_{(k,l)\in A_q} \tau_{kl} \leq u_i + u_j. \tag{2.24}$$

To construct valid inequalities based on Observation 2.2, we make use of Theorem (4.5) in Ascheuer et al. (2000), which we restate for the TWAVRP as the following lemma.

Lemma 2.3. *For any integer feasible TWAVRP solution, a set of clients $S \subseteq V'$ and two vertices $i, j \in V' \backslash S$ $(i \neq j)$, a single vehicle visits i first, then all clients in S and then j consecutively in scenario $\omega \in \Omega$ if and only if:*

$$\sum_{l\in S} x^{\omega}_{il} + \sum_{k\in S}\sum_{l\in S} x^{\omega}_{kl} + \sum_{k\in S} x^{\omega}_{kj} + x^{\omega}_{ij} = |S| + 1. \tag{2.25}$$

Lemma 2.3 gives us a criterion for testing whether there is a path visiting client i, then visiting a subset of other clients, and then visiting client j. Combining this lemma with Observation 2.2 allows us to formulate the precedence inequalities.

Let us denote by $(S : T)$ the set of arcs in A which start in S and end in T, for vertex sets S and T. If S or T is a singleton, we just write the element, e.g. $(i : T)$. For notational convenience we introduce the sets $\mathcal{S}(i, j) = \{S \mid S \subseteq V' \backslash \{i, j\}\}$ for all $i, j \in V'$. That is, $\mathcal{S}(i, j)$ is the set of all possible subsets of clients not containing clients i and j. When traveling from i to j, visiting exclusively clients from S, only the arcs in $(i : S) \cup (S : S) \cup (S : j) \cup (i : j)$ are relevant. Therefore, we introduce $\mathcal{F}(i, S, j) = \{F \mid F \subseteq (i : S) \cup (S : S) \cup (S : j) \cup (i : j)\}$, which for given $i, j \in V'$ and $S \in \mathcal{S}(i, j)$ is the set of all possible subsets of these arcs.

Furthermore, let $\delta_{ij}(S, F)$ be the shortest possible travel time from client $i \in V'$ to client $j \in V'$, visiting all clients in $S \in \mathcal{S}(i, j)$ in between, using only arcs from

set $F \in \mathcal{F}(i, S, j)$. If no such path exists $\delta_{ij}(S, F) = \infty$. We then arrive at the main theorem for the precedence inequalities:

Theorem 2.4. ***Precedence inequalities:*** *For given scenarios $\omega, \omega' \in \Omega$ $(\omega \neq \omega')$, given clients $i, j \in V'$ $(i \neq j)$, given vertex set $S \in \mathcal{S}(i, j)$, corresponding to clients visited in scenario ω, and vertex set $S' \in \mathcal{S}(j, i)$ corresponding to clients visited in scenario ω', and given arc sets $F \in \mathcal{F}(i, S, j)$ and $F' \in \mathcal{F}(j, S', i)$ such that $\delta_{ij}(S, F) + \delta_{ji}(S', F') > u_i + u_j$, the following are valid inequalities:*

$$\sum_{(k,l)\in F} x_{kl}^{\omega} + \sum_{(k,l)\in F'} x_{kl}^{\omega'} \leq |S| + |S'| + 1. \tag{2.26}$$

Proof. This is a direct application of Observation 2.2. Lemma 2.3 shows that Observation 2.2 is contradicted if and only if $\sum_{(k,l)\in F} x_{kl}^{\omega} + \sum_{(k,l)\in F'} x_{kl}^{\omega'} = |S| + |S'| + 2$. By integrality of the x-variables, the theorem follows. □

It is possible to generalize this result by redefining $\delta_{ij}(S, F)$ to be the minimum travel time to visit client i, all clients in S and then client j using only arcs of F, but only using paths that can be feasible when considering the exogenous time windows. We choose not to present this generalization, as we consider an application in which the exogenous time windows are in general very wide. The proposed generalization is then unlikely to add much value, while making it more complex to identify violated inequalities.

Clearly, the number of possible precedence inequalities is exponential in the number of clients. Hence, it is not efficient to add all precedence inequalities to the formulation directly. Instead, we separate the precedence inequalities in a cutting plane fashion. Note that exact separation of the precedence inequalities is difficult, as finding violated precedence inequalities is co-NP-hard, which is proven in Appendix 2.B. For this reason, we separate subsets of the precedence inequalities exactly, and we present heuristics for more general precedence inequalities.

Before we consider these subsets, we state the following two lemmas, which will be useful when deriving our separation algorithms. We provide a lower bound on the flow in F for violated inequalities and we show that for every violated inequality F is cyclic or F contains an elementary (i, j)-path visiting all vertices in S.

Lemma 2.5. *Let $\omega, \omega' \in \Omega$, $i, j \in V'$, $S \in \mathcal{S}(i, j)$, $S' \in \mathcal{S}(j, i)$, $F \in \mathcal{F}(i, S, j)$ and $F' \in \mathcal{F}(j, S', i)$ correspond to a violated precedence inequality, for a feasible solution*

to the LP relaxation of the formulation (2.1)-(2.16), (2.17) *and* (2.19). *Then both*

$$\sum_{(k,l)\in F} x^{\omega}_{kl} > |S| \tag{2.27}$$

and

$$\sum_{(k,l)\in F'} x^{\omega'}_{kl} > |S'|. \tag{2.28}$$

Proof. By definition of $\mathcal{F}(i,S,j)$ all directed arcs in F point to vertex j, or to a vertex in S. Thus, by the flow conservation constraints it follows that the total flow in F is bounded by $|S|+1$. Hence, we have $\sum_{(k,l)\in F} x^{\omega}_{kl} \leq |S|+1$ and similarly $\sum_{(k,l)\in F'} x^{\omega'}_{kl} \leq |S'|+1$. From Theorem 2.4 it follows that for a violated precedence inequality $\sum_{(k,l)\in F} x^{\omega}_{kl} + \sum_{(k,l)\in F'} x^{\omega'}_{kl} > |S|+|S'|+1$. Combining these facts proves the lemma. □

Lemma 2.6. *Let* $\omega,\omega' \in \Omega$, $i,j \in V'$, $S \in \mathcal{S}(i,j)$, $S' \in \mathcal{S}(j,i)$, $F \in \mathcal{F}(i,S,j)$ *and* $F' \in \mathcal{F}(j,S',i)$ *correspond to a violated precedence inequality, for a feasible solution to the LP relaxation of the formulation* (2.1)-(2.16), (2.17) *and* (2.19). *Then* F *contains a cycle or* F *contains an elementary* (i,j)*-path through all vertices of* S. *Also* F' *contains a cycle or* F' *contains an elementary* (j,i)*-path through all vertices of* S'.

Proof. We prove this statement for F, as for F' the proof is analogous. If F contains a cycle, the lemma holds. Next, we assume F is acyclic. Moreover, we assume that F does not contain an elementary path from i to j through all vertices of S, and we show that this leads to a contradiction.

Because F is acyclic, the vertices of S can be relabeled $v_1, v_2, \ldots, v_{|S|}$ such that if $l < k$ then $(v_k, v_l) \notin F$ (see Kahn (1962)). By assumption, there is no elementary path from i through all $v_1, v_2, \ldots, v_{|S|}$ to j. Hence, there exists an integer $g \in \{1,2,\ldots,|S|-1\}$ such that there is no arc from v_g to v_{g+1}.

Let $U_1 = \{i, v_1, v_2, \ldots, v_{g-1}\}$ and let $U_2 = \{v_{g+2}, v_{g+3}, \ldots, v_{|S|}, j\}$. By construction, we have that

$$\begin{aligned}\sum_{(k,l)\in F} x^{\omega}_{kl} &= \sum_{(k,l)\in \left(U_1:(U_1\cup v_g\cup v_{g+1}\cup U_2)\right)\bigcap F} x^{\omega}_{kl} + \sum_{(k,l)\in \left((v_g\cup v_{g+1}\cup U_2):U_2\right)\bigcap F} x^{\omega}_{kl} \\ &\leq |U_1| + |U_2| = |S|. \end{aligned} \tag{2.29}$$

This follows because the total outflow of the vertices in U_1 is bounded by $|U_1|$ due to the flow conservation constraints. Similarly, the total inflow of the vertices in U_2

is bounded by $|U_2|$. Hence, it follows that the total flow captured by F is bounded by $|U_1| + |U_2| = |S|$.

This contradicts Lemma 2.6, which states that $\sum_{(k,l)\in F} x^{\omega}_{kl} > |S|$. Thus, our assumption that F does not contain an elementary path from i through S to j is false. Hence, we have proven that F contains a cycle, or contains an elementary (i,j)-path visiting all clients in S. □

2.4.1 Path precedence inequalities

The first subset of precedence inequalities we consider, is the subset for which F and F' both form a single elementary path, which we refer to as the path precedence inequalities. We make use of the following proposition:

Proposition 2.7. *For any integer feasible solution to the TWAVRP:*

$$\sum_{(k,l)\in A_p} \tau_{kl} + \sum_{(k,l)\in A_q} \tau_{kl} > u_i + u_j \implies \sum_{(k,l)\in A_p} x^{\omega}_{kl} + \sum_{(k,l)\in A_q} x^{\omega'}_{kl} \leq |A_p| + |A_q| - 1$$

$$\forall (i,j) \in A, p \in \mathscr{P}_{ij}, q \in \mathscr{P}_{ji}, \omega \in \Omega, \omega' \in \Omega. \tag{2.30}$$

Proof. This is a direct application of Theorem 2.4 with $F = A_p$ and $F' = A_q$. Note that $\delta_{ij}(S,F) = \sum_{(k,l)\in A_p} \tau_{kl}$, as following the path p is the only way to visit all vertices in S using only vertices in F. Analogously, $\delta_{ji}(S',F') = \sum_{(k,l)\in A_q} \tau_{kl}$. Finally, note that $|S| = |A_p| - 1$ and $|S'| = |A_q| - 1$, and hence $|S| + |S'| + 1 = |A_p| + |A_q| - 1$. □

Proposition 2.7 defines valid inequalities for paths only, instead of for arbitrary sets of vertices and arcs. Note that the precedence inequality is completely determined by the two scenarios and the two paths. Next, we show some properties of the path precedence inequalities. These properties are then used to prove that for a given solution to the LP relaxation, all violated path precedence inequalities can be found in polynomial time.

Lemma 2.8. *All violated path precedence inequalities adhere to the following two inequalities:*

$$\sum_{(k,l)\in A_p} x^{\omega}_{kl} > |A_p| - 1, \tag{2.31}$$

$$\sum_{(k,l)\in A_q} x^{\omega'}_{kl} > |A_q| - 1. \tag{2.32}$$

Proof. This is a direct application of Lemma 2.5 with $F = A_p$ and $F' = A_q$. □

Lemma 2.9. *Let p and q correspond to a violated path precedence inequality. Path p in graph G contains at most one arc (k,l) for which $x^w_{kl} \leq \frac{1}{2}$. Path q contains at most one arc (k',l') for which $x^{\omega'}_{k'l'} \leq \frac{1}{2}$.*

Proof. Suppose p has $m \geq 2$ arcs for which $x^\omega_{kl} \leq \frac{1}{2}$. This implies $\sum_{(k,l)\in A_p} x^\omega_{kl} \leq |A_p| - \frac{1}{2}m \leq |A_p| - 1$. Hence (2.31) is not satisfied. It follows that p has at most one arc for which $x^\omega_{kl} \leq \frac{1}{2}$. The proof for path q is analogous. □

Proposition 2.10. *All violated path precedence inequalities can be found in polynomial time.*

Proof. To find all violated path precedence inequalities, we generate an exhaustive list of candidate paths from i to j which meet the necessary condition given by Lemma 2.9. If we do the same for all candidate paths from j to i, we can check for all combinations of the candidates whether (2.30) is violated.

To generate a list of candidates, we first use Lemma 2.9, which states that for scenario $\omega \in \Omega$ a candidate uses at most one arc for which $x^\omega_{kl} \leq \frac{1}{2}$. Starting at i, the path thus first uses a (possibly zero) number of arcs for which $x^\omega_{kl} > \frac{1}{2}$, followed by zero or one arcs for which $x^\omega_{kl} \leq \frac{1}{2}$. After that, we visit another (possibly zero) number of arcs for which $x^\omega_{kl} > \frac{1}{2}$ before we reach j.

By the flow conservation constraints, the total outflow and the total inflow of a vertex are both equal to one. Hence, at each vertex there can be at most one incoming arc and one outgoing arc for which $x^\omega_{kl} > \frac{1}{2}$. This implies there is at most one elementary path leaving i for which all arcs have an x value larger than $\frac{1}{2}$. Analogously there is at most one elementary path entering j for which all x values are larger than $\frac{1}{2}$. Finding these two elementary paths takes $O(n^2)$ time, as the paths contain $O(n)$ vertices, and, for a single vertex, determining which arc has value larger than $\frac{1}{2}$ takes $O(n)$ time.

All candidate paths from i to j can thus be constructed by starting in i, following the arcs with x values larger than $\frac{1}{2}$ up to a certain point after which an arc with x value less or equal to $\frac{1}{2}$ is taken to arrive at the path of arcs with x values larger than $\frac{1}{2}$ that arrives at j, which is followed until we reach j. That is, without loss of generality we sequentially visit the vertices $i = v_1, v_2, \ldots, v_f, w_g, w_{g-1}, \ldots, w_1 = j$ for integers f and g between 1 and n.

For given f and g, the total flow of the candidate path is given by $\sum_{i=1}^{f-1} x_{v_i v_{i+1}} + x_{v_f w_g} + \sum_{i=1}^{g-1} x_{w_{i+1} w_i}$, and the total travel time is given by $\sum_{i=1}^{f-1} \tau_{v_i v_{i+1}} + \tau_{v_f w_g} +$

$\sum_{i=1}^{g-1} \tau_{w_{i+1}w_i}$. By precalculating the summations for all f and g in $O(n^2)$ time, this part only takes constant time.

For each scenario, there are $O(n^2)$ combinations of f and g. We thus find $O(|\Omega|n^2)$ candidates from i to j. Then, we check for all combinations of candidates from i to j and candidates from j to i if the condition in Proposition 2.7 is satisfied. Per combination, this takes constant time, as we only sum the predetermined values of total flow and total travel time. There are $O((|\Omega|n^2)^2)$ such combinations. As we repeat the procedure for all combinations of two vertices i and j, the total time complexity is $O(|\Omega|^2 n^6)$. □

2.4.2 Tournament precedence inequalities

In the previous section, we have introduced the path precedence inequalities, which are precedence inequalities based on elementary paths. In this section we present a broader subset of the precedence inequalities, in which both (S, F) and (S', F') represent a directed acyclic graph, which leads to stronger valid inequalities. Furthermore, we show that to satisfy all these valid inequalities it is sufficient to restrict ourselves to those directed acyclic graphs F and F' obtained by taking the transitive closures of an elementary path. The transitive closure of a set of arcs $F \subseteq A$ in graph G, is defined as follows:

$$\text{trcl}(F) := \{(k,l) \in A : l \text{ can be reached from } k \text{ using only arcs in } F\}. \tag{2.33}$$

These inequalities are similar to the tournament constraints of Ascheuer et al. (2000), hence we call this class the tournament precedence inequalities. In Ascheuer et al. (2000), the tournament inequalities are introduced for the ATSPTW, which are obtained by bounding the total flow on the transitive closure of a simple path which violates (exogenous) time window constraints or cannot be extended without violating time window constraints. Next we present the tournament precedence inequalities, discuss how to separate them, and show that if all tournament precedence inequalities are satisfied then so are all precedence inequalities based on directed acyclic graphs.

Proposition 2.11. *For any integer feasible solution to the TWAVRP:*

$$\sum_{(k,l)\in A_p} \tau_{kl} + \sum_{(k,l)\in A_q} \tau_{kl} > u_i + u_j \implies \sum_{(k,l)\in \text{trcl}(A_p)} x^{\omega}_{kl} + \sum_{(k,l)\in \text{trcl}(A_q)} x^{\omega'}_{kl} \leq |A_p| + |A_q| - 1$$

$$\forall (i,j) \in A, p \in \mathscr{P}_{ij}, q \in \mathscr{P}_{ji}, \omega \in \Omega, \omega' \in \Omega. \tag{2.34}$$

Proof. This is a direct application of Theorem 2.4 with $F = \text{trcl}(A_p)$ and $F' = \text{trcl}(A_q)$. Observe that taking the transitive closure of an elementary path yields a directed acyclic graph, which contains only a single path visiting all vertices. In particular, $\delta_{ij}(S,F) = \delta_{ij}(S,\text{trcl}(A_p)) = \delta_{ij}(S,A_p) = \sum_{(k,l)\in A_p} \tau_{kl}$. Similarly, $\delta_{ji}(S,F) = \sum_{(k,l)\in A_q} \tau_{kl}$. Note that $|S| = |A_p| - 1$ and $|S'| = |A_q| - 1$, and hence $|S| + |S'| + 1 = |A_p| + |A_q| - 1$. □

Corollary 2.12. *If a path precedence inequality is violated, then its corresponding tournament precedence inequality (by taking transitive closures) is violated as well.*

Proof. Follows directly from Proposition 2.11, the non-negativity of the x variables and $F \subseteq \text{trcl}(F)$ for all $F \subseteq A$. □

As a result, we can find violated tournament precedence inequalities by separating path precedence inequalities. However, not all violated tournament precedence inequalities can be found in this way. Hence, to separate all tournament precedence inequalities, we next present another algorithm.

First, per scenario, we make a list of all elementary paths in G, not involving the depot vertices. By definition, each tournament precedence inequality is characterized by two elementary paths and two scenarios. Hence, after we generate the lists, we can separate the tournament precedence inequalities by combining elementary paths from the lists, and checking the condition given in Proposition 2.11 for each pair.

To construct the list per scenario we use a procedure similar to that described in Ascheuer et al. (2001) to detect violated tournament constraints for the ATSPTW. We enumerate all paths but backtrack as soon as $\sum_{(k,l)\in\text{trcl}(A_p)} x^{\omega}_{kl} \leq |A_p| - 1$. It is suggested in Ascheuer et al. (2001) that only a polynomial number of paths is generated this way, which would imply that our separation routine, involving multiple scenarios, also requires only a polynomial number of iterations.

We have mentioned that, for separating tournament precedence inequalities, restricting to transitive closures of elementary paths still allows us to capture all precedence inequalities based on directed acyclic graphs. We state this formally in the following lemma.

Lemma 2.13. *Let $\omega, \omega' \in \Omega$, $i, j \in V'$, $S \in \mathcal{S}(i,j)$, $S' \in \mathcal{S}(j,i)$, $F \in \mathcal{F}(i,S,j)$ and $F' \in \mathcal{F}(j,S',i)$ correspond to a precedence inequality, for a feasible solution to the LP relaxation of the formulation* (2.1)-(2.16), (2.17) *and* (2.19). *Furthermore,*

assume F and F' are acyclic. If all tournament precedence inequalities are satisfied, then this precedence inequality is also satisfied.

Proof. As F is assumed to be acyclic, by Lemma 2.6 it contains an elementary path $p \in \mathcal{P}_{ij}$ through all vertices of S. By definition of the transitive closure, it follows that $F \subseteq \text{trcl}(A_p)$. Analogously we have $F' \subseteq \text{trcl}(A_q)$ for some $q \in \mathcal{P}_{ji}$ visiting all vertices of S'.

We have $\sum_{(k,l)\in F} x^{\omega}_{kl} + \sum_{(k,l)\in F'} x^{\omega'}_{kl} \leq \sum_{(k,l)\in \text{trcl}(A_p)} x^{\omega}_{kl} + \sum_{(k,l)\in \text{trcl}(A_q)} x^{\omega'}_{kl} \leq |S| + |S'| + 1$, as all tournament precedence inequalities are assumed to be satisfied. It follows that if all tournament precedence inequalities are satisfied, each precedence inequality based on directed acyclic graphs is satisfied as well. □

2.4.3 Additional strategies

We have discussed two subsets of the precedence inequalities with corresponding separation strategies. There are, however, some additional strategies that can be utilized.

Note that both the separation algorithm for the path precedence inequalities and the separation algorithm for the tournament precedence inequalities first generate a list of viable candidates $i, j \in V$, $S \in \mathcal{S}(i,j)$ and $F \in \mathcal{F}(i,S,j)$ per scenario, after which all combinations of candidates are checked to find violated inequalities. An additional strategy is to opportunistically alter these candidates when combining them to create stronger inequalities.

Recall that one way to do this, is by separating path precedence inequalities and taking transitive closures, resulting in tournament precedence inequalities (Corollary 2.12). Another strategy is to complete F by adding arcs. That is, let F^* be the maximum cardinality element of $\mathcal{F}(i,S,j)$. By definition, $F^* = (i:S) \cup (S:S) \cup (S:j)$. Note that F^* is not acyclic, and hence in general $\delta_{ij}(S,F^*) \neq \delta_{ij}(S,F)$. Furthermore, $\delta_{ij}(S,F^*)$ is hard to calculate.

Therefore, we introduce an easy to calculate lower bound on $\delta_{ij}(S,F^*)$. Note that any violated precedence inequality found while using this lower bound is valid for the actual value of $\delta_{ij}(S,F^*)$ as well. It is well known that the weights of a minimum spanning tree can be used as a lower bound on the length of the shortest elementary path visiting all vertices. It thus follows that:

$$\delta_{ij}(S,F^*) \geq \min_{k\in S}\{\tau_{ik}\} + \text{MST}(S) + \min_{k\in S}\{\tau_{kj}\}, \tag{2.35}$$

in which $\text{MST}(S)$ represents the weight of the minimum weight spanning tree of an

undirected complete graph with vertex set S and edge weight $\min\{\tau_{kl}, \tau_{lk}\}$ for each edge (k, l).

We consider the following strategies. First, separate either path precedence inequalities or tournament precedence inequalities. The path precedence inequalities may be converted to tournament precedence inequalities. Each resulting tournament precedence inequality corresponds to sets $i, j \in V'$, $S \in \mathcal{S}(i, j)$, $F \in \mathcal{F}(i, S, j)$, $S' \in \mathcal{S}(j, i)$ and $F \in \mathcal{F}(j, S, i)$. Now try whether a violated precedence inequality can be obtained by replacing F by F^* and/or F' by F'^*, using the lower bounds on travel time given by (2.35). If so, use these stronger valid inequalities.

2.5 Numerical experiments

In this section we present the results of our numerical experiments to test the effectiveness of our new formulation, the precedence inequalities and the branching strategy. Furthermore, we present experiments in which our algorithm is compared to the branch-price-and-cut algorithm of Spliet and Gabor (2015).

All experiments are run on an Intel i7 3.5GHz computer with 16GB of RAM. To allow for a fair comparison between algorithms, we restrict all experiments to a single thread on a single core. As a basis for our implementation, we use the commercial solver CPLEX version 12.5, with default settings. We disable all CPLEX's built in valid inequalities, so we can more accurately test the effect of the valid inequalities discussed in this chapter.

Our own valid inequalities will be generated in a callback, which is called each time the LP relaxation has been solved, or re-solved after adding valid inequalities. In this callback we separate rounded capacity inequalities and precedence inequalities, and only afterwards the LP is resolved. We use the built-in 'traditional branch-and-cut' in combination with our own branching strategy as discussed in Section 2.3.

For our branch-and-cut algorithm we use the 64 bit version of CPLEX, which allows for the full 16GB of memory to be used. The algorithm of Spliet and Gabor (2015) requires less memory, and hence we use the 32 bit version of CPLEX, which gives a slightly better performance.

We use a one hour time limit per instance in all experiments. From preliminary tests we have found that almost every instance is unsolvable within the time limit without separating rounded capacity inequalities, so we separate those in all experiments.

2.5.1 Test-instances

First, we introduce the different sets of test-instances which we use for our numerical experiments.

2.5.1.1 Small instances

We use forty instances introduced by Spliet and Gabor (2015). These instances are randomly generated instances, inspired by a Dutch retail chain. The set contains ten instances of 10, 15, 20 and 25 clients, respectively. The clients are uniformly distributed over a square with sides of length five. Both the starting depot and the ending depot are located in the center of the square. The travel cost and the travel time in hours between two points in the square is equal to the Euclidean distance.

Each instance includes three demand scenarios, each with equal probability of occurrence. The average demand is about 1/6 vehicle load. The exogenous time windows are rather wide: On average the exogenous time window of the client has width 10.8, compared to an endogenous time window width of 2.

2.5.1.2 Large instances

To be able to test our branch-and-cut algorithm on larger instances as well, we have generated fifty additional instances in the same way that the small instances have been generated. That is, we created ten instances of 30, 35, 40, 45 and 50 clients respectively. All instances are available online.

2.5.2 Branch-and-cut experiments

Next, we compare the branching strategies and the separation algorithms for the precedence inequalities, using the forty small instances. We consider six different strategies to separate precedence inequalities, which have been detailed in Section 2.4.3:

N Do not separate precedence inequalities.

P Separate path precedence inequalities.

P2T 1) Separate path precedence inequalities.
2) Turn them into tournament precedence inequalities.

P2C 1) Separate path precedence inequalities.
2) Turn them into tournament precedence inequalities.

3) Complete F and/or F' by adding additional arcs and add corresponding violated precedence inequalities if they are found.

T Separate tournament precedence inequalities.

T2C 1) Separate tournament precedence inequalities.
2) Complete F and/or F' by adding additional arcs and add corresponding violated precedence inequalities if they are found.

For the branching strategy, we conduct experiments for $\rho \in \{0, 0.1, ..., 1\}$. Recall that this corresponds to a strategy in which we branch on connections for the the fraction ρ of arcs with the shortest travel time. For the other connections, we branch on arcs.

In Table 2.1 we have reported solution times for all combinations of separation and branching strategies. These numbers are aggregated values obtained by computing the average over the forty instances. All instances were solved to optimality.

Seconds	Separation strategy					
ρ	N	P	P2T	P2C	T	T2C
0.0	31.8	27.2	20.2	18.6	33.9	36.2
0.1	13.7	10.4	7.1	7.6	8.2	8.7
0.2	13.5	5.7	9.3	5.4	9.3	7.3
0.3	13.2	6.8	6.5	5.5	7.5	8.8
0.4	12.0	10.1	6.3	6.9	8.4	8.0
0.5	11.1	7.2	5.7	6.5	8.7	7.8
0.6	18.0	9.3	8.8	4.8	7.6	7.0
0.7	10.4	8.5	7.9	6.8	10.2	7.8
0.8	21.4	9.8	9.9	7.1	9.9	10.6
0.9	12.3	9.5	8.7	5.3	8.6	6.0
1.0	22.5	11.9	9.6	6.2	9.0	5.4

Table 2.1: Average solution times for various strategies.

We see that strategies P2T and P2C are the best performers for most choices of ρ. Setting $\rho \neq 0$ yields a positive effect for all separation strategies, although the exact value of ρ does not seem to be that important. In the remainder of the experiments, we will use the combination of $\rho = 0.6$ and P2C, as this combination yields the lowest average solution time on our test set. If we compare the solution time of the combination of $\rho = 0.6$ and P2C to the solution time of the combination of N and $\rho = 0$, we see that introducing the precedence inequalities and the branching strategy together yields a factor 6.6 improvement in solution time.

If we allow CPLEX to use up to eight threads and if we enable all the default cuts, the combination of N and $\rho = 0$ results in an average solution time of 9.7 seconds per instance. This is a significantly better than the original 31.8 seconds, as shown in Table 2.1. However, the combination of P2C and $\rho = 0.6$ still results in a better average solution time of 4.8 seconds, and only requires a single thread and none of the default CPLEX cuts.

In Table 2.2, results of using the different separation strategies are presented for $\rho = 0.6$. The rows present the average solution time in seconds, the average number of visited nodes of the search trees, the average number of precedence inequalities and rounded capacity inequalities added, and the percentages of the total solution time used for the separation of the precedence inequalities and for the rounded capacity inequalities. The last row displays for how many out of the forty instances the used separation strategy yielded the shortest computation time. If multiple strategies have the same solution time (rounded to milliseconds), they are all counted as best strategies.

	N	P	P2T	P2C	T	T2C
Seconds	18.0	9.3	8.8	4.8	7.6	7.0
Nodes	2,481	1,406	1,144	537	848	696
Precedence inequalities	0	117	107	66	97	101
Rounded capacity inequalities	575	426	413	341	405	390
% of time sep. prec.	0.0%	5.4%	6.2%	4.4%	8.0%	8.0%
% of time sep. cap.	2.7%	3.3%	3.0%	2.7%	3.0%	2.6%
Best strategy	6/40	9/40	9/40	14/40	7/40	14/40

Table 2.2: Average branch-and-cut statistics for various separation strategies ($\rho = 0.6$).

If we look at the total solution time, it becomes clear that all strategies yield an improvement over strategy N. Based on the tested instances, strategy P2C yields the lowest computation time; a factor 3.8 better than strategy N.

Looking at the number of nodes in the search trees, we see that all strategies allow the number of nodes to be greatly reduced compared to strategy N. We see that, on average, using T2C allows for the largest number of violated precedence inequalities to be found per node in the search tree. Still, strategy P2C is more effective. This observation cannot be explained by the increase in separation time alone; the average time spend per instance on separating precedence inequalities is 0.2 seconds for P2C, and 0.6 seconds for T2C.

There are two other effects that may explain the difference between P2C and T2C. First, as more precedence inequalities are found, larger LP relaxations have to be solved, which takes more time. Second, as more precedence inequalities can be found, it can happen that the LP relaxation is resolved more often. It seems that this additional work does not add much value over P2C.

Table 2.2 shows that P2C is the best strategy for 14 of the test-instances. For T2C, this number is the same. Looking at the disaggregated data (Table 2.5 in Appendix 2.C) we see that for the instances with 20 clients, T2C is the best strategy 5 out of 10 times, while P2C is never the best strategy. For the instances with 25 clients, however, P2C is the best strategy 5 out of 10 times, while T2C is the best only once.

Surprisingly, after processing only the root node, per instance there is almost no difference in lower bounds between strategy N and the other strategies. For instance 32, P2C is able to close the root gap, where the other strategies remain with a gap of 0.27%. For four other instances, differences between lower bounds of at most 0.04% are observed. Finally, for 35 out of the 40 instances, the lower bounds are exactly the same for all six strategies.

The power of the precedence inequalities really shows further down in the search tree. Indeed, their inclusion decreases both the solution time, and the number of nodes in the search tree, as can be seen from Table 2.2. This could be explained by the nature of the precedence inequalities: they disallow certain combinations of paths. If a solution is very fractional, not many paths can be detected, and hence the precedence inequalities are of little use. Deeper in the tree, where more variables are fixed, they become more effective.

2.5.3 Comparison with branch-price-and-cut

Next, we compare the performance of our branch-and-cut algorithm, using strategy P2C and $\rho = 0.6$, to the performance of the branch-price-and-cut algorithm in Spliet and Gabor (2015).

To this end, we run their implementation on the same computer as on which our algorithm is run. The computation times are thus directly comparable. The results we present are based on their branch-price-and-cut algorithm with 2-cycle elimination and adding rounded capacity cuts, which is the solution method in Spliet and Gabor (2015) yielding the best average time performance on the test set.

The results per instance can be found in Table 2.3. Per instance, data is provided on five different categories, both for the branch-price-and-cut algorithm (BP&C) and

the branch-and-cut algorithm (B&C). The columns labeled 'Seconds' indicate the times in seconds for solving the instance to optimality, the maximum time allowed being one hour. The columns 'Nodes' state the number of nodes of the search tree that have been explored during that time. When the algorithm terminates, the percentual deviation from the optimum is given in the column 'Optimality gap'. A value of zero indicates the problem was solved to optimality. The column 'Root gap' shows a similar value, indicating the optimality gap after processing only the root node. The optimality gap and the root gap are calculated ex-post, using the actual optimal value. Finally, the column 'Value' gives the optimal objective value for that instance.

What immediately stands out is the enormous decrease in computation time of our new algorithm with respect to the previous algorithm. The instances that can be solved to optimality by the branch-price-and-cut algorithm, can be solved to optimality by the branch-and-cut algorithm 89.6 times faster on average. In total, 9 instances cannot be solved to optimality by the branch-price-and-cut algorithm after a total of 9 hours of computation time. With the branch-and-cut algorithm, all these instances can be solved to optimality in 137.9 seconds. Hence, this is a speedup of at least a factor 234.9.

The new algorithm is faster for all tested instances by at least a factor 7.6 and even up to a factor 2997 for instance 21. If we consider the time necessary to attempt to solve all instances combined, the total time decreases from 37,255.3 seconds in total to 192.1 seconds; a speedup factor of 193.9. We point out that this difference is not the result of using rounded capacity cuts, as both the branch-and-cut algorithm and the branch-price-and-cut algorithm use these valid inequalities.

It can be seen that an advantage of the branch-price-and-cut algorithm is the stronger LP bound it provides, as for almost all instances the root gap is smaller than the gap given by the branch-and-cut algorithm. However, this strength is less apparent in the remainder of the search tree, as on average the branch-and-cut algorithm processes only twice the number of nodes in the branch-price-and-cut algorithm. There are some extreme instances, however, where processing a lot of nodes is necessary, e.g., instances 12 and 36. Here, we observe that the branch-and-cut algorithm is able to process a large number of nodes in little time (14000+ in less than 40 seconds for instance 12). The branch-price-and-cut algorithm generates stronger bounds, but cannot process enough nodes in the given time to solve the problem.

		Seconds		Nodes		Optimality gap		Root gap		Value
Inst.	Clients	BP&C	B&C	BP&C	B&C	BP&C	B&C	BP&C	B&C	B&C
1	10	0.7	0.0	1	1	0	0	0	0	17.65
2	10	121.9	0.1	483	18	0	0	0.17	0.28	15.56
3	10	3.7	0.0	1	1	0	0	0	0	17.42
4	10	28.5	0.1	193	6	0	0	0.14	0.14	18.51
5	10	2.3	0.3	2	42	0	0	0	0.34	16.07
6	10	1.5	0.0	2	1	0	0	0	0	18.00
7	10	4.9	0.0	4	1	0	0	0	0	17.02
8	10	3.5	0.1	29	21	0	0	0.65	0.96	23.89
9	10	3.0	0.0	7	1	0	0	0	0	20.31
10	10	5.9	0.0	5	1	0	0	0	0	16.31
11	15	87.4	0.1	22	1	0	0	0	0	17.78
12	15	3,600.0	39.1	889	14,037	0.15	0	0.67	2.36	27.10
13	15	3,600.0	2.6	684	587	0.59	0	1.10	1.78	29.37
14	15	58.0	0.2	45	1	0	0	0	0.03	23.18
15	15	29.4	0.6	36	11	0	0	0	0.17	24.15
16	15	92.4	0.3	98	1	0	0	0.10	0.17	21.03
17	15	22.9	0.1	15	1	0	0	0	0	22.04
18	15	105.3	0.8	98	124	0	0	0.20	0.47	22.30
19	15	133.3	1.3	133	210	0	0	0.56	0.97	26.52
20	15	41.6	0.4	28	11	0	0	0	0	22.11
21	20	3,600.0	1.2	864	48	0.02	0	0.57	1.11	28.08
22	20	152.3	9.0	62	658	0	0	0.03	0.19	29.80
23	20	99.6	0.4	40	1	0	0	0	0.12	30.30
24	20	112.2	1.7	27	58	0	0	0.03	0.86	24.16
25	20	3,600.0	6.9	712	389	0.08	0	0.61	1.09	29.84
26	20	65.4	0.2	16	1	0	0	0	0	29.72
27	20	85.5	0.3	24	1	0	0	0	0	26.48
28	20	106.5	1.1	36	11	0	0	0	0.08	26.14
29	20	65.5	0.5	17	1	0	0	0	0.05	26.61
30	20	45.1	0.3	4	1	0	0	0	0	26.36
31	25	610.9	2.3	121	20	0	0	0.13	0.57	31.43
32	25	840.0	1.3	164	4	0	0	0.07	0	30.71
33	25	3,600.0	9.4	413	395	0.33	0	0.45	1.03	33.71
34	25	193.2	11.1	36	391	0	0	0	0.33	33.34
35	25	640.0	6.1	119	201	0	0	0	0.85	29.05
36	25	3,600.0	39.3	1,662	1,733	0.13	0	0.43	1.49	30.50
37	25	3,600.0	22.4	278	1,472	0.29	0	0.29	0.43	28.68
38	25	3,600.0	9.7	1,259	337	0.12	0	0.30	0.59	35.69
39	25	3,600.0	7.2	2,294	219	0	0	0.50	0.94	32.55
40	25	1,093.2	15.2	521	471	0	0	0.30	0.62	32.14
Average		931.4	4.8	286.1	537.2	0.04	0	0.18	0.45	25.29

Table 2.3: Comparison of branch-price-and-cut and branch-and-cut.

2.5.4 Performance on larger instances

As our branch-and-cut algorithm is able to solve all 40 test-instances used in Spliet and Gabor (2015), we also test our algorithm on larger instances, as introduced in Section 2.5.1. Recall that these instances are generated in a similar way as the first forty instances. The only difference is that the number of clients is larger: between 30 and 50.

In Table 2.4 we report results for the instances with 30 and 35 clients. This table is structured in the same way as Table 2.3. The column 'Value' is replaced by the columns 'Lower bound' and 'Upper bound', as not all instances can be solved to optimality. The reported root gaps are no longer ex post, but based on the best known upper bound.

It can be seen that all but one of the instances with 30 clients can be solved to optimality within one hour of computation time, and for the remaining instance we find a solution with an optimality gap of 1.66%.

The instances with 35 clients are more difficult: 6 out of the 10 instances can be solved to optimality within one hour. The remaining instances have an optimality gap of less than 1.32%.

The instances with 40, 45 and 50 clients cannot be solved consistently by our branch-and-cut algorithm, which proves optimality of the found solution for only two of these instances in one hour of computation time. In Table 2.6 in Appendix 2.C, we report the results for these instances, including best found lower and upper bounds. The instances with 40 clients have optimality gaps below 2.09%. The instances with 45 clients have optimality gaps between 0.48% and 7.12%. The instances with 50 clients have optimality gaps between 3.09% and 7.23%.

2.6 Conclusion

In this chapter we present a compact formulation for the TWAVRP based on the 2-commodity flow formulation introduced by Baldacci et al. (2004) and the MTZ-inequalities introduced by Miller et al. (1960). We use this formulation in a branch-and-cut algorithm in which rounded capacity cuts are separated in each node of the search tree.

To further improve the performance of our algorithm, we introduce a branching rule and a novel class of valid inequalities: the precedence inequalities. These TWAVRP specific inequalities are hard to separate in general. Therefore, we introduce exact separation algorithms for two subsets, the path precedence inequalities

Inst.	Clients	Seconds	Nodes	Optimality gap	Root gap	Lower bound	Upper bound
41	30	137.0	3,281	0	1.99	36.38	36.38
42	30	3,600.0	40,701	1.66	3.48	34.16	34.74
43	30	187.6	8,317	0	1.91	35.48	35.48
44	30	60.6	1,296	0	1.46	35.88	35.88
45	30	110.8	4,158	0	1.78	35.55	35.55
46	30	7.7	165	0	0.58	37.47	37.47
47	30	17.8	292	0	0.70	32.54	32.54
48	30	357.9	9,657	0	1.84	36.32	36.32
49	30	930.3	24,789	0	2.34	35.30	35.30
50	30	30.7	710	0	0.72	40.27	40.27
51	35	18.2	114	0	0.78	43.46	43.46
52	35	14.0	69	0	0.06	41.84	41.84
53	35	3,600.0	32,201	1.32	2.86	44.54	45.14
54	35	3,600.0	50,601	0.87	3.06	41.20	41.57
55	35	68.5	983	0	0.68	37.92	37.92
56	35	3,600.0	42,289	0.92	2.58	44.08	44.49
57	35	3,600.0	31,201	0.44	1.85	40.65	40.83
58	35	127.9	1,871	0	0.89	41.22	41.22
59	35	245.1	2,485	0	0.93	43.43	43.43
60	35	443.3	6,655	0	1.32	42.27	42.27

Table 2.4: Result for the branch-and-cut algorithm on instances 41 to 60.

and the tournament precedence inequalities. Furthermore, we extend these algorithms to separation heuristics for general precedence inequalities. Using the branching rule and separating precedence inequalities makes our algorithm 6.6 times faster.

The new algorithm is superior to the best known algorithm in the literature, the algorithm of Spliet and Gabor (2015), for all tested instances. Overall, an average speedup factor of 193.9 is achieved.

Finally, we test our algorithm on larger instances. Of the instances with 30 clients, 9 out of 10 instances could be solved to optimality within the one hour time limit. For the instances with 35 clients, we found the optimal solution for 6 out of 10 instances. The instances that could not be solved to optimality all have an optimality gap of less than 1.66%. Instances with 40 clients and more, however, could not be solved to optimality consistently.

In this chapter, we compare our results to the branch-price-and-cut algorithm of Spliet and Gabor (2015). Even without the use of precedence inequalities, our algorithm shows a substantial speedup over the branch-price-and-cut algorithm. It it still interesting, though, to investigate the effect of incorporating the precedence inequal-

ities in a branch-price-and-cut algorithm. Similarly, it is interesting to see how the precedence inequalities would perform on the DTWAVRP. Proposition 2.7 explains which paths directly exclude each other. When the paths themselves are variables in the formulation, as is the case in a set partitioning formulation, this proposition might be used to derive stronger valid inequalities.

Another interesting topic of further research concerns algorithms for separating precedence inequalities. In the future, TWAVRP algorithms would benefit from new algorithms for separating the remaining class of precedence inequalities, corresponding to directed cyclic graphs.

Appendix

2.A Proof of Proposition 2.1

Proposition 1. *All integer feasible solutions to* (2.1)-(2.14)*,* (2.16)*,* (2.17)*,* (2.19)*, and* (2.20)-(2.23) *satisfy*

$$x_{ij}^{\omega} \in \mathbb{B} \ \forall (i,j) \in A, \omega \in \Omega. \tag{2.15}$$

Proof. Constraints (2.20) and (2.21) state that $x_{0j}^{\omega}, x_{j,n+1}^{\omega} \in \mathbb{B}$ for all $j \in V'$. Next, we will prove that if $x_{ij}^{\omega} = 1$ with $i \in V' \cup \{0\}$ and $j \in V'$, then there exists a unique $k \in V' \cup \{n+1\}$, $k \neq i$ such that $x_{jk}^{\omega} = 1$.

Suppose that $x_{ij}^{\omega} = 1$ for some $i \in V' \cup \{0\}$ and $j \in V'$. Due to the flow conservation constraints $x_{lj}^{\omega} = 0$ for all $l \neq i$ and furthermore there exists a vertex $k \in V' \cup \{n+1\}$ such that $x_{jk}^{\omega} > 0$. If $k = n+1$, then by integrality of the flows going into the depot, we have $x_{jk}^{\omega} = 1$ and by the flow conservation constraints we have $x_{jl}^{\omega} = 0$ for all $l \neq k$. Finally, suppose $k \neq n+1$. Note that $k \neq i$ as $x_{jk} > 0$ while $x_{ji} = 0$ because $x_{ij} + x_{ji} = 1$. Constraints (2.22) and (2.23) state that $x_{jk}^{\omega} + x_{kj}^{\omega} = 1$ and since $x_{lj}^{\omega} = 0$ for all $l \neq i$, it follows that $x_{jk}^{\omega} = 1$. Using again the flow conservation constraints, $x_{jl} = 0$ for all $l \neq k$.

We know that all flows out of the depot are equal to one. We have just proven that any vertex in V' with a single inflow of 1 also has a single outflow of 1. It follows that all flows between the depots are of size 1.

What remains to be proven is that all clients are contained in the integral flows between the depots. Because of the flow conservation constraints (2.2)-(2.3), the only other possibility is that there exists a cycle, given by edges $(1,2),(2,3),...,(k-$

$1,k),(k,1)$ such that for each arc (i,j) in this cycle we have $x_{ij}^{\omega} + x_{ji}^{\omega} = 1$. Using the flow conservation constraints (2.2)-(2.3), we have that all arcs adjacent to this cycle have zero flow. Constraints (2.4) and (2.16) together state that if $x_{ij} + x_{ji} = 0$, then $z_{ij} = z_{ji} = 0$. Hence, if a cycle exists, (2.5) contains the following constraints:

$$\begin{aligned} z_{k,1}^{\omega} - z_{1,k}^{\omega} + z_{2,1}^{\omega} - z_{1,2}^{\omega} &= 2d_1^{\omega} \\ z_{1,2}^{\omega} - z_{2,1}^{\omega} + z_{3,2}^{\omega} - z_{2,3}^{\omega} &= 2d_2^{\omega} \\ &\vdots \\ z_{k-1,k}^{\omega} - z_{k,k-1}^{\omega} + z_{1,k}^{\omega} - z_{k,1}^{\omega} &= 2d_k^{\omega} \end{aligned}$$

Summing these constraints gives $0 = 2\sum_{i=1}^{k} d_i^{\omega} > 0$, which is a contradiction. Hence, all clients are contained in the integer flows between the depots. It follows that all x-variables are integer and that (2.15) is satisfied. □

2.B Separating precedence inequalities is co-NP-hard

We prove that separating precedence inequalities is co-NP-hard. First, we present a brief outline of the proof.

We construct a specific instance of the TWAVRP and a corresponding optimal solution to the LP relaxation of (2.1)-(2.16), (2.17) and (2.19). This optimal solution to the LP relaxation is an instance of the separation problem of finding violated precedence inequalities. We will refer to this instance as instance I. Next, we characterize this instance of the separation problem as a decision problem. Finally, we show that separating precedence inequalities is co-NP-hard. We show this by a polynomial time reduction from Euclidean TSP.

TWAVRP instance

Consider $n \geq 4$ clients, travel times τ_{ij} adhering to the triangle inequality, and endogenous time window widths u_i. Let τ^{max} be the maximum of the given travel times, and let u^{max} be the maximum of the given endogenous time window widths.

We set $s_i = 0$ and $e_i = 2\tau^{max} + u^{max}$ for all locations $i \in V$ and we define two scenarios $\Omega = \{1,2\}$, both with probability 0.5 of occurring. In both scenarios, we set the demand of every client equal to 1. The vehicle capacity Q is set equal to n. We set all travel costs equal to zero, such that any feasible solution to the LP

relaxation is also an optimal solution.

Solution to the LP relaxation

Next, we describe an optimal solution to the LP relaxation. We start with setting the x-variables. For scenario 1, set $x^1_{1i} = x^1_{ij} = x^1_{in} = \frac{1}{n-2}$ for all $i = 2, 3, \ldots, n-1$ and $j = 2, 3, \ldots, n-1$ $(i \neq j)$ and set $x^1_{01} = x^1_{n,n+1} = 1$. All other x-variables for scenario 1 are set to zero.

For scenario 2, we set $x^2_{01} = x^2_{n,n+1} = 1 - \frac{1}{n-1}$. Furthermore, we set $x^2_{0i} = x^2_{i,n+1} = 1$ for all $i = 2, 3, \ldots n-1$. Moreover, we set $x^2_{0n} = x^2_{1,n+1} = 1$ and $x^2_{n1} = \frac{1}{n-1}$. The remaining flow variables are set to zero.

As an example, Figures 2.1 and 2.2 present the flows given by the x-variables when $n = 4$.

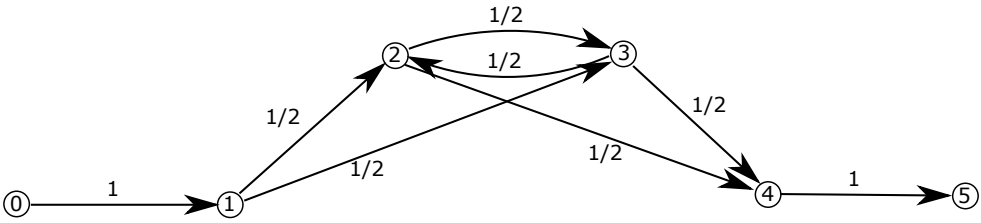


Figure 2.1: Flows given by the x-variables in scenario 1 when $n = 4$.

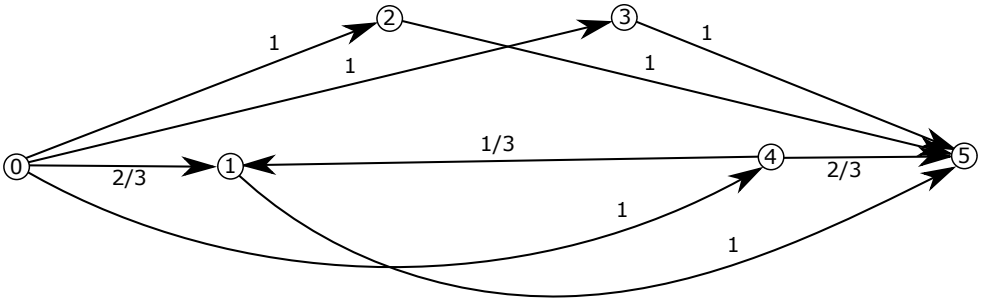


Figure 2.2: Flows given by the x-variables in scenario 2 when $n = 4$.

Next, we also present an assignment for the z-variables. For scenario 1 we set $z^1_{01} = z^1_{n+1,n} = n$. Furthermore, we set $z^1_{1i} = z^1_{ni} = \frac{n-1}{n-2}$ and $z^1_{i1} = z^1_{in} = \frac{1}{n-2}$ for all $i = 2, 3, \ldots, n-1$. Finally, we set $z^1_{ij} = \frac{n}{n-2}$ for all $i = 2, 3, \ldots, n-1$ and $j = 2, 3, \ldots, n-1$ $(i \neq j)$. All other z-variables are set to zero.

For scenario 2 we set $z^2_{01} = \frac{n-2}{n-1}$, $z^2_{10} = n-2$, $z^2_{0n} = \frac{n}{n-1}$, $z^2_{n0} = n - \frac{n}{n-1}$, $z^2_{1n} = 1$, $z^2_{n1} = \frac{1}{n-1}$, $z^2_{n+1,1} = n$, $z^2_{n+1,n} = \frac{n(n-2)}{n-1}$ and finally $z^2_{0i} = 1$, $z^2_{i0} = n-1$ and

$z^2_{n+1,i} = n$ for all $i = 2, 3, \ldots, n-1$. All remaining z-variables are set to zero.

Finally, set $y_i = \tau^{max}$ for all $i \in V'$ and $t^\omega_i = \tau^{max}$ for all $i \in V'$ and all $\omega \in \Omega$.

It is straightforward to check that the described variables give a feasible solution to the LP relaxation of (2.1)-(2.16), (2.17) and (2.19). As all costs are equal to zero, this solution is also optimal. By definition, our optimal solution to the LP relaxation is an instance of the separation problem. In the remainder, we refer to this instance of the separation problem as instance I.

Characterization as a decision problem

Next, we characterize instance I of the separation problem as a decision problem. First, we prove the following lemma.

Lemma 2.14. *If instance I contains a violated precedence inequality, then the only violated precedence inequality is given by (using the notation of Theorem 2.4) $i = 1$, $j = n$, $S = \{2, 3, \ldots, n-1\}$, $F = (1 : S) \cup (S : S) \cup (S : n)$, $S' = \emptyset$ and $F' = \{(n, 1)\}$.*

Proof. By definition, F' cannot contain arcs involving the depot vertices. Therefore, considering the assigned values of the x-variables in scenario 2, we have that the arc $(n, 1)$ is the only arc with non-zero flow that may be in F'. Hence, $\sum_{(k,l)\in F'} x^2_{kl} = 0$ if $(n, 1) \notin F'$ or $0 < \sum_{(k,l)\in F'} x^2_{kl} = \frac{1}{n-1} < 1$ if $(n, 1) \in F'$. By Lemma 2.5 we have that $\sum_{(k,l)\in F'} x^2_{kl} > |S'| \geq 0$. Thus, $(n, 1)$ is contained in F'. Using that $\sum_{(k,l)\in F'} x^2_{kl} < 1$ and $\sum_{(k,l)\in F'} x^2_{kl} > |S'|$ it follows that $|S'| = 0$ and hence $S' = \emptyset$. As $S' = \emptyset$ and $(n, 1) \in F'$, it follows that $i = 1$, $j = n$ and $F' = \{(n, 1)\}$. It remains to be shown that $S = \{2, 3, \ldots, n-1\}$ and $F = (1 : S) \cup (S : S) \cup (S : n)$.

Suppose by contradiction that $S \in \mathcal{S}(1, n)$ is such that $|S| \leq n - 3$. In this case, the largest set $F \in \mathcal{F}(1, S, n)$ is given by $F = (1 : S) \cup (S : S) \cup (S : n)$. Therefore, the number of arcs in F is bounded by $|1 : S| + |S : S| + |S : n| = |S| + |S|(|S| - 1) + |S| = |S|(|S| + 1)$. For instance I, all these arcs have flow equal to $\frac{1}{n-2}$. It follows that $\sum_{(k,l)\in F} x^1_{kl} \leq |S|(|S| + 1)\frac{1}{n-2}$ for all $S \in \mathcal{S}(1, n)$ and $F \in \mathcal{F}(1, S, n)$. Recall that $\sum_{(k,l)\in F'} x^2_{kl} = \frac{1}{n-1}$ and $|S'| = 0$. Using $|S| + 1 \leq n - 2$ we derive $\sum_{(k,l)\in F} x^1_{kl} + \sum_{(k,l)\in F'} x^2_{kl} \leq |S|(|S| + 1)\frac{1}{n-2} + \frac{1}{n-1} \leq |S| + |S'| + 1$ for all $|S| \leq n - 3$. By Theorem 2.4, the corresponding precedence inequality is not violated, which is a contradiction. It follows that $|S| = n - 2$, which implies $S = \{2, 3, \ldots, n-1\}$.

Finally, we show that $F = (1 : S) \cup (S : S) \cup (S : n)$ by contradiction. Suppose that $F \neq (1 : S) \cup (S : S) \cup (S : n)$. Recall that $S = \{2, 3, \ldots, n-1\}$ and thus $|S| = n - 2$. Therefore, $|F| \leq |S|(|S| + 1) - 1 = (n - 2)(n - 1) - 1$. Hence, $\sum_{(k,l)\in F} x^1_{kl} + \sum_{(k,l)\in F} x^2_{kl} \leq ((n-2)(n-1) - 1)\frac{1}{n-2} + \frac{1}{n-1} = n - 1 - \frac{1}{n-2} + \frac{1}{n-1} \leq$

$n-1 = |S| + |S'| + 1$, which is a contradiction of the precedence inequality being violated. It follows that $F = (1:S) \cup (S:S) \cup (S:n)$.

Hence, the only potentially violated precedence inequality is given by $i = 1$, $j = n$, $S = \{2, 3, \ldots, n-1\}$, $F = (1:S) \cup (S:S) \cup (S:n)$, $S' = \emptyset$ and $F' = \{(n, 1)\}$. By assumption, a violated precedence inequality exists, hence, this is the only one. □

The next corollary follows from Lemma 2.14 and characterizes instance I of the separation problem as a decision problem.

Corollary 2.15. *Instance I contains a violated precedence inequality if and only if*

$$\delta_{1n}(S, F) > u_1 + u_n - \tau_{n1} \tag{2.36}$$

with $S = \{2, 3, \ldots, n-1\}$ and $F = (1:S) \cup (S:S) \cup (S:n)$.

Proof. Assume instance I contains a violated precedence inequality. By Lemma 2.14, $i = 1$, $j = n$, $S = \{2, 3, \ldots, n-1\}$, $F = (1:S) \cup (S:S) \cup (S:n)$, $S' = \emptyset$ and $F' = \{(n, 1)\}$. By Theorem 2.4 all precedence inequalities satisfy $\delta_{1n}(S, F) + \delta_{n1}(S', F') > u_1 + u_n$. Substituting $\delta_{n1}(S', F') = \delta_{n1}(\emptyset, \{(n, 1)\}) = \tau_{n1}$ then gives (2.36).

For the reverse implication, let $S = \{2, 3, \ldots, n-1\}$, $F = (1:S) \cup (S:S) \cup (S:n)$, $S' = \emptyset$ and $F' = \{(n, 1)\}$ and assume that $\delta_{1n}(S, F) > u_1 + u_n - \tau_{n1}$, which is equivalent to $\delta_{1n}(S, F) + \delta_{n1}(S', F') > u_1 + u_n$ as shown earlier. Similar as in the proof of Lemma 2.14, we have $\sum_{(k,l)\in F} x^1_{kl} + \sum_{(k,l)\in F'} x^2_{kl} = |S|(|S|+1)\frac{1}{n-2} + \frac{1}{n-1} = n - 1 + \frac{1}{n-1} > |S| + |S'| + 1$. Hence, by Theorem 2.4, we have that i, j, S, F, S' and F' define a violated precedence inequality. □

The decision problem is co-NP-complete

Next, we show that the decision problem $\delta_{1n}(S, F) > u_1 + u_n - \tau_{n1}$ is co-NP-complete. First, we note that the decision problem is in co-NP. A polynomial certificate for the *NO*-answer is given by a path from 1 to n visiting all vertices of S and using only arcs of F, of length $u_1 + u_n - \tau_{n1}$ or less. Next, we show that the decision problem is co-NP complete. We show this with a reduction from Euclidean TSP.

Consider the Euclidean TSP problem: given n distinct points in Euclidean space, determine whether there exists a tour visiting all vertices with total travel time α or less. The travel time between two points is given by their Euclidean distance. Let these travel times be τ_{ij}. Clearly, the travel times are positive and satisfy the triangle inequality.

Next, we show that the Euclidean TSP problem can be reduced to answering a polynomial number of instances of the complement of the decision problem $\delta_{1n}(S,F) > u_1 + u_n - \tau_{n1}$. For every edge (k,l) we do the following. Without loss of generality, relabel the vertices such that $k = 1$ and $l = n$. Now construct an instance I of the separation problem by using the same n and t, and by arbitrarily choosing $u_i \geq 0$ for all $i \in V'$ in such a way that $u_1 + u_n = \alpha$. Next, determine whether $\delta_{1n}(S,F) \leq u_1 + u_n - \tau_{n1}$, which is the complement of $\delta_{1n}(S,F) > u_1 + u_n - \tau_{n1}$. If for any (k,l) the answer is YES, this implies that there exists a tour of length at most $u_1 + u_n - \tau_{n1} + \tau_{n1} = \alpha$, and hence the answer to the Euclidean TSP problem is YES. If the answer is NO for all (k,l), it follows that it is impossible to construct a tour that is sufficiently short. Hence, the answer to the Euclidean TSP problem is NO.

As the number of edges is polynomial, we solve the Euclidean TSP problem by answering the complement of the decision problem $\delta_{1n}(S,F) > u_1 + u_n - \tau_{n1}$ a polynomial number of times. As Euclidean TSP is NP-complete (Papadimitriou, 1977), it follows that answering $\delta_{1n}(S,F) > u_1 + u_n - \tau_{n1}$ is co-NP-complete.

Separating precedence inequalities is co-NP-hard

Finally, consider a separation algorithm that returns a violated precedence inequality if one exists, or returns that no violated precedence inequality exists. Now use this separation algorithm on instance I. By Corollary 2.15, there is only one potential precedence inequality in I that may be violated. Hence, if a violated precedence inequality is found, it is established that $\delta_{1n}(S,F) > u_1 + u_n - \tau_{n1}$, for a given S and F. Since determining whether $\delta_{1n}(S,F) > u_1 + u_n - \tau_{n1}$ is co-NP-complete, it follows that the problem of separating precedence inequalities is co-NP-hard in general.

2.C Additional tables

Below, we present additional tables.

		Seconds					
Inst.	Clients	N	P	P2T	P2C	T	T2C
1	10	0.0	0.0	0.0	0.0	0.0	0.0
2	10	0.6	0.1	0.1	0.1	0.1	0.1
3	10	0.0	0.0	0.0	0.0	0.0	0.0
4	10	0.8	0.1	0.1	0.1	0.1	0.1
5	10	0.2	0.1	0.1	0.3	0.1	0.1
6	10	0.0	0.0	0.0	0.0	0.0	0.0
7	10	0.0	0.0	0.0	0.0	0.0	0.0
8	10	0.4	0.1	0.1	0.1	0.2	0.1
9	10	0.0	0.0	0.0	0.0	0.0	0.0
10	10	0.0	0.0	0.0	0.0	0.0	0.0
11	15	0.1	0.1	0.1	0.1	0.1	0.1
12	15	68.0	143.2	95.5	39.1	60.8	63.8
13	15	8.7	3.9	3.7	2.6	4.1	3.0
14	15	0.2	0.2	0.2	0.2	0.7	0.7
15	15	1.6	0.5	0.9	0.6	0.5	0.6
16	15	0.9	0.3	0.3	0.3	0.3	0.3
17	15	0.1	0.1	0.1	0.1	0.1	0.1
18	15	1.7	0.8	1.7	0.8	2.2	1.7
19	15	0.7	0.7	0.8	1.3	1.3	1.3
20	15	0.2	1.2	0.7	0.4	0.7	0.4
21	20	1.2	1.1	1.2	1.2	1.1	1.2
22	20	8.1	1.9	5.9	9.0	5.9	1.3
23	20	0.3	0.4	0.4	0.4	0.4	0.5
24	20	1.8	2.1	3.5	1.7	3.6	1.7
25	20	13.5	12.2	7.1	6.9	8.3	4.4
26	20	0.3	0.3	0.3	0.2	0.3	0.2
27	20	2.1	0.4	0.4	0.3	0.5	0.3
28	20	1.1	1.0	1.2	1.1	1.2	1.1
29	20	0.6	0.5	0.5	0.5	0.5	0.5
30	20	0.2	0.3	0.3	0.3	0.3	0.3
31	25	12.0	15.7	18.2	2.3	5.0	15.0
32	25	5.5	4.3	8.0	1.3	2.8	5.3
33	25	35.3	9.2	37.8	9.4	11.3	14.4
34	25	2.6	8.4	2.3	11.1	2.4	15.8
35	25	26.6	4.6	15.1	6.1	19.0	6.2
36	25	33.9	39.0	31.9	39.3	31.5	47.7
37	25	339.9	42.7	70.7	22.4	75.4	23.6
38	25	79.4	12.9	14.5	9.7	17.7	23.2
39	25	51.3	35.5	10.1	7.2	17.1	37.0
40	25	20.5	27.0	18.3	15.2	27.2	6.7
Average		18.0	9.3	8.8	4.8	7.6	7.0

Table 2.5: Solution times for various strategies, using $\rho = 0.6$.

Inst.	Clients	Seconds	Nodes	Opt. gap	Root gap	Lower bound	Upper bound
61	40	3,600.0	17,282	2.09	3.29	45.38	46.35
62	40	550.3	8,479	0	1.08	48.35	48.35
63	40	3,600.0	36,554	0.23	1.83	44.38	44.48
64	40	3,169.7	18,069	0	1.88	43.75	43.75
65	40	3,600.0	37,915	0.77	2.22	43.13	43.46
66	40	3,600.0	32,601	1.22	2.79	44.14	44.68
67	40	3,600.0	27,201	0.67	1.89	46.64	46.96
68	40	3,600.0	16,210	1.76	2.88	44.23	45.02
69	40	3,600.0	17,822	1.88	3.25	42.39	43.20
70	40	3,600.0	16,901	0.68	1.76	42.71	43.00
71	45	3,600.0	5,741	4.36	5.08	49.52	51.78
72	45	3,600.0	9,201	2.69	3.40	50.73	52.13
73	45	3,600.0	14,001	0.48	1.70	41.50	41.70
74	45	3,600.0	4,789	1.23	2.18	47.25	47.84
75	45	3,600.0	9,501	2.19	2.90	48.77	49.86
76	45	3,600.0	6,401	7.12	7.61	48.38	52.09
77	45	3,600.0	12,064	2.13	2.87	50.09	51.18
78	45	3,600.0	8,501	3.58	4.18	52.02	53.95
79	45	3,600.0	20,601	1.58	2.71	47.45	48.21
80	45	3,600.0	10,097	1.99	2.75	49.57	50.57
81	50	3,600.0	5,898	3.46	4.08	56.81	58.85
82	50	3,600.0	5,701	3.19	3.85	51.50	53.20
83	50	3,600.0	4,001	5.31	5.61	57.45	60.67
84	50	3,600.0	4,401	7.23	7.71	52.31	56.38
85	50	3,600.0	3,146	4.17	4.92	53.74	56.07
86	50	3,600.0	3,585	5.63	6.14	51.68	54.76
87	50	3,600.0	6,611	3.09	3.65	52.47	54.14
88	50	3,600.0	6,801	3.68	4.39	54.82	56.91
89	50	3,600.0	4,001	3.71	4.06	59.23	61.51
90	50	3,600.0	4,901	3.15	3.69	57.68	59.55

Table 2.6: Results for the branch-and-cut algorithm on instances 61 to 90.

Chapter 3

Addressing orientation-symmetry in the Time Window Assignment Vehicle Routing Problem

Abstract

The Time Window Assignment Vehicle Routing Problem (TWAVRP) is the problem of assigning time windows for delivery before demand volume becomes known. This implies that vehicle routes in different demand scenarios have to be synchronized, such that the same client is visited around the same time in each scenario. For TWAVRP instances that are relatively difficult to solve, we observe many similar solutions in which one or more routes have a different orientation, i.e., the clients are visited in the reverse order. We introduce an edge-based branching method combined with additional components to eliminate orientation-symmetry from the search tree, and we present enhancements to make this method efficient in practice. Next, we present a branch-price-and-cut algorithm based on this branching method. Our computational experiments show that addressing orientation-symmetry significantly improves our algorithm: the number of nodes in the search tree is reduced by 92.6% on average, and 25 additional benchmark instances are solved to optimality. Furthermore, the resulting algorithm is competitive with the state-of-the-art. The main ideas of this chapter are not TWAVRP specific and can be applied to other vehicle routing problems with consistency considerations or synchronization requirements.

This chapter is based on Dalmeijer and Desaulniers (2018).

3.1 Introduction

We consider the Time Window Assignment Vehicle Routing Problem (TWAVRP), the problem of assigning time windows to clients in a distribution network before the demand of the clients is revealed. It is assumed, however, that a list of possible demand scenarios and corresponding probabilities is given. The TWAVRP asks for an assignment of time windows to the clients and, for each scenario, a set of feasible routes that adhere to the assigned time windows and the scenario specific demand, such that the expected cost of distribution is minimized. That is, each client is assigned a single time window, which should be respected, regardless of which scenario occurs. Each assigned time window, or endogenous time window, is of fixed width. Furthermore, the endogenous time windows must fall within given exogenous time windows.

The TWAVRP was first introduced by Spliet and Gabor (2015), and was inspired by distribution networks of retail chains. In this setting, the clients are retail stores that are supplied from a central depot and the exogenous time windows represent the opening hours of the stores. In retail, stores are often assigned a time window that does not change for a longer period of time (e.g. one year). The TWAVRP was introduced to assign time windows in this setting, taking the uncertainty of the future demand into account.

Spliet and Gabor (2015) present a branch-price-and-cut (BPC) algorithm to solve TWAVRP instances with up to 25 clients and three demand scenarios, within one hour of computation time. Spliet and Desaulniers (2015) present a BPC algorithm for the DTWAVRP, a variant of the TWAVRP where each endogenous time window is chosen from a discrete set of options. With their exact method, the authors are able to solve instances of similar size as the instances by Spliet and Gabor (2015), and they present a heuristic for larger instances. Another variant of the TWAVRP is the TWAVRP with time-dependent travel times, for which a BPC algorithm is presented in Spliet et al. (2018).

In Chapter 2, we have introduced a branch-and-cut (BC) algorithm for the TWAVRP, which makes use of a new class of valid inequalities; the precedence inequalities. This algorithm solves the benchmark instances introduced by Spliet and Gabor (2015) on average about 200 times faster than the original BPC algorithm, and allows for instances with up to 35 clients and three demand scenarios to be solved.

Subramanyam et al. (2018) present a scenario decomposition algorithm to solve strategic time window assignment vehicle routing problems. The decomposition re-

sults in multiple vehicle routing problems which can be solved in a "black box" fashion. When applied to the TWAVRP, their algorithm outperforms the BC algorithm presented in Chapter 2 and instances with up to 50 clients and three demand scenarios are solved to optimality.

For an increasing number of companies, client satisfaction has become a priority, as satisfied clients provide a better lifetime value. As client satisfaction is often the result of consistent service, various types of consistency have been considered in the literature (see Kovacs et al. (2014) for a recent survey). In the case of the TWAVRP, clients are always visited within the assigned time window, which can be classified as *time-consistency*.

A problem closely related to the TWAVRP is the Consistent Vehicle Routing Problem (ConVRP), introduced by Groër et al. (2009). Where the TWAVRP only imposes time-consistency, the ConVRP is more restrictive and imposes both time-consistency and driver-consistency, i.e., each client should always be visited by the same driver. Lian (2017) presents a branch-and-price algorithm for the ConVRP, in which driver-consistency is enforced in the pricing subproblem. Subramanyam and Gounaris (2018) focus on the single vehicle variant of the ConVRP, and present a decomposition algorithm.

The TWAVRP can also be seen as a vehicle routing problem with *operation synchronization* constraints (Drexl, 2012). That is, the routes in the different scenarios have to be synchronized, such that the visits to a client (the operations) in different scenarios are at approximately the same time.

It is well known that symmetry has a negative impact on branch-and-bound algorithms, as it can lead to multiple branches for which symmetric optimal solutions are computed. In this chapter, we introduce the concept of *orientation-symmetry* and we show that orientation-symmetry has a big impact on both the algorithm by Spliet and Gabor (2015) and the algorithm presented in Chapter 2.

We then propose an edge-based branching method combined with additional components to eliminate orientation-symmetry from the search tree. The first additional component is an algorithm to find the best solution to the TWAVRP out of a set of orientation-symmetric solutions. We make use of the precedence inequalities to speed up this algorithm and make it efficient in practice. The second additional component is a novel constraint that is used to cut off groups of orientation-symmetric solutions.

For the TWAVRP, branching only on the edge flows is not sufficient to explore the whole search tree, as opposed to several other vehicle routing problems where this is the case (e.g., the symmetric Capacitated Vehicle Routing Problem). By

combining edge-based branching with the additional components, we are able to solve the TWAVRP while addressing orientation-symmetry.

Based on our method to address orientation-symmetry, we construct a BPC algorithm to solve the TWAVRP to optimality. Our computational experiments show that addressing orientation-symmetry significantly improves our algorithm: the number of nodes in the search tree is reduced by 92.6% on average (from 145.0 to 10.7), and 25 additional benchmark instances are solved to optimality. Furthermore, the resulting algorithm is competitive with the scenario decomposition algorithm by Subramanyam et al. (2018).

Our experiments also show that addressing orientation-symmetry improves the BC algorithm presented in Chapter 2. Finally, we conduct experiments on instances with additional clients and instances with additional demand scenarios.

This chapter is structured as follows. In Section 3.2, we present the set-partitioning formulation for the TWAVRP. We also state the precedence inequalities, as they will be used in a later section of the chapter. In Section 3.3 we define orientation-symmetry and present an edge-based branching method combined with additional components to eliminate orientation-symmetry from the search tree. Section 3.4 presents the BPC algorithm, and Section 3.5 details our computational experiments. In the final section, we give a conclusion and present some directions for further research.

3.2 Mathematical formulation and precedence inequalities

In this section, we present the set-partitioning formulation for the TWAVRP introduced by Spliet and Gabor (2015) and the precedence inequalities proposed in Chapter 2.

3.2.1 Set-partitioning formulation

The TWAVRP is defined on a graph $G = (V, A)$ with $V = \{0, 1, \ldots, n+1\}$. Vertices 0 and $n+1$ correspond to the depot and are referred to as the *starting depot* and the *ending depot*, respectively. The other vertices correspond to the n clients. For convenience, let $V' = \{1, \ldots, n\}$. The set A consists of all arcs from the starting depot to the clients, all arcs between clients, and all arcs from the clients to the ending depot. Each arc in $(i, j) \in A$ is associated with a cost c_{ij} and a travel time

τ_{ij}. Both the costs and the travel times are non-negative and satisfy the triangle inequality. Service times at the clients are included in the travel times by adding the service time at client i to the travel time of all outgoing arcs.

The demand uncertainty is modeled through Ω, a finite set of demand scenarios. Each demand scenario $\omega \in \Omega$ occurs with probability p_ω. Naturally, $\sum_{\omega\in\Omega} p_\omega = 1$. For client $i \in V'$, demand in scenario $\omega \in \Omega$ is given by $d_i^\omega \geq 0$. This is a slight generalization of the assumption by Spliet and Gabor (2015), who assume that demand is strictly positive. Let $V'_\omega \subseteq V'$ be the set of clients with non-zero demand, i.e., the clients that have to be visited in scenario ω. We have access to an unlimited number of identical vehicles with capacity Q. For all $i \in V'$ and $\omega \in \Omega$, we assume that $d_i^\omega \leq Q$.

Each vertex $i \in V$ is associated with an exogenous time window with starting time $s_i \geq 0$ and ending time $e_i > s_i$. Vertices 0 and $n+1$ both refer to the depot and thus $s_0 = s_{n+1}$ and $e_0 = e_{n+1}$. Each client $i \in V'$ has to be assigned a time window of given non-negative width u_i, which has to be within the exogenous time window. Service must start, but not necessarily complete, within the assigned time window.

In this chapter, a *route* refers to a pair (P, t), with P an elementary path in G from the starting depot to the ending depot, and t a vector of times at which service starts at the clients. We define t_r^i to be the time at which service starts at client i if route r is used. To allow for a concise formulation, t_r^i is defined to be equal to zero for all clients i that are not in the path P corresponding to route r. For each $\omega \in \Omega$, $\mathcal{R}(\omega)$ is the set of all feasible routes in scenario ω. Route r is feasible in scenario ω if the exogenous time windows and the vehicle capacity are respected, and only clients with non-zero demand are visited (i.e., clients in V'_ω). For all routes r and clients i we define a_r^i to be equal to one if client i is served by route r, and zero otherwise. Similarly, let b_r^{ij} be equal to one if arc (i, j) is contained in route r, and zero otherwise. The cost of route r is given by c_r.

For each client $i \in V'$ the variable y_i indicates the start of the endogenous time window. As the endogenous time window is of fixed width u_i, it follows that the time window ends at $y_i + u_i$. Recall that each endogenous time window should be contained in the exogenous time window, and hence $y_i \in [s_i, e_i - u_i]$.

The route variables θ_r^ω give the contribution of route $r \in \mathcal{R}(\omega)$ to the solution in scenario ω. The flow on arc (i, j) in scenario ω is given by the flow variables x_{ij}^ω. We obtain a feasible solution to the TWAVRP if all flow variables are integral, even if the route variables are fractional. This follows from the fact that a convex combination of routes corresponding to the same path is feasible.

Given this notation, the TWAVRP can be expressed as the following set-partitioning model:

$$\min \sum_{\omega\in\Omega} p_\omega \sum_{r\in\mathcal{R}(\omega)} c_r\theta_r^\omega, \tag{3.1}$$

$$\text{s.t.} \sum_{r\in\mathcal{R}(\omega)} a_r^i\theta_r^\omega = 1 \qquad \forall\omega\in\Omega,\ i\in V'_\omega, \tag{3.2}$$

$$\sum_{r\in\mathcal{R}(\omega)} t_r^i\theta_r^\omega \geq y_i \qquad \forall\omega\in\Omega,\ i\in V'_\omega, \tag{3.3}$$

$$\sum_{r\in\mathcal{R}(\omega)} t_r^i\theta_r^\omega \leq y_i + u_i \quad \forall\omega\in\Omega,\ i\in V'_\omega, \tag{3.4}$$

$$y_i \in [s_i, e_i - u_i] \qquad \forall i\in V', \tag{3.5}$$

$$\theta_r^\omega \geq 0 \qquad \forall\omega\in\Omega,\ r\in\mathcal{R}(\omega), \tag{3.6}$$

$$x_{ij}^\omega = \sum_{r\in\mathcal{R}(\omega)} b_r^{ij}\theta_r^\omega \qquad \forall\omega\in\Omega,\ (i,j)\in A, \tag{3.7}$$

$$x_{ij}^\omega \in \{0,1\} \qquad \forall\omega\in\Omega,\ (i,j)\in A. \tag{3.8}$$

The objective function (3.1) is to minimize the expected cost of distribution. Constraints (3.2) ensure that in each scenario, every client with non-zero demand is visited exactly once. Note that clients with zero demand are never visited, due to our definition of $\mathcal{R}(\omega)$, the set of feasible routes in scenario ω. Constraints (3.3) and (3.4) guarantee that in each scenario, the time of service for each visited client is within the endogenous time window. Constraints (3.5) define the continuous variables that indicate the starting times of the endogenous time windows. The route variables are defined by Constraints (3.6) and are linked to the flow variables by Constraints (3.7). Finally, Constraints (3.8) state that the flow variables are subject to binary requirements.

Recall that each route r consists of a pair (P,t). As waiting is allowed, and t is a continuous vector of times of service, it follows that the number of routes, and thus the number of variables in the above formulation, is typically infinite.

3.2.2 Precedence inequalities

The precedence inequalities are inequalities based on the following fact. Consider an integral solution to the TWAVRP that contains route $r=(P,t)\in R(\omega)$ in scenario ω and route $r'=(P',t')\in R(\omega')$ in scenario ω'. Assume that there exist two client

vertices $i, j \in V'$ such that P contains a subpath p from i to j and P' contains a subpath p' from j to i. Let A_p and $A_{p'}$ be the arc sets of p and p', respectively. It then holds that

$$\sum_{(k,l)\in A_p} \tau_{kl} + \sum_{(k,l)\in A_{p'}} \tau_{kl} \leq u_i + u_j. \tag{3.9}$$

This fact follows from the following observation. For route r to visit both i and j within their endogenous time windows, client i should be served after time y_i and client j should be served before time $y_j + u_j$. Hence, an upper bound on the time between the visits to i and j is given by $y_j + u_j - y_i$. That is, $\sum_{(k,l)\in A_p} \tau_{kl} \leq y_j + u_j - y_i$. Analogously, route r' implies that $\sum_{(k,l)\in A_{p'}} \tau_{kl} \leq y_i + u_i - y_j$. By adding up these two inequalities, y_i and y_j cancel, and Inequality (3.9) follows.

Though not presented in Chapter 2, it is straightforward to extend this idea to the case where the depot is involved. Define an endogenous time window for the depot that has the same width as the exogenous time window. That is, define $u_0 = e_0 - s_0$. Now for a subpath p of P from $i = 0$ to $j \in V'$ and a subpath p' of P' from $j \in V'$ to $n + 1$, Inequality (3.9) holds by the same argument as before.

In Chapter 2, we use inequalities of the form (3.9) to derive the precedence inequalities. In this chapter, we only state the *path precedence inequalities*, a subclass of the precedence inequalities.

Consider a pair of elementary paths p and p', as described above, such that Inequality (3.9) does *not* hold. Furthermore, consider a pair of distinct scenarios $\omega, \omega' \in \Omega$. The path-precedence inequalities can then be stated as:

$$\sum_{(k,l)\in A_p} x^{\omega}_{kl} + \sum_{(k,l)\in A_{p'}} x^{\omega'}_{kl} \leq |A_p| + |A_{p'}| - 1. \tag{3.10}$$

As p and p' are chosen such that Inequality (3.9) does not hold, it must be that $\sum_{(k,l)\in A_p} x^{\omega}_{kl} < |A_p|$ or $\sum_{(k,l)\in A_{p'}} x^{\omega'}_{kl} < |A_{p'}|$. By integrality of the x-variables, Inequality (3.10) follows.

3.3 Orientation-symmetry

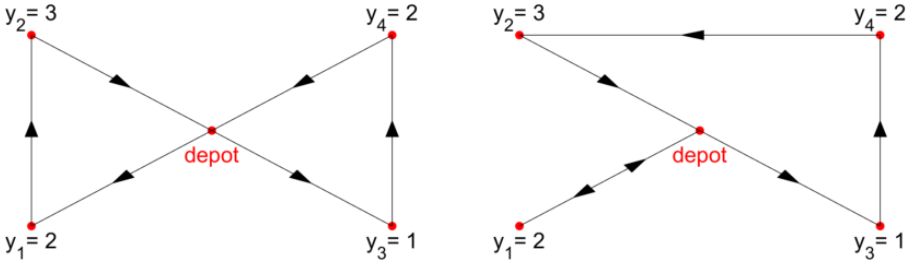
Given a feasible solution to the TWAVRP, it is often possible to find another feasible solution by first changing the orientation of one or more routes, and then reassigning the endogenous time windows. *Changing the orientation of a route* here means that the clients are visited in the reverse order. E.g., if we change the orientation of a route in scenario ω that starts at the depot, visits clients 1, 2 and 3, and then returns

to the depot, we obtain a route in scenario ω that starts at the depot, visits clients 3, 2 and 1, and then returns to the depot.

If one feasible solution can be turned into another feasible solution by changing route orientations and reassigning the endogenous time windows, then we say that these solutions are *orientation-symmetric*. Reassigning the endogenous time windows may indeed be necessary: by changing the orientation of one or more routes, the current endogenous time windows can become infeasible. If the arc costs are symmetric, then orientation-symmetric solutions have the same objective value.

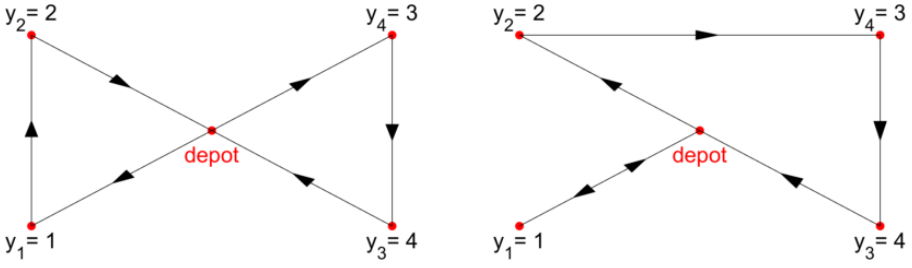
We now present an example of two orientation-symmetric solutions. Consider an instance of the TWAVRP with four clients and two demand scenarios. Every arc has a travel time of one and all endogenous time window widths are set to zero. Recall that the start of the endogenous time window of client $i \in V'$ is given by y_i.

It can be seen that Solution 1 (Figure 3.1) and Solution 2 (Figure 3.2) are orientation-symmetric solutions. Solution 1 can be turned into Solution 2 by reversing the route visiting clients 3 and 4 in scenario 1, reversing the route visiting clients 3, 4 and 2 in scenario 2, and by reassigning the endogenous time windows.



(a) Scenario 1 (b) Scenario 2

Figure 3.1: Example solution 1.



(a) Scenario 1 (b) Scenario 2

Figure 3.2: Example solution 2.

Out of the forty benchmark instances introduced by Spliet and Gabor, instances 12, 36 and 37 are among the most difficult to solve with both the algorithm by Spliet and Gabor (2015) and the algorithm presented in Chapter 2. Interestingly, these instances all contain a number of orientation-symmetric optimal solutions: 11, 112 and 40, respectively.

It is well known that the presence of symmetric solutions has a negative impact on the performance of branch-and-bound methods, which may in part explain why instances 12, 36 and 37 are difficult to solve. In the remainder of Section 3.3, we propose an edge-based branching method, combined with additional components, that ensures that orientation-symmetric solutions are always in the same branch, thereby eliminating orientation-symmetry in the search tree. In Section 3.5 we show computationally that addressing orientation-symmetry improves algorithmic performance.

3.3.1 Eliminating orientation-symmetry

We make the distinction between (directed) arcs and (undirected) edges. An edge between i and j, with $i < j$ and $i, j \in \{0\} \cup V'$, is denoted by $[i, j]$. Note that we use square brackets for edges and parentheses for arcs.

Each edge $[i, j]$ corresponds to at most two arcs in A. When i and j are client vertices ($i, j \in V'$), edge $[i, j]$ corresponds to the arcs (i, j) and (j, i), if they exist. When i is the depot vertex ($i = 0$) and j is a client vertex ($j \in V'$), edge $[0, j]$ corresponds to the arcs $(0, j)$ and $(j, n + 1)$, if they exist. We use E to denote the set of all edges.

Let the variable z_{ij}^{ω} represent the total flow on edge $[i, j] \in E$ in scenario $\omega \in \Omega$. The flow on an edge in a given scenario is obtained by summing the flows on the corresponding arcs in the same scenario. For example, for edge $[1, 3] \in E$ in scenario ω we have $z_{13}^{\omega} = x_{13}^{\omega} + x_{31}^{\omega}$, assuming that both arcs $(1, 3)$ and $(3, 1)$ exist.

Observation 3.1. *The edge flows of any feasible solution to the TWAVRP are integral and correspond, per scenario, to disjoint undirected elementary paths that start and end at the depot.*

If branching decisions are based on z_{ij}^{ω}, then orientation-symmetric solutions always end up in the same branch. This follows from the fact that, by definition, orientation-symmetric solutions have the same edge flows. In Spliet and Gabor (2015) and in Chapter 2, branching decisions are based on x_{ij}^{ω} instead of z_{ij}^{ω}.

To solve the TWAVRP, it is insufficient to branch on fractional z_{ij}^{ω} variables only. Recall that a solution with integral arc flows corresponds to a feasible solution to the

TWAVRP. However, if the edge flows are integral, the arc flows may be fractional. In general, we do not obtain a feasible solution to the TWAVRP if the arc flows are fractional, even if the edge flows are integral and correspond to disjoint undirected elementary paths. Undirected paths do not properly represent the passing of time. As a result, we cannot guarantee that the endogenous time windows are satisfied.

Consider a node in the search tree for which we compute a solution to the linear programming (LP) relaxation of (3.1)-(3.8). If the edge flows are not integral, we branch on the variable z_{ij}^{ω} that corresponds to the non-depot edge that has flow closest to 0.5. It can be shown that if the non-depot edges have integral flows, then the depot edges also have integral flows. If the edge flows and the arc flows are both integral, we have obtained a feasible solution to the TWAVRP and the current node can be pruned by integrality. Finally, if the edge flows are integral and the arc flows are fractional, we use Procedure 1 to continue the search.

Procedure 1 Processing a node with integral edge flows and fractional arc flows

1: Solve a restricted TWAVRP in which the edge flows are fixed to their current values.
2: **if** Restricted TWAVRP is feasible and has cost lower than the incumbent solution **then**
3: Make the optimal solution to the restricted TWAVRP the incumbent solution.
4: **end if**
5: Add a constraint to forbid the current integral edge flows.

In Step 1, we solve a restricted TWAVRP in which the edge flows are fixed. By definition, solutions that have the same edge flows are orientation-symmetric. Hence, solving the restricted TWAVRP amounts to finding the best solution among a set of orientation-symmetric solutions. In Section 3.3.2 we present an algorithm for solving the restricted TWAVRP.

In Step 5, a constraint is added to forbid the current integral edge flows. This constraint forces at least one of the z_{ij}^{ω} variables to take a different value. On the other hand, solutions to the TWAVRP with different integral edge flows should not be cut off. In Section 3.3.3, we present this nontrivial constraint.

Note that we do not assume that the arc costs are symmetric. If they are symmetric, however, then the objective value can be obtained directly from the edge flows. It follows that if the restricted TWAVRP is feasible, we obtain an incumbent solution with cost equal to the lower bound of the current node. As a result, the current node can immediately be pruned by optimality.

3.3.2 Solving the restricted TWAVRP with fixed edge flows

In this section, we present an algorithm for solving the TWAVRP with fixed edge flows. We will refer to this problem as the *restricted TWAVRP*. From Observation 3.1 it follows directly that if the edge flows do not correspond to disjoint undirected elementary paths, then there does not exist a solution to the TWAVRP that is consistent with the fixed edge flows. We therefore assume without loss of generality that the integral edge flows can be represented by m disjoint undirected elementary paths.

It follows that the restricted TWAVRP can be seen as a TWAVRP on a restricted arc set, and can thus be solved by any algorithm for the TWAVRP. In this section, however, we exploit the fact that all solutions to the restricted TWAVRP are orientation-symmetric to construct an algorithm that is more efficient in practice.

To obtain arc flows consistent with the fixed edge flows, all m undirected paths must be given an orientation. First, arbitrarily assign a default orientation to each of the undirected paths. Then, for all $k \in \{1, \dots, m\}$, let the variable o_k be equal to 1 if the orientation of path k is equal to the default orientation, or -1 otherwise. For single client paths, we set $o_k = 1$ without loss of generality. Note that the edge flows z_{ij}^{ω} and the path orientations o_k together define the arc flows x_{ij}^{ω}. For example, if edge $[1, 3]$ in scenario ω is on path k, then the value of o_k determines whether $x_{13}^{\omega} = 1$ or $x_{31}^{\omega} = 1$.

We can now enumerate all possible assignments of the o_k variables. Each assignment $(o_1, o_2, \dots, o_m) \in \{-1, 1\}^m$ defines the arc flows of a potential solution. When the arc flows are fixed, the MIP formulation in Chapter 2 reduces to a linear program, which can be solved to find a feasible solution to the TWAVRP if one exists. In Appendix 3.A we propose an alternative method to solve the TWAVRP for fixed arc flows that does not rely on linear programming.

When all possible assignments of the o_k variables have been enumerated, we have either found the minimum cost solution to the restricted TWAVRP, or we have proven that no solution exists. Hence, we have solved the restricted TWAVRP, as required. Recall that for instances with symmetric costs, all orientation-symmetric solutions yield the same objective value. In that case, the enumeration can be stopped after the first feasible solution has been found.

3.3.2.1 More efficient enumeration

The enumeration algorithm presented above enumerates all 2^m route orientations. Some assignments of the o_k variables, however, lead to violated path precedence

inequalities. As the path precedence inequalities are valid, these assignments cannot lead to a feasible solution to the TWAVRP and can thus be ignored. In this section, we use this observation to significantly reduce the number of route orientations that has to be enumerated.

We introduce a *conflict graph* to represent conflicts between the o_k variables due to violated path precedence inequalities. The vertex set of the conflict graph is given by $\{1, \ldots, m\}$, in which vertex k corresponds to o_k.

The edges of the conflict graph correspond to violations of the path precedence inequalities. Recall that o_k and o_l define the arc flows on paths k and l, respectively. This implies that for given values of o_k and o_l we can test for a violation of Inequality (3.10). If $o_k \neq o_l$ results in a violated path precedence inequality (i.e., both $(o_k, o_l) = (1, -1)$ and $(o_k, o_l) = (-1, 1)$ result in a violation), then k and l are connected by a *positive edge*: an edge with value 1. If $o_k = o_l$ results in a violated path precedence inequality (i.e., both $(o_k, o_l) = (1, 1)$ and $(o_k, o_l) = (-1, -1)$ result in a violation), then k and l are connected by a *negative edge*: an edge with value -1. If neither case is applicable, then edge $[k, l]$ does not exist.

An assignment of the o_k variables is said to be *conflict-free* if for every edge $[k, l]$ the value of $o_k o_l$ is equal to the edge value. If the conflict graph contains a positive edge $[k, l]$, we require $o_k = o_l$ in a conflict-free assignment. Similarly, for a negative edge $[k, l]$, we require $o_k \neq o_l$ for the assignment to be conflict-free.

Figure 3.3 presents an example of a conflict graph which allows for different conflict-free assignments. Positive edges are shown as dotted lines and negative edges are shown as solid lines. One of the conflict-free assignments is given by $o_1 = o_2 = o_4 = o_5 = 1$ and $o_3 = o_6 = -1$.

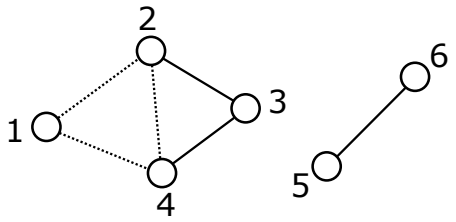


Figure 3.3: Example of a conflict graph.

Observation 3.2. *Only a conflict-free assignment of the o_k variables can lead to a feasible solution to the TWAVRP.*

Observation 3.2 follows directly from the definition of a conflict-free assignment: if an assignment is *not* conflict-free, then at least one path precedence inequality is

violated. As the path precedence inequalities are valid, it follows that the current assignment of the o_k variables cannot lead to a feasible solution to the TWAVRP.

The conflict graph contains a positive edge if both $(o_k, o_l) = (1, -1)$ and $(o_k, o_l) = (-1, 1)$ result in a violated path precedence inequality. Note that if the travel times are symmetric, it follows from (3.9) and (3.10) that $(o_k, o_l) = (1, -1)$ leads to a violation if and only if $(o_k, o_l) = (-1, 1)$ leads to a violation. A similar result is true for the negative edges. When solving asymmetric instances, it can be beneficial to use a directed conflict graph that considers the pairs $(o_k, o_l) = (1, -1)$ and $(o_k, o_l) = (-1, 1)$ separately. However, as we focus on instances that are difficult due to symmetry, we do not present this extension here.

The conflict graph can be constructed in polynomial time. This follows from the fact that a feasible solution contains at most $|\Omega|n$ paths. Hence, there are $\mathcal{O}(|\Omega|^2 n^2)$ vertex pairs, and for each pair, at most four possible assignments of (o_k, o_l) have to be tested for violated path precedence inequalities. As path precedence inequalities can be separated in polynomial time (Chapter 2), the conflict graph can be constructed in polynomial time.

By Observation 3.2, only a conflict-free assignment of the o_k variables can lead to a feasible solution to the TWAVRP. Hence, after constructing the conflict graph, the next step is to enumerate all conflict-free assignments.

First, we modify the conflict graph by replacing each positive edge $[k, l]$ by a dummy vertex that is connected to both k and l by negative edges. Note that this forces $o_k = o_l$, just like the original positive edge. Figure 3.4 shows the result of modifying the conflict graph presented in Figure 3.3. Clearly, there is a bijection between the conflict-free assignments of the original graph and those of the modified graph. It follows that it is sufficient to find all conflict-free assignments in the modified conflict graph.

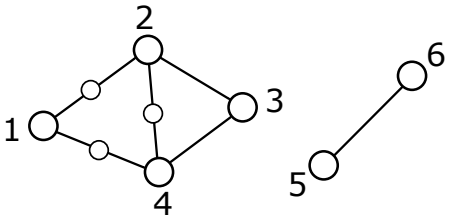


Figure 3.4: Example of a modified conflict graph.

In the modified conflict graph, adjacent vertices are connected by a negative edge, and must thus be assigned different values. If we associate the value 1 with the color

red, and the value -1 with the color blue, then every conflict-free assignment in the modified conflict graph corresponds to a vertex coloring using at most two colors.

It is well known that we can use breadth-first search to either find a two-coloring, or to find an odd cycle, which proves that no two-coloring exists (e.g., see Kleinberg and Tardos (2006), Chapter 3.4). This algorithm takes polynomial time.

When a single two-coloring is known, we can easily find additional two-colorings by switching the roles of red and blue in any of the connected components. In fact, all two-colorings can be obtained in this way. This follows from the fact that any connected graph allows for at most two distinct two-colorings.

In conclusion, we can now enumerate all two-colorings of the modified conflict graph, or equivalently, enumerate all conflict-free assignments of the original conflict graph. The number of conflict-free assignments is equal to two to the power of the number of connected components of the conflict graph. Note that this can be significantly smaller than the total number of possible assignments, which is equal to 2^m.

If no conflict-free assignment can be found, we obtain an odd cycle in the modified conflict graph. This cycle indicates which undirected paths cause the TWAVRP to be infeasible. In Section 3.3.3, we use this information to strengthen the constraint that forbids the current integral edge flows, which is added in Step 5 of Procedure 1.

3.3.3 Cutting off integral edge flows

In this section we present constraints that cut off the current integral edge flows. A constraint of this type is needed in Step 5 of Procedure 1. As in the previous section, we assume that integral edge flows are represented by disjoint undirected elementary paths. Let the current edge flows be given by $z^\omega_{ij} = \bar{z}^\omega_{ij}$ for all $[i,j] \in E$ and $\omega \in \Omega$.

Figure 3.5 gives an example of the edge flows in a single scenario. Vertices 1 up to 7 correspond to the clients, and vertex 0 corresponds to the depot. This example contains three disjoint undirected elementary paths. Note that client 1 is contained in a single client path. By definition $z^\omega_{01} = x^\omega_{01} + x^\omega_{1,n+1}$, and hence in the example we have $\bar{z}^\omega_{01} = 2$. The other undirected paths contain multiple clients. As these paths are elementary, it follows that the corresponding edge flows are equal to one.

To obtain a constraint that cuts off the current integral edge flows, we make use of the more general Proposition 3.3. For a given subset of edges with non-zero flow, Proposition 3.3 provides an inequality that can only be satisfied by changing at least one of the flows on the selected edges. For technical reasons we impose that if a depot edge is selected that is part of a path with multiple clients, then the adjacent

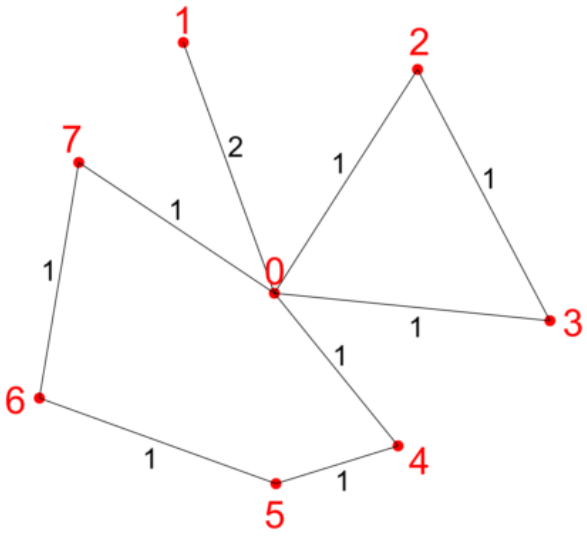


Figure 3.5: Example current edge flows $\bar{z}_{ij}^{\omega}$ for a given scenario ω.

non-depot edge is also selected.

Let $\hat{E} \subseteq E$ be a set of edges. Given a pair of clients $i, j \in V'$ $(i \neq j)$ we define $\lambda_{ij}^{\hat{E}}$ to be the number of depot edges in $\hat{E}$ that are incident to i or j. That is, $\lambda_{ij}^{\hat{E}} = |\{[0,i],[0,j]\} \cap \hat{E}|$. Given this notation, Proposition 3.3 can be stated as follows.

Proposition 3.3. *Let the current integral edge flows be given by $\bar{z}_{ij}^{\omega}$ for all $[i,j] \in E$ and $\omega \in \Omega$. For each scenario $\omega \in \Omega$, let $E^{\omega} \subset E$ be any subset of edges with non-zero flow in scenario ω, satisfying the following technical condition: if $[0,j] \in E^{\omega}$ and $\bar{z}_{0j}^{\omega} = 1$ then E^{ω} contains an edge adjacent to $[0,j]$. Then, the constraint*

$$\sum_{\omega \in \Omega} \left(\sum_{[i,j] \in E^{\omega}} z_{ij}^{\omega} + \sum_{[i,j] \in E^{\omega} | i,j \in V'} \lambda_{ij}^{E^{\omega}} z_{ij}^{\omega} \right) \leq \sum_{\omega \in \Omega} \left(\sum_{[i,j] \in E^{\omega}} \bar{z}_{ij}^{\omega} + \sum_{[i,j] \in E^{\omega} | i,j \in V'} \lambda_{ij}^{E^{\omega}} \bar{z}_{ij}^{\omega} \right) - 1 \tag{3.11}$$

is violated by an integral edge flow if and only if $z_{ij}^{\omega} = \bar{z}_{ij}^{\omega}$ for all $[i,j] \in E^{\omega}$ and $\omega \in \Omega$.

Proof. See Appendix 3.B. □

To illustrate Proposition 3.3, we construct an inequality for the example in Figure 3.5. Note that $\lambda_{ij}^{E^{\omega}}$ can be seen as an additional weight that is assigned to non-depot edges that are adjacent to depot edges. To forbid the current flow on

$E^\omega = \{[0,2],[2,3],[0,3]\}$, we obtain the constraint $(z^\omega_{02} + z^w_{23} + z^w_{03}) + 2z^\omega_{23} \leq 4$. The edge $[2,3]$ gets additional weight ($\lambda^{E^\omega}_{23} = 2$) because it is adjacent to two depot edges in E^ω. This additional weight is necessary: the constraint $z^\omega_{02} + z^w_{23} + z^w_{03} \leq 2$ (obtained by setting $\lambda^{E^\omega}_{ij} = 0$) would cut off the current solution but is not valid, as it would also cut off the solution $z^\omega_{02} = z^\omega_{03} = 2$.

To obtain a constraint that cuts off the current integral edge flows, we can now apply Proposition 3.3. For each scenario $\omega \in \Omega$, choose E^ω to be the set of *all* edges with non-zero flow in scenario ω. Proposition 3.3 then provides a constraint that can be used in Step 5 of Procedure 1. For example, the scenario shown in Figure 3.5 contributes $z^\omega_{01} + (z^\omega_{02} + z^\omega_{23} + z^\omega_{03}) + 2z^\omega_{23} + (z^\omega_{04} + z^\omega_{45} + z^\omega_{56} + z^\omega_{67} + z^\omega_{07}) + z^\omega_{45} + z^\omega_{67}$ to the left-hand side of this constraint and 14 to the right-hand side. The technical condition in Proposition 3.3 is satisfied because the edges with non-zero flow make disjoint undirected elementary paths.

It is often possible to add a constraint that is stronger than the one proposed in the previous paragraph. Recall from Section 3.3.2.1 that if there is no conflict-free assignment, we obtain a cycle of conflicting undirected paths. These paths conflict due to the path precedence inequalities.

Let a *conflict edge* be an edge that is contained in at least one of these path precedence inequalities. It can be seen that if we replace each path in the conflict by its smallest subpath containing all conflict edges, then the conflict remains.

It follows that, for each scenario $\omega \in \Omega$, we may choose E^ω to represent these smallest subpaths. For example, if $[2,3]$, $[4,5]$ and $[6,7]$ are conflict edges in Figure 3.5, then we can choose $E^\omega = \{[2,3],[4,5],[5,6],[6,7]\}$. If necessary, additional edges may be added to E^ω to ensure that the technical condition is satisfied. As the selected edge flows lead to a conflict, Proposition 3.3 provides a valid inequality.

The constraint that we construct based on the conflict is stronger than the constraint that we obtain in case that no conflict is found. This follows from the fact that the latter only cuts off the current integral edge flows, while the former can also cut off other integral edge flows that would generate the same conflict in the conflict graph.

3.4 Branch-price-and-cut algorithm

In this section we present the proposed BPC algorithm to solve the TWAVRP. First, we detail how we solve the LP relaxation of (3.1)-(3.8) with column generation. Next, we discuss the route relaxations that we use and the valid inequalities that we apply.

We end this section with an overview of the BPC algorithm.

3.4.1 Column generation

The name *branch-and-price* is due to Barnhart et al. (1998), who introduce the term for branch-and-bound algorithms that make use of column generation to solve the LP relaxations. In our case, model (3.1)-(3.8) can be seen as the *integer master problem*, which contains a huge number of variables. Its LP relaxation, given by (3.1)-(3.6), is referred to as the *master problem*. The master problem is solved by column generation.

Column generation is iterative. In each iteration we first solve a *restricted master problem* (RMP), which is obtained by restricting the master problem to a subset of the variables. Next, we solve a subproblem, the *pricing problem*, to obtain variables with negative reduced costs that can be added to the RMP. If no variable with negative reduced cost can be found, the solution to the RMP is an optimal solution to the master problem.

For all $\omega \in \Omega$ and $i \in V'_\omega$, let β_i^ω, $\gamma_i^\omega \geq 0$ and $\delta_i^\omega \leq 0$ be the dual variables corresponding to Constraints (3.2), (3.3) and (3.4), respectively. For convenience, let $\pi_i^\omega = \gamma_i^\omega + \delta_i^\omega$. Note that both β_i^ω and π_i^ω are unrestricted. The reduced cost $\bar{c}_r^\omega$ corresponding to the route variable θ_r^ω can now be expressed as

$$\bar{c}_r^\omega = p_w c_r - \sum_{i \in V'_\omega} \beta_i^\omega a_r^i - \sum_{i \in V'_\omega} \pi_i^\omega t_r^i. \tag{3.12}$$

Recall that $a_r^i = 0$ and $t_r^i = 0$ if client i is not visited by route r. Hence, in Equation (3.12), we can sum over all clients in V'_ω. The pricing problem is to find a route variable θ_r^ω for which $\bar{c}_r^\omega$ is negative.

Note that the pricing problem decomposes into a separate problem for each scenario. For a given scenario $\omega \in \Omega$, we model the pricing problem as an Elementary Shortest Path Problem with Resource Constraints (ESPPRC) and linear node costs.

Let $G_\omega = (V_w, A_w)$ be the subgraph of G that is obtained by removing all clients with zero demand and all adjacent arcs. The goal is to find a minimum cost elementary path in G_ω from the starting depot to the ending depot such that both vehicle capacity and the exogenous time windows are respected. Note that we may reduce the size of the arc set A_ω by removing arcs that are infeasible due to capacity or time constraints, without affecting any of the feasible paths in G_ω.

To each arc $(i, j) \in A_\omega$ we assign the cost $\bar{c}_{ij}(t^j) = p_w c_{ij} - \beta_j^\omega - \pi_j^\omega t^j$ if $j \in V'_\omega$

and $\bar{c}_{ij}(t^j) = p_w c_{ij}$ otherwise, with t^j the time at which vertex j is visited. It follows that a path with negative costs corresponds to a route with negative reduced cost. Note that $\bar{c}_{ij}(t^j)$ depends linearly on t^j, which is a decision variable in the pricing problem.

Liberatore et al. (2011) encounter a similar pricing problem in the context of the Vehicle Routing Problem with Soft Time Windows and they present a bi-directional labeling algorithm for solving it. To solve the pricing problem of the TWAVRP, we implement a mono-directional variant of their algorithm with a simplified dominance rule.

Labeling algorithms maintain a set of *labels* that each corresponds to a path starting at the depot. These paths are extended along all arcs, which results in new labels being generated according to the *extension functions*. Labels that are infeasible are eliminated. Furthermore, a *dominance rule* is applied to eliminate labels that are non Pareto-optimal. Eventually, we obtain a set of feasible elementary paths from the starting depot to the ending depot that includes the shortest elementary path.

Next, we present the definition of the labels, the extension functions and the dominance rule. Our notation is consistent with Contardo et al. (2015), to which we refer for more information on labeling algorithms for vehicle routing problems.

A label is given by a tuple $L = (i, \bar{c}(T), d, \tau, U, p)$, where $i \in V_\omega$ is the end vertex of the path associated with L, $\bar{c}(T)$ is the cost function that returns the minimum possible cost of the path if i is visited at or before time T, d is the cumulative vehicle load, τ is the earliest arrival time in i, $U \subseteq V'$ is the set of clients that are visited by the path or that are unreachable due to capacity or time constraints, and p is the predecessor label of L, i.e., the label that has been extended to obtain L. The components of label L are denoted by $i(L)$, $\bar{c}(L,T)$, $d(L)$, $\tau(L)$, $U(L)$ and $p(L)$, respectively.

The function $\bar{c}(L,T)$ is defined on the domain $T \in [\tau(L), e_{i(L)}]$, where $e_{i(L)}$ is the ending time of the exogenous time window of vertex $i(L)$, the end vertex of the path associated with L. It is shown by Ioachim et al. (1998) that $\bar{c}(L,T)$ is convex, non-increasing and piece-wise linear in T. Hence, $\bar{c}(L,T)$ can be represented by a list of coordinates.

The initial label is given by $L = (0, 0, 0, s_0, \emptyset, \emptyset)$. A label for which $i(L) = i$ can be extended to a label L' over the arc $(i,j) \in A_\omega$ if $j \notin U(L)$. For the extended label, $i(L')$, $d(L')$, $\tau(L')$, $U(L')$ and $p(L')$ are given by the following straightforward

extension functions:

$$
\begin{aligned}
i(L') &= j && (3.13)\\
d(L') &= d(L) + d^{\omega}_{j} && (3.14)\\
\tau(L') &= \max\{s_j, \tau(L) + \tau_{ij}\} && (3.15)\\
U(L') &= U(L) \cup \{j\} \cup \{k \in V'_{\omega} | d(L') + d^{\omega}_{k} > Q \vee \tau(L') + \tau_{jk} > e_k\} && (3.16)\\
p(L') &= L. && (3.17)
\end{aligned}
$$

The label L' is feasible if $d(L') \leq Q$ and $\tau(L') \leq e_j$.

Next, we give the extension function for $\bar{c}(L', T)$. Let $T^* \in [\tau(L'), e_j]$ be the time at which vertex j must be visited to minimize the cost of the path corresponding to L'. If vertex j is visited at time t^j then vertex i must be visited at or before time $t^j - \tau_{ij}$. Furthermore, vertex i must be visited at or before time e_i. It follows that

$$
T^* = \underset{t^j \in [\tau(L'), e_j]}{\arg\min} \{\bar{c}(L, \min\{t^j - \tau_{ij}, e_i\}) + \bar{c}_{ij}(t^j)\}. \tag{3.18}
$$

Hence, T^* can be determined by minimizing a one-dimensional, convex and piecewise linear function. This is straightforward, as there is always a minimum at one of the breakpoints of the function or at one of the two boundaries of the domain.

If vertex j must be visited at or before some time T, Ioachim et al. (1998) show that it is optimal to visit vertex j at time $t^j = \min\{T, T^*\}$. It follows that the extension function for $\bar{c}(L', T)$ is given by

$$
\bar{c}(L', T) = \begin{cases} \bar{c}(L, \min\{T - \tau_{ij}, e_j\}) + \bar{c}_{ij}(T) & \text{if } \tau(L') \leq T \leq T^* \\ \bar{c}(L, \min\{T^* - \tau_{ij}, e_j\}) + \bar{c}_{ij}(T^*) & \text{if } T^* \leq T \leq e_j. \end{cases} \tag{3.19}
$$

To reduce the number of labels, we apply a dominance rule to remove non Pareto-optimal labels. We say that label L_1 dominates label L_2 if all of the following conditions are met:

i. $i(L_1) = i(L_2)$

ii. $\bar{c}(L_1, T) \leq \bar{c}(L_2, T) \quad \forall T \in [\tau(L_1), e_{i(L_1)}] \cap [\tau(L_2), e_{i(L_2)}]$

iii. $d(L_1) \leq d(L_2)$

iv. $\tau(L_1) \leq \tau(L_2)$

v. $U(L_1) \subseteq U(L_2)$.

Conditions (i)-(v) imply that L_2 is not Pareto-optimal for any $T \in [\tau(L_1), e_{i(L_1)}] \cap [\tau(L_2), e_{i(L_2)}]$ (Liberatore et al., 2011). By Condition (i) we have that $e_{i(L_1)} = e_{i(L_2)}$ and by Condition (iv) we have that $\tau(L_1) \leq \tau(L_2)$. Hence, Conditions (i) and (iv) together imply that $[\tau(L_1), e_{i(L_1)}] \cap [\tau(L_2), e_{i(L_2)}] = [\tau(L_2), e_{i(L_2)}]$. It follows that L_2 is not Pareto-optimal for any $T \in [\tau(L_2), e_{i(L_2)}]$, which is the complete domain of $\bar{c}(L_2, T)$. As such, label L_2 can be eliminated.

With our dominance rule, label L_1 can only dominate label L_2 if $\bar{c}(L_1, T) \leq \bar{c}(L_2, T)$ for all T in the intersection of the two domains. The dominance rule presented by Liberatore et al. (2011) also allows L_1 to dominate L_2 on a subinterval of $[\tau(L_1), e_{i(L_1)}]$, which results in more labels being eliminated. The advantage of our dominance rule is that comparing labels is easier, as no subintervals have to be determined. Furthermore, if the exogenous time windows and the travel times are all integers, it can be shown that the breakpoints of $\bar{c}(L, T)$ are integer (Ioachim et al., 1998). This improves the numerical stability of the algorithm.

3.4.2 Route relaxations

Enforcing elementarity in the pricing problem can result in a large amount of Pareto-optimal labels, because there are 2^n possible subsets of the clients. To reduce the number of labels, elementarity is often (partially) relaxed and cyclic routes are allowed to be added to the RMP. For a cyclic route $r = (P, t)$, a_r^i is the number of times that client $i \in V'$ is visited, b_r^{ij} is the number of times that arc $(i, j) \in A$ is used and t_r^i is the sum of all the times at which client $i \in V'$ is visited.

Adding cyclic routes to the RMP may decrease the value of the lower bound provided by (3.1)-(3.8). In an integral solution to the TWAVRP, however, cyclic routes cannot be selected due to Constraints (3.2). It follows that the branch-price-and-cut algorithm remains exact.

We incorporate the *ng*-route relaxation introduced by Baldacci et al. (2011) and the Strong Degree Constraints (SDCs) introduced by Contardo et al. (2014). Both have been used to solve various vehicle routing problem. E.g., the *ng*-route relaxation has been applied to the TWAVRP with Time-Dependent Travel Times by Spliet et al. (2018) and the combination of *ng*-route relaxation and SDCs has been shown to be effective for the Vehicle Routing Problem with Time Windows by Contardo et al. (2015).

We initialize the *ng*-route relaxation with neighborhoods of size 10. As proposed by Roberti and Mingozzi (2014), we allow for dynamically adding clients to these neighborhoods to further eliminate cycles. The number of times that a client can be

added to a neighborhood is limited to 5 per scenario. If cycles remain after increasing the neighborhoods, we add at most 30 SDCs per scenario, as long as they are violated by at least 0.05.

To use the *ng*-route relaxation and the SDCs, both the RMP and the labeling algorithm have to be modified. These modifications are not TWAVRP specific and are detailed in Contardo et al. (2015).

3.4.3 Heuristic pricing

To speed up the column generation algorithm we make use of heuristic dynamic programming as described by Desaulniers et al. (2008), among others.

At first, we ignore arcs that do not seem promising according to their reduced cost. The reduced cost of an arc $(i,j) \in A_\omega$ in scenario $\omega \in \Omega$ is given by $\bar{c}_{ij}(t^j) = p_w c_{ij} - \beta_j^\omega - \pi_j^\omega t^j$ if $j \in V'_\omega$ and $\bar{c}_{ij}(t^j) = p_w c_{ij}$ otherwise. For each vertex, we sort the incoming and outgoing arcs by reduced cost, ignoring the time-dependent term. Then, we keep for each vertex the ξ incoming arcs with the least reduced cost and the ξ outgoing arcs with the least reduced cost, where ξ is a preset parameter equal to 10, and increased to 15 if no negative reduced cost columns can be found.

Second, we ignore some of the resources when using the dominance rule. This simplifies the pricing problem, which can now be solved quickly by our labeling algorithm. If the labeling algorithm identifies a route with negative reduced cost, it can be added to the RMP.

If no more routes with negative reduced cost can be found, we take back into account some of the arcs and some of the resources that we previously ignored. Eventually we solve the full pricing problem, and as a result our column generation algorithm remains exact.

3.4.4 Valid inequalities

In this section we introduce the valid inequalities that we use as cutting planes to strengthen the lower bound of (3.1)-(3.8).

The first class of valid inequalities that we use is the rounded capacity inequalities, or the *capacity cuts* for short, which are known to be effective for vehicle routing problems (Baldacci et al., 2012). The capacity cuts are given by

$$\sum_{(i,j)\in A_\omega | i\in S, j\in V_\omega \setminus S} x_{ij}^\omega \geq \left\lceil \frac{\sum_{i\in S} d_i^\omega}{Q} \right\rceil \quad \forall \omega \in \Omega, S \subseteq V'_\omega, |S| \geq 2. \tag{3.20}$$

The left-hand side of Inequality (3.20) is the total arc flow from the clients in a given subset S to the other vertices, which is an upper bound on the number of vehicles visiting the clients in S. This follows from the fact that the same vehicle can enter and exit S multiple times. The right-hand side of Inequality (3.20) is a lower bound on the number of vehicles that is required to satisfy the demand of the clients in S.

We use the heuristic separation algorithm by Lysgaard (2003) to find violated capacity cuts. The details of this algorithm are presented in Lysgaard et al. (2004). Any inequality defined on the x-variables can be included in the master problem without complicating the pricing problem, see Desaulniers et al. (2011) for details.

Next to the capacity cuts, we make use of the three client subset-row inequalities (3SR-inequalities), which is a subclass of the more general class introduced by Jepsen et al. (2008). The 3SR-inequalities can be expressed as follows:

$$\sum_{r \in \mathcal{R}(\omega)} \left\lfloor \frac{\sum_{i \in S} a_r^i}{2} \right\rfloor \theta_r^\omega \leq 1 \quad \forall \omega \in \Omega, S \subseteq V'_\omega, |S| = 3, \tag{3.21}$$

that is, for each subset of three clients, there can be at most one route that visits at least two of them.

3SR-inequalities can readily be added to the master problem, but they do complicate the pricing problem as their duals cannot be incorporated in the modified arc costs. Such inequalities are said to be non-robust (Pessoa et al., 2008). Jepsen et al. (2008) detail how the labeling algorithm can be modified to allow for 3SR-inequalities in the master problem. This involves introducing additional resources to correctly model the reduced cost.

Each time that the master problem is solved to optimality, we first add all capacity cuts with a violation of at least 0.05. After adding the capacity cuts, the master problem is resolved. If no capacity cuts can be added, we try increasing the *ng*-route relaxation neighborhoods and adding SDCs, as discussed in Section 3.4.2. If neither can be done, we add per iteration at most ten 3SR-inequalities with a minimum violation of 0.1.

We do not add precedence inequalities as valid inequalities. The precedence inequalities are used in Section 3.3.2.1 to create a conflict graph for the restricted TWAVRP, and in Section 3.3.3 to derive stronger constraints to cut off integral edge flows. Preliminary experiments showed that if we already address orientation-symmetry, then adding precedence inequalities as cutting planes does not have a significant effect on the performance of our algorithm.

3.4.5 Branching

We explore the nodes of the search tree using best-first search. We first try to branch on the number of vehicles used in a given scenario. Next, we use our edge-based branching method combined with additional components, as introduced in Section 3.3.1, to address orientation-symmetry. Note that branching on the number of vehicles does not introduce orientation-symmetry into the search tree because orientation-symmetric solutions use the same number of vehicles.

To address orientation-symmetry as in Section 3.3.1 we either branch on a fractional edge or we add a constraint to forbid the current integral edge flow. Enforcing that the edge $[i, j] \in E$ is not used in scenario $\omega \in \Omega$ is achieved by removing the edge from the pricing problem. To force flow on edge $[i, j] \in E$ in scenario $\omega \in \Omega$ we use the constraint $z_{ij}^{\omega} = 1$. The constraints that are used to cut off the integral edge flows are also stated in terms of z_{ij}^{ω}. By definition, constraints in terms of the z-variables can be rewritten in terms of the x-variables, and can thus be added to the master problem without complicating the pricing problem.

3.5 Computational experiments

The BPC algorithm is implemented in GENCOL, which is a general purpose solver for solving routing and scheduling problems through decomposition and column generation. GENCOL is coded in C and C++.

All experiments are run on a server with an Intel Xeon E3-1226 v3 3.30GHz processor and 16GB of RAM. For a fair comparison with earlier work we use only a single thread and we set a time limit of one hour per instance for all experiments. All linear programs are solved with the commercial solver CPLEX version 12.6.3. To prevent problems with numerical stability of the simplex method, we enable the *numerical emphasis* setting.

3.5.1 Test instances

We make use of the benchmark instances for the TWAVRP as introduced by Spliet and Gabor (2015) and extended in Chapter 2. These instances are available in the vehicle routing problem repository VRP-REP (Mendoza et al., 2014).

In total there are 90 benchmark instances, consisting of ten instances with 10 clients, ten instances with 15 clients, etc., up to 50 clients. These clients are uniformly distributed over a square with sides of length five. The starting depot and the ending

depot are both located at the center of the square. The travel costs and the travel times in hours are given by the Euclidean distances between the locations.

Each instance contains three demand scenarios with the same probability of occurrence. The demands of different clients are uncorrelated and the average demand is about 1/6 of the vehicle capacity. The exogenous time windows have a width of 10.8 hours on average, while the endogenous time windows have a width of two hours.

To also be able to test our algorithm on larger instances, we extend the instance set that was used in Chapter 2. The new instances are generated in the same way, but contain more clients and demand scenarios. The extended instance set are available on VRP-REP.

3.5.2 Comparison with other algorithms

We first test our BPC algorithm on the benchmark instances of Chapter 2 and we compare our results. Note that the results in Chapter 2 were obtained on an Intel i7 3.5GHz processor, which is comparable to, if not better than, the processor that is used for the results in this chapter.

The results are summarized in Table 3.1 and the full table is presented as Table 3.5 in Appendix 3.C. The algorithm presented in Chapter 2 is denoted by BC. The BPC algorithm presented in this chapter is denoted by BPC+OS to stress the fact that this algorithm addresses orientation-symmetry.

The columns labeled 'Seconds' state the average times in seconds to (attempt to) solve the benchmark instances to optimality, with a maximum time of 3600 seconds allowed per instance. The number of nodes in the search tree that have been explored during this time is given by the 'Nodes' columns.

The 'Optimality gap' columns present the percentage gap between the optimal objective value and the lower bound after the algorithm terminates. Similarly, 'Root gap' presents the percentage gap between the optimal objective value and the lower bound after processing the root node. If the optimal objective value is unavailable we use the best known upper bound from either BC or BPC+OS to calculate the gaps. Finally, the columns labeled 'Solved' state the total number of instances that could be solved within one hour of computation time.

Table 3.1 shows that for the given benchmark instances, BPC+OS outperforms BC. By using BPC+OS instead of BC, the average solution time is decreased by 78%. All instances that can be solved by BC can also be solved by BPC+OS (Table 3.5). Additionally, 29 instances that could not be solved in Chapter 2 are now solved to optimality. Only four of the benchmark instances remain unsolved.

	Seconds		Nodes		Optimality gap		Root gap		Solved	
Clients	**BC**	**BPC+OS**	**BC**	**BPC+OS**	**BC**	**BPC+OS**	**BC**	**BPC+OS**	**BC**	**BPC+OS**
10	0.1	0.2	9.3	1.2	0	0	0.17	0.04	10	10
15	4.6	2.3	1,498.4	10.8	0	0	0.59	0.10	10	10
20	2.2	2.5	116.9	2.6	0	0	0.35	0.05	10	10
25	12.4	11.6	524.3	5.4	0	0	0.69	0.03	10	10
30	544.0	70.6	9,336.6	12.5	0.15	0	1.67	0.13	9	10
35	1531.7	421.4	16,846.9	23.3	0.33	0.02	1.47	0.12	6	9
40	3252.0	542.6	22,903.4	15.3	0.82	0.03	2.18	0.12	2	9
45	3600.0	705.8	10,089.7	9.3	1.72	0.03	2.53	0.09	0	9
50	3600.0	1028.7	4,904.6	16.0	2.35	0.14	2.90	0.23	0	9
Avg.	1394.1	309.5	7,358.9	10.7	0.60	0.02	1.39	0.10	57/90	86/90

Table 3.1: Comparison between BC and BPC+OS on the benchmark instances from Chapter 2.

BC and BPC+OS use different strategies to solve the TWAVRP. BC has a relatively weak LP relaxation that is easy to calculate, while BPC+OS relies on a strong bound that takes more computational effort to determine. It is thus no surprise that BPC+OS explores less nodes and has smaller root gaps than BC. What is interesting to observe, however, is how big these differences are. The average root gap for BPC+OS is only 0.10% and on average only 10.7 nodes have to be explored to close this gap. This is in contrast to the 1.39% average root gap for BC, which requires 7,358.9 nodes on average to close the gap.

Next, we compare the solution times of BPC+OS with those of the scenario decomposition algorithm by Subramanyam et al. (2018). The computational experiments by Subramanyam et al. (2018) are based on the same benchmark instances and are performed on an Intel Xeon E5-2687W 3.1GHz processor using a single thread. As such, the computation times allow for a fair comparison. For convenience, the solution times reported by Subramanyam et al. (2018) are available in Table 3.6 in Appendix 3.C.

With the scenario decomposition algorithm, it takes 207.2 seconds on average to (attempt to) solve the benchmark instances to optimality. In total, 89 out of the 90 instances are solved to optimality within one hour of computation time. For BPC+OS, the average solution time is 309.5 seconds, and 86 instances can be solved to optimality. If we only consider the instances that are solved to optimality by both algorithms, the average for the scenario decomposition is 129.1 seconds, compared to 156.5 seconds for BPC+OS.

The scenario decomposition algorithm has the best solution time for 69 instances, while BPC+OS has the best solution time for 24 instances (note that the numbers do not sum to 90 due to draws). The instances for which BPC+OS has better performance tend to be large: out of the 20 largest instances, BPC+OS is faster than the scenario decomposition algorithm for eight of them.

We conclude that our BPC algorithm is competitive with the state-of-the-art when orientation-symmetry is properly addressed. The scenario decomposition algorithm uses several elements as described in Pecin et al. (2017) that have not yet been added to our BPC algorithm, including *bidirectional labeling*, *variable fixing*, *route enumeration*, and *limited-memory subset row cuts*. Adding such elements to BPC+OS is likely to improve performance further. Combining the two algorithms to get the best of both worlds is not trivial and may be an interesting topic for future research.

3.5.3 Effect of addressing orientation-symmetry

In this section we consider the isolated effect of addressing orientation-symmetry. To this end, we also adapt BC to address orientation-symmetry (denoted by BC+OS) and we test BPC+OS without addressing orientation-symmetry (denoted by BPC).

For BPC we add precedence inequalities in the same way as for BC (see Chapter 2). For BC+OS we do not add precedence inequalities; preliminary experiments suggest that adding precedence inequalities does not improve performance when addressing orientation-symmetry. The four algorithms are compared in Table 3.2, which summarizes data from Table 3.6 in Appendix 3.C.

	Seconds				**Solved**			
Clients	**BC**	**BC+OS**	**BPC**	**BPC+OS**	**BC**	**BC+OS**	**BPC**	**BPC+OS**
10	0.1	0.0	0.3	0.2	10	10	10	10
15	4.6	1.1	22.5	2.3	10	10	10	10
20	2.2	0.4	364.4	2.5	10	10	9	10
25	12.4	3.4	751.5	11.6	10	10	8	10
30	544.0	461.6	1894.7	70.6	9	9	5	10
35	1531.7	1208.3	1514.3	421.4	6	7	6	9
40	3252.0	3069.3	2548.0	542.6	2	2	4	9
45	3600.0	3600.0	2203.2	705.8	0	0	6	9
50	3600.0	3600.0	3002.8	1028.7	0	0	3	9
Avg.	1394.1	1327.1	1366.8	309.5	57/90	58/90	61/90	86/90

Table 3.2: Comparison between BC, BC+OS, BPC and BPC+OS on the benchmark instances of Chapter 2.

Table 3.2 shows that, for the given instances, addressing orientation-symmetry has a positive effect on the performance of both BC and BPC. For BC the average solution time decreases from 1394.1 seconds to 1327.1 seconds. Note, however, that the average is heavily influenced by the instances that cannot be solved before the time limit. If we only consider the instances that can be solved by both BC and BC+OS, we actually see a decrease of 37.5% in average solution time (Table 3.6).

For BPC, addressing orientation-symmetry clearly has a bigger effect: the average solution time decreases from 1366.8 seconds to 309.5 seconds. Furthermore, we see that the number of instances that can be solved to optimality increases from 61 to 86.

If we compare BC with BPC we observe that BC outperforms BPC if the number of clients is at most 30. For 35 clients or more, BPC has a better performance on average. If we compare BC+OS with BPC+OS we see that instances with up to

25 clients only take a short time to solve, and for instances with 30 clients or more, BPC+OS outperforms BC+OS.

One of the reasons that BPC+OS is so effective, is because addressing orientation-symmetry significantly reduces the amount of nodes that have to be processed. For BPC+OS, the average number of nodes that is processed for each benchmark instance is only 10.7, while for BPC the average is 145.0. This shows that orientation-symmetry is indeed prominent, and that properly addressing orientation-symmetry reduces the required computational effort.

3.5.4 Instances with additional clients

We have shown that the BPC algorithm that addresses orientation-symmetry can solve 86 out of the 90 benchmark instances of Chapter 2. To test the limits of our algorithm, we also perform computational experiments with the extended instance set. In this section, we first increase the number of clients while keeping the number of demand scenarios constant at three. In the next section, we increase the number of scenarios.

The results for instances with 55 clients up to 65 clients are presented in Table 3.3. We have chosen to only report the optimality gap and the root gap for instances that are solved to optimality. Note that we do not put effort into generating good upper bounds: the upper bound is only updated when we encounter a feasible solution. As such, the optimality gap and the root gap are uninformative if the problem is not solved to optimality.

Based on Table 3.3 we conclude that our algorithm cannot consistently solve the benchmark instances with more than 50 clients. We manage to solve four out of the ten instances with 55 clients, two out of the ten instances with 60 clients, and a single instance with 65 clients.

Only a small number of nodes can be processed within the one hour time limit. The long time per node is due to the more complicated pricing problems, but also due to the many valid inequalities that are added. In Section 3.5.3 we have seen that addressing orientation-symmetry reduces the number of nodes that have to be processed. This becomes even more important as the time spent per node increases.

3.5.5 Instances with additional scenarios

In this section we test the performance of our algorithm when the number of demand scenarios increases. We consider instances from the extended instance set with 10

Inst.	Clients	Seconds	Nodes	Optimality gap	Root gap
91	55	3600.0	17	-	-
92	55	3600.0	17	-	-
93	55	3600.0	38	-	-
94	55	3600.0	7	-	-
95	55	306.0	10	0	0.01
96	55	2082.7	9	0	0.06
97	55	1938.0	21	0	0.09
98	55	3600.0	5	-	-
99	55	3216.5	25	0	0.14
100	55	3600.0	8	-	-
101	60	3600.0	16	-	-
102	60	2042.6	7	0	0.04
103	60	3600.0	5	-	-
104	60	3600.0	15	-	-
105	60	3600.0	19	-	-
106	60	358.4	1	0	0
107	60	3600.0	5	-	-
108	60	3600.0	16	-	-
109	60	3600.0	5	-	-
110	60	3600.0	14	-	-
111	65	3600.0	2	-	-
112	65	3600.0	13	-	-
113	65	3600.0	3	-	-
114	65	3600.0	7	-	-
115	65	2742.1	7	0	0.02
116	65	3600.0	7	-	-
117	65	3600.0	12	-	-
118	65	3600.0	3	-	-
119	65	3600.0	13	-	-
120	65	3600.0	7	-	-

Table 3.3: Computational results for BPC+OS on instances with 55 up to 65 clients and three demand scenarios.

clients up to 50 clients, and with either three, five or seven demand scenarios.

The results of this computational experiment are summarized in Table 3.4. For two of the instances with five scenarios, CPLEX reported that one of the linear programs could not be solved due to numerical issues. These instances have been reported as unsolved with a solution time of 3600 seconds.

	Seconds			**Solved**		
Clients	**3 scen.**	**5 scen.**	**7 scen.**	**3 scen.**	**5 scen.**	**7 scen.**
10	0.2	0.5	0.9	10	10	10
15	2.3	364.2	1033.4	10	9	8
20	2.5	14.7	56.0	10	10	10
25	11.6	398.2	1294.7	10	9	7
30	70.6	514.5	1281.1	10	9	8
35	421.4	899.5	2531.8	9	9	5
40	542.6	1547.0	2926.9	9	7	5
45	705.8	2507.2	3308.5	9	5	1
50	1028.7	3427.2	3600.0	9	1	0
Avg.	309.5	1074.8	1781.5	86/90	69/90	54/90

Table 3.4: Computational results for BPC+OS on instances with 10 up to 50 clients and three up to seven demand scenarios.

As expected, Table 3.4 shows that the complexity of the TWAVRP increases with the number of scenarios. Out of the 90 instances with three scenarios, 86 instances can be solved to optimality in one hour of computation time. For five scenarios and seven scenarios the number of solved instances is 69 and 54, respectively.

Our algorithm can solve almost all instances with five scenarios and up to 35 clients in one hour of computation time. Out of the instances with seven scenarios, more than half of the instances up to 30 clients can be solved to optimality.

3.6 Conclusion

In this chapter we define orientation-symmetry for the TWAVRP and we observe that orientation-symmetry is common, especially for instances that are difficult to solve by exact methods. To overcome the problem of orientation-symmetry, we introduce an edge-based branching method combined with additional components that eliminates orientation-symmetry from the search tree. We then present a branch-price-and-cut algorithm to solve the TWAVRP while addressing orientation-symmetry.

Our computational experiments suggest that addressing orientation-symmetry significantly improves the BPC algorithm. On the benchmark set, the average solution time decreases from 1366.8 seconds to 309.5 per instance, and 25 additional instances are solved to optimality. Addressing orientation-symmetry also greatly reduces the number of nodes in the search tree: the average number of nodes per instance decreases from 145.0 to 10.7, or by 92.6%. The resulting algorithm is competitive with the scenario decomposition algorithm by Subramanyam et al. (2018).

Our experiments also show that addressing orientation-symmetry improves the BC algorithm presented in Chapter 2. Finally, we report computational results on instances with additional clients and instances with additional demand scenarios.

For future work it can be interesting to consider heuristics based on the current algorithm. Another direction for further research is to analyze how many demand scenarios are sufficient to obtain a good time window assignment under various assumptions about the demand distribution and the structure of the network.

Finally, we remark that the main ideas in this chapter are not TWAVRP specific and may be applied to other vehicle routing problems with consistency considerations or synchronization requirements. The algorithm that we propose for the restricted TWAVRP (Section 3.3.2) is based on a conflict graph. By redefining the conflict graph, the same algorithm can be used to solve the restricted version of other problems. Proposition 3.3 provides a general constraint to cut off integer edge flows, which is not unique to the TWAVRP. Given the benefit of addressing orientation-symmetry in the TWAVRP, applying our method to other problems is an interesting direction for further research.

Appendix

3.A Solving the TWAVRP for fixed arc flows

In this section we present a simple algorithm for solving the TWAVRP for fixed arc flows, which does not rely on linear programming.

For each scenario, we assume that the arc flows are integral and give m disjoint directed elementary paths from the starting depot to the ending depot. This implies that a potential solution consists of m routes. The k'th route is given by $r_k = (P_{r_k}, t_{r_k})$ and is used in scenario ω_k. Note that P_{r_k} follows directly from the arc flows. As P_{r_k} is known, we also know $a^i_{r_k}$ for all $i \in V'_{\omega_k}$ and $b^{ij}_{r_k}$ for all $(i,j) \in A_{\omega_k}$. By definition, $\theta^{\omega_k}_{r_k} = 1$ for all $k \in \{1, \ldots, m\}$ and all other route variables are equal

to zero.

Furthermore, we assume that the routes are such that, per scenario, the capacity constraint is satisfied and that each client is visited exactly once. If one of these conditions does not hold, we can immediately conclude that the TWAVRP is infeasible for the given arc flows.

Given these assumptions, the goal is to find values for $t^j_{r_k}$ for all $k \in \{1, \ldots, m\}$, $j \in V'_{\omega_k}$ and values for y_j for all $j \in V'_{\omega_k}$ such that the routes $r_k = (P_{r_k}, t_{r_k})$ are feasible and the Constraints (3.2)-(3.8) are satisfied. For convenience, let $t^0_{r_k}$ be the time that the vehicle assigned to route k leaves the depot and let $t^{n+1}_{r_k}$ be the time that the vehicle returns to the depot.

We present Algorithm 2 for solving the TWAVRP for fixed arc flows. This algorithm repeatedly calculates a lower bound $\underline{t}^j_{r_k}$ on the final value of $t^j_{r_k}$, and updates $t^j_{r_k}$ accordingly. The TWAVRP is proven to be infeasible for the given arc flows if the lower bound on $t^j_{r_k}$ exceeds the upper bound of value e_j. The loop ends after an iteration in which none of the $t^j_{r_k}$ values are updated. Finally, values for y_j are calculated.

Algorithm 2 Solving the TWAVRP for fixed arc flows

1: Set $t^0_{r_k} = s_0$ for all $k \in \{1, \ldots, m\}$.
2: Set $t^j_{r_k} = 0$ for all $k \in \{1, \ldots, m\}$ and $j \in V_{\omega_k} \setminus \{0\}$.
3: updated $= true$
4: **while** updated is $true$ **do**
5: updated $= false$
6: **for** $k = 1, \ldots, m$ **do**
7: **for** $(i, j) \in P_{r_k}$ **do**
8: $\underline{t}^j_{r_k} = \max\{s_j, t^i_{r_k} + \tau_{ij}, \max_{l \in \{1,\ldots,m\}} t^j_{r_l} - u_j\}$
9: **if** $\underline{t}^j_{r_k} > e_j$ **then**
10: **return** "Problem infeasible"
11: **else if** $t^j_{r_k} < \underline{t}^j_{r_k}$ **then**
12: $t^j_{r_k} = \underline{t}^j_{r_k}$
13: updated $= true$
14: **end if**
15: **end for**
16: **end for**
17: **end while**
18: Set $y_j = \max\{s_j, \max_{l \in \{1,\ldots,m\}} t^j_{r_l} - u_j\}$ for all $j \in V'_{\omega_k}$.

In Steps 1 and 2 the values of $t^j_{r_k}$ are initialized. Vehicles are set to leave the depot at time s_0. If client j is not contained in route r_k then, by definition, $t^j_{r_k} = 0$.

Hence, we initialize with the value zero and we only update $t^j_{r_k}$ if j is contained in route r_k.

In Step 8 we determine $\underline{t}^j_{r_k}$, which is a lower bound on the final value of $t^j_{r_k}$. Note that $(i,j) \in P_{r_k}$, which implies that route r_k first visits $i \in V_{\omega_k}$ and then visits $j \in V_{\omega_k}$ immediately after. For a route to be feasible, we require that $s_j \leq t^j_{r_k} \leq e_j$ if client j is contained in route r_k. Hence, $t^j_{r_k} \geq s_j$. Furthermore, we require that at least time τ_{ij} has passed between the visits to i and j. As the current value of $t^i_{r_k}$ is a lower bound on its final value, it follows that $t^j_{r_k} \geq t^i_{r_k} + \tau_{ij}$. Time $\max_{l\in\{1,\dots,m\}} t^j_{r_l}$ is the latest time that j is visited in any of the scenarios. Due to the width of the endogenous time window, the earliest time that j can be visited is $\max_{l\in\{1,\dots,m\}} t^j_{r_l} - u_j$. Hence, $t^j_{r_k} \geq \max_{l\in\{1,\dots,m\}} t^j_{r_l} - u_j$. Combining the three lower bounds shows that $\underline{t}^j_{r_k}$ as defined in Step 8 is a lower bound on the final value of $t^j_{r_k}$.

Finally, we need to show that Algorithm 2 produces a feasible solution if the TWAVRP is feasible for the given arc flows. If the algorithm does not return "Problem infeasible" we know that $s_j \leq t^j_{r_k} \leq e_j$ for all $k \in \{1,\dots,m\}$ and all j contained in route r_k. The requirement that $t^j_{r_k} \geq t^i_{r_k} + \tau_{ij}$ if $(i,j) \in P_{r_k}$ is enforced in Step 8. It follows that all $r_k = (P_{r_k}, t_{r_k})$ are valid routes.

It remains to show that Constraints (3.2)-(3.8) are satisfied. By assumption, Constraints (3.2), (3.6), (3.7) and (3.8) are already satisfied. In Step 18 we assign values to the y_j variables. It follows immediately that $y_j \geq s_j$ for all $j \in V'_{\omega_k}$. Because all routes are valid, we have $t^j_{r_k} \leq e_j$ for all $k \in \{1,\dots,m\}$. It then follows that $y_j = \max\{s_j, \max_{l\in\{1,\dots,m\}} t^j_{r_l} - u_j\} \leq \max\{s_j, e_j - u_j\}$. By definition of the parameters, $e_j - u_j \geq s_j$ and thus $y_j \leq e_j - u_j$ for all $j \in V'_{\omega_k}$, which shows that Constraints (3.5) are satisfied.

The while-loop only terminates when $t^j_{r_k} \geq \underline{t}^j_{r_k}$ for all $k \in \{1,\dots,m\}$ and all j contained in route k. From Steps 8 and 18 it follows that $\underline{t}^j_{r_k} \geq y_j$, and thus $t^j_{r_k} \geq \underline{t}^j_{r_k} \geq y_j$, if j is contained in route k. It follows that Constraints (3.3) are satisfied.

By Step 18 we have that $y_j + u_j = \max\{s_j + u_j, \max_{l\in\{1,\dots,m\}} t^j_{r_l}\} \geq \max_{l\in\{1,\dots,m\}} t^j_{r_l}$ $\geq t^j_{r_l}$ for all $l \in \{1,\dots,m\}$ and all j contained in route l. It follows immediately that Constraints (3.4) are satisfied.

We conclude that Algorithm 2 terminates with a feasible solution if one exists and states that the problem is infeasible otherwise.

3.B Proof of Proposition 3.3

Proposition. *Let the current integral edge flows be given by $\bar{z}_{ij}^{\omega}$ for all $[i,j] \in E$ and $\omega \in \Omega$. For each scenario $\omega \in \Omega$, let $E^{\omega} \subset E$ be any subset of edges with non-zero flow in scenario ω, satisfying the following technical condition: if $[0,j] \in E^{\omega}$ and $\bar{z}_{0j}^{\omega} = 1$ then E^{ω} contains an edge adjacent to $[0,j]$. Then, the constraint*

$$\sum_{\omega\in\Omega}\left(\sum_{[i,j]\in E^{\omega}} z_{ij}^{\omega} + \sum_{[i,j]\in E^{\omega}|i,j\in V'} \lambda_{ij}^{E^{\omega}} z_{ij}^{\omega}\right) \leq \sum_{\omega\in\Omega}\left(\sum_{[i,j]\in E^{\omega}} \bar{z}_{ij}^{\omega} + \sum_{[i,j]\in E^{\omega}|i,j\in V'} \lambda_{ij}^{E^{\omega}} \bar{z}_{ij}^{\omega}\right) - 1 \tag{3.11}$$

is violated by an integral edge flow if and only if $z_{ij}^{\omega} = \bar{z}_{ij}^{\omega}$ for all $[i,j] \in E^{\omega}$ and $\omega \in \Omega$.

Proof. It is trivial that if $z_{ij}^{\omega} = \bar{z}_{ij}^{\omega}$ for all $[i,j] \in E^{\omega}$ and $\omega \in \Omega$, then Constraint (3.11) is violated. It remains to prove the converse implication: if Constraint (3.11) is violated for an integral edge flow then it must be that $z_{ij}^{\omega} = \bar{z}_{ij}^{\omega}$ for all $[i,j] \in E^{\omega}$ and $\omega \in \Omega$.

Partition each E^{ω} into $u < \infty$ sets $E_1^{\omega}, E_2^{\omega}, \ldots, E_u^{\omega}$ such that every non-depot edge is in the same partition as its adjacent depot edges. By definition, $\lambda_{ij}^{E^{\omega}}$ is the number of depot edges in E^{ω} that is adjacent to i or j. Hence, by the choice of partition, $\lambda_{ij}^{E_k^{\omega}} = \lambda_{ij}^{E^{\omega}}$ if $[i,j] \in E_k^{\omega}$ and $\lambda_{ij}^{E_k^{\omega}} = 0$ otherwise. It follows that Constraint (3.11) is equivalent to

$$\sum_{\omega\in\Omega}\sum_{k\in\{1,\ldots,u\}}\left(\sum_{[i,j]\in E_k^{\omega}} z_{ij}^{\omega} + \sum_{[i,j]\in E_k^{\omega}|i,j\in V'} \lambda_{ij}^{E_k^{\omega}} z_{ij}^{\omega}\right) \leq \sum_{\omega\in\Omega}\sum_{k\in\{1,\ldots,u\}}\left(\sum_{[i,j]\in E_k^{\omega}} \bar{z}_{ij}^{\omega} + \sum_{[i,j]\in E_k^{\omega}|i,j\in V'} \lambda_{ij}^{E_k^{\omega}} \bar{z}_{ij}^{\omega}\right) - 1. \tag{3.22}$$

The partition of E^{ω} can be chosen such that every E_k^{ω} belongs to one of the following five classes:

1. $E_k^{\omega} = \emptyset$.
2. $E_k^{\omega} = \{[0,j]\}$ for some $j \in V'$ and $\bar{z}_{0j}^{\omega} = 2$.
3. $E_k^{\omega} = \{[i,j]\}$ for some $i,j \in V'$ and $\bar{z}_{ij}^{\omega} = 1$.
4. E_k^{ω} consists of the edge $[i,j]$ and one adjacent depot edge. That is, either $E_k^{\omega} = \{[0,i],[i,j]\}$ and $\bar{z}_{0i}^{\omega} = \bar{z}_{ij}^{\omega} = 1$, or $E_k^{\omega} = \{[0,j],[i,j]\}$ and $\bar{z}_{0j}^{\omega} = \bar{z}_{ij}^{\omega} = 1$.

5. E_k^ω consists of the edge $[i,j]$ and two adjacent depot edges. That is, $E_k^\omega = \{[0,i],[0,j],[i,j]\}$ and $\bar{z}_{0i}^\omega = \bar{z}_{0j}^\omega = \bar{z}_{ij}^\omega = 1$.

That such a partition is possible follows directly from Observation 3.1 and the conditions stated in Proposition 3.3.

Lemma 3.4. *If E_k^ω belongs to one of the five classes, then*

$$\sum_{[i,j]\in E_k^\omega} z_{ij}^\omega + \sum_{[i,j]\in E_k^\omega} \lambda_{ij}^{E_k^\omega} z_{ij}^\omega \leq \sum_{[i,j]\in E_k^\omega} \bar{z}_{ij}^\omega + \sum_{[i,j]\in E_k^\omega} \lambda_{ij}^{E_k^\omega} \bar{z}_{ij}^\omega. \tag{3.23}$$

Furthermore, equality holds if and only if $z_{ij}^\omega = \bar{z}_{ij}^\omega$ for all $[i,j] \in E_k^\omega$.

Proof. For the first class, $E_k^\omega = \emptyset$, the result is trivial. For the second class, Inequality (3.23) reduces to $z_{0j}^\omega \leq 2$ for which the lemma trivially holds. The third class is also trivial: in this case, Inequality (3.23) reduces to $z_{ij}^\omega \leq 1$.

For the fourth class, we assume $E_k^\omega = \{[0,i],[i,j]\}$ and $\bar{z}_{0i}^\omega = \bar{z}_{ij}^\omega = 1$. The proof for $E_k^\omega = \{[0,j],[i,j]\}$ and $\bar{z}_{0j}^\omega = \bar{z}_{ij}^\omega = 1$ is analogous. Inequality (3.23) now reduces to $z_{0i}^\omega + 2z_{ij}^\omega \leq 3$. If we have $z_{ij}^\omega = 0$, then the left-hand side of this inequality is maximized by choosing $z_{0i}^\omega = 2$, which gives value 2. If we have $z_{ij}^\omega = 1$, then client i is not contained in a single client path in scenario ω, which implies that $z_{0i}^\omega \neq 2$. It follows that the left-hand side is maximized by choosing $z_{0i} = 1$, which gives the value 3. Hence, Inequality (3.23) is satisfied for all integral edge flows and we have equality if and only if $z_{0i} = 1 = \bar{z}_{0i}$ and $z_{ij} = 1 = \bar{z}_{ij}$.

The fifth class is given by $E_k^\omega = \{[0,i],[0,j],[i,j]\}$ and $\bar{z}_{0i}^\omega = \bar{z}_{0j}^\omega = \bar{z}_{ij}^\omega = 1$. Inequality (3.23) then reduces to $z_{0i}^\omega + z_{0j}^\omega + 3z_{ij}^\omega \leq 5$. If $z_{ij}^\omega = 0$ then we maximize the left-hand side by choosing $z_{0i}^\omega = z_{0j}^\omega = 2$, which results in the value 4. If $z_{ij}^\omega = 1$ then $z_{0i}^\omega \neq 2$ and $z_{0j}^\omega \neq 2$. Hence, the left-hand side is maximized by choosing $z_{0i}^\omega = z_{0j}^\omega = 1$, which results in the value 5. It follows that Inequality (3.23) is satisfied for E_k^ω and we have equality if and only if $z_{0i} = 1 = \bar{z}_{0i}$, $z_{0j} = 1 = \bar{z}_{0j}$ and $z_{ij} = 1 = \bar{z}_{ij}$. □

By assumption, Constraint (3.11) is violated. Equivalently, Constraint (3.22) is violated. By integrality of the edge flow variables this implies

$$\sum_{\omega\in\Omega}\sum_{k\in\{1,\dots,u\}}\left(\sum_{[i,j]\in E_k^\omega} z_{ij}^\omega + \sum_{[i,j]\in E_k^\omega | i,j\in V'} \lambda_{ij}^{E_k^\omega} z_{ij}^\omega\right) \geq$$
$$\sum_{\omega\in\Omega}\sum_{k\in\{1,\dots,u\}}\left(\sum_{[i,j]\in E_k^\omega} \bar{z}_{ij}^\omega + \sum_{[i,j]\in E_k^\omega | i,j\in V'} \lambda_{ij}^{E_k^\omega} \bar{z}_{ij}^\omega\right). \tag{3.24}$$

Lemma 3.4 then implies that for every $\omega \in \Omega$ and $k \in \{1, \dots, u\}$ Inequality (3.23) holds with equality. By the same lemma this implies that $z_{ij}^\omega = \bar{z}_{ij}^\omega$ for all $[i, j] \in E_k^\omega$. As $E_1^\omega, \dots, E_u^\omega$ partition E^ω, it follows that $z_{ij}^\omega = \bar{z}_{ij}^\omega$ for all $[i, j] \in E^\omega$ and $\omega \in \Omega$. □

3.C Additional tables

Below, we present additional tables.

		Seconds		Nodes		Optimality gap		Root gap	
Inst.	**Clients**	**BC**	**BPC+OS**	**BC**	**BPC+OS**	**BC**	**BPC+OS**	**BC**	**BPC+OS**
1	10	0	0	1	1	0	0	0	0
2	10	0	0	18	1	0	0	0.28	0
3	10	0	0	1	1	0	0	0	0
4	10	0	0	6	1	0	0	0.14	0
5	10	0	0	42	1	0	0	0.34	0
6	10	0	0	1	1	0	0	0	0
7	10	0	0	1	1	0	0	0	0
8	10	0	0	21	3	0	0	0.96	0.41
9	10	0	0	1	1	0	0	0	0
10	10	0	0	1	1	0	0	0	0
11	15	0	1	1	1	0	0	0	0
12	15	39	13	14,037	93	0	0	2.36	0.07
13	15	3	3	587	7	0	0	1.78	0.88
14	15	0	1	1	1	0	0	0.03	0
15	15	1	0	11	1	0	0	0.17	0
16	15	0	1	1	1	0	0	0.17	0
17	15	0	0	1	1	0	0	0	0
18	15	1	1	124	1	0	0	0.47	0
19	15	1	2	210	1	0	0	0.97	0
20	15	0	1	11	1	0	0	0	0
21	20	1	3	48	3	0	0	1.11	0.15
22	20	9	2	658	11	0	0	0.19	0
23	20	0	1	1	1	0	0	0.12	0
24	20	2	2	58	1	0	0	0.86	0
25	20	7	8	389	5	0	0	1.09	0.34
26	20	0	2	1	1	0	0	0	0
27	20	0	4	1	1	0	0	0	0
28	20	1	1	11	1	0	0	0.08	0
29	20	1	2	1	1	0	0	0.05	0
30	20	0	2	1	1	0	0	0	0
31	25	2	5	20	1	0	0	0.57	0
32	25	1	6	4	1	0	0	0	0
33	25	9	16	395	5	0	0	1.03	0.04
34	25	11	6	391	3	0	0	0.33	0
35	25	6	3	201	5	0	0	0.85	0
36	25	39	5	1,733	3	0	0	1.49	0
37	25	22	51	1,472	31	0	0	0.43	0.25
38	25	10	7	337	3	0	0	0.59	0.05
39	25	7	9	219	1	0	0	0.94	0
40	25	15	8	471	1	0	0	0.62	0
41	30	137	12	3,281	1	0	0	1.99	0
42	30	3600	291	40,701	45	1.51	0	3.34	0.54
43	30	188	194	8,317	23	0	0	1.91	0.30
44	30	61	17	1,296	4	0	0	1.46	0
45	30	111	33	4,158	13	0	0	1.78	0.04
46	30	8	8	165	1	0	0	0.58	0
47	30	18	13	292	1	0	0	0.70	0
48	30	358	55	9,657	27	0	0	1.84	0.19
49	30	930	23	24,789	3	0	0	2.34	0.11
50	30	31	60	710	7	0	0	0.72	0.07
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮

		Seconds		Nodes		Optimality gap		Root gap	
Inst.	**Clients**	**BC**	**BPC+OS**	**BC**	**BPC+OS**	**BC**	**BPC+OS**	**BC**	**BPC+OS**
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
51	35	18	20	114	1	0	0	0.78	0
52	35	14	11	69	1	0	0	0.06	0
53	35	3600	185	32,201	11	1.08	0	2.63	0.36
54	35	3600	55	50,601	7	0.81	0	3.00	0.11
55	35	69	19	983	4	0	0	0.68	0
56	35	3600	40	42,289	7	0.92	0	2.58	0.05
57	35	3600	3600	31,201	170	0.44	0.24	1.85	0.39
58	35	128	127	1,871	15	0	0	0.89	0.16
59	35	245	33	2,485	2	0	0	0.93	0
60	35	443	124	6,655	15	0	0	1.32	0.11
61	40	3600	118	17,282	7	1.61	0	2.81	0.04
62	40	550	34	8,479	1	0	0	1.08	0
63	40	3600	208	36,554	5	0.23	0	1.83	0.09
64	40	3170	129	18,069	4	0	0	1.88	0.01
65	40	3600	199	37,915	5	0.62	0	2.07	0
66	40	3600	304	32,601	21	1.21	0	2.79	0.20
67	40	3600	354	27,201	19	0.51	0	1.73	0.15
68	40	3600	3600	16,210	78	1.76	0.28	2.88	0.49
69	40	3600	126	17,822	8	1.58	0	2.95	0.17
70	40	3600	356	16,901	5	0.68	0	1.76	0.03
71	45	3600	36	5,741	2	2.23	0	2.97	0
72	45	3600	173	9,201	9	1.95	0	2.66	0.03
73	45	3600	529	14,001	5	0.48	0	1.70	0.02
74	45	3600	113	4,789	7	1.08	0	2.03	0.01
75	45	3600	231	9,501	7	1.25	0	1.97	0.06
76	45	3600	342	6,401	5	2.91	0	3.41	0.06
77	45	3600	3600	12,064	25	2.13	0.29	2.87	0.58
78	45	3600	177	8,501	3	2.47	0	3.08	0
79	45	3600	1719	20,601	27	1.34	0	2.47	0.10
80	45	3600	139	10,097	3	1.38	0	2.15	0
81	50	3600	419	5,898	9	2.24	0	2.87	0.29
82	50	3600	1971	5,701	21	2.10	0	2.76	0.15
83	50	3600	1176	4,001	30	1.94	0	2.25	0.10
84	50	3600	331	4,401	5	2.98	0	3.49	0.07
85	50	3600	1182	3,146	17	2.23	0	2.99	0.06
86	50	3600	620	3,585	14	2.18	0	2.70	0.01
87	50	3600	222	6,611	2	2.31	0	2.87	0
88	50	3600	3600	6,801	44	3.68	1.43	4.39	1.56
89	50	3600	574	4,001	9	1.66	0	2.02	0.05
90	50	3600	193	4,901	9	2.13	0	2.68	0.01

Table 3.5: Detailed comparison between BC and BPC+OS on the benchmark instances of Chapter 2.

		Seconds				
Inst.	**Clients**	**BC**	**BC+OS**	**BPC**	**BPC+OS**	**SWG**
1	10	0.0	0.0	0.1	0.1	0.0
2	10	0.1	0.0	0.5	0.2	0.2
3	10	0.0	0.0	0.3	0.2	0.0
4	10	0.1	0.0	0.8	0.1	0.2
5	10	0.3	0.0	0.3	0.2	0.1
6	10	0.0	0.0	0.1	0.1	0.0
7	10	0.0	0.0	0.1	0.1	0.0
8	10	0.0	0.0	0.2	0.2	0.1
9	10	0.0	0.0	0.3	0.2	0.1
10	10	0.0	0.0	0.2	0.1	0.1
11	15	0.1	0.1	2.6	1.4	0.6
12	15	39.1	9.3	74.5	13.4	1.3
13	15	2.6	0.8	139.0	2.9	0.4
14	15	0.2	0.1	0.9	0.6	1.9
15	15	0.6	0.1	1.1	0.4	0.3
16	15	0.3	0.1	0.6	0.6	0.3
17	15	0.1	0.1	0.7	0.4	0.3
18	15	0.8	0.2	2.3	1.1	0.4
19	15	1.3	0.2	1.8	1.7	0.4
20	15	0.4	0.1	1.6	0.6	0.3
21	20	1.2	0.6	4.4	2.9	0.8
22	20	9.0	0.9	4.8	1.5	0.7
23	20	0.4	0.2	5.9	1.4	1.0
24	20	1.7	0.4	3.7	1.6	1.3
25	20	6.9	1.2	3600.0	7.5	7.8
26	20	0.2	0.1	4.6	1.8	0.8
27	20	0.3	0.2	9.4	3.5	1.0
28	20	1.1	0.3	2.5	1.0	0.8
29	20	0.5	0.2	4.4	1.9	0.6
30	20	0.3	0.1	4.2	2.1	0.6
31	25	2.3	1.1	14.0	5.1	2.6
32	25	1.3	0.7	28.0	6.1	3.0
33	25	9.4	3.5	163.3	15.7	9.9
34	25	11.1	1.9	19.0	5.6	3.4
35	25	6.1	4.1	9.0	3.3	3.1
36	25	39.3	11.7	25.3	4.9	3.9
37	25	22.4	4.1	3600.0	50.8	17.2
38	25	9.7	1.2	3600.0	7.0	38.5
39	25	7.2	3.1	26.6	9.3	2.5
40	25	15.2	2.4	29.3	8.3	2.0
41	30	137.0	49.4	47.5	12.1	2.7
42	30	3600.0	3600.0	3600.0	290.7	9.7
43	30	187.6	205.0	3600.0	193.6	285.4
44	30	60.6	23.2	49.0	16.7	19.9
45	30	110.8	48.2	775.9	33.4	26.1
46	30	7.7	2.7	30.0	8.3	11.8
47	30	17.8	12.0	44.5	13.0	5.7
48	30	357.9	155.8	3600.0	54.6	7.0
49	30	930.3	513.1	3600.0	23.4	67.6
50	30	30.7	6.8	3600.0	60.3	46.2
⋮	⋮	⋮	⋮	⋮	⋮	⋮

		Seconds				
Inst.	**Clients**	**BC**	**BC+OS**	**BPC**	**BPC+OS**	**SWG**
⋮	⋮	⋮	⋮	⋮	⋮	⋮
51	35	18.2	7.7	101.5	20.4	102.9
52	35	14.0	1.3	66.3	10.5	25.5
53	35	3600.0	3600.0	3600.0	184.5	39.3
54	35	3600.0	3600.0	3600.0	55.3	42.8
55	35	68.5	14.0	61.8	19.1	12.7
56	35	3600.0	3600.0	103.1	39.6	17.9
57	35	3600.0	1148.3	3600.0	3600.0	3600.0
58	35	127.9	15.4	262.7	127.4	64.1
59	35	245.1	28.5	147.5	33.3	14.8
60	35	443.3	67.9	3600.0	123.5	143.1
61	40	3600.0	3600.0	3600.0	117.5	9.7
62	40	550.3	153.9	311.9	33.5	27.8
63	40	3600.0	3600.0	928.9	207.6	18.2
64	40	3169.7	1738.9	993.2	128.5	192.2
65	40	3600.0	3600.0	1646.3	199.1	20.2
66	40	3600.0	3600.0	3600.0	303.9	62.7
67	40	3600.0	3600.0	3600.0	354.2	110.2
68	40	3600.0	3600.0	3600.0	3600.0	2506.0
69	40	3600.0	3600.0	3600.0	125.6	31.1
70	40	3600.0	3600.0	3600.0	356.1	445.3
71	45	3600.0	3600.0	537.3	36.1	20.4
72	45	3600.0	3600.0	2624.6	172.6	69.4
73	45	3600.0	3600.0	3600.0	529.2	39.6
74	45	3600.0	3600.0	786.6	113.0	228.6
75	45	3600.0	3600.0	3600.0	230.8	582.0
76	45	3600.0	3600.0	2274.8	342.4	1082.5
77	45	3600.0	3600.0	3600.0	3600.0	1241.8
78	45	3600.0	3600.0	198.5	176.5	102.2
79	45	3600.0	3600.0	3600.0	1718.7	99.4
80	45	3600.0	3600.0	1209.8	138.6	146.8
81	50	3600.0	3600.0	3600.0	418.7	559.8
82	50	3600.0	3600.0	3600.0	1970.8	211.0
83	50	3600.0	3600.0	3600.0	1175.9	2826.9
84	50	3600.0	3600.0	1127.3	331.2	115.7
85	50	3600.0	3600.0	3600.0	1181.5	1113.0
86	50	3600.0	3600.0	3600.0	619.7	306.1
87	50	3600.0	3600.0	3600.0	222.3	93.3
88	50	3600.0	3600.0	3600.0	3600.0	203.3
89	50	3600.0	3600.0	2604.3	574.3	1299.0
90	50	3600.0	3600.0	1096.1	192.8	231.6

Table 3.6: Detailed comparison between BC, BC+OS, BPC, BPC+OS, and SWG on the benchmark instances of Chapter 2. The SWG column displays the results by Subramanyam, Wang, and Gounaris (Subramanyam et al., 2018).

Chapter 4

Dynamic Time Window Adjustment

Abstract

To improve customer satisfaction in a delivery network with uncertain travel times, we propose to communicate time window adjustments to the customers throughout the day. We refer to these updates as dynamic time window adjustments. Dynamic time window adjustments are often used in practice, but have not yet been considered in the scientific literature. We provide a general model and we present the Dynamic Time Window Adjustment Problem (DTWAP). The DTWAP is the problem of optimizing the dynamic time window adjustments to maximize the expected customer satisfaction for a given route. Instead of solving the DTWAP in a specific setting, we derive general properties and we present three different solution methods. We also introduce the simple DTWAP, which is a special case that we analyze in more detail. The use of our results is demonstrated with an illustrative example concerning attended home delivery.

This chapter is based on Dalmeijer et al. (2019).

4.1 Introduction

In this chapter, we study the problem of communicating time window estimates to customers in a delivery network with uncertain travel times. For logistics service providers, it is common to update the customers about the estimated time window of delivery throughout the day, e.g., through text messages, app notifications, or by providing tracking information online. For example, major logistics companies FedEx, UPS and DHL all provide such services.

We use the term *dynamic time window adjustment* for the act of changing the time window that has been communicated to a customer. Dynamic time window adjustments are often used in practice. However, to the best of our knowledge, optimizing the dynamic time window adjustments has not been considered in the current scientific literature.

We introduce a general model for dynamic time window adjustment, and we consider the problem of determining adjustments that maximize the expected customer satisfaction for a given route. This optimization problem is referred to as the *Dynamic Time Window Adjustment Problem (DTWAP)*.

Two types of adjustment are considered: time window extension (EXT) and time window postponement (POS). For EXT, the start of the time window is fixed, and only the deadline can be adjusted. For example, a customer may be told that the original 1pm-3pm time window is extended to 1pm-4pm. For POS, the width of the time window is fixed, and the time window can be moved. For example, the 1pm-3pm time window may be postponed to 2pm-4pm.

We do not restrict ourselves to specific customer satisfaction functions. Instead, we discuss multiple solution methods, and we present an overview of which methods can be applied, depending on the properties of the customer satisfaction functions. To demonstrate how our work can be applied, an illustrative example is presented for which we optimize the customer satisfaction in an attended home delivery setting.

In the literature, we see an increasing interest in improving customer satisfaction, with different lines of research directly or indirectly addressing this topic. Examples are studies considering *consistent delivery* (Groër et al., 2009), *self-imposed time windows* (Jabali et al., 2015), *attended home delivery* (Agatz et al., 2011), *arrival time prediction* (Ulmer and Thomas, 2019), and *stochastic vehicle routing* (Gendreau et al., 1996).

It should be noted that in the literature, improving customer satisfaction is often combined with vehicle routing. In this chapter, we limit ourselves to studying the DTWAP, for which the route is assumed to be given. This choice is motivated by

the fact that, for general customer satisfaction functions, the DTWAP is already a challenging problem in itself. To be more precise, it is strongly NP-hard to solve the DTWAP, even for deterministic travel times. This result is proven in Section 4.2.

4.1.1 Literature review

To improve customer satisfaction, Groër et al. (2009) propose consistent delivery over multiple days or scenarios, e.g., always serve a customer with the same driver, or at approximately the same time. It is assumed that when the day or the scenario is known, the remaining vehicle routing problem is deterministic. This differs from our setting, where we assume that information is revealed during the day, and we are able to react dynamically.

Jabali et al. (2015) introduce the concept of self-imposed time windows. Self-imposed time windows are time windows that are chosen and communicated by the distributor to improve customer satisfaction. The main difference between assigning self-imposed time windows and dynamic time window adjustment, is that the former is done once, while the latter can be repeated throughout the day.

Han et al. (2017) consider randomness in attended home delivery due to customer no-show and customer response time, e.g., the time to answer the door. After completing an appointment, the authors dynamically determine the starting time of the next appointment and the maximum waiting time. This appointment scheduling subproblem has similarities with the special cases that we consider in Section 4.5. In general, an important difference with our work is that we communicate to all customers, not just to the one that will be visited next.

Ulmer and Thomas (2019) study the case where customers arrive dynamically, and immediately have to be given a (static) estimate of the arrival time. This differs from our setting, as we assume that the customers are static, and we can dynamically adjust the time windows.

Authors who study the Stochastic Vehicle Routing Problem (SVRP, Gendreau et al. (1996)) propose various ways to improve customer satisfaction. Zhang et al. (2013) select routes that guarantee a minimum on-time arrival probability at each customer location, while Taş et al. (2014) use linear penalties for early and late deliveries. Zhang et al. (2016) consider the SVRP with stochastic demands and propose different models to benefit the customer. They propose, among other things, to maximize the sum of the on-time probabilities and to consider earliness and tardiness penalties.

Jaillet et al. (2016) introduce a riskiness measure for time window violations based on the work of Aumann and Serrano (2008). Zhang et al. (2019) propose the

'essential riskiness index' which has properties similar to the index used by Jaillet et al. (2016) but gives some computational advantages. In both papers, minimizing the total riskiness is part of the objective when constructing the routes.

A different approach is taken by Errico et al. (2016). The authors consider the SVRP with stochastic service times, but explicitly do not allow the time windows to be adjusted. To deal with uncertainty, they introduce two recourse actions: *skip the current customer* and *skip the next customer*. Customers that are skipped receive emergency service, or are postponed to the following day.

For the SVRP variants mentioned above, determining the optimal actions for a given route is relatively easy. The main difficulty is due to integrating these decisions into the routing phase. For the DTWAP, we do not consider routing decisions. However, determining the optimal actions for a given route is significantly more difficult in this case.

The SVRP and the DTWAP are complementary, as one may first solve the SVRP to obtain high quality routes, and then solve the DTWAP for each route to improve customer satisfaction further. Integrating the DTWAP and the SVRP may yield additional benefits, and is an interesting direction for future research.

4.1.2 Contribution and outline

In this chapter, we introduce dynamic time window adjustment into the literature. Dynamic time window adjustment is common in practice, but has not yet attracted the attention of researchers. We provide a general model and we present the DTWAP, which is the problem of optimizing the dynamic time window adjustments to maximize the expected customer satisfaction.

Instead of solving the DTWAP in a specific setting, we present three different solution methods. We discuss when each method is applicable, and we consider its advantages and disadvantages. As a result, our work can be applied to different settings, including parcel delivery, attended home delivery, and retailer distribution. We illustrate this with an attended home delivery example, for which we use the results of this chapter to maximize customer satisfaction.

In Section 4.2, we formally introduce the DTWAP, and we prove that the problem is strongly NP-hard. In Section 4.3, we formulate the DTWAP as a multi-stage stochastic programming problem, and we present three solution methods. Section 4.4 is used to discuss the properties of the DTWAP, and to state sufficient conditions such that the methods in Section 4.3 can be applied.

In Section 4.5, we analyze the *simple DTWAP*, which is a special case of the

DTWAP with additional assumptions, including assumptions on the structure of the customer satisfaction functions. By exploiting this structure, the solution methods can be made more efficient. In Section 4.6, we consider the effect of discretizing time. We show that discretization may lead to suboptimal solutions in general. However, under certain assumptions, we prove that discretizing time yields an optimal solution for the simple DTWAP.

Our illustrative example is presented in Section 4.7. This example demonstrates how the results of this chapter can be used to maximize customer satisfaction in an attended home delivery setting. We present a model with customer dissatisfaction functions that are inspired by Section 4.5, and we make use of Section 4.3 and Section 4.4 to construct a solution method. We also present heuristics motivated by Sections 4.5 and 4.6. In the final section, we give a conclusion and we present some directions for further research.

4.2 Dynamic Time Window Adjustment Problem

In this section, we formally introduce the DTWAP. Assume that a fixed route is given, starting at the depot, visiting locations 1 up to n (in order), and returning to the depot. Locations $V' = \{1, \ldots, n\}$ correspond to the customers and locations 0 and $n+1$ correspond to the depot. Let $V = V' \cup \{0, n+1\}$.

The travel time between location i and location $i+1$ for $i \in V' \cup \{0\}$ is given by the non-negative random variable t_{ii+1}. We assume that all travel times are *stagewise independent*, i.e., t_{ii+1} is independent of $t_{01}, \ldots, t_{i-1,i}$, for all $i \in V' \cup \{0\}$ (Shapiro et al., 2009).

Let $[a_i, b_i]$ be the initial time window of customer $i \in V'$, with $a_i, b_i \in \mathbb{R}$ and $a_i \leq b_i$. One-sided time windows can be expressed by setting $a_i = -\infty$ or $b_i = \infty$. Time windows can be adjusted in two different ways: they can be extended (EXT) or postponed (POS). Let $\mathcal{P}_i$ be the set of allowed adjustments for customer i. For a given adjustment $p \in \mathcal{P}_i$, the EXT time window is given by $[a_i, b_i + p]$. We assume that $a_i \leq b_i + p$ for all $p \in \mathcal{P}_i$. The POS time window is given by $[a_i + p, b_i + p]$. We assume that the set $\mathcal{P}_i$ is closed and that $0 \in \mathcal{P}_i$ for all $i \in V'$. Note that $|\mathcal{P}_i|$ is allowed to be infinite.

On arrival at customer location $i \in V'$ (possibly before the time window opens), the following decisions are made. Let $p_j^i \in \mathcal{P}_j$ for $i < j$, $i \in V' \cup \{0\}$, $j \in V'$, be the adjustment of the time window of customer j, as decided when arriving at customer i. Note that the adjustment is defined with respect to the initial time window, i.e., the

time window before any adjustments are made. If $i = 0$, then p_j^i is the adjustment for customer j as decided before leaving the depot. Let $p^i = (p_{i+1}^i, \ldots, p_n^i)$ be the vector of adjustments as determined at location $i \in \{0, 1, \ldots, n-1\}$. By definition, $p^i \in \prod_{j=i+1}^{n} \mathcal{P}_j$. For convenience, we define $p_j^i = 0$ for $i = -1$.

The arrival time at customer $i \in V'$ is denoted by t_i and the departure time from customer i is denoted by x_i. Note that t_i and x_i both depend on earlier decisions and realizations of the random travel times. For now, we assume that the service times are equal to zero, such that customer i is served at time x_i. Let the parameter $x_0 \geq 0$ be the time at which the vehicle leaves the depot. For convenience, we define $t_0 = x_0$.

When the vehicle arrives at customer i, we consider three possible waiting behaviors.

1. Never Wait (NW): customer i is served immediately upon arrival, which may be outside of the time window.

2. Always Wait (AW): if necessary, the vehicle waits until the time window opens, and then immediately serves customer i. That is, the vehicle waits until a_i for EXT and until $a_i + p_i^{i-1}$ for POS.

3. Voluntarily Wait (VW): we decide dynamically how long the vehicle waits before serving customer i. Note that it is allowed to serve customer i early, even after waiting.

In all cases, customer i may be served after the deadline if the vehicle arrives late or waits.

We assume that the time window adjustment type (EXT or POS) and the waiting behavior (NW, AW, or VW) are the same for all customers. This results in six cases, which we abbreviate by EXT-NW, EXT-AW, EXT-VW, POS-NW, POS-AW, and POS-VW, respectively.

After the adjustments for the customers $j > i$ are decided, i.e., the vector p^i is chosen, the adjustments are communicated to the customers. We assume this communication to be instantaneous. Next, in the case of waiting behavior VW, we decide on how long to wait before serving customer i. We call this the voluntary waiting time $w_i \in W_i$, with $W_i = \mathbb{R}_{\geq 0}$. For notational convenience, we define the same decision for NW and AW, but with $W_i = \{0\}$, such that the voluntary waiting time is zero. At the depot, only the adjustment vector p^0 is determined, and no voluntary waiting decision is made.

For a given customer $i \in V'$, let $d_i(t_i, w_i, p_i^{i-1})$ return the departure time x_i, given the arrival time t_i, the voluntary waiting time w_i, and the adjustment p_i^{i-1}. The function $d_i(t_i, w_i, p_i^{i-1})$ depends on the type of adjustment (EXT or POS) and on the waiting behavior (NW, AW, or VW). Explicitly, we have:

$$d_i^{EXT-NW}(t_i, w_i, p_i^{i-1}) = t_i, \tag{4.1}$$
$$d_i^{EXT-AW}(t_i, w_i, p_i^{i-1}) = \max\{t_i, a_i\}, \tag{4.2}$$
$$d_i^{EXT-VW}(t_i, w_i, p_i^{i-1}) = t_i + w_i, \tag{4.3}$$
$$d_i^{POS-NW}(t_i, w_i, p_i^{i-1}) = t_i, \tag{4.4}$$
$$d_i^{POS-AW}(t_i, w_i, p_i^{i-1}) = \max\{t_i, a_i + p_i^{i-1}\}, \tag{4.5}$$
$$d_i^{POS-VW}(t_i, w_i, p_i^{i-1}) = t_i + w_i, \tag{4.6}$$

with w_i on the relevant domain W_i.

Without loss of generality, we minimize *customer dissatisfaction*, instead of maximizing customer satisfaction. We assume that the dissatisfaction of customer j can be written as

$$\sum_{i=0}^{j-1} g_j(t_i, p_j^{i-1}, p_j^i) + h_j(x_j, p_j^{j-1}), \tag{4.7}$$

for some functions g_j and h_j for all $j \in V'$.

The function g_j models the dissatisfaction that results from changing the adjustment of customer j from p_j^{i-1} to p_j^i. Note that the adjustment is communicated immediately upon arrival at customer i. The function h_j represents dissatisfaction due to serving customer j outside of the communicated time window. As such, it is dependent on the time that customer j is served and on the final adjustment p_j^{j-1}. We assume that the total dissatisfaction of the customers is given by the sum of the individual dissatisfaction functions.

We now formally introduce the DTWAP. Let a *state* be a triple of location $i \in V' \cup \{0\}$, arrival time t_i, and adjustment vector p^{i-1}. An *action* consists of choosing a new adjustment vector p^i and a voluntary waiting time w_i. The DTWAP is the problem of finding an optimal action for every state that is encountered by the vehicle, such that the total expected customer dissatisfaction is minimized.

We point out that driver costs and stochastic service times can easily be incorporated into the model. To incur driver costs, we add a dummy customer n' after customer n and we choose an appropriate function $h_{n'}$. Service time at customer i can be included by inserting a dummy customer i' after customer i. The travel time

between customers i and i' is taken to be the (stochastic) service time, and customer i' is given the non-adjustable time window $[0, \infty)$. Note that in this way, we can only model service times that are stagewise independent.

Finally, we prove that the DTWAP is strongly NP-hard, even for deterministic travel times, and for dissatisfaction functions that can be evaluated in polynomial time. Note that if we do not assume that the dissatisfaction functions can be evaluated in polynomial time, then the DTWAP is obviously hard to solve.

Proposition 4.1. *The DTWAP is strongly* NP*-hard, even for deterministic travel times, and for dissatisfaction functions that can be evaluated in polynomial time.*

Proof. See Appendix 4.A. □

4.3 Formulations and solution methods

The DTWAP can be seen as a *multi-stage stochastic programming problem*. We formalize this by stating the DTWAP in recursive form, i.e., as a stochastic dynamic program. Next, we consider three solution methods for multi-stage stochastic programming problems. We do so to identify properties that need to be satisfied by the DTWAP such that existing methods can be applied. For more details, we make references to the books by Bertsekas (2005), Kall and Mayer (2011), and Shapiro et al. (2009).

We consider the following three solution methods. In Section 4.3.2, we present a deterministic equivalent mathematical program for the DTWAP (Shapiro et al., 2009). In certain cases, this mathematical program can be solved efficiently by standard solvers. In Section 4.3.3, we consider dual decomposition methods (Kall and Mayer, 2011). In Section 4.3.4, we discuss discretizing time, such that the stochastic dynamic programming recursions can be solved directly (Bertsekas, 2005).

4.3.1 Stochastic dynamic programming formulation

Let $c_i(t_i, p^{i-1})$ be the *value function* of customer i. The value function is defined as the minimum expected total dissatisfaction of customers i through n, given that the vehicle arrives at customer i at time t_i, with adjustment vector p^{i-1}.

In the remainder, we assume that the DTWAP instances are *sufficiently expensive*. That is, for every attainable state, we assume that $c_i(t_i, p^{i-1}) > -\infty$. This condition is satisfied, for example, when the dissatisfaction functions are non-negative. Furthermore, we assume that the optimal action is attainable, i.e., there exist finite optimal

adjustments and voluntary waiting times. We do not consider these assumptions to be restrictive in practice.

The DTWAP can then be stated as follows.

$$c_n(t_n, p^{n-1}) = \min_{w_n \in W_n} \left\{ h_n \left(x_n, p_n^{n-1} \right) \right\}, \qquad \forall t_n \geq 0,\ p^{n-1} \in \mathcal{P}_n, \tag{4.8}$$

$$c_i\left(t_i, p^{i-1}\right) = \min_{\substack{p^i \in \prod_{j=i+1}^{n} \mathcal{P}_j \\ w_i \in W_i}} \left\{ \sum_{j=i+1}^{n} g_j(t_i, p_j^{i-1}, p_j^i) + h_i\left(x_i, p_i^{i-1}\right) + \mathbb{E}\left[c_{i+1}\left(x_i + t_{ii+1}, p^i\right)\right] \right\},$$
$$\forall i \in V' \backslash \{n\},\ t_i \geq 0,\ p^{i-1} \in \prod_{j=i}^{n} \mathcal{P}_j, \tag{4.9}$$

$$c_0\left(t_0\right) = \min_{p^0 \in \prod_{j=1}^{n} \mathcal{P}_j} \left\{ \sum_{j=1}^{n} g_j(t_0, 0, p_j^0) + \mathbb{E}\left[c_1\left(x_0 + t_{01}, p^0\right)\right] \right\}, \tag{4.10}$$

with $x_i = d_i(t_i, w_i, p_i^{i-1})$ for all $i \in V'$. Recall that x_0 and t_0 are parameters with the same value.

Equations (4.9) define the value function at customer $i \in V' \backslash \{n\}$. At customer i, and based on t_i and p^{i-1}, we decide on the optimal amount of voluntary waiting, w_i, and on the new adjustment vector p^i. The first term within the minimum represents the cost of all changes to the adjustment vector. The h_i term gives the cost of early and late deliveries at customer i. The third term is the expected cost from arriving at customer $i+1$ onwards.

Equations (4.8) and (4.10) are the boundary cases for the last customer and the depot, respectively. For the last customer, we only incur the penalty for early and late delivery. For the depot, we only determine the initial adjustment vector.

The DTWAP can be seen as a multi-stage stochastic optimization problem with $n+1$ stages. The first stage problem is to find the optimal adjustment vector p^0 before the vehicle leaves the depot. This corresponds to Equation (4.10). Next, the travel time to the first customer is revealed, and the vehicle arrives at customer 1 at time t_1 with postponement vector p^0. The second stage problem is to determine the optimal adjustment vector p^1 and voluntary waiting time w_1, which corresponds to Equations (4.9). Stages three up to n also correspond to Equations (4.9). The $n+1$'th stage problem is to decide w_n, as in Equations (4.8).

4.3.2 Solving the deterministic equivalent mathematical program

If the travel time distributions are discrete with finite support, we can formulate a deterministic mathematical program that is equivalent to (4.8)-(4.10). This construction is based on the scenario tree, which is discussed in more detail by Shapiro et al. (2009).

A *scenario* is a vector of realizations of all random variables. We denote scenario $k \in K$ by $\xi^k = (\xi^k_j)_{j=2,\dots,n+1}$, with ξ^k_j the information that is revealed before making the j'th stage decision. In our case, ξ^k_j is the travel time from customer $j-2$ to customer $j-1$ under scenario k. For example, at customer two we make the third stage decision, and the travel time from customer one to customer two is known at this point. The probability that a random scenario ξ is equal to scenario ξ^k is given by $\mathbb{P}(\xi = \xi^k) = \prod_{j=2}^{n+1} \mathbb{P}(t_{j-2,j-1} = \xi^k_j)$. This follows from the stagewise independence of the random travel times.

Next, we present the deterministic equivalent mathematical program. Each decision variable is duplicated $|K|$ times to represent the decisions in the different scenarios. That is, $p^i(\xi^k)$ is the adjustment vector that is determined at customer i under scenario ξ^k and $w_i(\xi^k)$ is the voluntary waiting time at customer i under scenario ξ^k. For clarity, we denote the travel time between customer i and customer $i+1$ by $t_{ii+1}(\xi^k)$. Note that $t_{ii+1}(\xi^k)$ is now a deterministic parameter. The variables $t_i(\xi^k)$ and $x_i(\xi^k)$ represent the arrival and departure times at customer i in scenario ξ^k.

The deterministic equivalent mathematical program can be stated as follows.

$$\min \sum_{k\in K}\left(\mathbb{P}(\xi=\xi^k)\sum_{j\in V'}\left(\sum_{i=0}^{j-1} g_j(t_i(\xi^k), p_j^{i-1}(\xi^k), p_j^i(\xi^k)) + h_j(x_j(\xi^k), p_j^{j-1}(\xi^k))\right)\right) \tag{4.11}$$

$$\text{s.t.,} \quad t_i(\xi^k) = x_{i-1}(\xi^k) + t_{i-1,i}(\xi^k), \qquad \forall i \in V', k \in K, \tag{4.12}$$

$$x_i(\xi^k) = d_i(t_i(\xi^k), w_i(\xi^k), p_i^{i-1}(\xi^k)), \quad \forall i \in V', k \in K, \tag{4.13}$$

$$p^i(\xi^k) = p^i(\xi^l), \qquad \forall i \in V'\backslash\{n\}, k, l \in K \text{ s.t. } \xi^k_j = \xi^l_j \ \forall j \le i+1, \tag{4.14}$$

$$w_i(\xi^k) = w_i(\xi^l), \qquad \forall i \in V', k, l \in K \text{ s.t. } \xi^k_j = \xi^l_j \ \forall j \le i+1, \tag{4.15}$$

$$p^i(\xi^k) \in \prod_{j=i+1}^{n} \mathcal{P}_j, \qquad \forall i \in \{0, \ldots, n-1\}, k \in K, \tag{4.16}$$

$$w_i(\xi^k) \in W_i, \qquad \forall i \in V', k \in K, \tag{4.17}$$

$$t_i(\xi^k) \in \mathbb{R}, \qquad \forall i \in V', k \in K, \tag{4.18}$$

$$x_i(\xi^k) \in \mathbb{R}, \qquad \forall i \in V', k \in K. \tag{4.19}$$

The Objective (4.11) is to minimize the expected total customer dissatisfaction over all scenarios. Constraints (4.12) define the arrival time at customer i for all $i \in V'$. Constraints (4.13) define the departure time from customer i, using the appropriate d_i function from (4.1)-(4.6).

Constraints (4.14)-(4.15) are the *nonanticipativity constraints* (Shapiro et al., 2009), which prevent decisions to be based on future information. For example, the value of $p^i(\xi^k)$ is a stage $i+1$ decision. As such, Constraints (4.14) enforce that if two scenarios k and l are the same up to stage $i+1$, then the decisions $p^i(\xi^k)$ and $p^i(\xi^l)$ are the same. It can be seen that nonanticipativity for $t_i(\xi^k)$ and $x_i(\xi^k)$ is implied by nonanticipativity for $p^i(\xi^k)$ and $w_i(\xi^k)$, and Equalities (4.12)-(4.13).

Solving the deterministic equivalent mathematical program (4.11)-(4.19) may be difficult. First, the number of scenarios, $|K|$, can be extremely large. For example, for n customers and only two possible travel time realizations per arc, we already have 2^n scenarios. To limit the number of scenarios, Sample Average Approximation (SAA, Kleywegt et al. (2002)) may be applied. However, especially for multi-stage problems, a large number of samples may still be necessary to obtain a good solution (Shapiro and Nemirovski, 2005).

Second, as a consequence of Proposition 4.1, the mathematical program (4.11)-(4.19) may already be difficult to solve for a single scenario. In Section 4.4, we give conditions under which the deterministic equivalent mathematical program can be solved efficiently for a moderate number of scenarios.

4.3.3 Dual decomposition methods

An important difficulty in solving stochastic dynamic programming recursions like (4.8)-(4.10) directly, is that the expected future value function is defined implicitly (Shapiro and Nemirovski, 2005). That is, for $F_{i+1}(x_i, p^i) = \mathbb{E}\left[c_{i+1}(x_i + t_{ii+1}, p^i)\right]$, we do not have a closed form analytic expression for the function F_{i+1}. On the other hand, if F_{i+1} were known explicitly, the DTWAP could be solved independently per stage as a straightforward mathematical program.

Dual decomposition methods construct an outer approximation of F_{i+1} by using linear inequalities, which is possible if F_{i+1} is convex. This idea goes back to Benders' decomposition (Benders, 1962) and the L-shaped method (Slyke and Wets, 1969). Pereira and Pinto (1991) introduce the Stochastic Dual Dynamic Programming (SDDP) method which includes Monte Carlo simulation to approximate F_{i+1}.

For more information on dual decompostion methods, we refer to Kall and Mayer (2011) and Pereira and Pinto (1991). For a detailed analysis of the SDDP method and the amount of samples that is needed to obtain good quality solutions, we refer to Shapiro (2011). For our discussion, the most important property to note is that dual decomposition methods rely on all functions F_{i+1} to be convex. In Section 4.4 we give sufficient conditions for the DTWAP such that this is the case.

4.3.4 Solving the discretized stochastic dynamic program

For specific models, it may be possible to solve the stochastic dynamic program (4.8)-(4.10) directly. For the appropriate solution methods, we refer to Bertsekas (2005). In general, however, solving the stochastic dynamic program is complicated due to the potentially infinite state-space, the potentially infinite action-space, and the potentially continuous travel time distributions.

By discretizing time, we obtain a DTWAP instance for which (4.8)-(4.10) is straightfoward to solve. Specifically, we approximate the vehicle starting time and the initial time windows by integers. Furthermore, we approximate each set of possible adjustments and each set of possible voluntary waiting times by a finite set of integer values. The travel time distributions are approximated by discrete distributions with finite support.

For the discretized instance, the number of possible states, possible actions, and possible travel time realizations are all finite. This allows us to solve the (4.8)-(4.10) directly by *backward recursion*. That is, we first use (4.8) to determine c_n. For every possible t_n and p^{n-1}, we enumerate $w_n \in W_n$ to find the minimum costs. When c_n is determined, we use (4.9) to determine c_{n-1}. In this case, we enumerate $p^{n-1} \in \mathcal{P}_n$ and $w_{n-1} \in W_{n-1}$. Note that, by assumption, the expectation can be written as a finite sum. We repeat the process until all value functions are completely determined.

Alternatively, one can solve (4.8)-(4.10) by *forward recursion*. We start by calculating $c_0(t_0)$, the optimal expected total dissatisfaction. To calculate this value, we must first calculate c_1 for all possible future states. For each such state, we calculate c_2 for all possible future states, etc. Once a value has been calculated, it is stored in memory to prevent that the same calculation is made twice. The advantage of

forward recursion is that we only calculate the values of the states that are necessary to determine the optimal dynamic time window adjustments. This is different from backward recursion, where we calculate the values of all states.

The discretization approach is straightforward and does not require any assumptions on the convexity of the dissatisfaction functions. For this reason, the same approach can be used when the model is extended, for example, by including time-dependent travel times (Ichoua et al., 2003).

The clear downside of the discretization approach is that the number of states may be very large, especially because the discretization must be sufficiently fine to obtain a good quality solution. Furthermore, solving the minimization problems by enumerating $p^i \in \mathcal{P}_{i+1} \times \ldots \times \mathcal{P}_n$ and $w_i \in W_i$ may be computationally expensive.

In Section 4.6, we consider the effect of discretization in more detail. We show that the discretization approach can result in suboptimal solutions, even if the vehicle starting time is integer, the initial time windows are integer, and the travel times have finitely many realizations that are all integer. On the other hand, we prove in the same setting that the discretization approach is exact for the simple DTWAP introduced in Section 4.5.

4.4 Properties

In Section 4.3 we have considered different solution methods for multi-stage stochastic programming problems, and we have identified the relevant properties such that they can be applied to the DTWAP. In this section, we give sufficient conditions under which these properties hold.

First, we consider properties such that the deterministic equivalent mathematical program (Section 4.3.2) can be solved as a convex or linear program, respectively. As in Section 4.3.2, we assume that the number of scenarios is finite.

Proposition 4.2. *Consider the following six conditions:*

1. $\mathcal{P}_i$ *is a closed continuous interval,* $\forall i \in V'$,
2. $g_j(t_i, p_j^{i-1}, p^i)$ *is convex,* $\forall j \in V'$,
3. $g_j(t_i, p_j^{i-1}, p^i)$ *is non-decreasing in* t_i, $\forall j \in V'$,
4. $h_i(x_i, p_i^{i-1})$ *is convex,* $\forall i \in V'$,
5. $h_i(x_i, p_i^{i-1})$ *is non-decreasing in* x_i *for* $x_i \geq a_i$, $\forall i \in V'$,
6. $h_i(x_i, p_i^{i-1})$ *is non-decreasing in* x_i *for* $x_i \geq a_i + p_i^{i-1}$, $\forall i \in V'$.

The deterministic equivalent mathematical program (4.11)-(4.19) *can be solved as a convex program in the following cases:*

- *The waiting behavior is NW or VW, and Conditions 1, 2, and 4 hold.*
- *The time window adjustment type is EXT, the waiting behavior is AW, and Conditions 1-5 hold.*
- *The time window adjustment type is POS, the waiting behavior is AW, and Conditions 1-4, and 6 hold.*

Proof. Under Conditions 2 and 4, we have that the Objective (4.11) is a convex function. By definition, Equations (4.12), (4.14)-(4.15), and (4.17) are linear constraints. Condition 1 ensures that (4.16) can be replaced by box constraints.

It remains to consider Constraints (4.13), i.e., $x_i(\xi^k) = d_i(t_i(\xi^k), w_i(\xi^k), p_i^{i-1}(\xi^k))$. Under waiting behavior NW or VW, we have that d_i is a linear function, which implies that (4.13) are linear constraints. Hence, (4.11)-(4.19) can be solved as a convex program in the first case.

Next, consider the combination of EXT and waiting behavior AW. We replace Constraints (4.13) by the two linear constraints $x_i(\xi^k) \geq t_i(\xi^k)$ and $x_i(\xi^k) \geq a_i$, for every $i \in V'$ and $k \in K$. By Conditions 3 and 5, it is optimal to choose $x_i(\xi^k)$ as small as possible, i.e., $x_i(\xi^k) = \max\{t_i(\xi^k), a_i\} = d_i(t_i(\xi^k), w_i(\xi^k), p_i^{i-1}(\xi^k))$. Hence, (4.11)-(4.19) can be solved as a convex program in the second case.

The proof for the third case is similar to the second case, and follows from replacing the Constraints (4.13) by $x_i(\xi^k) \geq t_i(\xi^k)$ and $x_i(\xi^k) \geq a_i + p_i^{i-1}$, for every $i \in V'$ and $k \in K$. □

Corollary 4.3. *If, in addition to the conditions of Proposition 4.2, the dissatisfaction functions are polyhedral convex functions, then the deterministic equivalent mathematical program* (4.11)-(4.19) *can be solved as a linear program.*

Proof. The convex programs that are constructed in the proof of Proposition 4.2 all have a convex objective function and linear constraints.

In the objective function, we may replace each $g_j(t_i(\xi^k), p_j^{i-1}(\xi^k), p_j^i(\xi^k))$ by a new variable $y_{ij}(\xi^k) \in \mathbb{R}$, if we also add the constraints $y_{ij}(\xi^k) \geq g_j(t_i(\xi^k), p_j^{i-1}(\xi^k), p_j^i(\xi^k))$. By the definition of polyhedral convex functions, the new constraints can be represented by a finite number of linear inequalities.

For the h_j terms, we can use a similar transformation. It follows that (4.11)-(4.19) can be solved as a linear program. □

Proposition 4.2 and Corollary 4.3 give conditions for which the deterministic equivalent mathematical program can be solved efficiently if the number of scenarios is not too big. In the linear case, general purpose solvers like CPLEX and Gurobi can be used immediately.

We remark that if the dissatisfaction functions can be modeled with integer variables and linear constraints, then the deterministic equivalent mathematical program can be solved as a mixed integer linear programming problem by the same commercial solvers.

Next, we consider conditions such that dual decomposition methods can be applied. As discussed in Section 4.3.3, we need that $F_{i+1}(x_i, p^i) = \mathbb{E}\left[c_{i+1}(x_i + t_{ii+1}, p^i)\right]$ is convex for all $i \in \{0, \ldots, n-1\}$. Note that we no longer assume that the travel times are discretely distributed.

Proposition 4.4. *Given the same conditions as in Proposition 4.2, the function $F_{i+1}(x_i, p^i)$ is convex for all $i \in \{0, \ldots, n-1\}$.*

Proof. We prove this by induction. That is, we assume that $F_{i+1}(x_i, p^i)$ is convex and we use (4.9) to prove that $F_i(x_{i-1}, p^{i-1})$ is convex. The proofs for the boundary cases F_n and F_1 are analogous.

If $g_j(t_i, p_j^{i-1}, p_j^i)$, $h_i(x_i, p_i^{i-1})$, and $F_{i+1}(x_i, p^i)$ are convex, then the right-hand side of (4.9) can be stated as a convex minimization problem. To do so, we replace $x_i = d_i(t_i, w_i, p_i^{i-1})$ by the appropriate linear constraints, similar as in the proof of Proposition 4.2, and we minimize over x_i. For EXT-AW, for example, we replace $x_i = d_i(t_i, w_i, p_i^{i-1})$ by $x_i \geq t_i$ and $x_i \geq a_i$. Note that for this proof, we do not require an explicit description of the function $F_{i+1}(x_i, p^i)$.

The function $c_i(t_i, p^{i-1})$ is the result of minimizing a convex function over a convex set for a subset of the variables. It follows that c_i is convex (Boyd and Vandenberghe, 2004). By definition, $F_i(x_{i-1}, p^{i-1}) = \mathbb{E}[c_i(x_{i-1} + t_{i-1,i}, p^{i-1})]$, which is a (possibly infinite) non-negative sum of convex functions, which is itself convex. □

Corollary 4.5. *If, in addition to the conditions of Proposition 4.2, the dissatisfaction functions are polyhedral convex functions and the travel time distributions are discrete with finite support, then the function $F_{i+1}(x_i, p^i)$ is polyhedral convex for all $i \in \{0, \ldots, n-1\}$.*

Proof. If $g_j(t_i, p_j^{i-1}, p_j^i)$, $h_i(x_i, p_i^{i-1})$, and $F_{i+1}(x_i, p^i)$ are polyhedral convex functions, then the minimization problems at the right-hand sides of (4.8)-(4.10) can be solved as linear programs. The required transformations are similar to those described in the proof of Corollary 4.5.

From linear programming duality, it follows that $c_i(t_i, p^{i-1})$ is a polyhedral convex function. By definition, $F_i(x_{i-1}, p^{i-1})$ is the sum of finitely many polyhedral convex functions, which is polyhedral convex. □

Proposition 4.4 and Corollary 4.5 give sufficient conditions such that dual decomposition methods, including SDDP, can be applied. These conditions are similar to the conditions that ensure that the deterministic equivalent mathematical program can be solved efficiently for a moderate number of scenarios. As discussed in Section 4.3.4, the discretization approach does not require any assumptions on the dissatisfaction functions.

4.5 Simple DTWAP

In this section, we introduce the *simple DTWAP*, a special case of the DTWAP, which is defined in Section 4.5.1. In Section 4.5.2, we show that the simple DTWAP can be solved relatively efficiently by the methods discussed in Section 4.3. Furthermore, we show that for EXT-NW, POS-NW, and EXT-AW, the simple DTWAP can be decomposed into n independent problems. This fact can be exploited by the solution methods. In Section 4.5.3, we demonstrate that ignoring one of the main assumptions of the simple DTWAP may result in a severely more complicated problem.

4.5.1 Assumptions

First, we assume that $\mathcal{P}_i$ is a continuous interval for all $i \in V'$. Next, we discuss our assumptions on the dissatisfaction functions.

We assume that the time at which an adjustment is communicated to the customer does not affect the amount of dissatisfaction received. That is, we can write $g_j(t_i, p_j^{i-1}, p_j^i)$ as $g_j(p_j^{i-1}, p_j^i)$. This assumption is restrictive, but may be appropriate if travel times are relatively long. In this case, updating the next customer before driving there may be sufficiently far in advance, such that the customer is indifferent about the timing of the information. In Section 4.5.3, we show that omitting the above assumption may result in a considerably more complicated problem.

For the simple DTWAP, we assume that increasing and decreasing the adjustment both yield a penalty that is linear in the amount of change. That is, we assume that

$$g_j(p_j^{i-1}, p_j^i) = \left(p_j^i - p_j^{i-1}\right)^+ \alpha_j + \left(p_j^{i-1} - p_j^i\right)^+ \beta_j, \tag{4.20}$$

for some parameters $\alpha_j, \beta_j \in \mathbb{R} \cup \{\infty\}$, and using $y^+ = \max\{0, y\}$.

If any of the parameters is chosen to be ∞, we replace the corresponding term by a domain restriction. For example, if decreasing the adjustment is not allowed, we set $\beta_j = \infty$ and (4.20) reduces to $g_j(p_j^{i-1}, p_j^i) = \left(p_j^i - p_j^{i-1}\right)\alpha_j$ on domain $p_j^{i-1} - p_j^i \leq 0$.

We assume that $\alpha_j + \beta_j \geq 0$, which implies that $g_j(p_j^{i-1}, p_j^i)$ is convex. This assumption ensures that the satisfaction of customer j cannot be improved by first increasing the adjustment and then decreasing it again.

In a similar way, we define linear penalties for early and late delivery, for both EXT and POS:

$$h_j^{EXT}(x_j, p_j^{j-1}) = \left(x_j - (b_j + p_j^{j-1})\right)^+ \gamma_j + (a_j - x_j)^+ \delta_j, \tag{4.21}$$

$$h_j^{POS}(x_j, p_j^{j-1}) = \left(x_j - (b_j + p_j^{j-1})\right)^+ \gamma_j + \left((a_j + p_j^{j-1}) - x_j\right)^+ \delta_j, \tag{4.22}$$

for some parameters $\gamma_j, \delta_j \in \mathbb{R} \cup \{\infty\}$. We assume that $\gamma_j, \delta_j \geq 0$, such that h_j is convex. That is, early and late delivery are both disliked by the customer.

We assume that the parameters α_j, β_j, γ_j, and δ_j are chosen such that the DTWAP is sufficiently expensive and that the optimal adjustments and voluntary waiting times can be attained. It is sufficient to make the additional assumption that $\alpha_j \geq 0$ and $\beta_j \geq 0$, or that $\mathcal{P}_j$ is bounded. For specific cases, weaker conditions may apply. In the EXT case, for example, it can be seen that $\beta_j < 0$ is allowed, even if $\mathcal{P}_j$ is unbounded.

It is straightforward to verify that the conditions of Proposition 4.2 are satisfied for every combination of EXT and POS with NW, AW and VW. By Proposition 4.2, it follows that the deterministic equivalent mathematical program can be solved as a convex program, given that a finite sample of scenarios is used. By Proposition 4.4, it follows that $F_{i+1}(x_i, p^i) = \mathbb{E}\left[c_{i+1}(x_i + t_{ii+1}, p^i)\right]$ is convex for all $i \in \{0, \ldots, n-1\}$.

Hence, the solution methods discussed in Section 4.3 can all be used to solve the simple DTWAP. Additionally, we note that g_j and h_j are polyhedral convex functions. It follows from Corollaries 4.3 and 4.5 that if the travel time distributions are discrete with finite support, then the deterministic equivalent mathematical program reduces to a linear program, and the functions $F_{i+1}(x_i, p^i)$ are polyhedral convex functions.

4.5.2 Properties of the simple DTWAP

In this section, we analyze the simple DTWAP. We first show that all solution methods discussed in Section 4.3 are more efficient for the simple DTWAP than in general. Afterwards, we consider every combination of EXT and POS with NW, AW and VW in more detail. We show which cases are decomposable, and thus allow for even more

efficient solution methods. Our analysis is also useful for Section 4.6, where we prove that the discretization approach (see Section 4.3.4) is exact for the simple DTWAP, under certain assumptions.

We make the following observation:

Observation 4.6. *The cost of informing customer j is not dependent on customer i or on the current time. Nor can we benefit from increasing and then decreasing the adjustment (Section 4.5.1). The longer we wait before informing the customer, the more stochastic variables have been realized. As such, it is optimal to determine the adjustment for customer j on arrival at customer $j-1$. In other words, there exists an optimal solution with $p_j^0 = p_j^1 = \ldots = p_j^{j-2} = 0$.*

Observation 4.6 implies that all value functions $c_i(t_i, p^{i-1})$ can be replaced by two-dimensional value functions $c_i(t_i, p_i^{i-1})$. Furthermore, for the simple DTWAP, we have that $g_j(0,0) = 0$ by definition. It follows that (4.8)-(4.10) simplify to

$$c_n(t_n, p_n^{n-1}) = \min_{w_n \in W_n} \left\{ h_n\left(x_n, p_n^{n-1}\right) \right\}, \qquad \forall t_n \geq 0,\ p_n^{n-1} \in \mathcal{P}_n, \tag{4.23}$$

$$c_i\left(t_i, p_i^{i-1}\right) = \min_{\substack{p_{i+1}^i \in \mathcal{P}_{i+1} \\ w_i \in W_i}} \left\{ g_{i+1}(0, p_{i+1}^i) + h_i\left(x_i, p_i^{i-1}\right) + \mathbb{E}\left[c_{i+1}\left(x_i + t_{ii+1}, p_{i+1}^i\right) \right] \right\},$$

$$\forall i \in V' \backslash \{n\},\ t_i \geq 0,\ p_i^{i-1} \in \mathcal{P}_i, \tag{4.24}$$

$$c_0\left(t_0\right) = \min_{p_1^0 \in \mathcal{P}_1} \left\{ g_1(0, p_1^0) + \mathbb{E}\left[c_1\left(x_0 + t_{ii+1}, p_1^0\right) \right] \right\}, \qquad \forall t_0 \geq 0. \tag{4.25}$$

with $x_i = d_i(t_i, w_i, p_i^{i-1})$.

Next, we define an equivalent stochastic dynamic program with one-dimensional value functions. Without loss of generality, we may reverse the order of determining the adjustment p_{i+1}^i and serving customer i. This follows from the fact that customer $i+1$ is indifferent about the timing of the information.

Let $\bar{c}_i(x_i)$ be the expected total dissatisfaction incurred after serving customer i at time x_i, but before determining the adjustment p_{i+1}^i. That is,

$$\bar{c}_n(x_n) = 0, \qquad \forall x_n \geq 0, \tag{4.26}$$

$$\bar{c}_i(x_i) = \min_{p_{i+1}^i \in \mathcal{P}_{i+1}} \left\{ g_{i+1}(0, p_{i+1}^i) + \mathbb{E}\left[c_{i+1}\left(x_i + t_{ii+1}, p_{i+1}^i\right) \right] \right\},$$

$$\forall i \in \{0, \ldots, n-1\}, x_i \geq 0. \tag{4.27}$$

From the definition, it is straightforward to show that $\bar{c}_i$ is convex. Furthermore, if the travel time distributions are discrete with finite support, then $\bar{c}_i$ is a polyhedral

convex function.

The stochastic dynamic program (4.23)-(4.25) can be rewritten as follows.

$$\bar{c}_n(x_n) = 0, \qquad \forall x_n \geq 0, \tag{4.28}$$

$$\bar{c}_i(x_i) = \min_{p^i_{i+1} \in \mathcal{P}_{i+1}} \left\{ g_{i+1}(0, p^i_{i+1}) + \mathbb{E}\left[\min_{w_{i+1} \in W_{i+1}} \left\{ \bar{c}_{i+1}\left(x_{i+1}\right) + h_{i+1}(x_{i+1}, p^i_{i+1}) \right\} \right] \right\},$$
$$\forall i \in \{0, \ldots, n-1\},\ x_i \geq 0,\ p_i^{i-1} \in \mathcal{P}_i, \tag{4.29}$$

with $x_{i+1} = d_{i+1}(t_{i+1}, w_{i+1}, p^i_{i+1})$ and $t_{i+1} = x_i + t_{ii+1}$ in the right-hand side of (4.29).

The structural properties of the simple DTWAP yield computational advantages for all solution methods discussed in Section 4.3. First, if we construct a deterministic equivalent mathematical program for (4.28)-(4.29), we require significantly less variables than in Section 4.3.2. This is due to Observation 4.6.

For the dual decomposition methods (Section 4.3.3), the expected future value function F_{i+1} is defined as $F_{i+1}(x_i, p^i) = \mathbb{E}\left[c_{i+1}(x_i + t_{ii+1}, p^i_{i+1})\right]$. Note that F_{i+1} only depends on x_i and on p^i_{i+1}, and can thus be seen as a two-dimensional function. Due to the lower dimensionality, one may expect it to be easier to approximate F_{i+1} for the simple DTWAP than in the general case.

Finally, for the discretization approach (Section 4.3.4), the number of possible states decreases significantly. Furthermore, the minimization problems that are solved by enumeration have a significantly smaller feasible set. As a result, the discretization approach is less computationally expensive for the simple DTWAP.

4.5.2.1 Never Wait

For waiting behavior NW, the set of possible voluntary waiting times is given by $W_i = \{0\}$, and the departure time is given by $d_{i+1}(t_{i+1}, w_{i+1}, p^i_{i+1}) = t_{i+1} = x_i + t_{ii+1}$. It follows that Equation (4.29) simplifies to

$$\begin{aligned}\bar{c}_i\left(x_i\right) &= \min_{p^i_{i+1} \in \mathcal{P}_{i+1}} \left\{ g_{i+1}(0, p^i_{i+1}) + \mathbb{E}\left[\bar{c}_{i+1}\left(x_i + t_{ii+1}\right) + h_{i+1}(x_i + t_{ii+1}, p^i_{i+1}) \right] \right\} \\ &= \min_{p^i_{i+1} \in \mathcal{P}_{i+1}} \left\{ g_{i+1}(0, p^i_{i+1}) + \mathbb{E}\left[h_{i+1}(x_i + t_{ii+1}, p^i_{i+1}) \right] \right\} + \mathbb{E}\left[\bar{c}_{i+1}\left(x_i + t_{ii+1}\right) \right].\end{aligned} \tag{4.30}$$

Equation (4.30) reveals that the simple DTWAP is decomposable in the NW case. This follows from the fact that the minimization problem in Equation (4.30) does not

depend on $\bar{c}_{i+1}$, the value function of the next customer. The optimal adjustments can then be determined independently per customer, and there is no need to calculate the value functions to determine the optimal actions. This is true for both EXT-NW and POS-NW.

Decomposing the problem makes it easier to solve. If the expectation of h_{i+1} can be evaluated efficiently, then the optimal action can be found efficiently for any given state. This follows from the convexity of g_{i+1} and h_{i+1}. In practice, this means that we do not require extensive calculations up front to determine the optimal actions. Instead, we can solve an easy problem for each state that we encounter.

The independence of the customers is a result of the following fact: if vehicles never wait, adjusting the time window of customer i changes the dissatisfaction of customer i, but has no affect on the arrival times at the other customers. A similar observation is made by Taş et al. (2014), who use this fact to specify the arrival time distributions at the customers.

4.5.2.2 Always Wait

For waiting behavior AW, we also have that the domain of the voluntary waiting time is given by $W_i = \{0\}$. We first consider EXT-AW specifically. In this case, we have

$$d_{i+1}(t_{i+1}, w_{i+1}, p^i_{i+1}) = \max\{t_{i+1}, a_{i+1}\} = \max\{x_i + t_{ii+1}, a_{i+1}\}, \tag{4.31}$$

such that (4.29) reduces to

$$\begin{aligned}\bar{c}_i\left(x_i\right) = \min_{p^i_{i+1} \in \mathcal{P}_{i+1}} &\left\{g_{i+1}(0, p^i_{i+1}) + \mathbb{E}\left[h_{i+1}(\max\{x_i + t_{ii+1}, a_{i+1}\}, p^i_{i+1})\right]\right\} + \\ &\mathbb{E}\left[\bar{c}_{i+1}\left(\max\{x_i + t_{ii+1}, a_{i+1}\}\right)\right].\end{aligned} \tag{4.32}$$

Equation (4.32) shows that the EXT-AW case decomposes into n independent problems, just like the EXT-NW case and the POS-NW case. This is a result of the fact that the vehicle always waits until the start of the time window. When the time window is extended, the start of the time window does not change. As a result, the arrival time distributions are not affected by the adjustments. It follows that the adjustments can be determined independently per customer.

Next we consider POS-AW. In this case, we have

$$d_{i+1}(t_{i+1}, w_{i+1}, p^i_{i+1}) = \max\{t_{i+1}, a_{i+1}+p^i_{i+1}\} = \max\{x_i+t_{ii+1}, a_{i+1}+p^i_{i+1}\}, \tag{4.33}$$

such that (4.29) simplifies to

$$\bar{c}_i(x_i) = \min_{p^i_{i+1} \in \mathcal{P}_{i+1}} \{ g_{i+1}(0, p^i_{i+1}) + \mathbb{E}\left[h_{i+1}(\max\{x_i + t_{ii+1}, a_{i+1} + p^i_{i+1}\}, p^i_{i+1})\right] + \mathbb{E}\left[\bar{c}_{i+1}\left(\max\{x_i + t_{ii+1}, a_{i+1} + p^i_{i+1}\}\right)\right]\}. \tag{4.34}$$

Note that we are not be able to take the $\mathbb{E}[\bar{c}_{i+1}]$ term out of the minimum, due to its dependency on p^i_{i+1}. Hence, the POS-AW case cannot be decomposed in the same way as EXT-NW, POS-NW, and EXT-AW.

4.5.2.3 Voluntarily Wait

In the VW case, the voluntary waiting time $w_i \in W_i = \mathbb{R}_{\geq 0}$ is a decision variable. The departure time $d_{i+1}(t_{i+1}, w_{i+1}, p^i_{i+1}) = t_{i+1} + w_{i+1}$ is the sum of the arrival time and the voluntary waiting time.

For both EXT-VW and POS-VW, we have that the simple DTWAP cannot be decomposed in the same way as EXT-NW, POS-NW, and EXT-AW. Voluntarily waiting at customer i affects the arrival time at customer $i+1$. As such, we are unable to determine the optimal waiting time independently per customer. Our observations on the VW cases are detailed in Appendix 4.B.

4.5.3 Time-dependent linear penalties

For the simple DTWAP, we have made the important assumption that the time at which an adjustment is communicated to the customer does not affect the amount of dissatisfaction received. That is, we have replaced $g_j(t_i, p^{i-1}_j, p^i_j)$ by $g_j(p^{i-1}_j, p^i_j)$. In this section, we prove that dropping this assumption may lead to a severely more complicated problem. We show this by introducing time-dependent linear penalties into the simple DTWAP.

Recall that

$$g_j(p^{i-1}_j, p^i_j) = \left(p^i_j - p^{i-1}_j\right)^+ \alpha_j + \left(p^{i-1}_j - p^i_j\right)^+ \beta_j, \tag{4.20}$$

for some parameters $\alpha_j, \beta_j \in \mathbb{R} \cup \{\infty\}$, $\alpha_j + \beta_j \geq 0$. If any of the parameters is chosen to be ∞, we replace the corresponding term by a domain restriction. Note that both increasing and decreasing the adjustment is penalized linearly.

Next, we consider time-dependent linear penalties of the form

$$g_j(t_i, p^{i-1}_j, p^i_j) = \left(p^i_j - p^{i-1}_j\right)^+ \alpha_j f_{1j}(t_i) + \left(p^{i-1}_j - p^i_j\right)^+ \beta_j f_{2j}(t_i), \tag{4.35}$$

for some weighting functions $f_{1j}(t_i)$ and $f_{2j}(t_i)$. For given t_i, changes are linearly penalized, and the slopes are time-dependent.

Proposition 4.7. *Let $f_{1j}(t_i)$ and $f_{2j}(t_i)$ be twice-differentiable real functions that are defined on the same real interval. If $g_j(t_i, p_j^{i-1}, p_j^i)$ as in (4.35) is convex, then $\alpha_j = 0$ or $\alpha_j = \infty$ or $f_{1j}(t_i)$ is constant. Furthermore, $\beta_j = 0$ or $\beta_j = \infty$ or $f_{2j}(t_i)$ is constant.*

Proof. Let $\bar{g}_j(t_i, y)$ be the restriction of $g_j(t_i, p_j^{i-1}, p_j^i)$ to $y = p_j^i - p_j^{i-1}$. That is, $\bar{g}_j(t_i, y) = y^+\alpha_j f_{1j}(t_i) + (-y)^+\beta_j f_{2j}(t_i)$.

If $\alpha_j < \infty$, then $\bar{g}_j$ is twice differentiable for all t_i in the domain of $f_{1j}(t_i)$ and $f_{2j}(t_i)$ and all $y > 0$. For any such point, we have

$$\nabla^2 \bar{g}_j(t_i, y) = \begin{bmatrix} y\alpha_j f''_{1j}(t_i) & \alpha_j f'_{1j}(t_i) \\ \alpha_j f'_{1j}(t_i) & 0 \end{bmatrix}.$$

If $\alpha_j \neq 0$ then $\nabla^2 \bar{g}_j(t_i, y)$ can only be positive semi-definite if $f'_{1j}(t_i) = 0$. This implies that $\bar{g}_j(t_i, y)$ cannot be convex, unless $f_{1j}(t_i)$ is a constant. As $\bar{g}_j$ is a restriction of g_j to a linear subspace, it follows that $g_j(t_i, p_j^{i-1}, p_j^i)$ cannot be convex either. If $\beta_j < \infty$ and $\beta_j \neq 0$ we can similarly show that $\bar{g}_j(t_i, y)$ cannot be convex, unless $f_{2j}(t_i)$ is a constant. □

Proposition 4.7 demonstrates that using time-dependent linear penalties results in $g_j(t_i, p_j^{i-1}, p_j^i)$ not being convex, except in pathological cases. As a result, the conditions of Proposition 4.2 and Proposition 4.4 are no longer satisfied. It follows that it is not clear if the deterministic equivalent mathematical program can be solved efficiently for even a single scenario. Furthermore, if the F_{i+1} functions are not convex, dual decomposition methods can no longer be used. We conclude that introducing time-dependent linear penalties considerably complicates the simple DTWAP, although the discretization approach remains an option.

4.6 Effect of discretization

In this section, we analyze the effect of discretizing the possible adjustments and the voluntary waiting times. Discretization was proposed in Section 4.3.4 to allow for solving the stochastic dynamic programming recursions directly. First, we show the negative result that discretization may lead to suboptimal solutions in general, even for instances with integer parameters and discrete travel time distributions with finite

support. On the other hand, we prove in the same setting that the discretization approach is exact for the simple DTWAP introduced in Section 4.5.

Specifically, we consider the following setting. We assume that the vehicle starting time and all initial time windows are integer. Furthermore, we assume that the travel time distributions are discrete with finite support, and all possible realizations are integer. If the sets of possible adjustments are bounded, we assume that they are bounded by integers. That is, $\min_{p\in\mathcal{P}_i}\{p\} \in \mathbb{Z}$ and $\max_{p\in\mathcal{P}_i}\{p\} \in \mathbb{Z}$ for all $i \in V'$, whenever the minimum and maximum are defined.

We do not force the actions to be integral: both the adjustments and the voluntary waiting times are assumed to be real. Given these assumptions, we are interested in whether there exist optimal time window adjustments and voluntary waiting times that are integer.

4.6.1 Discretization for general instances

We now construct a DTWAP instance that satisfies the assumptions above, but does not admit optimal actions that are integer. Consider an $n = 1$ customer instance with dissatisfaction functions $g_1(t_0, 0, p_1^0) = (p_1^0)^2$ and $h_1(x_1, p_1^0) = (1 - p_1^0)^2$. Let the vehicle depart the depot at time $x_0 = t_0 = 0$, and let the initial time window of customer 1 be given by $[a_i, b_i] = [0, 0]$. The travel time to customer 1 is deterministic and equal to 0. The adjustment type (EXT or POS) and the waiting behavior (NW, AW, or VW) can be chosen arbitrarily. The set of possible postponements is given by $\mathcal{P}_1 = [0, 1]$.

For the described instance, the DTWAP reduces to the following problem:

$$\min_{p_1^0\in[0,1]} \left\{(p_1^0)^2 + (1 - p_1^0)^2\right\}. \tag{4.36}$$

It can be seen that the only possible integer actions, $p_1^0 = 0$ and $p_1^0 = 1$, result in strictly higher dissatisfaction than the action $p_1^0 = \frac{1}{2}$. Hence, this instance proves that discretization may lead to suboptimal solutions in general, even under the current assumptions.

4.6.2 Discretization for the simple DTWAP

Under the same assumptions, we now prove that the simple DTWAP admits optimal actions that are integer, regardless of the adjustment type (EXT or POS) and the waiting behavior (NW, AW, or VW). It follows that discretizing the actions into

integers still yields an optimal solution.

In Proposition 4.8, we prove our claim for the cases EXT-NW, POS-NW, and EXT-AW. As shown in Section 4.5.2, the simple DTWAP is decomposable in these cases. This significantly simplifies our proof, compared to the cases POS-AW, EXT-VW, and POS-VW.

Proposition 4.8. *The simple DTWAP with discretized time allows for optimal integer decisions in the cases of EXT-NW, POS-NW, and EXT-AW.*

Proof. See Appendix 4.C. □

Next, we consider the non-decomposable cases POS-AW, EXT-VW, and POS-VW. In these cases, the optimal action p_{i+1}^i additionally depends on $\bar{c}_{i+1}$. In Appendix 4.D, we prove several lemmas before we prove Proposition 4.9.

Proposition 4.9. *The simple DTWAP with discretized time allows for optimal integer decisions in the cases of POS-AW, EXT-VW, and POS-VW.*

Proof. See Appendix 4.D. □

Propositions 4.8 and 4.9 together prove that for the simple DTWAP, under the assumptions stated at the beginning of this section, the optimal actions may be assumed to be integer. Hence, in these cases, the discretization approach as introduced in Section 4.3.4 provides a straightforward and exact algorithm to solve the simple DTWAP.

4.7 Illustrative example

In this section, we provide an illustrative example of how the results of this chapter can be used to improve customer satisfaction in a practical application. In Section 4.7.1, we present an attended home delivery problem with adjustment type POS and waiting behavior AW. To model dissatisfaction, we use the customer dissatisfaction functions of the simple DTWAP (see Section 4.5) as a basis, and we modify them to better fit this specific setting.

In Section 4.7.2, we make use of Sections 4.3 and 4.4 to construct a solution method. Our computational results are presented in Section 4.7.3. Finally, we present three heuristics in Section 4.7.4, which use ideas from Section 4.5 and Section 4.6.

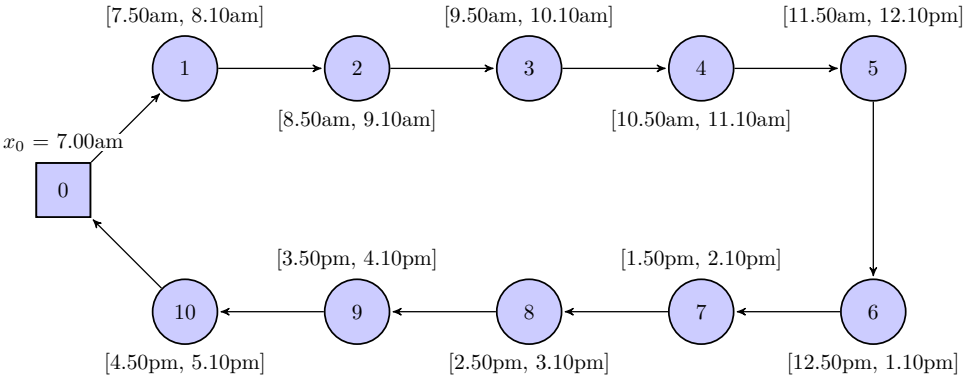


Figure 4.1: Example instance.

4.7.1 DTWAP instance

For our example, we consider an *attended home delivery* setting. That is, the customers have to be present at the time of delivery. Agatz et al. (2008) list different reasons that necessitate attended home delivery. For example, attended home delivery is necessary for goods that are perishable or physically large. The customers also need to be present if a service is performed, e.g., repairs.

We assume that the vehicle leaves the depot at 7.00 am and performs a route with ten customers (see Figure 4.1). That is, $t_0 = x_0 = 420$ and $n = 10$. Note that time is measured in minutes since midnight, i.e., 420 corresponds to 7.00 am.

The travel times t_{ii+1} for all $i \in V' \cup \{0\}$ are assumed to be independent and discrete uniformly distributed between 50 and 70 minutes, and the service times are assumed to be negligible. The initial time windows of the customers are 20 minutes in width, and are centered around the time of arrival when all travel times are equal to their expected values, and no waiting is necessary. For example, the initial time window of customer two is given by [8.50 am, 9.10 am], which is centered around 9.00 am. For each customer $j \in V'$, this is modeled as $a_j = 420 + 60j - 10$ and $b_j = 420 + 60j + 10$.

In our attended home delivery setting, the customers are present from the start of the current time window onwards, but are not available earlier. In the case of late delivery, customers stay at home until the delivery is made. Because the customers are not available earlier, we use waiting behavior AW to always wait until the time window opens.

We allow the time windows to be postponed by 0, 5, 10, 15, 20, 25 or 30 minutes,

i.e., we use adjustment type POS and $\mathcal{P}_j = \{0, 5, 10, 15, 20, 25, 30\}$ for all $j \in V'$. By postponing the time windows in steps of five minutes, we obtain time windows that are easy to remember for the customers. To prevent that customers have to rush back home due to time window adjustments, we only allow time windows to be moved forward in time, which will be incorporated in our customer dissatisfaction functions.

Our customers dislike adjustments, but are not influenced by the time of communication until three hours before the current deadline. For example, consider customer 5. If the current time window is given by [11.50 am, 12.10 pm], then customer 5 does not care at what time a postponement is communicated, as long as it is before 9.10 am. After 9.10 am, it becomes increasingly more difficult for customer 5 to deal with time window adjustments. As such, the dissatisfaction is larger when the postponement is communicated later. After the current deadline, the delivery is considered to be late and the time window can no longer be adjusted.

4.7.1.1 Customer dissatisfaction functions

We use the customer dissatisfaction functions of the simple DTWAP as a basis, and we modify them to fit our setting. The simple DTWAP has the advantage that we can perform the analysis in Section 4.5.2. However, without modification, the associated customer dissatisfaction functions are not appropriate for our practical application.

First, we consider the functions $g_j(t_i, p_j^{i-1}, p_j^i)$, which model the dissatisfaction of customer j due to changing the postponement from p_j^{i-1} to p_j^i at time t_i. The function g_j that is used in the simple DTWAP, Equation (4.20), is not appropriate for our example. First, Equation (4.20) assumes that customers are indifferent about the timing of information, which is not the case here. Second, Equation (4.20) allows for adjusting the time window of customer j when t_i is past the current deadline of customer j, which we do not allow in our example.

Based on Section 4.5.3, we propose the following dissatisfaction function:

$$g_j\left(t_i, p_j^{i-1}, p_j^i\right) = \left(p_j^i - p_j^{i-1}\right)^+ \alpha_j f_j(t_i), \tag{4.37}$$

on domain $p_j^{i-1} \leq p_j^i$ and $t_i \leq b_j + p_j^{i-1}$, with

$$f_j(t_i) = 1 + \nu_j \left(t_i - \left(b_j + p_j^{i-1} - L_j\right)\right)^+, \tag{4.38}$$

and non-negative parameters α_j, ν_j, and L_j.

Equation (4.37) provides a time-dependent linear penalty for changing the ad-

justment, as introduced in Section 4.5.3. The domain restrictions correspond to our assumptions: the condition $p_j^{i-1} \leq p_j^i$ ensures that the adjustments can only be increased, and $t_i \leq b_j + p_j^{i-1}$ prevents adjustments to be made after the deadline has passed. For convenience, we consider g_j to evaluate to ∞ outside of its domain.

The function $f_j(t_i)$ models how the dissatisfaction due to time window adjustments is affected by time. If the time at which the adjustment is communicated is at least L time units before the current deadline $b_j + p_j^{i-1}$, then $f_j(t_i) = 1$, and the time of communication does not affect the dissatisfaction. In our example, $L_j = 180$ for every customer, which corresponds to three hours. After time $b_j + p_j^{i-1} - L_j$, the parameter ν_j indicates how strongly customer j is affected by the timing of information.

Next, we consider the functions $h_j(x_j, p_j^{j-1})$, which model the dissatisfaction due to missing the deadline. We modify the function h_j used for the simple DTWAP, Equation (4.22), by including a fixed dissatisfaction κ_j for missing the deadline. We obtain the following function:

$$h_j(x_j, p_j^{j-1}) = \left(x_j - (b_j + p_j^{j-1})\right)^+ \gamma_j + I(x_j > b_j + p_j^{j-1})\kappa_j, \tag{4.39}$$

with $I(.)$ the indicator function. Note that we do not define a penalty for early delivery, because the vehicle always waits until the time window opens.

For our computational experiments, we assume that the parameters for customer $j \in V'$ are given by $\alpha_j = 0.1$, $\nu_j = 0.1$, $L_j = 180$, $\gamma_j = 1$, and $\kappa_j = 100$. Note that these values are chosen for demonstrative purposes, and may differ in practical applications.

Finally, we verify that our DTWAP instance satisfies the two general assumptions stated in Section 4.3.1. The dissatisfaction functions g_j and h_j are all non-negative, which implies that the DTWAP instance is sufficiently expensive. By definition, only finite postponements are allowed, which implies that the optimal actions are attainable. It follows that the assumptions are satisfied.

4.7.2 Solution method

In this section, we describe a solution method to solve the DTWAP constructed in Section 4.7.1. Recall that Section 4.3 presents three potential solution methods: solving the deterministic equivalent mathematical program, using dual decomposition methods, and solving the discretized stochastic dynamic program. In Section 4.4, we discuss the properties of the DTWAP, and we state conditions such that the methods

in Section 4.3 can be applied.

For the current example, we have that the g_j functions are not convex. This follows from the use of time-dependent linear penalties, as shown in Proposition 4.7. As a result, Proposition 4.2 and Proposition 4.4 in Section 4.4 are not applicable.

It follows that it is not clear if the deterministic equivalent mathematical program can be solved efficiently for even a single scenario. Furthermore, it is unclear if dual decomposition methods can be applied. For these reasons, we apply the third method: solving the discretized stochastic dynamic program. As discussed in Section 4.3, this method does not require convexity assumptions.

Section 4.3.4 details how the DTWAP instance can be modified such that the stochastic dynamic program (4.8)-(4.10) can be solved directly. In our case, we discretize time in minutes, and we observe that all relevant parameters already have integer values. The same holds true for the travel time distributions, which have finite support and all realizations are integer. It follows that the stochastic dynamic program (4.8)-(4.10) can be solved directly by backward or forward recursion, as explained in Section 4.3.4.

In the attended home delivery setting, solving the stochastic dynamic program can be accelerated by making an observation similar to Observation 4.6, which is used in Section 4.5 to speed up the solution methods for the simple DTWAP. This observation states that if the customer dissatisfaction is independent of the time of communication, then it is optimal to delay adjustment decisions.

In our setting, customer j is indifferent about the timing of the information, given that it is at least three hours before the current deadline. Hence, if the next opportunity to change the adjustments is, *with certainty*, at least three hours before the deadline, then we can delay the adjustment decision. Mathematically speaking, if the arrival time at customer i is t_i, and the probability that $t_{i+1} \leq b_j - L_j$ is one, then $p_j^i = 0$ is optimal.

4.7.3 Computational results

We solve the stochastic dynamic program (4.8)-(4.10) by forward recursion, as described in Section 4.3.4, including the acceleration strategy discussed in Section 4.7.2. The algorithm is coded in C++, and our computational results are obtained with an Intel Core i7-8550U processor.

In Table 4.1, we compare the results of the DTWAP with two benchmarks: *no adjustment* and *a priori*. For the no adjustment benchmark, we do not make any time window adjustments. That is, the action at customer i is given by $p^i = 0$ for

every state. For the a priori benchmark, we only allow adjusting the time windows before leaving the depot. That is, all adjustments are made a priori, and the time windows cannot be updated dynamically.

Recall that the set of possible adjustments is given by $\mathcal{P}_j = \{0, 5, 10, 15, 20, 25, 30\}$ for all $j \in V'$. To obtain a fair comparison with the DTWAP, we use the same possible adjustments for the a priori benchmark.

The a priori benchmark is calculated by enumerating all possible adjustment vectors. For each adjustment vector, we obtain a DTWAP instance by fixing the adjustment options accordingly. For example, if $p^0 = (5, 10, \ldots, 10)$, then we set $\mathcal{P}_1 = \{5\}$, $\mathcal{P}_2 = \{10\}, \ldots, \mathcal{P}_{10} = \{10\}$. We then use our DTWAP algorithm to obtain the expected total dissatisfaction for the given adjustment vector. By enumeration, the best a priori adjustment vector is found.

The reported statistics are the expected values over all $21^{10} \approx 17$ trillion scenarios. To calculate these statistics efficiently, we first decide on the action for every state. We then obtain the statistics recursively by solving stochastic dynamic programs that are similar to (4.8)-(4.10), but with the minimization over p^i replaced by the action chosen earlier.

Before discussing the results in Table 4.1, we want to emphasize that we only consider a single DTWAP instance. To make general statements, a more extensive computational study is required. Instead, we use our illustrative example to demonstrate the potential benefit of dynamic time window adjustment in a practical application.

If no adjustments are made, in expectation 21.5% of the customers is served after the communicated deadline. Even though the expected average lateness per customer is only 2.0 minutes, the dissatisfaction is high because the customers really dislike late deliveries, even if they are just a couple of minutes late. This characteristic corresponds to the parameter $\kappa_j = 100$ in the h_j function (Equation (4.39)).

If we compare the no adjustment benchmark to the DTWAP, we see that using dynamic time window adjustments can significantly reduce the expected total dissatisfaction from 235.3 to 27.9. In expectation, for the average customer, the time window is adjusted 1.2 times, and the final average postponement is 14 minutes. The worst-off customer, which turns out to be customer 10, can expect the time window to be adjusted 1.8 times, and the final postponement to be 26 minutes. The expected percentage of deadlines missed decreases from 21.5% to only 0.6%, and the expected average lateness per customer decreases from 2 minutes to 2 seconds.

Next, we compare the a priori benchmark to the DTWAP. For the a priori bench-

	No adjustment	A priori	DTWAP
Total expected dissatisfaction	235.3	37.6	27.9
Expected dissatisfaction due to missed deadlines (% of total)	100.0%	33.6%	23.7%
Expected dissatisfaction due to adjustments (% of total)	0.0%	66.4%	76.3%
Expected percentage of deadlines missed	21.5%	1.2%	0.6%
Expected average lateness per customer	120 seconds	3 seconds	2 seconds
Expected lateness for the worst-off customer	233 seconds	12 seconds	8 seconds
Expected average postponement time per customer	0 minutes	20 minutes	14 minutes
Expected postponement for the worst-off customer	0 minutes	30 minutes	26 minutes
Expected average number of postponements per customer	0	0.9	1.2
Expected number of postponements for the worst-off customer	0	1	1.8
Solution time	0 seconds	9962 seconds	86 seconds

Table 4.1: Comparison between no adjustment, a priori adjustment, and dynamic time window adjustment.

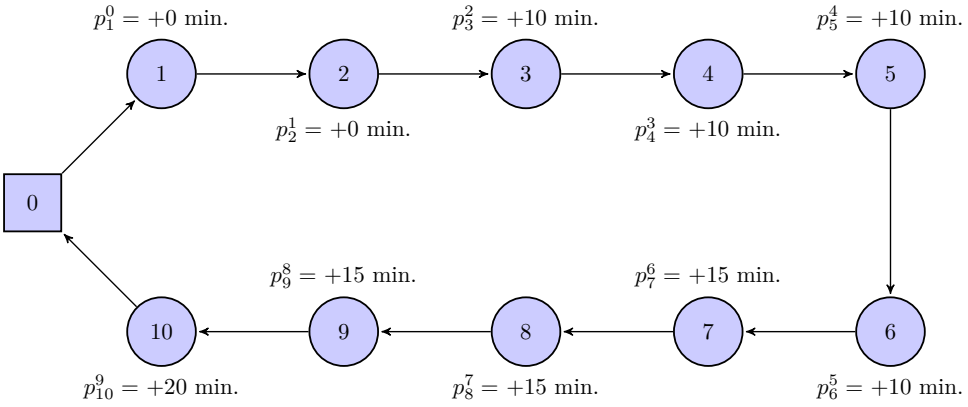


Figure 4.2: Optimal adjustments for the scenario with expected travel times.

mark, the expected probability of missing a deadline is 1.2%, which is twice the probability of 0.6% for the DTWAP. The average postponement time per customer is also higher for the a priori benchmark, with 20 minutes versus 14 minutes. We conclude that dynamic time window adjustments may simultaneously reduce the probability of missing a deadline, and reduce the average time window postponement.

There is a large difference in solution time between calculating the a priori benchmark and solving the DTWAP: 9962 seconds versus 86 seconds. This difference is due to the large number of adjustment vectors that are enumerated in the a priori case. Ten customers with seven adjustment options each results in $7^{10} \approx 282$ million possible vectors. For the DTWAP, this number is significantly smaller, because many decisions can be delayed (see Section 4.7.2). For example, if we can delay all decisions, except for the postponements of the next four customers, then only $7^4 = 2401$ adjustment vectors need to be considered at the current state.

To gain more insight into the optimal DTWAP solution, Figure 4.2 presents the final postponements for the scenario in which all travel times are equal to their expected value. That is, all travel times turn out to be 60 minutes. If we would have known in advance that this scenario would occur, no time window adjustment would have been necessary. Without foresight, we use time windows adjustments to anticipate the uncertainty in the travel times.

In Figure 4.2, the postponements are larger for later customers, even though all travel times are equal to their expected value. This is because the vehicle always waits until the time window opens, and because time windows can only be postponed further. As a result, postponements that are used to anticipate randomness cause

delays themselves, prompting larger time window adjustments for the later customers. The attended home delivery example shows that, even if such a snowball effect occurs, time window adjustment can be beneficial.

4.7.4 Heuristic solution methods

Based on the analysis in this chapter, we can construct various heuristic solution methods. We present three such heuristics, and we apply them to our attended home delivery example. The heuristics are compared on solution time and on expected total dissatisfaction.

In Section 4.5, we have analyzed the simple DTWAP. For this problem, by Observation 4.6, it is optimal to only decide on the adjustment for the next customer. Based on this fact, we have explained how to speed up the solution methods. We can use the same idea as a heuristic: only consider the adjustments for the next given number of customers. We implement this heuristic by making straightforward changes to our DTWAP algorithm.

In Section 4.6, we have considered the effect of discretization. For the simple DTWAP, we have shown that continuous adjustments can be replaced by integer adjustments without loss of solution quality. As a heuristic, we can use a more coarse discretization for the possible adjustments. For our example, we will consider replacing $\mathcal{P}_j = \{0, 5, 10, 15, 20, 25, 30\}$ by $\mathcal{P}_j = \{0, 10, 20, 30\}$.

For our third heuristic, we solve the stochastic dynamic program (4.8)-(4.10) heuristically by assuming that all travel times are deterministic and equal to their expected value. That is, given the current state, we solve a DTWAP in which each travel time distribution is replaced by a degenerate distribution that is always equal to the expected travel time. Note that the difference in dissatisfaction between applying this heuristic and solving the DTWAP can be considered as the *value of stochastic information*.

In Table 4.2, we compare the two benchmarks, the DTWAP, and the three heuristics. For the first heuristic, we present two variants: only consider the next customer, and consider the next three customers.

The first heuristic is very fast, but does not produce a good solution. Only considering the next customer results in communicating each time window adjustment at the latest possible time. Because customers appreciate being informed timely, the resulting solution has a relatively high expected total dissatisfaction.

This is different for considering the next three customers. In our example, customers do not care about the exact time at which time window adjustments are

	Seconds	Dissatisfaction
No adjustment	0.0	235.3
A priori	9962.4	37.6
DTWAP	86.3	27.9
Only consider the next customer	0.0	81.7
Only consider the next three customers	5.1	31.2
Coarse discretization	1.7	32.0
No stochastic information	0.0	134.6

Table 4.2: Comparison of heuristics on solution time and total expected dissatisfaction.

communicated, given that they are communicated at least three hours before the current deadline. Note that the expected travel time is 60 minutes, such that three hours roughly corresponds to visiting the next three customers. We see that this heuristic is able to find a solution that is relatively close to the optimal solution, in a fraction of the time that we need to solve the DTWAP.

For our example, the coarse discretization heuristic is the second best heuristic in terms of solution quality. By limiting the number of adjustment options, the number of possible adjustment vectors decreases exponentially. This explains the relatively fast solution time. We also want to point out that the coarse discretization heuristic provides a better solution than the a priori benchmark. This demonstrates that, in our example, allowing adjustments to be dynamic is more important than providing more adjustment options.

Finally, we consider the heuristic in which we do not use any stochastic information. This heuristic is fast, but the solution quality is bad. Our attended home delivery example thus demonstrates that not accounting for random travel times can be harmful to the satisfaction of the customers.

4.8 Conclusion

In this chapter, we introduce dynamic time window adjustment to the literature. To the best of our knowledge, this topic has not yet attracted the attention of researchers. On the other hand, dynamic time window adjustments are highly relevant in practice, and are often used to improve customer satisfaction.

We provide a general model and we introduce the DTWAP to optimize the dynamic time window adjustments. We consider three different solution methods for

the DTWAP, and we discuss their advantages and disadvantages.

We also introduce the simple DTWAP, which we analyze in more detail. We explain why the simple DTWAP can be solved more efficiently than in general, and for three variants we show that the simple DTWAP decomposes into independent problems per customer. Furthermore, under certain assumptions, we prove that discretizing time still yields an optimal solution for the simple DTWAP.

In our illustrative example on attended home delivery, we demonstrate how the results of this chapter can be used to solve the DTWAP in a practical application. Based on our analysis, we choose the solve our problem by discretization and forward recursion. Ideas developed throughout the chapter are used to construct and test three different heuristics.

Our computational results show that dynamic time window adjustment has the potential to improve customer satisfaction. If we compare a priori (non-dynamic) time window adjustment to dynamic time window adjustment, we see for our example that using the latter results in missing less deadlines, while the time windows are adjusted by smaller amounts.

For future work, it can be interesting to use dynamic time window adjustments in specific settings, including parcel delivery, attended home delivery, and retailer distribution. This includes the construction of suitable customer dissatisfaction functions and metrics to measure the actual improvement. Another direction for further research is to search for special cases that allow the DTWAP to be solved relatively efficiently, as is the case for the simple DTWAP, and to develop specialized algorithms.

Finally, we may integrate the DTWAP with other problems. The initial time windows, for example, are now assumed to be fixed. Integrating the initial time window assignment with the DTWAP is a relevant practical problem that has not yet been considered.

Another possibility is combining the DTWAP with stochastic vehicle routing. Our computational experiments indicate that after making a priori decisions, dynamic time window adjustments can improve customer satisfaction further. As such, we see an interesting opportunity in improving customer satisfaction by combining dynamic time window adjustment and stochastic vehicle routing.

Appendix

4.A Proof of Proposition 4.1

Proposition 4.1. *The DTWAP is strongly* NP*-hard, even for deterministic travel times, and for dissatisfaction functions that can be evaluated in polynomial time.*

Proof. We first observe the following. If $n = 1$, $t_0 = 0$, $\mathcal{P}_1 = \mathbb{N}_{\geq 0}$, and $h_1(x_1, p_1^0) = 0$, then the DTWAP simplifies to

$$\min_{p_1^0 \in \mathbb{N}_{\geq 0}} g_1(0, 0, p_1^0). \tag{4.40}$$

That is, solving the DTWAP is at least as hard as minimizing a general one-dimensional function over $\mathbb{N}_{\geq 0}$. Note that this fact is independent of the adjustment type (EXT or POS) and the waiting behavior (NW, AW, or VW).

Next, we present a polynomial-time reduction from the MAX-CUT problem to Problem (4.40). For our purpose, it is sufficient to note the following two facts. First, MAX-CUT is defined on a graph with m vertices and is strongly NP-hard (Garey and Johnson, 1979). Second, MAX-CUT can be stated as $\min_{y \in \{0,1\}^m} f(y)$, for some quadratic function f (Boros and Hammer, 1991). The values of the coefficients of f are polynomial in the input values.

Now define the function $g_1(0, 0, p_1^0)$, as follows. Without loss of generality, we assume that p_1^0 is encoded as a binary number. If p_1^0 has more than m digits, then $g_1(0, 0, p_1^0) = \infty$. If p_1^0 has at most m digits, then pad the left with zeros to obtain a binary number with exactly m digits. Interpret this binary number as a binary vector y and return $f(y)$. Clearly, every binary vector of length m can be constructed in this way.

Hence, we have shown that MAX-CUT reduces to Problem (4.40), which is a special case of the DTWAP. It can be verified that this is a polynomial-time reduction. Furthermore, it can be seen that our reduction does not introduce large numerical parameters. It follows that the DTWAP is strongly NP-hard. □

4.B Properties of the simple DTWAP - Voluntarily Wait

In the VW case, the voluntary waiting time $w_i \in W_i = \mathbb{R}_{\geq 0}$ is a decision variable. The departure time $d_{i+1}(t_{i+1}, w_{i+1}, p_{i+1}^i) = t_{i+1} + w_{i+1}$ is the sum of the arrival time

and the voluntary waiting time.

Let $w_i^*(t_i, p_i^{i-1})$ be an optimal waiting time for when the vehicle arrives at customer i, at time t_i, with time window adjustment p_i^{i-1}. That is, we define

$$w_i^*(t_i, p_i^{i-1}) = \arg\min_{w_i \geq 0} \left\{ \bar{c}_i(t_i + w_i) + h_i(t_i + w_i, p_i^{i-1}) \right\}. \tag{4.41}$$

In Equation (4.29), we may then replace the minimization over w_{i+1} with an optimal value w_{i+1}^*. We obtain

$$\bar{c}_i(x_i) = \min_{p_{i+1}^i \in \mathcal{P}_{i+1}} \left\{ g_{i+1}(0, p_{i+1}^i) + \mathbb{E}\left[\bar{c}_{i+1}\left(x_{i+1}\right) + h_{i+1}(x_{i+1}, p_{i+1}^i) \right] \right\}, \tag{4.42}$$

with $x_{i+1} = t_{i+1} + w_{i+1}^*(t_{i+1}, p_{i+1}^i)$ and $t_{i+1} = x_i + t_{ii+1}$ in the right-hand side.

Proposition 4.10. *Let $X_i^* = \arg\min_{X_i \in \overline{\mathbb{R}}} \{\bar{c}_i(X_i) - X_i\delta_i\}$ be the preferred departure time from customer i. Then $w_i^*(t_i, p_i^{i-1})$ can be stated as follows.*

- *For EXT-VW, $w_i^*(t_i, p_i^{i-1}) = (\min\{X_i^*, a_i\} - t_i)^+$.*
- *For POS-VW, $w_i^*(t_i, p_i^{i-1}) = \left(\min\{X_i^*, a_i + p_i^{i-1}\} - t_i\right)^+$.*

Proof. Recall that the value function $\bar{c}_i$ is a convex function. Additionally, under VW, the value function is non-decreasing in the departure time x_i. This follows from the fact that the vehicle can always depart later by waiting.

The function $h_i(x_i, p_i^{i-1})$ is non-increasing in the departure time x_i until the start of the (adjusted) time window, zero within the time window, and non-decreasing after the end of the time window. As the value function $\bar{c}_i$ is non-decreasing in the departure time, it is optimal to not wait further than the beginning of the time window. That is, it is optimal to assume that $w_i^*(t_i, p_i^{i-1}) \leq (a_i - t_i)^+$ for EXT-VW and $w_i^*(t_i, p_i^{i-1}) \leq (a_i + p_i^{i-1} - t_i)^+$ for POS-VW.

In the case of EXT-VW, we show that

$$\begin{aligned} w_i^*(t_i, p_i^{i-1}) &= \arg\min_{w_i \geq 0} \left\{ \bar{c}_i(t_i + w_i) + h_i(t_i + w_i, p_i^{i-1}) \right\} && (4.43) \\ &= \arg\min_{0 \leq w_i \leq (a_i - t_i)^+} \left\{ \bar{c}_i(t_i + w_i) - w_i\delta_i \right\}. && (4.44) \end{aligned}$$

Note that (4.43) is simply the definition of $w_i^*(t_i, p_i^{i-1})$.

To go from (4.43) to (4.44), we first add the upper bound $w_i \leq (a_i - t_i)^+$ as discussed above. If $t_i \geq a_i$, then we arrive at customer i after the start of the time

window. It is then optimal not to wait, i.e., $w_i = 0$, which is forced by $0 \leq w_i \leq (a_i - t_i)^+$.

If $t_i < a_i$, then $w_i \leq (a_i - t_i)^+$ implies that $x_i = t_i + w_i \leq a_i$. That is, the vehicle departs from customer i before or at the time that the time window opens. By definition, the function h_i is non-increasing on this domain with a fixed slope of δ_i. Because we are only interested in the argument, and not in the value of the minimum, we may ignore the other variables and replace $h_i(t_i + w_i, p_i^{i-1})$ by $-w_i\delta_i$.

To solve the problem in (4.44), we first make a change of variables from w_i to $X_i = t_i + w_i$. Let X_i^* be the unconstrained minimum over the extended real number line, denoted by $\overline{\mathbb{R}} = \mathbb{R} \cup \{-\infty, \infty\}$. That is,

$$X_i^* = \arg\min_{X_i \in \overline{\mathbb{R}}} \{\bar{c}_i(X_i) - (X_i - t_i)\delta_i\} \tag{4.45}$$

$$= \arg\min_{X_i \in \overline{\mathbb{R}}} \{\bar{c}_i(X_i) - X_i\delta_i\}. \tag{4.46}$$

Note that we may remove $t_i\delta_i$ because this term is not dependent on X_i.

The value X_i^* is the preferred departure time from customer i. That is, if there exists a waiting time $0 \leq w_i \leq (a_i - t_i)^+$ such that the vehicle departs at X_i^*, then this is optimal. If no such waiting time w_i satisfies the constraints, it is optimal to select the closest feasible point. This is a property of one-dimensional convex minimization problems. It can be shown that this results in $w_i^*(t_i) = (\min\{X_i^*, a_i\} - t_i)^+$ for EXT-VW. For POS-VW, a similar argument can be given to demonstrate that $w_i^*(t_i, p_i^{i-1}) = \left(\min\{X_i^*, a_i + p_i^{i-1}\} - t_i\right)^+$. □

By Proposition 4.10, we may simplify (4.42) further. For EXT-VW, we obtain

$$\bar{c}_i(x_i) = \min_{p_{i+1}^i \in \mathcal{P}_{i+1}} \left\{g_{i+1}(0, p_{i+1}^i) + \mathbb{E}\left[h_{i+1}(x_{i+1}, p_{i+1}^i)\right]\right\} + \mathbb{E}\left[\bar{c}_{i+1}\left(x_{i+1}\right)\right], \tag{4.47}$$

with $x_{i+1} = t_{i+1} + \left(\min\{X_{i+1}^*, a_{i+1}\} - t_{i+1}\right)^+$ and $t_{i+1} = x_i + t_{ii+1}$ on the right-hand side.

Note that in this case, we *cannot* decompose the simple DTWAP in the same way as before. This is because the departure time x_{i+1} depends on the preferred departure time X_{i+1}^*, which depends on the following customers.

It is true, however, that the optimal adjustment vector p^i only depends on the future customers through X_{i+1}^*. It is possible that this fact can be exploited by solution methods, for example by estimating X_{i+1}^*, instead of using the value function $\bar{c}_{i+1}$.

For POS-VW, we are unable to simplify Equation (4.42) further. In this case, the departure time x_{i+1} depends on the optimal waiting time $w_i^*(t_i, p_i^{i-1})$, which depends on the adjustment p_i^{i-1}. As such, the $\mathbb{E}[\bar{c}_{i+1}]$ term cannot be taken out of the minimum.

4.C Proof of Proposition 4.8

Proposition 4.8. *The simple DTWAP with discretized time allows for optimal integer decisions in the cases of EXT-NW, POS-NW, and EXT-AW.*

Proof. First consider EXT-NW and POS-NW. In Section 4.5.2, we have shown that in the cases of EXT-NW and POS-NW, the function $\bar{c}_i$ simplifies to

$$\bar{c}_i\left(x_i\right) = \min_{p_{i+1}^i \in \mathcal{P}_{i+1}} \left\{ g_{i+1}(0, p_{i+1}^i) + \mathbb{E}\left[h_{i+1}(x_i + t_{ii+1}, p_{i+1}^i)\right] \right\} + \mathbb{E}\left[\bar{c}_{i+1}\left(x_i + t_{ii+1}\right)\right]. \tag{4.30}$$

For convenience, we define

$$f(p_{i+1}^i) = g_{i+1}(0, p_{i+1}^i) + \mathbb{E}\left[h_{i+1}(x_i + t_{ii+1}, p_{i+1}^i)\right], \tag{4.48}$$

such that an optimal action can be found by minimizing $f(p_{i+1}^i)$ over $p_{i+1}^i \in \mathcal{P}_{i+1}$.

Note that f is a piece-wise linear function: the functions g_{i+1} and h_{i+1} are polyhedral convex by definition, and by assumption, the expectation can be written as a finite sum. It follows that the sum is a polyhedral convex function, which is piece-wise linear by definition.

Now assume that $x_i \in \mathbb{Z}$. By piece-wise linearity, there exists an optimal solution p_{i+1}^i that is a breakpoint of f, or a boundary point of $\mathcal{P}_{i+1}$. If p_{i+1}^i is a breakpoint of f, then p_{i+1}^i must also be a breakpoint of one of its summands. It is easily verified that for $x_i \in \mathbb{Z}$ and integer travel times, the functions g_{i+1} and h_{i+1} only have breakpoints for integer p_{i+1}^i. This follows immediately from the definitions (4.20) and (4.21)-(4.22), respectively, and from the integrality of the parameters. If p_{i+1}^i is a boundary point of $\mathcal{P}_{i+1}$, then p_{i+1}^i is integer by assumption.

The departure time from the depot, x_0, is integer by assumption. By the argument above, there exists an optimal integer action p_1^0. By assumption, the time windows and the travel times are integer. It follows that the arrival and departure time at customer 1 are again integer, which leads to an optimal integer action p_2^1, etc. By induction, we have proven that the simple DTWAP allows for optimal integer

decisions in the cases of EXT-NW and POS-NW.

The proof for EXT-AW is similar to the proof for EXT-NW and POS-NW, but with a different function f. In this case, we use Equation (4.32) to obtain

$$f(p_{i+1}^i) = g_{i+1}(0, p_{i+1}^i) + \mathbb{E}\left[h_{i+1}(\max\{x_i + t_{ii+1}, a_{i+1}\}, p_{i+1}^i)\right]. \tag{4.49}$$

Note that the function f remains piece-wise linear in p_{i+1}^i. It is again easily verified that for $x_i \in \mathbb{Z}$, integer travel times, and integer time windows, the functions g_{i+1} and h_{i+1} only have breakpoints for integer p_{i+1}^i. The other parts of the proof are the same as in the NW case. □

4.D Proof of Proposition 4.9

Before proving Proposition 4.9, we prove Lemmas 4.11 to 4.16. For convenience, we define functions $f(p_{i+1}^i)$, similar as in the proof of Proposition 4.8. Using Equation (4.27), it is straightforward to show that all f are polyhedral convex functions. For POS-AW, we use Equation (4.34) to obtain

$$\begin{aligned} f(p_{i+1}^i) = g_{i+1}(0, p_{i+1}^i) + \mathbb{E}\left[h_{i+1}(\max\{x_i + t_{ii+1}, a_{i+1} + p_{i+1}^i\}, p_{i+1}^i)\right] + \\ \mathbb{E}\left[\bar{c}_{i+1}\left(\max\{x_i + t_{ii+1}, a_{i+1} + p_{i+1}^i\}\right)\right]. \end{aligned} \tag{4.50}$$

For EXT-VW and POS-VW, we use Equation (4.47) to obtain

$$f(p_{i+1}^i) = g_{i+1}(0, p_{i+1}^i) + \mathbb{E}\left[h_{i+1}(x_{i+1}, p_{i+1}^i)\right] + \mathbb{E}\left[\bar{c}_{i+1}\left(x_{i+1}\right)\right], \tag{4.51}$$

with in the right-hand side

- $x_{i+1} = t_{i+1} + \left(\min\{X_{i+1}^*, a_{i+1}\} - t_{i+1}\right)^+$ and $t_{i+1} = x_i + t_{ii+1}$ for EXT-VW,
- $x_{i+1} = t_{i+1} + \left(\min\{X_{i+1}^*, a_{i+1} + p_{i+1}^i\} - t_{i+1}\right)^+$ and $t_{i+1} = x_i + t_{ii+1}$ for POS-VW.

For brevity, we define Assumption 1, which we refer to in the lemmas.

Assumption 1. *Assume that the combination of adjustment type and waiting behavior is POS-AW, EXT-VW, or POS-VW. Let $i \in \{0, \ldots, n-1\}$ and assume that $\bar{c}_{i+1}$ is piece-wise linear and all breakpoints are integer.*

Lemma 4.11. *Under Assumption 1, every breakpoint p^i_{i+1} of f satisfies at least one of the equations as marked in the table below:*

Description	***Equation***	***POS-AW***	***EXT-VW***	***POS-VW***
No adjustment:	$p^i_{i+1} = 0$	×	×	×
Arrive at start time window POS for some t_{ii+1}:	$x_i + t_{ii+1} = a_{i+1} + p^i_{i+1}$	×		×
Arrive at end time window for some t_{ii+1}:	$x_i + t_{ii+1} = b_{i+1} + p^i_{i+1}$	×	×	×
Start time window POS is integer:	$a_{i+1} + p^i_{i+1} \in \mathbb{Z}$	×		×
Time window POS start at preferred departure time:	$X^*_{i+1} = a_{i+1} + p^i_{i+1}$			×
Time window ends at preferred departure time:	$X^*_{i+1} = b_{i+1} + p^i_{i+1}$		×	×
Time window has width zero:	$a_{i+1} = b_{i+1} + p^i_{i+1}$		×	

Proof. If p^i_{i+1} is a breakpoint of f, then p^i_{i+1} must also be a breakpoint of one of its summands. Recall that by assumption, the expectation can be written as a finite sum.

Consider the case POS-AW. If p^i_{i+1} is a breakpoint of g_{i+1}, then it follows from the definition of g_{i+1}, Equation (4.20), that $p^i_{i+1} = 0$. Next, consider the h_{i+1} term. Note that h_{i+1} is defined by Equation (4.22). Without loss of generality, we may assume that $\delta_{i+1} = 0$, i.e., there is no penalty for early delivery. This is possible because we always wait until the time window opens. It follows that the h_{i+1} term within the expectation can be written as $\left(\left(\max\{x_i + t_{ii+1}, a_{i+1} + p^i_{i+1}\} - (b_{i+1} + p^i_{i+1})\right)^+ \gamma_{i+1}\right.$.

We observe two potential breakpoints for each h_{i+1} term. First, we may have a breakpoint when the two arguments of the maximum are equal, i.e., $x_i + t_{ii+1} = a_{i+1} + p^i_{i+1}$. Second, if $x_i + t_{ii+1} > a_{i+1} + p^i_{i+1}$ then $x_i + t_{ii+1} = b_{i+1} + p^i_{i+1}$ is a potential breakpoint. Note that if $x_i + t_{ii+1} < a_{i+1} + p^i_{i+1}$, then the h_{i+1} term is equal to $\left(a_{i+1} + p^i_{i+1} - (b_{i+1} + p^i_{i+1})\right)^+ \gamma_{i+1} = (a_{i+1} - b_{i+1})^+ \gamma_{i+1}$, which is constant in p^i_{i+1} and does not result in any breakpoints.

Next, we consider breakpoints due to the $\bar{c}_{i+1}$ term. Again, we can have a breakpoint if the arguments of the minimum are equal. By Assumption 1, the function $\bar{c}_{i+1}$ only has integer breakpoints. Hence, if $x_i + t_{ii+1} < a_{i+1} + p^i_{i+1}$, we have a potential breakpoint for $a_{i+1} + p^i_{i+1} \in \mathbb{Z}$. Note that if $x_i + t_{ii+1} > a_{i+1} + p^i_{i+1}$, then the $\bar{c}_{i+1}$ term is constant in p^i_{i+1}. This concludes the proof for POS-AW.

Next, consider the case EXT-VW. For the g_{i+1} term, we again obtain $p^i_{i+1} =$

0. In the definition of f, Equation (4.51), we have

$$x_{i+1} = \begin{cases} x_i + t_{ii+1} & \text{if } x_i + t_{ii+1} \geq \min\{X^*_{i+1}, a_{i+1}\} \\ \min\{X^*_{i+1}, a_{i+1}\} & \text{if } x_i + t_{ii+1} \leq \min\{X^*_{i+1}, a_{i+1}\}. \end{cases} \tag{4.52}$$

That is, if the vehicle arrives at customer $i+1$ before $\min\{X^*_{i+1}, a_{i+1}\}$, then it waits until that time. If the vehicle arrives later, customer $i+1$ is served immediately.

Now consider the h_{i+1} terms. From the definition, Equation (4.21), it follows that these terms are given by $\left(x_{i+1} - (b_{i+1} + p^i_{i+1})\right)^+ \gamma_j + (a_{i+1} - x_{i+1})^+ \delta_j$. If $x_{i+1} = x_i + t_{ii+1}$, we have a potential breakpoint for $x_i + t_{ii+1} = b_{i+1} + p^i_{i+1}$. Note that $(a_{i+1} - (x_i + t_{ii+1}))^+ \delta_j$ is constant in p^i_{i+1}, and thus does not yield breakpoints. If $x_{i+1} = \min\{X^*_{i+1}, a_{i+1}\}$, then depending on whether X^*_{i+1} or a_{i+1} is the minimum, we obtain potential breakpoints for $X^*_{i+1} = b_{i+1} + p^i_{i+1}$ and $a_{i+1} = b_{i+1} + p^i_{i+1}$. Finally, we consider $x_i + t_{ii+1} = \min\{X^*_{i+1}, a_{i+1}\}$, i.e., the value for which (4.52) switches cases. Again, we find potential breakpoints for $X^*_{i+1} = b_{i+1} + p^i_{i+1}$ and $a_{i+1} = b_{i+1} + p^i_{i+1}$.

Next, we consider breakpoints due to the $\bar{c}_{i+1}$ term. It follows from the definition of x_{i+1}, Equation (4.52), that $\bar{c}_{i+1}(x_{i+1})$ is constant in p^i_{i+1}. Hence, the $\bar{c}_{i+1}$ term does not provide additional potential breakpoints. This completes the proof for EXT-VW.

Finally, we consider POS-VW. For the g_{i+1} term, we again obtain $p^i_{i+1} = 0$. In the definition of f, Equation (4.51), we have

$$x_{i+1} = \begin{cases} x_i + t_{ii+1} & \text{if } x_i + t_{ii+1} \geq \min\{X^*_{i+1}, a_{i+1} + p^i_{i+1}\} \\ \min\{X^*_{i+1}, a_{i+1} + p^i_{i+1}\} & \text{if } x_i + t_{ii+1} \leq \min\{X^*_{i+1}, a_{i+1} + p^i_{i+1}\}. \end{cases} \tag{4.53}$$

That is, if the vehicle arrives at customer $i+1$ before $\min\{X^*_{i+1}, a_{i+1} + p^i_{i+1}\}$, then it waits until that time. If the vehicle arrives later, customer $i+1$ is served immediately.

The h_{i+1} terms are defined by Equation (4.22). As in the POS-AW case, we assume without loss of generality that $\delta_{i+1} = 0$. It follows that the h_{i+1} terms are given by $\left(x_{i+1} - (b_{i+1} + p^i_{i+1})\right)^+ \gamma_j$. If $x_{i+1} = x_i + t_{ii+1}$, we obtain a potential breakpoint for $x_i + t_{ii+1} = b_{i+1} + p^i_{i+1}$. Similarly, if $x_{i+1} = X^*_{i+1}$, we obtain a potential breakpoint for $X^*_{i+1} = b_{i+1} + p^i_{i+1}$. For $x_{i+1} = a_{i+1} + p^i_{i+1}$ we have $\left(x_{i+1} - (b_{i+1} + p^i_{i+1})\right)^+ \gamma_j = (a_{i+1} - b_{i+1})^+ \gamma_j$, which is constant in p^i_{i+1}.

Next, we consider the points where x_{i+1} switches cases. From $x_i + t_{ii+1} = \min\{X^*_{i+1}, a_{i+1} + p^i_{i+1}\}$ we obtain a potential breakpoint for $x_{i+1} + t_{ii+1} = a_{i+1} + p^i_{i+1}$.

Note that if $X^*_{i+1} < a_{i+1} + p^i_{i+1}$, then the minimum is constant in p^i_{i+1} and does not yield potential breakpoints. Finally, there can be a breakpoint if the arguments of the minimum are equal, i.e., $X^*_{i+1} = a_{i+1} + p^i_{i+1}$.

Now consider the breakpoints due to the $\bar{c}_{i+1}$ term. By Assumption 1, we have a potential breakpoint due to $\bar{c}_{i+1}(x_{i+1})$ is x_{i+1} is not constant in p^i_{i+1}, and x_{i+1} is integer. It follows that we can only get a breakpoint if $x_{i+1} = a_{i+1} + p^i_{i+1}$. The potential breakpoints that we find are given by $a_{i+1} + p^i_{i+1} \in \mathbb{Z}$. This completes the proof of POS-VW. After proving the cases POS-AW, EXT-VW, and POS-VW, we have proven the lemma. □

Lemma 4.12. *Under Assumption 1, the preferred departure time X^*_{i+1} from customer $i+1$ (see Proposition 4.10, Appendix 4.B) can be chosen such that $X^*_{i+1} \in \mathbb{Z} \cup \{-\infty, \infty\}$.*

Proof. By definition, $X^*_{i+1} = \arg\min_{X_{i+1} \in \bar{\mathbb{R}}} \{\bar{c}_{i+1}(X_{i+1}) - X_{i+1}\delta_{i+1}\}$. Under Assumption 1, the function $\bar{c}_{i+1}(X_{i+1}) - X_{i+1}\delta_{i+1}$ is piece-wise linear and only has breakpoints on the integers. As X^*_{i+1} is the arg min of this function, we may assume that $X^*_{i+1} \in \mathbb{Z} \cup \{-\infty, \infty\}$. □

Lemma 4.13. *Based on the parameter x_i, let $p^*(x_i) = \arg\min_{p^i_{i+1} \in \mathcal{P}_{i+1}} f(p^i_{i+1})$ be an optimal adjustment. Under Assumption 1, if $\bar{x}_i \notin \mathbb{Z}$, then $p^*(x_i) = C$ or $p^*(x_i) = x_i + C$ on the domain $x_i \in (\lfloor \bar{x}_i \rfloor, \lceil \bar{x}_i \rceil)$, for some integer value C.*

Proof. For convenience, we define $\bar{p}^i_{i+1} = p^*(\bar{x}_i)$ to be the optimal adjustment that corresponds to the departure time $\bar{x}_i$. Throughout this proof, we assume that $x_i \in (\lfloor \bar{x}_i \rfloor, \lceil \bar{x}_i \rceil)$. We refer to this set as the *neighborhood of* $\bar{x}_i$. We use $\bar{f}$ to denote the function f in which x_i is replaced by $\bar{x}_i$.

We define Q to be the set of *potential breakpoints* of f. That is, for a given value of x_i, the set Q contains all values of p^i_{i+1} that satisfy at least one of the relevant equations in Lemma 4.11. Similarly, we define $\bar{Q}$ to be the set of potential breakpoints of $\bar{f}$. By definition, there is a bijection between Q and $\bar{Q}$.

Consider $q \in Q$ and $\bar{q} \in \bar{Q}$, both defined by the same equation. By going over the equations in Lemma 4.11, and using that $\bar{x}_i \notin \mathbb{Z}$ and $x_i \in (\lfloor \bar{x}_i \rfloor, \lceil \bar{x}_i \rceil)$, the following three implications can be verified.

1. If $\bar{q} \in \mathbb{Z}$ then $q - \bar{q} = 0$, i.e., the difference between q and $\bar{q}$ is zero.

2. If $\bar{q} \notin \mathbb{Z}$ then $\bar{q} - \bar{x}_i \in \mathbb{Z}$, i.e., $\bar{q}$ and $\bar{x}_i$ have the same fractional part.

3. If $\bar{q} \notin \mathbb{Z}$ then $q - \bar{q} = x_i - \bar{x}_i$, i.e., the difference between q and $\bar{q}$ is equal to the difference between x_i and $\bar{x}_i$.

For example, consider the third equation in Lemma 4.11. We obtain $x_i + t_{ii+1} = b_{i+1} + q$ and $\bar{x}_i + t_{ii+1} = b_{i+1} + \bar{q}$. We first verify the first statement. If $\bar{q} \in \mathbb{Z}$, then it follows from the integrality of t_{ii+1} and b_{i+1} that $\bar{x}_i \in \mathbb{Z}$. This contradicts the assumption that $\bar{x}_i \notin \mathbb{Z}$, and the implication trivially holds. Next, we consider $\bar{q} \notin \mathbb{Z}$. Rewriting the equations yields $q - x_i = t_{ii+1} - b_{i+1}$ and $\bar{q} - \bar{x}_i = t_{ii+1} - b_{i+1}$, which shows $\bar{q} - \bar{x}_i \in \mathbb{Z}$ and $q - \bar{q} = x_i - \bar{x}_i$. Hence, both implications hold. The verification of the three implications for the other equations in Lemma 4.11 is similar.

It can be seen that the subdifferential $\partial f(p^i_{i+1})$ is uniquely determined by the set of potential breakpoints q such that $q < p^i_{i+1}$, and whether p^i_{i+1} is a potential breakpoint itself. This follows from the fact that the slopes of the summands of f can only change at the potential breakpoints, and that replacing x_i by $\bar{x}_i$ only changes the locations of the potential breakpoints, and not the slopes before and after the potential breakpoints.

Next, we show that the ordering of the potential breakpoints is the same for f and $\bar{f}$. Earlier, we have shown that if $\bar{q} \in \mathbb{Z}$, then $q - \bar{q} = 0$, i.e., the potential breakpoints are equal (Implication 1). In the case of $\bar{q} \notin \mathbb{Z}$ we have shown that $q - \bar{q} = x_i - \bar{x}_i$ (Implication 3) and that $\bar{q}$ and $\bar{x}_i$ have the same fractional part (Implication 2). By assumption, $\bar{x} \notin \mathbb{Z}$, and $x_i \in (\lfloor \bar{x}_i \rfloor, \lceil \bar{x}_i \rceil)$. It follows immediately that $q \in (\lfloor \bar{q} \rfloor, \lceil \bar{q} \rceil)$. In summary, f and $\bar{f}$ have the same integer breakpoints, and all fractional breakpoints change by the same amount. The fractional breakpoints remain fractional, which implies that the ordering of the potential breakpoints is the same for f and $\bar{f}$.

As an immediate consequence, we have $\partial f(q) = \partial \bar{f}(\bar{q})$ for corresponding potential breakpoints q and $\bar{q}$. Furthermore, the subdifferential of f at a point between two sequential potential breakpoints q and q' is equal to the subdifferential of $\bar{f}$ at a point between $\bar{q}$ and $\bar{q}'$. A similar argument can be made for a point between a boundary point of $\mathcal{P}_{i+1}$ and the closest potential breakpoint. Note that the lemma is trivially satisfied if $\mathcal{P}_{i+1}$ is a singleton.

We are now ready to prove that if $\bar{x}_i \notin \mathbb{Z}$, then $p^*(x_i) = C$ or $p^*(x_i) = x_i + C$, for x_i in the neighborhood of $\bar{x}_i$, for an integer value C. Consider an optimal adjustment $\bar{p}^i_{i+1} \in \mathbb{R}$ for a given departure time $\bar{x}_i \notin \mathbb{Z}$. Because $\bar{f}$ is a polyhedral convex function, we may assume that $\bar{p}^i_{i+1}$ is a breakpoint of $\bar{f}$ with $0 \in \partial \bar{f}(\bar{p}^i_{i+1})$, or $\bar{p}^i_{i+1}$ is a boundary point of $\mathcal{P}_{i+1}$.

We first consider the breakpoints of $\bar{f}$. If $\bar{p}^i_{i+1}$ is a breakpoint of $\bar{f}$ and $\bar{p}^i_{i+1} \in \mathbb{Z}$,

then $p^i_{i+1} = \bar{p}^i_{i+1}$ is the corresponding breakpoint of f. It follows that $\partial \bar{f}(\bar{p}^i_{i+1}) = \partial f(p^i_{i+1}) \ni 0$, which implies by convexity that p^i_{i+1} is optimal for x_i. Hence, we have $p^*(x_i) = C$ with $C = \bar{p}^i_{i+1}$, which is integer by assumption. Similarly, if $\bar{p}^i_{i+1}$ is breakpoint of $\bar{f}$ and $\bar{p}^i_{i+1} \notin \mathbb{Z}$, then $p^i_{i+1} - \bar{p}^i_{i+1} = x_i - \bar{x}_i$ (Implication 3) $\Leftrightarrow$ $p^i_{i+1} = x_i + \bar{p}^i_{i+1} - \bar{x}_i$ is optimal for x_i. That is, $p^*(x_i) = x_i + C$, with $C = \bar{p}^i_{i+1} - \bar{x}_i$. Note that C is integer by Implication 2.

Next, we consider the case that $\bar{p}^i_{i+1}$ is a boundary point of $\mathcal{P}_{i+1}$. By assumption, the boundary points of $\mathcal{P}_{i+1}$ are integral. It follows that $p^i_{i+1} = \bar{p}^i_{i+1}$. Hence, we can use $p^*(x_i) = C$, with $C = \bar{p}^i_{i+1}$ integer. This completes the proof. □

Lemma 4.14. *Under Assumption 1, for case POS-AW, we have that $\bar{c}_i$ is piece-wise linear and all breakpoints are integer.*

Proof. From the definition of $\bar{c}_i$, Equation (4.27), it is straightforward to show that $\bar{c}_i$ is a polyhedral convex function. Hence, $\bar{c}_i$ is piece-wise linear. It remains to prove that $\bar{c}_i$ has no breakpoints on the integers.

We prove this fact by contradiction. First, we assume that $x_i \notin \mathbb{Z}$ is a breakpoint of $\bar{c}_i(x_i)$. By Lemma 4.13, we have that the optimal adjustment in the neighborhood of x_i is given by $p^*(x_i) = C$ or $p^*(x_i) = x_i + C$ for some integer value C.

If we substitute the optimal actions into the definition of $\bar{c}_i(x_i)$, we obtain a straightforward expression for this function that is valid in the neighborhood of the current state x_i. By analyzing this expression, we show that it only has breakpoints on the integers. By contradiction, $x_i \notin \mathbb{Z}$ cannot be a breakpoint.

By definition, $\bar{c}_i(x_i) = \min_{p^i_{i+1} \in \mathcal{P}_{i+1}} f(p^i_{i+1})$. Instead of minimizing over p^i_{i+1}, we use the optimal adjustment $p^*(x_i)$. We then obtain $\bar{c}_i(x_i) = f(p^*(x_i))$.

First consider $p^*(x_i) = C$. Recall that $C \in \mathbb{Z}$ by Lemma 4.13. It follows from Equation (4.50) that

$$\begin{aligned}\bar{c}_i(x_i) = g_{i+1}(0, C) + \mathbb{E}\left[h_{i+1}(\max\{x_i + t_{ii+1}, a_{i+1} + C\}, C)\right] + \\ \mathbb{E}\left[\bar{c}_{i+1}\left(\max\{x_i + t_{ii+1}, a_{i+1} + C\}\right)\right]. \qquad (4.54)\end{aligned}$$

Next, we consider the breakpoints of $\bar{c}_i(x_i)$. Note that the g_{i+1} term is constant in x_i, and does not result in any breakpoints. It is straightforward to verify that h_{i+1} only has breakpoints for integer x_i. This follows from the fact that all parameters are integer. By Assumption 1, $\bar{c}_{i+1}$ only has breakpoints for the integers. If $x_i + t_{ii+1} < a_{i+1} + C$, then the $\bar{c}_{i+1}$ term is constant in x_i and does not give any potential breakpoints. If $x_i + t_{ii+1} > a_{i+1} + C$, then potential breakpoints are given by

$x_i + t_{ii+1} \in \mathbb{Z}$, which implies that the breakpoints x_i of $\bar{c}_i$ are integer. Finally, we may have a breakpoint for $x_i + t_{ii+1} = a_{i+1} + C$, which also corresponds to $x_i \in \mathbb{Z}$.

Hence, for $p^*(x_i) = C$, we have shown that if $x_i \notin \mathbb{Z}$ is a breakpoint of $\bar{c}_i$, then the function $\bar{c}_i$ defined in the neighborhood of x_i only has breakpoints on the integers. By contradiction, $x_i \notin \mathbb{Z}$ cannot be a breakpoint. It follows that $\bar{c}_i$ only has breakpoints on the integers.

We have to show the same result for $p^*(x_i) = x_i + C$. In this case, we have

$$\begin{aligned}\bar{c}_i(x_i) = g_{i+1}(0, x_i + C) + \mathbb{E}\left[h_{i+1}(\max\{x_i + t_{ii+1}, a_{i+1} + x_i + C\}, x_i + C)\right] + \\ \mathbb{E}\left[\bar{c}_{i+1}\left(\max\{x_i + t_{ii+1}, a_{i+1} + x_i + C\}\right)\right].\end{aligned} \tag{4.55}$$

Again, by combining Assumption 1 with the fact that all parameters are integer, it is straightforward to show that $\bar{c}_i$ only has breakpoints on the integers. Applying the same contradiction argument as for $p^*(x_i) = C$ completes the proof. □

Lemma 4.15. *Under Assumption 1, for case EXT-VW, we have that $\bar{c}_i$ is piece-wise linear and all breakpoints are integer.*

Proof. We use the same argument as in Lemma 4.14, but for a different function $\bar{c}_i$. Specifically, we show that the function $\bar{c}_i(x_i) = f(p^*(x_i))$ only has breakpoints for integer x_i, with f as in Equation (4.51). The other parts of the proof are identical.

In this case, we obtain

$$\begin{aligned}\bar{c}_i(x_i) = g_{i+1}(0, p^*(x_i)) + \\ \mathbb{E}\left[h_{i+1}\left(x_i + t_{ii+1} + \left(\min\{X^*_{i+1}, a_{i+1}\} - (x_i + t_{ii+1})\right)^+, p^*(x_i)\right)\right] + \\ \mathbb{E}\left[\bar{c}_{i+1}\left(x_i + t_{ii+1} + \left(\min\{X^*_{i+1}, a_{i+1}\} - (x_i + t_{ii+1})\right)^+\right)\right].\end{aligned} \tag{4.56}$$

It then needs to be verified that $\bar{c}_i$ only has integer breakpoints when $p^*(x_i) = C$ and when $p^*(x_i) = x_i + C$, for some integer C. This can be shown using the integrality of the parameters, and using that $\bar{c}_{i+1}$ only has breakpoints on the integers (Assumption 1). The arguments are straightforward, and similar to those in Lemma 4.11 and Lemma 4.14. As such, we omit them here.

After it is shown that (4.56) only has integer breakpoints, the proof for the EXT-VW case is the same as the proof for the POS-AW case (Lemma 4.14). □

Lemma 4.16. *Under Assumption 1, for case POS-VW, we have that $\bar{c}_i$ is piece-wise linear and all breakpoints are integer.*

Proof. Similar to the proof of Lemma 4.15, but for a different function $\bar{c}_i$. From Equation (4.51) we obtain In this case, we obtain

$$\begin{aligned}\bar{c}_i(x_i) = {} & g_{i+1}(0, p^*(x_i)) + \\ & \mathbb{E}\left[h_{i+1}\left(x_i + t_{ii+1} + \left(\min\{X^*_{i+1}, a_{i+1} + p^*(x_i)\} - (x_i + t_{ii+1})\right)^+, p^*(x_i)\right)\right] + \\ & \mathbb{E}\left[\bar{c}_{i+1}\left(x_i + t_{ii+1} + \left(\min\{X^*_{i+1}, a_{i+1} + p^*(x_i)\} - (x_i + t_{ii+1})\right)^+\right)\right].\end{aligned} \tag{4.57}$$

Similar as in Lemma 4.11, Lemma 4.14, and Lemma 4.15, it can be proven that $\bar{c}_i$ only has integer breakpoints. These steps are omitted here. Using the argument of Lemma 4.14 completes the proof. □

Proposition 4.9. *The simple DTWAP with discretized time allows for optimal integer decisions in the cases of POS-AW, EXT-VW, and POS-VW.*

Proof. By definition, $\bar{c}_n = 0$, which is linear and does not have breakpoints. Lemmas 4.14 to 4.16 show for all $i \in \{0, \ldots, n-1\}$ that if $\bar{c}_{i+1}$ is piece-wise linear and only has integer breakpoints, then the same is true for $\bar{c}_i$. By induction, it follows that $\bar{c}_i$ is piece-wise linear and only has integer breakpoints for all $i \in \{0, \ldots, n\}$.

It then follows from Lemma 4.11 that for a given $x \in \mathbb{Z}$, there exists an optimal adjustment $p^i_{i+1} \in \mathbb{Z}$. It follows from Lemma 4.12, and from the definition of the optimal waiting time, that there also exist optimal voluntary waiting times that are integer.

Hence, for a given integer state, there exist optimal integer actions. If integer actions are taken, the next state will again be integer. This can be seen from Equation (4.29), the definition of $\bar{c}_i$. It follows that the simple DTWAP with discretized time allows for optimal integer decisions in the cases of POS-AW, EXT-VW, and POS-VW. □

Chapter 5

Summary and conclusion

In this thesis, we consider two main problems: the Time Window Assignment Vehicle Routing Problem (TWAVRP) and the Dynamic Time Window Adjustment Problem (DTWAP). In the Introduction (Chapter 1), we explain the motivation and the practical relevance behind studying these problems. Chapters 2 and 3 are dedicated to developing solution methods for the TWAVRP, whereas in Chapter 4 we analyze the DTWAP.

In Chapter 2, we present our branch-and-cut algorithm for the TWAVRP. This algorithm marks a significant improvement for the computational tractability of the TWAVRP. In our experiments, we demonstrate that the new algorithm is about 200 times faster than the branch-price-and-cut algorithm by Spliet and Gabor (2015), which was the best known algorithm at the time. Furthermore, we can handle networks with more customers than previously possible.

The success of the algorithm in Chapter 2 is in part due to a novel set of valid inequalities: the precedence inequalities. We present multiple heuristics to separate the precedence inequalities, and we show experimentally that using precedence inequalities speeds up our algorithm by a factor 3.8. The precedence inequalities represent an important mathematical property of the TWAVRP, which is used in a different context in Chapter 3.

In Chapter 3, we present our branch-price-and-cut algorithm for the TWAVRP. This new algorithm outperforms the algorithm in Chapter 2 by properly addressing *orientation-symmetry*. Solutions are orientation-symmetric when they use the same routes, but one or more routes have a different orientation, i.e., the clients are visited in the reverse order. We observe that TWAVRP instances that are difficult to solve,

often have many orientation-symmetric solutions.

To address the issue of symmetry, we introduce an edge-based branching method combined with additional components to eliminate orientation-symmetry from the search tree. Furthermore, we present enhancements to make this method efficient in practice. For the enhancements, we make use of the precedence inequalities presented in Chapter 2.

Applying the edge-based branching method to our branch-price-and-cut algorithm significantly improves computational performance. Our experiments show that the number of nodes in the search tree is reduced by 92.6% on average. This demonstrates that addressing orientation-symmetry indeed plays an important role in solving the TWAVRP.

Even though we focus on solving the TWAVRP, the idea of addressing orientation-symmetry is not TWAVRP specific. For example, our branching method can be applied to other vehicle routing problems with consistency considerations or synchronization requirements. Given the benefit of addressing orientation-symmetry in the TWAVRP, applying our method to other problems is an interesting direction for further research.

In Chapter 4, we derive general properties of the DTWAP, and we present three different solution methods. We do not restrict ourselves to specific customer dissatisfaction functions. As such, this chapter can serve as a guide for solving the DTWAP in a variety of settings, including parcel delivery, attended home delivery, and retailer distribution.

Our illustrative example about attended home delivery demonstrates how the results of Chapter 4 can be used to solve practical problems. Our computational results show that dynamic time window adjustments have the potential to improve customer satisfaction by effectively communicating delays to the customers. Finally, we demonstrate how the ideas developed throughout Chapter 4 can be used to construct different heuristics.

The DTWAP is a novel problem with the potential to improve customer satisfaction in a variety of settings. As such, there are plenty of opportunities for further research, including practical applications of the DTWAP, and integration of the DTWAP with vehicle routing problems.

References

Niels Agatz, Ann Campbell, Moritz Fleischmann, and Martin W. P. Savelsbergh (2008). Challenges and Opportunities in Attended Home Delivery. *The Vehicle Routing Problem: Latest Advances and New Challenges*. Ed. by Bruce L. Golden, S. Raghavan, and Edward Wasil. Springer, pages 379–396.

Niels Agatz, Ann Campbell, Moritz Fleischmann, and Martin W. P. Savelsbergh (2011). Time Slot Management in Attended Home Delivery. *Transportation Science*, 45(3):435–449.

Norbert Ascheuer, Matteo Fischetti, and Martin Grötschel (2000). A Polyhedral Study of the Asymmetric Traveling Salesman Problem with Time Windows. *Networks*, 36(2):69–79.

Norbert Ascheuer, Matteo Fischetti, and Martin Grötschel (2001). Solving the Asymmetric Travelling Salesman Problem with time windows by branch-and-cut. *Mathematical Programming*, 90(3):475–506.

Robert J. Aumann and Roberto Serrano (2008). An Economic Index of Riskiness. *Journal of Political Economy*, 116(5):810–836.

Roberto Baldacci, Eleni Hadjiconstantinou, and Aristide Mingozzi (2004). An Exact Algorithm for the Capacitated Vehicle Routing Problem Based on a Two-Commodity Network Flow Formulation. *Operations Research*, 52(5):723–738.

Roberto Baldacci, Aristide Mingozzi, and Roberto Roberti (2011). New Route Relaxation and Pricing Strategies for the Vehicle Routing Problem. *Operations Research*, 59(5):1269–1283.

Roberto Baldacci, Aristide Mingozzi, and Roberto Roberti (2012). Recent exact algorithms for solving the vehicle routing problem under capacity and time window constraints. *European Journal of Operational Research*, 218(1):1–6.

Michel L. Balinski and Richard E. Quandt (1964). On an Integer Program for a Delivery Problem. *Operations Research*, 12(2):300–304.

Cynthia Barnhart, Ellis L. Johnson, George L. Nemhauser, Martin W. P. Savelsbergh, and Pamela H. Vance (1998). Branch-and-Price: Column Generation for Solving Huge Integer Programs. *Operations Research*, 46(3):316–329.

Richard Bellman (1966). Dynamic Programming. *Science*, 153(3731):34–37.

Jacques F. Benders (1962). Partitioning procedures for solving mixed-variables programming problems. *Numerische Mathematik*, 4(1):238–252.

Dimitri P. Bertsekas (2005). *Dynamic Programming and Optimal Control, Vol. I*. 3rd. Athena Scientific.

Endre Boros and Peter L. Hammer (1991). The max-cut problem and quadratic 0–1 optimization; polyhedral aspects, relaxations and bounds. *Annals of Operations Research*, 33(3):151–180.

Stephen Boyd and Lieven Vandenberghe (2004). *Convex Optimization*. Cambridge University Press.

Claudio Contardo, Jean-François Cordeau, and Bernard Gendron (2014). An Exact Algorithm Based on Cut-and-Column Generation for the Capacitated Location-Routing Problem. *INFORMS Journal on Computing*, 26(1):88–102.

Claudio Contardo, Guy Desaulniers, and François Lessard (2015). Reaching the Elementary Lower Bound in the Vehicle Routing Problem with Time Windows. *Networks*, 65(1):88–99.

Kevin Dalmeijer (2016). Time window assignment in an uncertain world. `https://www.erim.eur.nl/centres/last-mile/blog/detail/3869-time-window-assignment-in-an-uncertain-world/`.

Kevin Dalmeijer (2017). Time window assignment and symmetry. `http://www.erim.eur.nl/last-mile/blog/detail/4094-time-window-assignment-and-symmetry/`.

Kevin Dalmeijer and Guy Desaulniers (2018). Addressing orientation-symmetry in the Time Window Assignment Vehicle Routing Problem. *Cahiers du Gerad*, G-2018-48.

Kevin Dalmeijer and Remy Spliet (2018). A branch-and-cut algorithm for the Time Window Assignment Vehicle Routing Problem. *Computers & Operations Research*, 89:140–152.

Kevin Dalmeijer, Remy Spliet, and Albert P. M. Wagelmans (2019). Dynamic Time Window Adjustment. *Econometric Institute Research Papers*, EI2019-22.

Guy Desaulniers, Jacques Desrosiers, and Simon Spoorendonk (2011). Cutting Planes for Branch-and-Price Algorithms. *Networks*, 58(4):301–310.

Guy Desaulniers, François Lessard, and Ahmed Hadjar (2008). Tabu Search, Partial Elementarity, and Generalized k-path Inequalities for the Vehicle Routing Problem with Time Windows. *Transportation Science*, 42(3):387–404.

Michael Drexl (2012). Synchronization in Vehicle Routing—A Survey of VRPs with Multiple Synchronization Constraints. *Transportation Science*, 46(3):297–316.

Fausto Errico, Guy Desaulniers, Michel Gendreau, Walter Rei, and Louis-Martin Rousseau (2016). A priori optimization with recourse for the vehicle routing problem with hard time windows and stochastic service times. *European Journal of Operational Research*, 249(1):55–66.

Michael R. Garey and David S. Johnson (1979). *Computers and Intractability: A Guide to the Theory of Np-Completeness*. W. H. Freeman and Company.

Michel Gendreau, Gilbert Laporte, and René Séguin (1996). Stochastic vehicle routing. *European Journal of Operational Research*, 88(1):3–12.

Chris Groër, Bruce L. Golden, and Edward Wasil (2009). The Consistent Vehicle Routing Problem. *Manufacturing & Service Operations Management*, 11(4):630–643.

Shuihua Han, Ling Zhao, Kui Chen, Zong-wei Luo, and Deepa Mishra (2017). Appointment scheduling and routing optimization of attended home delivery system with random customer behavior. *European Journal of Operational Research*, 262(3):966–980.

Soumia Ichoua, Michel Gendreau, and Jean-Yves Potvin (2003). Vehicle dispatching with time-dependent travel times. *European Journal of Operational Research*, 144(2):379–396.

Irina Ioachim, Sylvie Gélinas, François Soumis, and Jacques Desrosiers (1998). A Dynamic Programming Algorithm for the Shortest Path Problem with Time Windows and Linear Node Costs. *Networks*, 31(3):193–204.

Ola Jabali, Roel Leus, Tom van Woensel, and Ton G. de Kok (2015). Self-imposed time windows in vehicle routing problems. *OR Spectrum*, 37(2):331–352.

Patrick Jaillet, Jin Qi, and Melvyn Sim (2016). Routing optimization under uncertainty. *Operations Research*, 64(1):186–200.

Mads Jepsen, Bjørn Petersen, Simon Spoorendonk, and David Pisinger (2008). Subset-Row Inequalities Applied to the Vehicle-Routing Problem with Time Windows. *Operations Research*, 56(2):497–511.

Arthur B. Kahn (1962). Topological sorting of large networks. *Communications of the ACM*, 5(11):558–562.

Peter Kall and János Mayer (2011). *Stochastic Linear Programming: Models, Theory, and Computation*. 2nd. Springer.

Jon Kleinberg and Éva Tardos (2006). *Algorithm Design*. Pearson Education.

Anton J. Kleywegt, Alexander Shapiro, and Tito Homem-de-Mello (2002). The Sample Average Approximation Method for Stochastic Discrete Optimization. *SIAM Journal on Optimization*, 12(2):479–502.

Attila A. Kovacs, Bruce L. Golden, Richard F. Hartl, and Sophie N. Parragh (2014). Vehicle Routing Problems in Which Consistency Considerations are Important: A Survey. *Networks*, 64(3):192–213.

Gilbert Laporte, Yves Nobert, and Martin Desrochers (1985). Optimal Routing under Capacity and Distance Restrictions. *Operations Research*, 33(5):1050–1073.

Kunlei Lian (2017). Service Consistency in Vehicle Routing. PhD thesis. University of Arkansas.

Federico Liberatore, Giovanni Righini, and Matteo Salani (2011). A column generation algorithm for the vehicle routing problem with soft time windows. *4OR*, 9(1):49–82.

Jens Lysgaard (2003). *CVRPSEP: A package of separation routines for the Capacitated Vehicle Routing Problem*. Working Paper. Aarhus School of Business.

Jens Lysgaard, Adam N. Letchford, and Richard W. Eglese (2004). A new branch-and-cut algorithm for the capacitated vehicle routing problem. *Mathematical Programming*, 100(2):423–445.

Jorge E. Mendoza, Christelle Guéret, Maxim Hoskins, Hubert Lobit, Victor Pillac, Thibaut Vidal, and Daniele Vigo (2014). VRP-REP: the vehicle routing community repository. *Third Meeting of the EURO Working Group on Vehicle Routing and Logistics Optimization (VeRoLog). Oslo, Norway.*

Clair E. Miller, Albert W. Tucker, and Richard A. Zemlin (1960). Integer Programming Formulation of Traveling Salesman Problems. *Journal of the ACM*, 7(4):326–329.

Christos H. Papadimitriou (1977). The Euclidean travelling salesman problem is NP-complete. *Theoretical Computer Science*, 4(3):237–244.

Diego Pecin, Artur Pessoa, Marcus Poggi de Aragão, and Eduardo Uchoa (2017). Improved branch-cut-and-price for capacitated vehicle routing. *Mathematical Programming Computation*, 9(1):61–100.

Mario V. F. Pereira and Leontina M. V. G. Pinto (1991). Multi-stage stochastic optimization applied to energy planning. *Mathematical Programming*, 52(1-3):359–375.

Artur Pessoa, Marcus Poggi de Aragão, and Eduardo Uchoa (2008). Robust Branch-Cut-and-Price Algorithms for Vehicle Routing Problems. *The Vehicle Routing Problem: Latest Advances and New Challenges*. Ed. by Bruce L. Golden, S. Raghavan, and Edward Wasil. Springer, pages 297–325.

Roberto Roberti and Aristide Mingozzi (2014). Dynamic ng-Path Relaxation for the Delivery Man Problem. *Transportation Science*, 48(3):413–424.

Alexander Shapiro (2011). Analysis of stochastic dual dynamic programming method. *European Journal of Operational Research*, 209(1):63–72.

Alexander Shapiro, Darinka Dentcheva, and Andrzej Ruszczyński (2009). *Lectures on stochastic programming: modeling and theory*. SIAM.

Alexander Shapiro and Arkadi Nemirovski (2005). On complexity of stochastic programming problems. *Continuous Optimization*. Ed. by Vaithilingam Jeyakumar and Alexander Rubinov. Springer, pages 111–146.

R. M. van Slyke and Roger Wets (1969). L-Shaped Linear Programs with Applications to Optimal Control and Stochastic Programming. *SIAM Journal on Applied Mathematics*, 17(4):638–663.

Remy Spliet, Said Dabia, and Tom van Woensel (2018). The Time Window Assignment Vehicle Routing Problem with Time-Dependent Travel Times. *Transportation Science*, 52(2):261–276.

Remy Spliet and Guy Desaulniers (2015). The discrete time window assignment vehicle routing problem. *European Journal of Operational Research*, 244(2):379–391.

Remy Spliet and Adriana F. Gabor (2015). The Time Window Assignment Vehicle Routing Problem. *Transportation Science*, 49(4):721–731.

Anirudh Subramanyam and Chrysanthos E. Gounaris (2018). A Decomposition Algorithm for the Consistent Traveling Salesman Problem with Vehicle Idling. *Transportation Science*, 52(2):386–401.

Anirudh Subramanyam, Akang Wang, and Chrysanthos E. Gounaris (2018). A scenario decomposition algorithm for strategic time window assignment vehicle routing problems. *Transportation Research Part B: Methodological*, 117:296–317.

Duygu Taş, Michel Gendreau, Nico Dellaert, Tom van Woensel, and Ton G. de Kok (2014). Vehicle routing with soft time windows and stochastic travel times: A column generation and branch-and-price solution approach. *European Journal of Operational Research*, 236(3):789–799.

Marlin W. Ulmer and Barrett W. Thomas (2019). Enough Waiting for the Cable Guy–Estimating Arrival Times for Service Vehicle Routing. *Transportation Science*:forthcoming.

Junlong Zhang, William H. K. Lam, and Bi Yu Chen (2013). A Stochastic Vehicle Routing Problem with Travel Time Uncertainty: Trade-Off Between Cost and Customer Service. *Networks and Spatial Economics*, 13(4):471–496.

Junlong Zhang, William H. K. Lam, and Bi Yu Chen (2016). On-time delivery probabilistic models for the vehicle routing problem with stochastic demands and time windows. *European Journal of Operational Research*, 249(1):144–154.

Yu Zhang, Roberto Baldacci, Melvyn Sim, and Jiafu Tang (2019). Routing optimization with time windows under uncertainty. *Mathematical Programming*, 175(1-2):263–305.

Abstract

In delivery networks, companies assign time windows to customers, informing them between which times they may expect service. Unfortunately, these time windows are often too wide to be informative, for example from 9am to 5pm. Another problem is that time windows may not be met due to unforeseen delays.

In this thesis, we study assigning time windows in distribution networks to improve customer satisfaction. More and more companies realize that the satisfaction of their customers is important. As such, the customers cannot be ignored when assigning time windows and constructing delivery routes.

We consider two optimization problems that explicitly incorporate the satisfaction of the customers. First, we study the Time Window Assignment Vehicle Routing Problem. This is the problem of assigning time windows for delivery before demand volume becomes known. In this case, the distributor guarantees service within the assigned time window, regardless of the realization of demand.

Second, we introduce and study the Dynamic Time Window Adjustment Problem. Here, the distributor is faced with travel time uncertainty. To deal with this uncertainty, the time windows that are communicated to the customers can be updated throughout the day. Adjustments may not be appreciated by the customers. However, informing the customer timely that a delivery will be made in a later time window is preferred to missing the deadline unannounced. We call these updates dynamic time window adjustments, and the Dynamic Time Window Adjustment Problem is the problem of optimizing these.

Nederlandse Samenvatting (Abstract in Dutch)

In distributienetwerken is het gebruikelijk dat bedrijven tijdvensters toekennen aan klanten om ze te informeren tussen welke tijden er wordt geleverd. Helaas zijn zulke tijdvensters vaak erg ruim, bijvoorbeeld van 9 uur 's ochtends tot 5 uur 's middags, waardoor ze weinig informatie geven. Een ander probleem is dat onvoorziene vertragingen ervoor kunnen zorgen dat de tijdvensters niet worden gehaald.

In dit proefschrift onderzoeken we het toekennen van tijdvensters in distributienetwerken, met als doel de klanttevredenheid te verbeteren. Steeds meer bedrijven realiseren zich dat de tevredenheid van hun klanten belangrijk is. Het volgt dat de belangen van de klant niet genegeerd kunnen worden bij het toekennen van tijdvensters en het plannen van bezorgroutes.

We beschouwen twee optimalisatieproblemen waarin klanttevredenheid expliciet wordt meegenomen. Het eerste probleem dat we bestuderen is het Tijdvenstertoekenning en Voertuigrouteringsprobleem (*Time Window Assignment Vehicle Routing Problem*). Dit probleem vraagt om tijdvensters toe te kennen voordat de groottes van de orders bekend zijn. In dit geval garandeert de distributeur dat levering plaatsvindt binnen het toegekende tijdvenster, ongeacht de daadwerkelijke ordergroottes.

Het tweede probleem dat we bestuderen is het Dynamische Tijdvensteraanpassingsprobleem (*Dynamic Time Window Adjustment Problem*). Voor dit probleem heeft de distributeur te maken met onzekere reistijden. Om hier mee om te gaan staan we toe dat de tijdvensters gedurende de dag worden bijgesteld. Het aanpassen van de tijdvensters wordt misschien niet door de klant gewaardeerd. Aan de andere kant: de klant tijdig informeren dat een levering is vertraagd is beter dan onaangekondigd te laat leveren. We noemen deze updates dynamische tijdvensteraanpassingen, en het Dynamische Tijdvensteraanpassingsprobleem vraagt om deze te optimaliseren.

About the author

Kevin holds a Bachelor's degree in Econometrics and Operations Research and a Master's degree in Econometrics and Management Science, with a specialization in Operations Research and Quantitative Logistics. Both degrees were obtained summa cum laude from the Erasmus University Rotterdam. Kevin's main research interest is logistics optimization, and in particular vehicle routing problems. During his PhD research, his focus was on how to assign and update time windows in distribution networks.

Kevin was awarded the *Best Econometric Thesis Award* for his Master's thesis at Erasmus University Rotterdam. He has presented his research on several international conferences. These conferences include the TRISTAN symposium on transportation analysis, the Odysseus workshop on freight transportation and logistics, and the VeRoLog workshop on vehicle routing and logistics. His research has been published in the journal *Computers & Operations Research*. In 2017, Kevin paid a four month research visit to the world-renowned research center GERAD in Montreal to collaborate with professor Desaulniers.

At the Erasmus University, Kevin helped redesign the course in non-linear optimization, and he gave lectures during five years. He has been involved in projects that create impact and improve education, for which he has been awarded the *Societal Impact Award* in 2015 and the *Educational Innovation Award* in 2017 by the Erasmus School of Economics. Kevin currently is a Postdoctoral Fellow at the H. Milton Stewart School of Industrial and Systems Engineering at the Georgia Institute of Technology.

Portfolio

Publications in peer-reviewed journals

Kevin Dalmeijer and Remy Spliet (2018). A branch-and-cut algorithm for the Time Window Assignment Vehicle Routing Problem. *Computers & Operations Research*, 89:140–152.

Working papers and reports

Kevin Dalmeijer and Guy Desaulniers (2018). Addressing orientation-symmetry in the Time Window Assignment Vehicle Routing Problem. *Cahiers du Gerad*, G-2018-48.

Kevin Dalmeijer, Remy Spliet, and Albert P. M. Wagelmans (2019). Dynamic Time Window Adjustment. *Econometric Institute Research Papers*, EI2019-22.

Ymro N. Hoogendoorn and Kevin Dalmeijer (2019). State-space relaxation dependent valid inequalities for set covering and set partitioning models. *Manuscript in preparation.*

Contributions Last-Mile Logistics blog

Kevin Dalmeijer (2016). Time window assignment in an uncertain world. `https://www.erim.eur.nl/centres/last-mile/blog/detail/3869-time-window-assignment-in-an-uncertain-world/`.

Kevin Dalmeijer (2017). Time window assignment and symmetry. `http://www.erim.eur.nl/last-mile/blog/detail/4094-time-window-assignment-and-symmetry/`.

Teaching (lecturing, course design, and supervision)

Jan Brinkhuis and Kevin Dalmeijer (2014-2015), *Non-linear Optimization*, Erasmus School of Economics, Econometrics and Operations Research.

Patrick J. F. Groenen and Kevin Dalmeijer (2015-2018), *Non-linear Optimization*, Erasmus School of Economics, Econometrics and Operations Research.

S. Ilker Birbil and Kevin Dalmeijer (2018-2019), *Non-linear Optimization*, Erasmus School of Economics, Econometrics and Operations Research.

Conference presentations

Odysseus 2015, Ajaccio, France.

Econometric Institute PhD Conference 2016, Rotterdam, The Netherlands.

LNMB conference 2016, Lunteren, The Netherlands.

TRISTAN 2016, Oranjestad, Aruba.

JOPT 2017, Montréal, Canada.

IFORS 2017, Québec, Canada.

Odysseus 2018, Cagliari, Italy.

INFORMS 2018, Phoenix, USA.

Last-Mile Delivery Workshop 2018, Rotterdam, The Netherlands.

VeRoLog 2019, Sevilla, Spain.

EURO 2019, Dublin, Ireland.

Other presentations

ESE take-off lecture 2015, Rotterdam, The Netherlands.

Econometric Institute seminar, 2015, Rotterdam, The Netherlands.

Publishing Papers in OR seminar, 2015, Rotterdam, The Netherlands.

ESE take-off lecture 2016, Rotterdam, The Netherlands.

QED seminar, 2016, Rotterdam, The Netherlands.

Constraint Programming seminar, 2016, Rotterdam, The Netherlands.

GERAD seminar, 2017, Montréal, Canada.

GERAD students day presentation, 2017, Montréal, Canada.

Community of Practice: digital grading, 2017, Rotterdam, The Netherlands.

Host ESE Bachelor Graduation Day, 2017, Rotterdam, The Netherlands.

Econometric Institute seminar, 2018, Rotterdam, The Netherlands.

PhD courses and certificates

Algorithms and Complexity.
Convex Analysis for Optimization.
Integer Programming Methods.
Interior Point Methods.
Markov Decision Processes.
Multi-class Queues and Stochastic Networks.
Networks and Polyhedra.
Networks and Semidefinite Programming.
Randomized Algorithms.
Robust Optimization.
Stochastic Programming.
Crafting and publishing papers in OR.
Reading group Constraint Programming.
Publishing Strategy.
Scientific Integrity.
Cambridge English: Proficiency.
University Teaching Qualification.

External positions

Consultant at S-ray Diagnostics (2015-2018), Rotterdam, The Netherlands.
PhD Representative at the LNMB (2016-2017), The Netherlands.

Awards

Kevin Dalmeijer (2014), *Best Econometric Thesis Award*, Awarded by Veneficus, Rotterdam, The Netherlands.

Marco de Haas, Daan L. van Knippenberg, Patrick J. F. Groenen, Kevin Dalmeijer, and Lisanne van Bunderen (2015), *ESE Societal Impact Award*, Awarded by Erasmus School of Economics, Rotterdam, The Netherlands.

Kevin Dalmeijer (2017), *ESE Educational Innovation Award*, Awarded by Erasmus School of Economics, Rotterdam, The Netherlands.

The ERIM PhD Series

The ERIM PhD Series contains PhD dissertations in the field of Research in Management defended at Erasmus University Rotterdam and supervised by senior researchers affiliated to the Erasmus Research Institute of Management (ERIM). All dissertations in the ERIM PhD Series are available in full text through the ERIM Electronic Series Portal: repub.eur.nl/pub. ERIM is the joint research institute of the Rotterdam School of Management (RSM) and the Erasmus School of Economics (ESE) at the Erasmus University Rotterdam (EUR).

Dissertations in the last four years

Ahmadi, S., *A motivational perspective to decision-making and behavior in organizations*, Promotors: Prof. J.J.P. Jansen & Prof. T.J.M. Mom, EPS-2019-477-S&E, repub.eur.nl/pub/116727

Akemu, O., *Corporate Responses to Social Issues: Essays in Social Entrepreneurship and Corporate Social Responsibility*, Promotors: Prof. G.M. Whiteman & Dr. S.P. Kennedy, EPS-2017-392-ORG, repub.eur.nl/pub/95768

Albuquerque de Sousa, J.A., *International stock markets: Essays on the determinants and consequences of financial market development*, Promotors: Prof. M.A. van Dijk & Prof. P.A.G. van Bergeijk, EPS-2019-465-F&A, repub.eur.nl/pub/115988

Alexiou, A., *Management of Emerging Technologies and the Learning Organization: Lessons from the Cloud and Serious Games Technology*, Promotors: Prof. S.J. Magala, Prof. M.C. Schippers & Dr. I. Oshri, EPS-2016-404-ORG, repub.eur.nl/pub/93818

Alserda, G.A.G., *Choices in Pension Management*, Promotors: Prof. S.G. van der Lecq & Dr. O.W. Steenbeek, EPS-2017-432-F&A, repub.eur.nl/pub/103496

Arampatzi, E., *Subjective Well-Being in Times of Crises: Evidence on the Wider Impact of Economic Crises and Turmoil on Subjective Well-Being*, Promotors: Prof. H.R. Commandeur, Prof. F. van Oort & Dr. M.J. Burger, EPS-2018-459-S&E, repub.eur.nl/pub/111830

Avci, E., *Surveillance of Complex Auction Markets: a Market Policy Analytics Approach*, Promotors: Prof. W. Ketter, Prof. H.W.G.M. van Heck & Prof. D.W. Bunn, EPS-2018-426-LIS, repub.eur.nl/pub/106286

Balen, T.H. van, *Challenges of Early Stage Entrepreneurs : the Roles of Vision Communication and Team Membership Change*, Promotors: Prof. J.C.M van den Ende & Dr. M. Tarakci, EPS-2018-468-LIS, repub.eur.nl/pub/115654

Benschop, N, *Biases in Project Escalation: Names, frames & construal levels*, Promotors: Prof. K.I.M. Rhode, Prof. H.R. Commandeur, Prof. M. Keil & Dr. A.L.P. Nuijten, EPS-2015-375-S&E, repub.eur.nl/pub/79408

Bernoster, I., *Essays at the Intersection of Psychology, Biology, and Entrepreneurship*, Promotors: Prof. A.R. Thurik, Prof. I.H.A. Franken & Prof. P.J.F Groenen, EPS-2018-463-S&E, repub.eur.nl/pub/113907

Beusichem, H.C. van, *Firms and Financial Markets: Empirical Studies on the Informational Value of Dividends, Governance and Financial Reporting*, Promotors: Prof. A. de Jong & Dr. G. Westerhuis, EPS-2016-378-F&A, repub.eur.nl/pub/93079

Bouman, P., *Passengers, Crowding and Complexity: Models for Passenger Oriented Public Transport*, Promotors: Prof. L.G. Kroon, Prof. A. Schöbel & Prof. P.H.M. Vervest, EPS-2017-420-LIS, repub.eur.nl/pub/100767

Bunderen, L. van, *Tug-of-War: Why and when teams get embroiled in power struggles*, Promotors: Prof. D.L. van Knippenberg & Dr. L. Greer, EPS-2018-446-ORG, repub.eur.nl/pub/105346

Burg, G.J.J. van den, *Algorithms for Multiclass Classification and Regularized Regression*, Promotors: Prof. P.J.F. Groenen & Dr. A. Alfons, EPS-2018-442-MKT, repub.eur.nl/pub/103929

Chammas, G., *Portfolio concentration*, Promotor: Prof. J. Spronk, EPS-2017-410-F&E, repub.eur.nl/pub/94975

Consiglio, I., *Others: Essays on Interpersonal and Consumer Behavior*, Promotor: Prof. S.M.J. van Osselaer, EPS-2016-366-MKT, repub.eur.nl/pub/79820

Cranenburgh, K.C. van, *Money or Ethics: Multinational corporations and religious organisations operating in an era of corporate responsibility*, Promotors: Prof. L.C.P.M. Meijs, Prof. R.J.M. van Tulder & Dr. D. Arenas, EPS-2016-385-ORG, repub.eur.nl/pub/93104

Darnihamedani, P., *Individual Characteristics, Contextual Factors and Entrepreneurial Behavior*, Promotors: Prof. A.R. Thurik & S.J.A. Hessels, EPS-2016-360-S&E, repub.eur.nl/pub/93280

Dennerlein, T., *Empowering Leadership and Employees' Achievement Motivations: the Role of Self-Efficacy and Goal Orientations in the Empowering Leadership Process*, Promotors: Prof. D.L. van Knippenberg & Dr. J. Dietz, EPS-2017-414-ORG, repub.eur.nl/pub/98438

Depecik, B.E., *Revitalizing brands and brand: Essays on Brand and Brand Portfolio Management Strategies*, Promotors: Prof. G.H. van Bruggen, Dr. Y.M. van Everdingen and Dr. M.B. Ataman, EPS-2016-406-MKT, repub.eur.nl/pub/93507

Duijzer, L.E., *Mathematical Optimization in Vaccine Allocation*, Promotors: Prof. R. Dekker & Dr. W.L. van Jaarsveld, EPS-2017-430-LIS, repub.eur.nl/pub/101487

Duyvesteyn, J.G., *Empirical Studies on Sovereign Fixed Income Markets*, Promotors: Prof. P. Verwijmeren & Prof. M.P.E. Martens, EPS-2015-361-F&A, repub.eur.nl/pub/79033

El Nayal, O.S.A.N., *Firms and the State: An Examination of Corporate Political Activity and the Business-Government Interface*, Promotor: Prof. J. van Oosterhout & Dr. M. van Essen, EPS-2018-469-S&E, repub.eur.nl/pub/114683

Elemes, A., *Studies on Determinants and Consequences of Financial Reporting Quality*, Promotor: Prof. E. Peek, EPS-2015-354-F&A, repub.eur.nl/pub/79037

Erlemann, C., *Gender and Leadership Aspiration: The Impact of the Organizational Environment*, Promotor: Prof. D.L. van Knippenberg, EPS-2016-376-ORG, repub.eur.nl/pub/79409

Faber, N., *Structuring Warehouse Management*, Promotors: Prof. M.B.M. de Koster & Prof. A. Smidts, EPS-2015-336-LIS, repub.eur.nl/pub/78603

Feng, Y., *The Effectiveness of Corporate Governance Mechanisms and Leadership Structure: Impacts on strategic change and firm performance*, Promotors: Prof. F.A.J. van den Bosch, Prof. H.W. Volberda & Dr. J.S. Sidhu, EPS-2017-389-S&E, repub.eur.nl/pub/98470

Fernald, K., *The Waves of Biotechnological Innovation in Medicine: Interfirm Cooperation Effects and a Venture Capital Perspective*, Promotors: Prof. E. Claassen, Prof. H.P.G. Pennings & Prof. H.R. Commandeur, EPS-2015-371-S&E, repub.eur.nl/pub/79120

Fisch, C.O., *Patents and trademarks: Motivations, antecedents, and value in industrialized and emerging markets*, Promotors: Prof. J.H. Block, Prof. H.P.G. Pennings & Prof. A.R. Thurik, EPS-2016-397-S&E, repub.eur.nl/pub/94036

Fliers, P.T., *Essays on Financing and Performance: The role of firms, banks and board*, Promotors: Prof. A. de Jong & Prof. P.G.J. Roosenboom, EPS-2016-388-F&A, repub.eur.nl/pub/93019

Frick, T.W., *The Implications of Advertising Personalization for Firms, Consumer, and Ad Platfroms*, Promotors: Prof. T. Li & Prof. H.W.G.M. van Heck, EPS-2018-452-LIS, repub.eur.nl/pub/110314

Fytraki, A.T., *Behavioral Effects in Consumer Evaluations of Recommendation Systems*, Promotors: Prof. B.G.C. Dellaert & Prof. T. Li, EPS-2018-427-MKT, repub.eur.nl/pub/110457

Gaast, J.P. van der, *Stochastic Models for Order Picking Systems*, Promotors: Prof. M.B.M de Koster & Prof. I.J.B.F. Adan, EPS-2016-398-LIS, repub.eur.nl/pub/93222

Ghazizadeh, P., *Empirical Studies on the Role of Financial Information in Asset and Capital Markets*, Promotors: Prof. A. de Jong & Prof. E. Peek. EPS-2019-470-F&A repub.eur.nl/pub/114023

Giurge, L., *A Test of Time; A temporal and dynamic approach to power and ethics*, Promotors: Prof. M.H. van Dijke & Prof. D. De Cremer, EPS-2017-412-ORG, repub.eur.nl/pub/98451

Gobena, L., *Towards Integrating Antecedents of Voluntary Tax Compliance*, Promotors: Prof. M.H. van Dijke & Dr. P. Verboon, EPS-2017-436-ORG, repub.eur.nl/pub/103276

Groot, W.A., *Assessing Asset Pricing Anomalies*, Promotors: Prof. M.J.C.M. Verbeek & Prof. J.H. van Binsbergen, EPS-2017-437-F&A, repub.eur.nl/pub/103490

Hanselaar, R.M., *Raising Capital: On pricing, liquidity and incentives*, Promotors: Prof. M.A. van Dijk & Prof. P.G.J. Roosenboom, EPS-2018-429-F&A-9789058925404, repub.eur.nl/pub/113274

Harms, J. A., *Essays on the Behavioral Economics of Social Preferences and Bounded Rationality*, Promotors: Prof. H.R. Commandeur & Dr. K.E.H. Maas, EPS-2018-457-S&E, repub.eur.nl/pub/108831

Hekimoglu, M., *Spare Parts Management of Aging Capital Products*, Promotor: Prof. R. Dekker, EPS-2015-368-LIS, repub.eur.nl/pub/79092

Hendriks, G., *Multinational Enterprises and Limits to International Growth: Links between Domestic and Foreign Activities in a Firm's Portfolio*, Promotors: Prof. P.P.M.A.R. Heugens & Dr. A.H.L Slangen, EPS-2019-464-S&E, repub.eur.nl/pub/114981

Hengelaar, G.A., *The Proactive Incumbent: Holy grail or hidden gem? Investigating whether the Dutch electricity sector can overcome the incumbent's curse and lead the sustainability transition*, Promotors: Prof. R.J. M. van Tulder & Dr. K. Dittrich, EPS-2018-438-ORG, repub.eur.nl/pub/102953

Hogenboom, A.C., *Sentiment Analysis of Text Guided by Semantics and Structure*, Promotors: Prof. U. Kaymak & Prof. F.M.G. de Jong, EPS-2015-369-LIS, repub.eur.nl/pub/79034

Hollen, R.M.A., *Exploratory Studies into Strategies to Enhance Innovation-Driven International Competitiveness in a Port Context: Toward Ambidextrous Ports*, Promotors: Prof. F.A.J. Van Den Bosch & Prof. H.W. Volberda, EPS-2015-372-S&E, repub.eur.nl/pub/78881

Jacobs, B.J.D., *Marketing Analytics for High-Dimensional Assortments*, Promotors: Prof. A.C.D. Donkers & Prof. D. Fok, EPS-2017-445-MKT, repub.eur.nl/pub/103497

Jia, F., *The Value of Happiness in Entrepreneurship*, Promotors: Prof. D.L. van Knippenberg & Dr. Y. Zhang, EPS-2019-479-ORG, repub.eur.nl/pub/115990

Kahlen, M. T., *Virtual Power Plants of Electric Vehicles in Sustainable Smart Electricity Markets*, Promotors: Prof. W. Ketter & Prof. A. Gupta, EPS-2017-431-LIS, repub.eur.nl/pub/100844

Kampen, S. van, *The Cross-sectional and Time-series Dynamics of Corporate Finance: Empirical evidence from financially constrained firms*, Promotors: Prof. L. Norden & Prof. P.G.J. Roosenboom, EPS-2018-440-F&A, repub.eur.nl/pub/105245

Karali, E., *Investigating Routines and Dynamic Capabilities for Change and Innovation*, Promotors: Prof. H.W. Volberda, Prof. H.R. Commandeur & Dr. J.S. Sidhu, EPS-2018-454-S&E, repub.eur.nl/pub/106274

Keko, E., *Essays on Innovation Generation in Incumbent Firms*, Promotors: Prof. S. Stremersch & Dr. N.M.A. Camacho, EPS-2017-419-MKT, repub.eur.nl/pub/100841

Kerkkamp, R.B.O., *Optimisation Models for Supply Chain Coordination under Information Asymmetry*, Promotors: Prof. A.P.M. Wagelmans & Dr. W. van den Heuvel, EPS-2018-462-LIS, repub.eur.nl/pub/109770

Khattab, J., *Make Minorities Great Again: a contribution to workplace equity by identifying and addressing constraints and privileges*, Promotors: Prof. D.L. van Knippenberg & Dr. A. Nederveen Pieterse, EPS-2017-421-ORG, repub.eur.nl/pub/99311

Kim, T. Y., *Data-driven Warehouse Management in Global Supply Chains*, Promotors: Prof. R. Dekker & Dr. C. Heij, EPS-2018-449-LIS, repub.eur.nl/pub/109103

Klitsie, E.J., *Strategic Renewal in Institutional Contexts: The paradox of embedded agency*, Promotors: Prof. H.W. Volberda & Dr. S. Ansari, EPS-2018-444-S&E, repub.eur.nl/pub/106275

Kong, L., *Essays on Financial Coordination*, Promotors: Prof. M.J.C.M. Verbeek, Dr. D.G.J. Bongaerts & Dr. M.A. van Achter. EPS-2019-433-F&A, repub.eur.nl/pub/114516

Koolen, D., *Market Risks and Strategies in Power Systems Integrating Renewable Energy*, Promotors: Prof. W. Ketter & Dr. R. Huisman, EPS-2019-467-LIS, repub.eur.nl/pub/115655

Krämer, R., *A license to mine? Community organizing against multinational corporations*, Promotors: Prof. R.J.M. van Tulder & Prof. G.M. Whiteman, EPS-2016-383-ORG, repub.eur.nl/pub/94072

Kyosev, G.S., *Essays on Factor Investing*, Promotors: Prof. M.J.C.M. Verbeek & Dr. J.J. Huij, EPS-2019-474-F&A, repub.eur.nl/pub/116463

Lamballais, T., *Optimizing the Performance of Robotic Mobile Fulfillment Systems*, Promotors: Prof. M.B.M de Koster & Prof. R. Dekker & Dr. D. Roy, EPS-2019-411-LIS, repub.eur.nl/pub/116477

Lee, C.I.S.G., *Big Data in Management Research: Exploring New Avenues*, Promotors: Prof. S.J. Magala & Dr. W.A. Felps, EPS-2016-365-ORG, repub.eur.nl/pub/79818

Legault-Tremblay, P.O., *Corporate Governance During Market Transition: Heterogeneous responses to Institution Tensions in China*, Promotor: Prof. B. Krug, EPS-2015-362-ORG, repub.eur.nl/pub/78649

Lenoir, A.S., *Are You Talking to Me? Addressing Consumers in a Globalised World*, Promotors: Prof. S. Puntoni & Prof. S.M.J. van Osselaer, EPS-2015-363-MKT, repub.eur.nl/pub/79036

Leung, W.L., *How Technology Shapes Consumption: Implications for Identity and Judgement*, Promotors: Prof. S. Puntoni & Dr. G. Paolacci, EPS-2019-485-MKT, repub.eur.nl/pub/117432

Li, D., *Supply Chain Contracting for After-sales Service and Product Support*, Promotor: Prof. M.B.M. de Koster, EPS-2015-347-LIS, repub.eur.nl/pub/78526

Li, X., *Dynamic Decision Making under Supply Chain Competition*, Promotors: Prof. M.B.M de Koster, Prof. R. Dekker & Prof. R. Zuidwijk. EPS-2018-466-LIS, repub.eur.nl/pub/114028

Liu, N., *Behavioral Biases in Interpersonal Contexts*, Promotors: Prof. A. Baillon & Prof. H. Bleichrodt, EPS-2017-408-MKT, repub.eur.nl/pub/95487

Ma, Y., *The Use of Advanced Transportation Monitoring Data for Official Statistics*, Promotors: Prof. L.G. Kroon & Dr. J. van Dalen, EPS-2016-391-LIS, repub.eur.nl/pub/80174

Maas, A.J.J., *Organizations and their external context: Impressions across time and space*, Promotors: Prof. P.P.M.A.R. Heugens & Prof. T.H. Reus, EPS-2019-478-S&E, repub.eur.nl/pub/116480

Maira, E., *Consumers and Producers*, Promotors: Prof. S. Puntoni & Prof. C. Fuchs, EPS-2018-439-MKT, repub.eur.nl/pub/104387

Mell, J.N., *Connecting Minds: On The Role of Metaknowledge in Knowledge Coordination*, Promotor: Prof. D.L. van Knippenberg, EPS-2015-359-ORG, repub.eur.nl/pub/78951

Meulen, D. van der, *The Distance Dilemma: the effect of flexible working practices on performance in the digital workplace*, Promotors: Prof. H.W.G.M. van Heck & Prof. P.J. van Baalen, EPS-2016-403-LIS, repub.eur.nl/pub/94033

Moniz, A, *Textual Analysis of Intangible Information*, Promotors: Prof. C.B.M. van Riel, Prof. F.M.G de Jong & Dr. G.A.J.M. Berens, EPS-2016-393-ORG, repub.eur.nl/pub/93001

Mulder, J., *Network design and robust scheduling in liner shipping*, Promotors: Prof. R. Dekker & Dr. W.L. van Jaarsveld, EPS-2016-384-LIS, repub.eur.nl/pub/80258

Neerijnen, P., *The Adaptive Organization: the socio-cognitive antecedents of ambidexterity and individual exploration*, Promotors: Prof. J.J.P. Jansen, P.P.M.A.R. Heugens & Dr. T.J.M. Mom, EPS-2016-358-S&E, repub.eur.nl/pub/93274

Okbay, A., *Essays on Genetics and the Social Sciences*, Promotors: Prof. A.R. Thurik, Prof. Ph.D. Koellinger & Prof. P.J.F. Groenen, EPS-2017-413-S&E, repub.eur.nl/pub/95489

Oord, J.A. van, *Essays on Momentum Strategies in Finance*, Promotor: Prof. H.K. van Dijk, EPS-2016-380-F&A, repub.eur.nl/pub/80036

Peng, X., *Innovation, Member Sorting, and Evaluation of Agricultural Cooperatives*, Promotor: Prof. G.W.J. Hendriks, EPS-2017-409-ORG, repub.eur.nl/pub/94976

Pennings, C.L.P., *Advancements in Demand Forecasting: Methods and Behavior*, Promotors: Prof. L.G. Kroon, Prof. H.W.G.M. van Heck & Dr. J. van Dalen, EPS-2016-400-LIS, repub.eur.nl/pub/94039

Petruchenya, A., *Essays on Cooperatives: Emergence, Retained Earnings, and Market Shares*, Promotors: Prof. G.W.J. Hendriks & Dr. Y. Zhang, EPS-2018-447-ORG, repub.eur.nl/pub/105243

Plessis, C. du, *Influencers: The Role of Social Influence in Marketing*, Promotors: Prof. S. Puntoni & Prof. S.T.L.R. Sweldens, EPS-2017-425-MKT, repub.eur.nl/pub/103265

Pocock, M., *Status Inequalities in Business Exchange Relations in Luxury Markets*, Promotors: Prof. C.B.M. van Riel & Dr. G.A.J.M. Berens, EPS-2017-346-ORG, repub.eur.nl/pub/98647

Pozharliev, R., *Social Neuromarketing: The role of social context in measuring advertising effectiveness*, Promotors: Prof. W.J.M.I. Verbeke & Prof. J.W. van Strien, EPS-2017-402-MKT, repub.eur.nl/pub/95528

Protzner, S., *Mind the gap between demand and supply: A behavioral perspective on demand forecasting*, Promotors: Prof. S.L. van de Velde & Dr. L. Rook, EPS-2015-364-LIS, repub.eur.nl/pub/79355

Reh, S.G., *A Temporal Perspective on Social Comparisons in Organizations*, Promotors: Prof. S.R. Giessner, Prof. N. van Quaquebeke & Dr. C. Troster, EPS-2018-471-ORG, repub.eur.nl/pub/114522

Riessen, B. van, *Optimal Transportation Plans and Portfolios for Synchromodal Container Networks*, Promotors: Prof. R. Dekker & Prof. R.R. Negenborn, EPS-2018-448-LIS, repub.eur.nl/pub/105248

Rietdijk, W.J.R., *The Use of Cognitive Factors for Explaining Entrepreneurship*, Promotors: Prof. A.R. Thurik & Prof. I.H.A. Franken, EPS-2015-356-S&E, repub.eur.nl/pub/79817

Roza, L., *Employee Engagement in Corporate Social Responsibility: A collection of essays*, Promotor: Prof. L.C.P.M. Meijs, EPS-2016-396-ORG, repub.eur.nl/pub/93254

Rösch, D., *Market Efficiency and Liquidity*, Promotor: Prof. M.A. van Dijk, EPS-2015-353-F&A, repub.eur.nl/pub/79121

Schie, R. J. G. van, *Planning for Retirement: Save More or Retire Later?*, Promotors: Prof. B. G. C. Dellaert & Prof. A.C.D. Donkers, EOS-2017-415-MKT, repub.eur.nl/pub/100846

Schoonees, P., *Methods for Modelling Response Styles*, Promotor: Prof. P.J.F. Groenen, EPS-2015-348-MKT, repub.eur.nl/pub/79327

Schouten, K.I.M., *Semantics-driven Aspect-based Sentiment Analysis*, Promotors: Prof. F.M.G. de Jong, Prof. R. Dekker & Dr. F. Frasincar, EPS-2018-453-LIS, repub.eur.nl/pub/112161

Schouten, M.E., *The Ups and Downs of Hierarchy: the causes and consequences of hierarchy struggles and positional loss*, Promotors; Prof. D.L. van Knippenberg & Dr. L.L. Greer, EPS-2016-386-ORG, repub.eur.nl/pub/80059

Sihag, V., *The Effectiveness of Organizational Controls: A meta-analytic review and an investigation in NPD outsourcing*, Promotors: Prof. J.C.M van den Ende & Dr. S.A. Rijsdijk, EPS-2019-476-LIS, repub.eur.nl/pub/115931

Smit, J., *Unlocking Business Model Innovation: A look through the keyhole at the inner workings of Business Model Innovation*, Promotor: Prof. H.G. Barkema, EPS-2016-399-S&E, repub.eur.nl/pub/93211

Straeter, L.M., *Interpersonal Consumer Decision Making*, Promotors: Prof. S.M.J. van Osselaer & Dr. I.E. de Hooge, EPS-2017-423-MKT, repub.eur.nl/pub/100819

Stuppy, A., *Essays on Product Quality*, Promotors: Prof. S.M.J. van Osselaer & Dr. N.L. Mead. EPS-2018-461-MKT, repub.eur.nl/pub/111375

Subaşi, B., *Demographic Dissimilarity, Information Access and Individual Performance*, Promotors: Prof. D.L. van Knippenberg & Dr. W.P. van Ginkel, EPS-2017-422-ORG, repub.eur.nl/pub/103495

Suurmond, R., *In Pursuit of Supplier Knowledge: Leveraging capabilities and dividing responsibilities in product and service contexts*, Promotors: Prof. J.Y.F Wynstra & Prof. J. Dul. EPS-2018-475-LIS, repub.eur.nl/pub/115138

Szatmari, B., *We are (all) the champions: The effect of status in the implementation of innovations*, Promotors: Prof. J.C.M van den Ende & Dr. D. Deichmann, EPS-2016-401-LIS, repub.eur.nl/pub/94633

Toxopeus, H.S., *Financing sustainable innovation: From a principal-agent to a collective action perspective*, Promotors: Prof. H.R. Commandeur & Dr. K.E.H. Maas. EPS-2019-458-S&E, repub.eur.nl/pub/114018

Turturea, R., *Overcoming Resource Constraints: The Role of Creative Resourcing and Equity Crowdfunding in Financing Entrepreneurial Ventures*, Promotors: Prof. P.P.M.A.R Heugens, Prof. J.J.P. Jansen & Dr. I. Verheuil, EPS-2019-472-S&E, repub.eur.nl/pub/112859

Valogianni, K., *Sustainable Electric Vehicle Management using Coordinated Machine Learning*, Promotors: Prof. H.W.G.M. van Heck & Prof. W. Ketter, EPS-2016-387-LIS, repub.eur.nl/pub/93018

Vandic, D., *Intelligent Information Systems for Web Product Search*, Promotors: Prof. U. Kaymak & Dr. Frasincar, EPS-2017-405-LIS, repub.eur.nl/pub/95490

Verbeek, R.W.M., *Essays on Empirical Asset Pricing*, Promotors: Prof. M.A. van Dijk & Dr. M. Szymanowska, EPS-2017-441-F&A, repub.eur.nl/pub/102977

Vermeer, W., *Propagation in Networks:The impact of information processing at the actor level on system-wide propagation dynamics*, Promotor: Prof. P.H.M.Vervest, EPS-2015-373-LIS, repub.eur.nl/pub/79325

Versluis, I., *Prevention of the Portion Size Effect*, Promotors: Prof. Ph.H.B.F. Franses & Dr. E.K. Papies, EPS-2016-382-MKT, repub.eur.nl/pub/79880

Vishwanathan, P., *Governing for Stakeholders: How Organizations May Create or Destroy Value for their Stakeholders*, Promotors: Prof. J. van Oosterhout & Prof. L.C.P.M. Meijs, EPS-2016-377-ORG, repub.eur.nl/pub/93016

Vlaming, R. de, *Linear Mixed Models in Statistical Genetics*, Promotors: Prof. A.R. Thurik, Prof. P.J.F. Groenen & Prof. Ph.D. Koellinger, EPS-2017-416-S&E, repub.eur.nl/pub/100428

Vries, H. de, *Evidence-Based Optimization in Humanitarian Logistics*, Promotors: Prof. A.P.M. Wagelmans & Prof. J.J. van de Klundert, EPS-2017-435-LIS, repub.eur.nl/pub/102771

Vries, J. de, *Behavioral Operations in Logistics*, Promotors: Prof. M.B.M de Koster & Prof. D.A. Stam, EPS-2015-374-LIS, repub.eur.nl/pub/79705

Wagenaar, J.C., *Practice Oriented Algorithmic Disruption Management in Passenger Railways*, Promotors: Prof. L.G. Kroon & Prof. A.P.M. Wagelmans, EPS-2016-390-LIS, repub.eur.nl/pub/93177

Wang, P., *Innovations, status, and networks*, Promotors: Prof. J.J.P. Jansen & Dr. V.J.A. van de Vrande, EPS-2016-381-S&E, repub.eur.nl/pub/93176

Wang, R., *Corporate Environmentalism in China*, Promotors: Prof. P.P.M.A.R Heugens & Dr. F. Wijen, EPS-2017-417-S&E, repub.eur.nl/pub/99987

Wasesa, M., *Agent-based inter-organizational systems in advanced logistics operations*, Promotors: Prof. H.W.G.M van Heck, Prof. R.A. Zuidwijk & Dr. A. W. Stam, EPS-2017-LIS-424, repub.eur.nl/pub/100527

Wessels, C., *Flexible Working Practices: How Employees Can Reap the Benefits for Engagement and Performance*, Promotors: Prof. H.W.G.M. van Heck, Prof. P.J. van Baalen & Prof. M.C. Schippers, EPS-2017-418-LIS, repub.eur.nl/pub/99312

Wiegmann, P.M., *Setting the Stage for Innovation: Balancing Diverse Interests through Standardisation*, Promotors: Prof. H.J. de Vries & Dr. K. Blind, EPS-2019-473-LIS, repub.eur.nl/pub/114519

Wijaya, H.R., *Praise the Lord! : Infusing Values and Emotions into Neo-Institutional Theory*, Promotors: Prof. P.P.M.A.R. Heugens & Prof. J.P. Cornelissen, EPS-2019-450-S&E, repub.eur.nl/pub/115973

Williams, A.N., *Make Our Planet Great Again: A Systems Perspective of Corporate Sustainability*, Promotors: Prof. G.M. Whiteman & Dr. S. Kennedy, EPS-2018-456-ORG, repub.eur.nl/pub/111032

Witte, C.T., *Bloody Business: Multinational investment in an increasingly conflict-afflicted world*, Promotors: Prof. H.P.G. Pennings, Prof. H.R. Commandeur & Dr. M.J. Burger, EPS-2018-443-S&E, repub.eur.nl/pub/104027

Ye, Q.C., *Multi-objective Optimization Methods for Allocation and Prediction*, Promotors: Prof. R. Dekker & Dr. Y. Zhang, EPS-2019-460-LIS, repub.eur.nl/pub/116462

Ypsilantis, P., *The Design, Planning and Execution of Sustainable Intermodal Port-hinterland Transport Networks*, Promotors: Prof. R.A. Zuidwijk & Prof. L.G. Kroon, EPS-2016-395-LIS, repub.eur.nl/pub/94375

Yuan, Y., *The Emergence of Team Creativity: a social network perspective*, Promotors: Prof. D. L. van Knippenberg & Dr. D. A. Stam, EPS-2017-434-ORG, repub.eur.nl/pub/100847

Yuferova, D., *Price Discovery, Liquidity Provision, and Low-Latency Trading*, Promotors: Prof. M.A. van Dijk & Dr. D.G.J. Bongaerts, EPS-2016-379-F&A, repub.eur.nl/pub/93017

Zhang, Q., *Financing and Regulatory Frictions in Mergers and Acquisitions*, Promotors: Prof. P.G.J. Roosenboom & Prof. A. de Jong, EPS-2018-428-F&A, repub.eur.nl/pub/103871

Zuber, F.B., *Looking at the Others: Studies on (un)ethical behavior and social relationships in organizations*, Promotor: Prof. S.P. Kaptein, EPS-2016-394-ORG, repub.eur.nl/pub/94388