New variants of the time-dependent vehicle routing problem with time windows

New Variants of the Time-dependent Vehicle Routing Problem with Time Windows

PENG SUN

New Variants of the Time-dependent Vehicle Routing Problem with Time Windows

This thesis is part of the Ph.D. thesis series of the Beta Research School for Operations Management and Logistics (onderzoeksschool–beta.nl) in which the following universities cooperate: Eindhoven University of Technology, Maastricht University, University of Twente, VU Amsterdam, Wageningen University and Research, and KU Leuven.

A catalogue record is available from the Eindhoven University of Technology Library.

ISBN:	978-90-386-5170-5
Author:	Peng Sun
Cover Design:	Mercedes Benjaminse
Printing:	Proefschriftmaken.nl

This research has been funded by the China Scholarship Council (CSC).

New Variants of the Time-dependent Vehicle Routing Problem with Time Windows

PROEFSCHRIFT

ter verkrijging van de graad van doctor aan de
Technische Universiteit Eindhoven, op gezag van de
rector magnificus, prof.dr.ir. F.P.T. Baaijens, voor een
commissie aangewezen door het College voor
Promoties, in het openbaar te verdedigen
op donderdag 03 december 2020 om 13:30 uur

door

Peng Sun

geboren te Hunan, China

Dit proefschrift is goedgekeurd door de promotoren en de samenstelling van de promotiecommissie is als volgt:

voorzitter:	prof.dr. I.E.J. Heynderickx
1e promotor:	prof.dr. T. van Woensel
co–promotor:	dr. L.P. Veelenturf
leden:	prof.dr. A.G. de Kok
	prof.dr. D. Feillet (Ecole des Mines Saint-Etienne)
	dr.ir. P. Vansteenwegen (KU Leuven)
	dr. M. Hewitt (Loyola University Chicago)
reserve lid:	dr.ir. N.P. Dellaert

Het onderzoek of ontwerp dat in dit proefschrift wordt beschreven is uitgevoerd in overeenstemming met de TU/e Gedragscode Wetenschapsbeoefening.

Acknowledgments

When I was a schoolchild, my interest in operation research began when I read a short story by a tea loving mathematician Luogeng Hua. I was intrigued to learn that you can save time by preparing the cups and tea leaves while the kettle is still on the burner. Then when the water has boiled, you can immediately brew your tea. Apparently, doing a PhD is far more difficult than preparing tea. At the start of my PhD journey in 2012, I would have not imagined it would take me eight years to complete this journey, which was filled with struggle, frustration, joy, and cheers. At this moment, I would like to express my sincere appreciation and deepest gratitude to those who assisted me in a myriad of ways.

First of all, I would like to thank my first promotor Tom van Woensel for his supervision and support during the past years. Thank you for providing me this great opportunity to work with you and opening this amazing world of optimization and operation research to me. I also appreciate your countless words of encouragement, knowledge and experience sharing, and valuable feedback on how to improve my skills especially in writing and time management. I am very grateful to have an easy-going and humorous first promotor. I would not have been able to finish this journey without your support and patience.

I was very lucky to have the chance to be the first PhD student supervised by Luuk Veelenturf. Luuk is not only a great mentor but also a good friend. Thank you for your great patience and endless support during my long PhD journey. You also spent a lot of time revising our papers, my thesis, our response letters, and so on. This thesis and those publications would not exist without your great efforts and ideas. Besides, I really enjoyed that, whenever I had some ideas to discuss with you, I could easily reach you either by knocking on your office door when I was still in TU/e or by Skype/WeChat when I moved to the Germany.

During my PhD I had the great honour to work with Mike Hewitt, who also hosted me during my three-month research visit in the Quinlan School of Business at Loyola University Chicago, USA. I am amazed with your knowledge and skills in optimization and operation research. I am also grateful to you for teaching me how to solve the optimization problem effectively and providing me with motivating thoughts and interesting further directions. Thank you for your great guidance and ideas and for sharing your knowledge with me. Without you, Chapter 3 of this thesis would not be able to be published in a top-level journal.

I would also like to thank Said Dabia. You supervised me for the first two years of my PhD

project and guided my entrance into this time-dependent world. It was also the toughest two years of my PhD journey. You suggested that I follow several optimization-related courses and patiently taught me how to appropriately formulate a problem and improve my coding skills. I am also grateful to you for your kindness in sharing parts of your exact algorithm code, which facilitated my being able to grasp the main ideas of the entire algorithm framework. Without you, Chapter 2 of this thesis would not have been as good as it is.

Next, I would like to thank the China Scholarship Council for sponsoring this research. Special thanks go to Asvin Goel, who hosted me during my one-year post-doc research in Kühne Logistics University, Hamburg, Germany. Also thank you for your valuable comments, which greatly improved Chapter 4 of this thesis.

Furthermore, I would like to thank Ton de Kok, Dominique Feillet, Pieter Vansteenwegen and Nico Dellaert for being part of my doctoral committee to read and assess my work. I highly appreciate your constructive comments and valuable suggestions.

In addition, I would like to thank all my colleagues in the OPAC group. A special word to Veaceslav Ghilas: As my both first and longest officemate, I truly enjoyed the time and all the interesting talks with you. In addition, I really enjoyed our live Jazz trip to New Orleans. Loe Schlicher, my second officemate: Thank you for providing me with the opportunity to get a taste of your research and sharing your life stories. Your enthusiasm on Mandarine learning is quite impressive. Farin Dashty Saridarq, Afonso Sampaio Oliveira, and Vincent Karels, my roommates in fifth year, thank you all for our chats about research. Big thanks to José van Dijk, Jolanda Verkuijlen, and Claudine Hulsman for your administrative assistance and kind support. I would like to thank Marco Slikker; it was a nice experience to work with you as the teaching assistant for your 'Introduction to Financial and Management Accounting' course for three years. I am also grateful to Maryam Steadie Seifi, Chiel van Oosterom, Emrah Demir, Taimaz Soltani, Taher Ahmadi, Zumbul Atan, and Engin Topan for the great times we had had during coffee breaks and lunch.

I am grateful to all my Chinese colleagues. Especially, I would like to thank Qianru Ge, who picked me up from the train station when I arrived in Eindhoven for the first time and for everything that followed. The same to Baoxiang Li, who provided me with a lot of help and support in research and studies. In addition, I am also grateful to Guanlian Xiao, Qiushi Zhu, Lei Jing, Yandong He, Xu Zhao, Lijia Tan, Yanfei Huang, and OPAC visiting students Jincheng Jiang, Tao Lu, Yixiao Huang, and Ruidian Song for all the good times spent together. All your

warm company made me feel like I was at home.

Finally, I would like to thank my parents for their unconditional support. My father was diagnosed with cancer of the tongue base just a few months before I started my PhD. They kept this news from me until I decided to take this great adventure. Thank you mom for helping me take care of my dad. When you both were bravely fighting against the disease, I was fighting for my PhD degree. Your spirit and optimism always cheer me up in my difficult moments. Now, you have won the battle against the disease, and I am in the last miles of my PhD journey. Last, my utmost thanks go to my girlfriend Li Wang for her constant support and love, especially during the final stages of my PhD journey. Her care is the most precious thing in my life.

Peng Sun

Berlin, September 2020

Contents

Chapter 1

Introduction

"The desire to write grows with writing."

Desiderius Erasmus

Nowadays, more people are living in cities than ever before. After decades of urban growth, urban populations have been continuously growing in both developed and developing countries. According to the United Nations, in 2018, 55% of the world's population was living in cities. Projections show that the urban share of the worldwide population will increase to 68% by 2050 (see UN.org [113]). Meanwhile, after years of profound technological change, an increasing number of urban individuals are immersed in ever-connected mobile devices and curated digital environments. Over the past decade, more people, especially younger generations, preferred to purchase goods and services using their mobile phones, which is convenient and fast. This trend led to the booming of e-commerce. In 2017, e-commerce sales jumped 24.8 percent to 2.3 trillion dollars and are projected to continue growing for the foreseeable future (eMarketer [37]). According to Internet Retailer's analysis (Young [130]), e-commerce made up 14.9% of total retail sales in 2019 in the United States.

Both trends bring more challenges to logistics systems. The transport demand for freight and passenger transport in the cities is expected to grow dramatically in the coming decades to serve the growing urban population and economy. As an example, the average number of daily deliveries to households in New York City tripled to more than 1.1 million shipments from 2009 to 2017 (Haag and Hu [51]). As another example, in 2017 the total freight transport volume within the European Union equaled 3616.6 billion tonne-kilometres, and the total movement of people within the European Union reached a new all-time high (6912.7 billion passenger-kilometres; see Statistical Pocketbook [105]). Both measures were 30 % higher than in 1995. At the same time, urbanization and e-commerce also directly changed the consumer expectations, as consumers now expect fast, free shipping and competitive pricing.

As customer expectations increase, so too will traffic congestion in cities. The rising traffic congestion becomes an inescapable phenomenon in large and growing metropolitan areas across the world. According to the annual summary by motoring organisation Royal Dutch Touring Club (ANWB), the Dutch roads saw a 17% increase in the volume of traffic jams in 2019 (Verhoeven [119]). At the same time, consumer expectations for faster delivery times are growing rapidly, especially as overnight and same-day delivery become more popular. This trend could lead to more traffic bottlenecks and capacity problems.

This all made transportation costs rise for the past few years, despite declining oil prices (Joo et al. [59]). Transportation costs are known to be one of the largest expenses for many logistics companies because transportation is a key logistics activity. In 2019, Amazon's transportation costs climbed to 37.9 billion dollars, which surged by 36.6% compared to the previous year (Clement [19]). Furthermore, last-mile logistics, more specifically short-haul transportation, causes disproportionately high costs of up to 28% of the total transportation costs [89].

To cope with this reality, innovative and effective transportation solutions need to be proposed and investigated. The efficiency of transport systems depends on the decisions to allocate scarce resources to perform a set of tasks (services). Generally, (short-haul) transportation encompasses different decision levels, such as strategic decisions (e.g., the location of distribution centres), tactical decisions (e.g., the type and size of fleet, public transport schedules), and finally, operational decisions (e.g., the routing and scheduling of the transport operations, crew scheduling).

More specifically, the primary research objective of this thesis is to investigate whether more efficient last-mile logistics can be achieved by taking travel-time fluctuations into account or not. Currently, in many routing applications, the travel time between two customers is assumed to be a constant value, such as a distance metric or an average travel time estimate. However, this assumption is not realistic. Due to the limited capacity of the road network and traffic intensification, the travel speed is seriously affected by the traffic fluctuations, which also results in great variations in travel times and transportation costs.

The rest of this chapter is organized as follows: Section 1.1 discusses the four research questions addressed in this thesis. Section 1.2 provides a brief introduction to the investigated problem. Finally, an overview of the thesis is given in Section 1.3.

1.1 Research questions

To achieve the aforementioned objective, we investigated in this thesis four research projects based on last-mile transportation solutions that consider fluctuations in travel time. More specifically, in the first three projects, we investigated several time-dependent extensions of the classical one-to-one pickup and delivery problem, where each customer request consists of transporting a load from one pickup location to one delivery location. We defined them as a family of time-dependent pickup and delivery problems with time windows. In the last project, we focused on one of the challenges in the laundry business and defined this problem as the time-dependent laundry routing problem with stochastic pickup demand. This problem generalizes the classic vehicle routing problem with simultaneous pickup and delivery in which each customer has commodities to be delivered from the depot and commodities to pick up to bring back to the depot.

Moreover, a set of research questions is formulated and addressed in the corresponding chapters of the thesis. The research questions are as follows:

Research question 1. Which exact and heuristic methods are effective to manage pickup and delivery service in a time-dependent environment?

Fundamentally, the problems considered in this thesis are optimization problems. Many algorithms can be used to solve an optimization problem, which can be classified into two main categories. Some of these algorithms are grouped as exact methods, and others belong to heuristic methods. The main advantage of an exact method is that it is able to produce the optimal solution to a given optimization problem. However, most of the exact methods are highly sophisticated and typically take more time to execute. In contrast, heuristic methods are easier to implement and can offer a quick and good solution, which is practical. The main drawback of this type of method is that its generated solution is not necessarily optimal. Therefore, in the first two projects, we focus on designing efficient exact methods for problems with small and medium-sized instances. The heuristics approaches are considered in the last two projects for problems with large-sized instances.

Research question 2. What are the benefits and advantages of taking fluctuating travel times into account while planning routes in pickup and delivery services?

According to Malandraki and Dial [72], two main components lead to travel-time fluctuations. The first component is associated with random events such as accidents and weather

conditions, which is referred to as stochastic travel time. The second component comprises the road congestion incurred due to daily predictable fluctuations of traffic density, which is referred to as time-dependent travel time. However, according to Barnhart and Laporte [10], the routing problem becomes quite challenging when both time-dependent and stochastic travel times are considered. A possibly exponential number of routes need to be evaluated to ensure an optimal route is found. Moreover, when vehicle arrivals are random but somewhat correlated with time-dependent travel time, the vehicle assignment becomes significantly more complicated (Hickman and Bernstein [54]). Therefore, in this thesis, we consider only time-dependent travel times. Under this setting, the travel time between locations is dependent on the time the driver departs (e.g. in rush hours traveling takes more time); a tour's cost depends on the tour's start time.

Research question 3. What is the value of different degrees of flexibility in pickup and delivery services when fluctuating travel times are considered?

On one hand, managing and owning a private fleet enables companies to control and shape their customers' journey and may lead to cost savings when managed well. However, the cost of recruiting and training enough drivers may not be affordable for smaller companies. On the other hand, outsourcing to a third-party logistics provider can eliminate the need to invest in warehouse space, technology, a delivery fleet, or the manpower needed to handle all the delivery processes. The companies can remain agile and responsive during periods of peak demand, maintaining the quality of their customer experience. However, huge potential risks are also associated with it; for example, the third-party logistics provider may not be able to fulfil all requirements in delivery quality or delivery time. Therefore, one strategy a logistics provider can employ for meeting the increasing demands and expectations is to complement and coordinate its fleet operations with those of for-hire, third-party logistics providers. For instance, e-commerce businesses Style Theory ([85]) and Vipshop.com ([128]) use a hybrid pickup and delivery model that consists of their own fleet and a partnership with third-party logistics providers Pickupp and SF Express, respectively. Therefore, in this thesis, we consider problems in which it can be decided to not serve certain customers (e.g., by outsourcing it to a third-party logistics provider).

Moreover, in some businesses, shift planning needs to be done before the daily vehicle routing starts. In these situations, start times of the routes are fixed. In other businesses the start times of the shifts can be flexible. In this thesis, we investigate the difference between variants where the start time is fixed or where it can be optimized.

Research question 4. Is a pickup and delivery service still efficient and effective when information regarding pickup demand is uncertain during the planning process?

Most research on vehicle routing assumes that all the necessary information is known and available to formulate and solve the problem. However, in practical applications, this assumption usually does not hold. The presence of uncertainty affects various aspects of the problem under study. In general, uncertainty generates a time lag, which separates the moments when a solution is planned and when it is executed. Therefore, in the last research project, we investigate a problem in which the pickup demands are uncertain at the moment of planning, given corresponding demand distributions. A planned route under a deterministic setting may turn out to be infeasible if the total observed demands for the customers scheduled on the route exceed the capacity of the vehicle. When such cases occur, a recourse action with additional costs is needed to produce a feasible solution.

1.2 Conceptual model

One of the most common routing problems for operational decisions in transportation, the vehicle routing problem (VRP), was first introduced by Danzig and Ramser [26]. It aims to construct optimal routes for multiple vehicles starting from a depot and visiting a set of geographically distributed customers. The time-dependent vehicle routing problem with time windows (TDVRPTW) is a generalization of the classical VRP. Generally, the TDVRPTW consists of designing a set of vehicle routes with least route duration cost, which start and end at a depot, to satisfy a set of requests. The travel time between any pair of locations varies over time. Therefore, the TDVRPTW aims to better represent the dynamic nature of the travel time, especially in urban areas where traffic congestion can have a significant impact on logistic operations. On one hand, because of its applicability and inherent complexity, most of the efforts have been spent on (meta-)heuristic algorithms (see, e.g., Donati et al. [33], Hashimoto et al. [52], Ichoua et al. [56], Malandraki and Daskin [71], Malandraki and Dial [72], Van Woensel et al. [114]). Generally, these algorithms trade optimality for computational time. The aim is to generate good-quality solutions within relatively short times. On the other hand, several exact algorithms have been proposed in the literature to optimally solve the TDVRPTW (see, e.g., Dabia et al. [25]). When it is not required to serve all requests and a specific reward is obtained if a request is served, the goal of the logistics provider changes to deciding which subset of requests to serve such that the collected reward is maximized within a maximum allowed travel time. This problem is called the orienteering problem (OP). Compared to the VRP or the OP, the time-dependent OP and its extensions have received very

little attention in the scientific literature (see, e.g., Fomin and Lingas [40], Li [66], Verbeeck et al. [117, 118]). This thesis focuses on solving different variants of time-dependent extensions to the pickup and delivery problem with time windows (PDPTW) and the vehicle routing problem with simultaneous pickup and delivery (VRPSPD).

The PDPTW is a generalization of the classical VRP. The PDPTW consists of designing a set of least cost vehicle routes, which start and end at a depot, to satisfy a set of *pickup and delivery requests*. The considered transportation network consists of depot and request locations. Each request is characterized by an origin location, a destination location, desired time windows for both locations, and demand. Due to time window constraints, the problem of finding a feasible pickup and delivery plan is NP-hard as discussed in Savelsbergh and Sol [96]. The applicability and inherent complexity of the PDPTW lead to an extensive amount of research. Most of the efforts have been spent on heuristic algorithms (see e.g., Nanry and Barnes [79], Røpke and Pisinger [93]). On the other hand, several exact algorithms have been proposed in the literature to optimally solve the PDPTW (see, e.g., Dumas et al. [34], Røpke and Cordeau [91], Røpke et al. [92]).

In this thesis, we first extend the single-vehicle case of the PDPTW by considering time-dependent travel times, and request selection (Chapter 2). Then, a family of time-dependent pickup and delivery problems with time windows (TDPDPTW) under two dimensions of operational flexibility are solved by an exact solution approach (Chapter 3). In Chapter 4, we focus on a heuristic for one variant of TDPDPTW, named as the time-dependent profitable pickup and delivery problem with time windows (TDPPDPTW). It considers time-dependent travel times and allows all routes to start at a flexible departure time and has an option to select only the profitable requests.

In general, vehicle routes need to satisfy the following constraints.

- Each route starts and ends at the predefined depot;
- Every request is served at most once. If it is served, the pickup and delivery nodes of the request should be visited by the same vehicle;
- Precedence relations: The pickup location of a request is required to be visited before its corresponding delivery location;
- If a request is served, its nodes must be visited in the imposed time windows;
- Vehicle capacity must not be violated at any time.

The VRPSPD has attracted a lot of attention in the scientific community. It was introduced by Min [73], inspired by the problem of distributing and collecting books in a library. Unlike classic PDPTW, each customer has delivery demands for the delivery commodity and pickup demands for the pickup commodity. The problem is to construct vehicle routes with minimum total cost, satisfying the pickup and delivery requests of each customer in a single visit while not exceeding the capacity of the vehicles. Angelelli and Mansini [2] considered time windows as additional constraints in the VRPSPD. The recent literature offers just a few contributions to the VRPSPD with stochastic demands (see, e.g., Dimitrakos and Kyriakidis [32], Minis and Tatarakis [75], Pandelis et al. [81], Wollenberg et al. [126], Zhu and Sheu [133]). In Chapter 5, we examine a real application in a laundry business and study a time-dependent laundry routing problem with stochastic pickup demands (TDLRPSPD). This problem generalizes the VRPSPD by extending it with soft time windows, time-dependent travel times, and stochastic pickup demands. In the TDLRPSPD, each customer has a chosen soft time window in which it wants to be served. Late servicing of customers incurs some penalty costs. Untimely response to the pickup failure influences customer service satisfaction level and consequently reduces market share. Furthermore, most research has focused on the case with the delivery of a single commodity, whereas the TDLRPSPD considers a pickup and delivery laundry case such that restaurants and hotels prefer to use textiles with their own logos (such as uniforms). Thus, the following constraints need to be satisfied:

- Each route starts and ends at a predefined depot;
- Every customer is served exactly once in the planned routes (it might be that due to failures in operations an extra visit is necessary);
- Each node is served no earlier than the earliest time at which service may start;
- The total delivery demands of each vehicle must not violate its capacity limit.

1.3 Overview of the thesis

In Chapter 2, we introduce the time-dependent capacitated profitable tour problem with time windows and precedence constraints and formally describe it as an arc-based mixed-integer program (MIP). First, the proposed MIP is solved using optimization software (i.e., Gurobi 5.6), considering small-size instances with up to 60 locations (30 pickup and delivery requests). To tackle larger instances, a tailored labeling algorithm is proposed. Several dominance criteria are also introduced to discard unpromising labels. Our computational results demonstrate that the algorithm is capable of solving instances with up to 150 locations

(75 pickup and delivery requests) to optimality. Additionally, we present a restricted dynamic programming heuristic to improve the computation time. This heuristic does not guarantee optimality, but is able to find the optimal solution for 32 instances out of the 34 instances in short computation times.

Chapter 3 studies a family of TDPDPTW, which extends the problem in Chapter 2 to the multiple-vehicle case. We aim to optimize the service of a transportation provider under two dimensions of operational flexibility. In the first, we consider problems wherein the transportation service provider can choose the transportation requests it serves to maximize profit. In the second, we consider problems wherein they can take advantage of periods of light traffic by dictating to drivers when their routes should begin. We also consider problems wherein these flexibilities are not present. We propose an exact solution approach for solving problems from this family that is based upon branch and price, wherein columns are generated via a tailored labeling algorithm. We augment the framework with adaptations of various speed-up techniques from the literature, including limited-memory subset-row cuts and route enumeration. With an extensive computational study, we assess the effectiveness of the proposed framework and the impact of the adapted techniques. We have shown that small to medium-size instances, with up to 45 freight requests (90 locations), can be optimally solved by the proposed exact algorithm within a reasonable run-time.

In Chapter 4, we focus on the TD-PPDP-TW. As one variant of the problem studied in Chapter 3, it allows all routes to start at a flexible departure time. It also decides which customers to serve, and orders the visits in each route. Moreover, we propose an adaptive large neighborhood search (ALNS) algorithm. The general idea of an ALNS algorithm is to iteratively improve a given solution by first partially deteriorating it and then repairing it. A destroy operator and an insertion operator are used, respectively. We use a total of ten removal and five insertion operators. Each operator is selected based on its past performance during the search process. Results of an extensive computational study show that the algorithm is able to quickly find high-quality solutions on instances with up to 75 transportation requests (150 locations). We also conduct a study of the impact on profits when explicitly recognizing traffic congestion during planning operations.

Chapter 5 investigates an important process for the commercial laundry business, the TDLRPSPD. While providing convenience and flexibility to customers, efficient laundry pickup and delivery will also result in reduction in cost and time and improvement in service quality. It ultimately results in customer loyalty, economic efficiency, and gain of competitive

advantage. Moreover, the pickup quantity for each hotel or restaurant is stochastic as it depends on how many customers this hotel or restaurant has had in the previous days. We assume that the laundry service provider can make a reasonable prediction based on each restaurant or hotel's historical performance. The literature on this problem is quite limited. To tackle this challenge, we present a two-stage stochastic programming with recourse model. We also propose a sample average approximation method together with an ALNS algorithm to obtain a heuristic solution. Several experiments are conducted to show the effect of our solution approach. Compared to a pure deterministic solution using expected pickup demands, it shows that an average cost savings of more than 50% can be achieved.

To help clarity, in Table 1.1, we summarize the general features of the research conducted in each chapter of this thesis. Column "Time-dependency" indicates whether the considered problem considers time-dependent travel time, "Request selection" indicates whether it is allowed to serve only a subset of the transportation requests, "Stochastic aspect" indicates which feature of the problem is considered to be stochastic in the considered problem, "Methodology" represents the types of algorithms used to solve the considered problem, and "Maximum instance size" indicates the largest (in terms of the number of requests) instance solved. Finally, column "Research questions" indicates which specific research question is answered in the corresponding chapter.

Table 1.1: An overview of the thesis

Chapter	Time-dependency	Requests selection	Stochastic aspect	Methodology	Maximum instance size	Research questions			
						1	2	3	4
2	✓	✓		Exact & Heuristic	75	✓			
3	✓	✓		Exact	45	✓		✓	
4	✓	✓		Heuristic	75	✓	✓		
5	✓		pickup demands	Heuristic	100	✓			✓

The work of Chapters 2 to 4, have already been published in the following papers:

Chapter 2: Peng Sun, Lucas P. Veelenturf, Said Dabia, Tom Van Woensel, The Time-Dependent Capacitated Profitable Tour Problem with Time windows and Precedence Constraints. *European Journal of Operational Research*, 264(3): 1058-1073, 2018.

Chapter 3: Peng Sun, Lucas P. Veelenturf, Mike Hewitt, Tom Van Woensel, The time-dependent pickup and delivery problems with time windows. *Transportation Research Part B: Methodological*, 116: 1-24, 2018.

Chapter 4: Peng Sun, Lucas P. Veelenturf, Mike Hewitt, Tom Van Woensel, An adaptive

large neighborhood search heuristic for the time-dependent profitable pickup and delivery problems with time windows. *Transportation Research Part E: Logistics and Transportation Review*, 138, 2020.

Chapter 2

The Time-Dependent Capacitated Profitable Tour Problem with Time Windows and Precedence Constraints

"It does not matter how slowly you go
as long as you do not stop."

Confucius

2.1 Introduction

The effective usage of empty vehicles' space is an important opportunity to increase the efficiency of urban transportation systems and to reduce traffic congestion, fuel consumption, and pollution. Comapnies (e.g., Uber) can generate extra income by renting out vehicles's empty space rising in their transportation processes. Several mobile applications are developed to improve the last-mile deliveries by involving the city's residents. For instance, Roadie created an on-the-way delivery network which is an online market where people post their required shipments and where anyone can offer to execute the shipment. DHL launched a platform called MyWays, enabling individuals to deliver packages with products ordered online directly to other end consumers. For those individuals with limited transportation resources, it is important to know which parcels are profitable to collect and deliver. On the one hand, serving a request may be attractive because it generates revenue. On the other hand, there is additional cost for serving the request. Consequently, it might not always be economically beneficial to serve a request. Moreover, taking time windows into consideration makes the routing problem more realistic, since, in daily operation, customers and freight requests are only available during the opening hours. Furthermore, due to the limited capacity of the road network and traffic intensification, the travel speed is serious affected by the traffic fluctuations, which also results in great variations in travel times and transportation costs.

Therefore, considering the time-dependent travel time has large potential for cost savings. Under this setting, the travel time between locations is depend on the time the driver departs (e.g. in rush hours traveling takes more time), a tour's cost depends on the tour's start time. Therefore, those independent drivers need a tool to help them out in making decisions on which request to serve and which time to depart. However, scarce literature can be found that focus on this area.

This chapter aims at building such a tool by modeling and solving a time-dependent capacitated profitable tour problem with time windows and precedence constraints. We consider a single vehicle with capacity limit and a set of requests which have a pick-up and a delivery node. Each pickup node and delivery node has its own time window in which it should be served. Moreover, a delivery node of a request can only be served after its pickup node is visited. For each served request a profit is collected. To capture travel speed variation during a day, a time-dependent travel time function is assigned to each edge linking two nodes. The objective is to determine the vehicle's tour starting and ending at the depot, and maximizing the difference between the total collected profits and total travel cost (See Figure 2.1). Following the literature in this area, the tour's total travel cost is equal to the tour's duration.

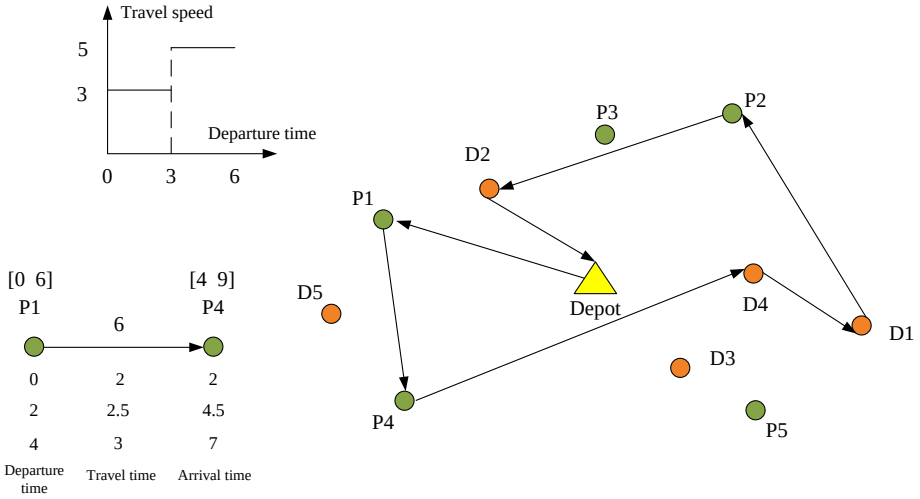

Figure 2.1: An illustration of the time-dependent capacitated profitable tour problem with time windows and precedence constraints. P_i and D_i are the pickup node and delivery node of request i respectively.

The described problem is NP-hard because it is an extension of the traveling salesman problem with pickup and delivery (TSPPD), which itself is an extension of the traveling salesman problem (TSP). In contrast to the TSPPD and the TSP, in our problem, it is not necessary to carry out all the requests.

The main contributions of this chapter are summarized as follows. First, we introduce a new model which extends the classical TSPPD by having time-dependent travel times and the option to reject requests. Secondly, we propose an exact solution method for this problem by developing a tailored labeling algorithm in which novel and strong dominance criteria are used. Finally, a restricted dynamic programming heuristic is proposed with a high solution quality and lower computation times than the labeling algorithm.

The remainder of the chapter is structured as follows. Section 2.2 provides a brief review of related existing work. Section 2.3 defines the problem and introduces a mathematical formulation of the problem. In Section 2.4, we present the tailored labeling algorithm and in section 2.5 the restricted dynamic programming heuristic is introduced. Finally, computational results are reported in Section 2.6, followed by conclusions in Section 2.7.

2.2 Literature review

There are three classes of problems closely related to the problem studied in this chapter: the traveling salesman problem with pickup and delivery (TSPPD), the time-dependent vehicle routing problem (TDVRP) and the traveling salesman problem with profits (TSP with profits).

2.2.1 The traveling salesman problem with pickup and delivery (TSPPD)

Our problem extends the TSPPD by considering time-dependent travel times. The TSPPD is firstly introduced by Ruland and Rodin [95] and is proven to be of great use in applications like dial-a-ride systems and courier services. Although the pickup and delivery problem (PDP), in which the TSPPD can appear as a subproblem, is extensively studied in the literature, only limited research focuses on the TSPPD.

Currently, the most popular methodology for solving the TSPPD is branch-and-cut. Ruland [94] and Ruland and Rodin [95] considered the undirected case of this problem and developed four classes of valid inequalities that are embedded in a branch-and-cut algorithm. The algorithm is tested on instances with up to 15 pickup and delivery requests. Recently,

Dumitrescu et al. [35] studied the same problem, the authors analyzed its polyhedral structure and proposed new valid inequalities that are shown to be facets for the TSPPD polytopes. Their algorithm is capable of solving instances with up to 35 pickup and delivery requests to optimality.

The TSPPD appears as pricing problem for the PDP and is in that situation usually named as the elementary shortest path problem with time windows, capacity and pickup and delivery (ESPPTWCPD). Sol [103], Sigurd and Pisinger [102] and Røpke and Pisinger [93] presented labeling algorithms with several different dominance rules to solve this problem to optimality.

2.2.2 The time-dependent vehicle routing problem (TDVRP)

Another related problem is the TDVRP. Although the TDVRP has attracted the attention of many researchers, literature on this subject remains scarce. The pioneering work is done by Malandraki and Daskin [71] and Malandraki and Dial [72]. In these papers mixed integer linear programs and several heuristics to solve the problem are proposed. The First-In-First-Out (FIFO) property, which implies that for every arc a later departure time results in a later (or equal) arrival time, is an intuitive and desirable property for time dependent routing problems. Ichoua et al. [56] and Dabia et al. [25] considered the TDVRP with travel time variability modelled by "constant speed" time periods, which ensures the FIFO property. The idea of constant speed time periods is adopted in our problem as well.

Due to the complexity of the time-dependent problem, most of the existing algorithms are based on heuristics. In Van Woensel et al. [114] a tabu search heuristic is used to solve the capacitated vehicle routing problem with time dependent travel times. An approximation based on queueing theory and the number of vehicles on a link is used to determine travel speeds. Donati et al. [33] developed a multi-ant colony system for the TDVRP and Ibaraki et al. [55] proposed an iterated local search heuristic for the time-dependent vehicle routing problem with time windows (TDVRPTW). Recently, Dabia et al. [25] developed a branch-and-price algorithm for the TDVRPTW, where a tailored labeling algorithm is presented to solve the time-dependent shortest path problem with resource constraint (TDSPPRC), which is the pricing problem in the algorithm.

2.2.3 The traveling salesman problem with profits (TSP with profits)

The proposed problem extends the traveling salesman problem with profits (TSP with profits), in which profits are associated with each request and the overall goal is to find athe shortest tour with the maximum collected profits. This means that in contrast to the original TSP, not all nodes have to be visited. In comparison with our study, it does not include time dependency and requests with pickup and delivery nodes.

According to Feillet et al. [39] and Vansteenwegen and Gunawan [115], TSPs with profits can be categorized into three generic problems depending on the way the terms *profits* and *travel time* are addressed in the objective function and constraints. They can be classified in the following categories: the profitable tour problem (PTP), the prize-collecting traveling salesman problem (PCTSP), and the orienteering problem (OP). Dell' Amico et al. [27] studied the profitable tour problem (PTP) where both the profits and the travel time are combined in the objective function. Our study builds upon the PTP by having the profits and the travel time in the objective as well. The difference between the prize-collecting TSP (PCTSP) and the PTP is that the profit collection is not a part of the objective function but a constraint, which ensures that a minimum amount of profits is collected within the tour. In the original definition of the PCTSP by Balas [5] there were also penalty values for unvisited nodes within the objective function. An abundant number of publications is devoted to the orienteering problem (OP), which aims to maximize the collected profits subject to a constraint on the maximum allowed tour length. This problem is also known as the selective traveling salesman problem (Laporte and Martello [63]).

A variant of the OP is the time-dependent orienteering problem (TDOP) which includes time-dependent travel times. Fomin and Lingas [40] provided a $(2+\epsilon)-$approximation algorithm for the TDOP which runs in polynomial time if the ratio between the minimum and maximum travel time between any two sites is constant. Li [66] designed a novel dynamic labeling algorithm for the TDOP in which time is measured in discrete units. Therefore, the FIFO property may not be satisfied in their model. Verbeeck et al. [117] provided a fast solution method for the TDOP based on an ant colony optimization algorithm. Recently, Verbeeck et al. [118] presented an ant colony optimization based algorithm for a stochastic variant of TDOP, which is addressed as the stochastic time-dependent orienteering problem with time windows. For more details about the OP and its variants, readers are referred to Vansteenwegen et al. [116] and Gunawan et al. [50].

To the best of our knowledge, no literature could be found that handle precedence in

pickup and delivery, profit-maximizing selection, and time-dependent travel time routing cost minimization at the same time. Thus, in this study, we introduce the time-dependent capacitated profitable tour problem with time windows and precedence constraints, which take care of these three challenges simultaneously. Moreover, both exact and heuristic methods are proposed to solve this problem.

2.3 Problem description and mathematical formulation

In this section, we first define the problem and introduce the notation used throughout the chapter. Afterwards, we present a mathematical formulation for the problem.

2.3.1 Problem definition

The time-dependent capacitated profitable tour problem with time windows and precedence constraints is defined as follows. We consider a set of n requests $R_1, ..., R_n$, where R_i $(i = 1, ..., n)$ is associated with the pickup node i and the corresponding delivery node $n + i$. Let $G = (N, A)$ be a directed graph, where $N = \{0, 1, ..., 2n + 1\}$ is the set of all nodes, and 0 and $2n + 1$ represent the origin and destination depot of the vehicle. We define the subsets $N_P = \{1, ..., n\}$ and $N_D = \{n + 1, ..., 2n\}$ as the pickup and delivery nodes, respectively. With each pickup node $i \in N_P$ a profit r_i and a load q_i are associated, and with each delivery node a load q_{n+i} is associated. For the requests, it must hold that $q_i = -q_{n+i}$. There is no inventory at the depots and therefore $q_0 = q_{2n+1} = 0$. To serve the requests we have one vehicle available with limited capacity Q.

A hard time window $[e_j, l_j]$ is associated with each node $j \in N_P \cup N_D$, where e_j and l_j represent the earliest and latest time, respectively, at which the service at node j may start. The service time is denoted by s_j. A vehicle needs to wait until time e_j, if it is arriving at node j before time e_j; and arriving later than l_j is not allowed. We denote $[e_0, l_0]$, $[e_{2n+1}, l_{2n+1}]$ as the time windows of the origin and the destination depot, respectively. Without loss of generality, we assume that $e_0 = 0$ and $s_0 = s_{2n+1} = 0$.

Let $\tau_{ij}(t_i)$ denote the travel time from node i to node j, which depends on the departure time t_i at node i. Then, we can define the set of feasible arcs as $A = \{(i, j) \in N \times N : i \neq j$ and $e_i + s_i + \tau_{ij}(e_i + s_i) \leq l_j\}$. This means that an arc from node i to node j is only included if it is possible to go from node i to j while respecting the time windows of both nodes.

The notation is summarized in Table 2.1.

Table 2.1: Parameters

Notation	Definition
$R = \{1, ..., n\}$	Set of requests
N	Set of nodes
A	Set of arcs
N_P	Set of pickup nodes
N_D	Set of delivery nodes
$(i, n+i)$	A transportation request R_i
r_i	Profit of request R_i
q_i	Demand of request R_i
Q	Carrying capacity of the vehicle
$[e_i, l_i]$	Time window of node i
s_i	Service time at node i
t_i	Departure time from node i
$\tau_{ij}(t_i)$	Travel time from node i to node j with departure time t_i at node i

The planning horizon is divided into several time periods. Each arc $(i, j) \in A$ has a speed profile associated with it, which consists of a constant speed within each time period. By using those stepwise speed functions, the FIFO property holds for every arc in the graph G (i.e. a later departure always leads to a later arrival and therefore overtaking will not occur). The speed profiles can be different for each arc.

Figure 2.2 depicts a speed profile and the corresponding travel time function for some arc (i, j). Using the idea described in Ichoua et al. [56], we denote the points a, b, c, d, and e where the speed changes as $speed\ breakpoints$. There are also $travel\ time\ breakpoints$ in the travel time function. These are the departure times which ensure an arrival at node j exactly at the time of a speed breakpoint (e.g., $a^{'}$ is the departure time at node i to arrive at node j at time a).

The travel time function is piecewise linear and can be represented by the breakpoints values. Note that in case of time-dependent travel times, the triangle inequality does not necessarily hold. Intuitively, when the direct link between node h and l is heavily congested, we may reach destination node l earlier by taking a diverted route (i.e., via one or several other nodes) than by the direct link from node h. One way to circumvent this, is to do pre-processing on the data, such that the travel time between node i and node j always represents the shortest travel time from i to j even if it is via a detour. However this requires to find time-dependent shortest routes from i to j for all possible departure times. This also results in increased computational

complexity and memory consumption. Another solution will be to allow nodes to be visited more than once. However allowing for non-elementary paths in the problem also leads to new mathematical challenges.

Because of the FIFO property of the travel time functions, a later departure at the depot 0 always results in a later arrival time at node i. Therefore, if a path is infeasible for a certain departure time t_0 at the origin depot (i.e., a time window of a node in the path is violated), it will also be infeasible for any departure time $t^{'} \geq t_0$ at the origin depot.

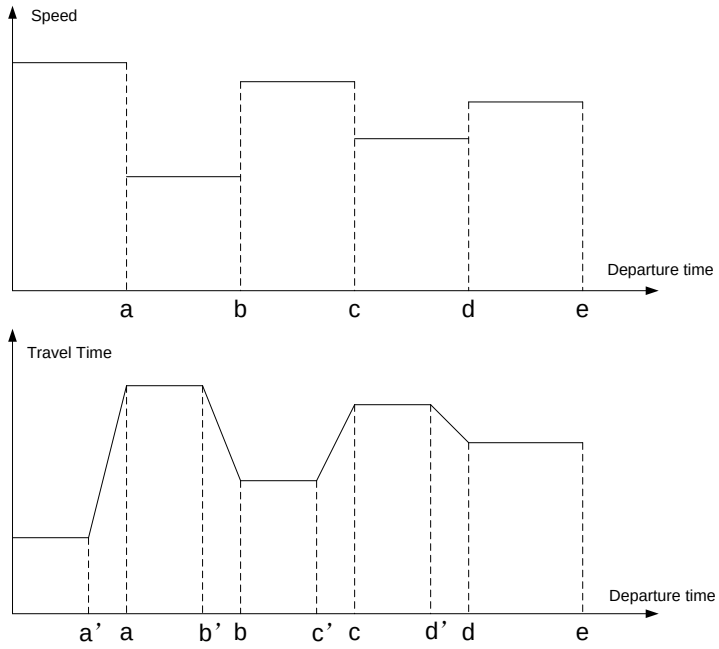

Figure 2.2: Speed and Travel Time Functions

Given a path $p = (v_0, v_1, ..., v_k)$ with $v_0 = 0$ and v_i being the node at position i in the path p, we define $\delta^p_{v_i}(t)$ as the ready time function at node v_i in path p given a departure time t at node 0. This ready time function is nondecreasing in t and can be calculated recursively for each node in the path as follows:

$$\delta^p_{v_i}(t) = \begin{cases} t & \text{if } i = 0, \\ \max\{e_{v_i} + s_{v_i}, \delta^p_{v_{i-1}}(t) + \tau_{v_{i-1},v_i}(\delta^p_{v_{i-1}}(t)) + s_{v_i}\} & \text{otherwise.} \end{cases} \tag{2.1}$$

The ready time function is piecewise linear and this means that we can represent the ready time function by using the $ready\ time\ function\ breakpoints$. These are the breakpoints of the ready time function of the predecessor node, breakpoints of the travel time function, and the boundary values of the time window of node v_i.

The duration of the path given a departure time t at node 0 can be calculated as $\delta^p_{v_k}(t) - t$, which is again a piecewise linear function. In this problem we minimize the total duration of the selected tour instead of the sum of the arc cost. As the duration is a piecewise linear function of the departure time, it is clear that the minimum duration of a tour can be computed by only considering the breakpoints of the ready time function.

2.3.2 Mathematical formulation

For every arc $(i, j) \in A$, we denote T_{ij} as the set of time periods of the corresponding travel time function $\tau_{ij}(t_i)$. A time period $T_m \in T_{ij}$, is defined by two consecutive travel time breakpoints, $T_m = [w_m, w_{m+1}]$. As $\tau_{ij}(t_i)$ is linear in each time period, using w_m, w_{m+1}, $\tau_{ij}(w_m)$ and $\tau_{ij}(w_{m+1})$, we can easily calculate the corresponding slope θ_m and its intersection η_m with the y-axis. Therefore

$$\tau_{ij}(t_i) = \theta_m t_i + \eta_m. \qquad \forall t_i \in T_m \tag{2.2}$$

Furthermore, let x^m_{ij} be a binary variable that takes value 1 if and only if the vehicle traverses the arc $(i, j) \in A$ with a departure time in time period m. A variable t^m_{ij} is introduced to denote this departure time of traveling from i to j in time period T_m. This means that t^m_{ij} is such that

$$t^m_{ij} = \begin{cases} t_i & \text{if } x^m_{ij} = 1, \\ 0 & \text{otherwise.} \end{cases} \tag{2.3}$$

Consequently, when traveling from i to j, we have that:

$$t_i = \sum_{j \in N \setminus \{0\}} \sum_{m=0}^{|T_{ij}|} t^m_{ij}. \tag{2.4}$$

It means that the travel time function $\tau_{ij}(t_i)$ of arc (i,j) at node i can be written as:

$$\tau_{ij}(t_i) = \sum_{m=0}^{|T_{ij}|} (\theta_m t_{ij}^m + \eta_m x_{ij}^m). \tag{2.5}$$

Let y_i be a binary variable that equals 1 if and only if node $i \in N_P \cup N_D$ is visited. Furthermore, let $Q_i, i \in N$ be a nonnegative integer variable that is the load of the vehicle upon departure from node i. Then the Mixed Integer Programming formulation is given as follows:

$$\max \sum_{j \in N_P} r_j y_j - (t_{2n+1} - t_0) \tag{2.6}$$

subject to

$$\sum_{j \in N_P} \sum_{m=0}^{|T_{0j}|} x_{0j}^m = 1 \tag{2.7}$$

$$\sum_{i \in N_D} \sum_{m=0}^{|T_{i,2n+1}|} x_{i,2n+1}^m = 1 \tag{2.8}$$

$$\sum_{i \in N \setminus \{2n+1\}} \sum_{m=0}^{|T_{ij}|} x_{ij}^m = y_j \qquad \forall j \in N \setminus \{0\} \tag{2.9}$$

$$\sum_{i \in N \setminus \{2n+1\}} \sum_{m=0}^{|T_{ik}|} x_{ik}^m - \sum_{j \in N \setminus \{0\}} \sum_{m=0}^{|T_{kj}|} x_{kj}^m = 0 \qquad \forall k \in N \setminus \{0, 2n+1\} \tag{2.10}$$

$$\sum_{j \in N \setminus \{0\}} \sum_{m=0}^{|T_{ij}|} x_{ij}^m - \sum_{j \in N \setminus \{0\}} \sum_{m=0}^{|T_{n+i,j}|} x_{n+i,j}^m = 0 \qquad \forall i \in N_P \tag{2.11}$$

$$t_j \geq (1+\theta_m) t_{ij}^m + \eta_m x_{ij}^m + s_j x_{ij}^m \qquad \forall i \in N \setminus \{2n+1\}, j \in N \setminus \{0\} \tag{2.12}$$

$$Q_i + q_j \leq Q_j + M(1 - x_{ij}^m) \qquad \forall i,j \in N, \forall m, m+1 \in \mid T_{ij} \mid \tag{2.13}$$

$$t_{n+i} \geq t_i \qquad \forall i \in N_P \tag{2.14}$$

$$t_i = \sum_{j \in N \setminus \{0\}} \sum_{m=0}^{|T_{ij}|} t_{ij}^m \qquad \forall i \in N \setminus \{2n+1\} \tag{2.15}$$

$$w_m x_{ij}^m \leq t_{ij}^m \leq w_{m+1} x_{ij}^m \qquad \forall i,j \in N, \forall m, m+1 \in \mid T_{ij} \mid \tag{2.16}$$

$$e_i y_i \leq t_i \leq l_i y_i \qquad \forall i \in N_P \cup N_D \tag{2.17}$$

$$\max\{0, q_i\} \leq Q_i \leq \min\{Q, Q + q_i\} \qquad \forall i \in N \tag{2.18}$$

$$x_{ij}^m, y_i \in \{0,1\} \qquad \forall i,j \in N, \forall m \in \mid T_{ij} \mid \tag{2.19}$$

The Objective Function (2.6) aims to find a tour that maximizes the collected profits minus the total traveling duration. Constraints (2.7)-(2.8) guarantee that the path starts at the origin depot 0 and ends at the destination depot $2n+1$. Constraints (2.9) guarantee that every node, except the nodes representing the start and end depots, is visited at most once. Constraints (2.10) keep the flow conservation. Constraints (2.11) ensure that it is not possible to visit only the pickup node or only the delivery node of a certain request. Constraints (2.12) guarantee that the departure time at a node in the route is larger or equal than the sum of the departure time from the previous node, the travel time between these two nodes and the required service time. Constraints (2.13) determine the consistency of the load variables. Precedence constraints (2.14) ensure that for each request i, the pickup node is visited before the delivery node. Constraints (2.15) are formulated as mentioned before in (2.4). Constraints (2.16)-(2.17) force the departure time of each request to be in the given time period and the given time window. Finally, Constraints (2.18) ensure that the load in the vehicle is never larger than the capacity of the vehicle.

Due to the computational inefficiency of solving large-scale instances with a commercial ILP solver, we develop a tailored labeling algorithm to solve this problem.

2.4 A tailored labeling algorithm

In order to solve our problem, we introduce a new exact dynamic programming algorithm, that is named as the tailored labeling algorithm. Røpke and Pisinger [93] developed a labeling algorithm to solve the pickup and delivery problems with time windows (PDPTW). However, this time-independent algorithm is only efficient if the triangle inequality holds. More recently, Dabia et al. [25] proposed another labeling algorithm for the time-dependent vehicle routing problem with time windows. Their algorithm has great potential for the time-dependent routing problem without precedence constraints.

In our situation the triangle inequality does not necessarily hold due to the time dependent travel times and furthermore precedence constraints are present. Therefore, we need to develop a new algorithm. Note that the proposed algorithm can be generalized to solve other time-dependent routing problems with precedence constraints.

The algorithm starts generating labels from the depot 0. It progressively extends all feasible labels until they reach the end depot $2n+1$. Moreover, to speed up our tailored labeling algorithm, instead of starting the label extension only from the origin depot 0 in a forward

direction, we simultaneously generate labels in backward direction from the destination depot to its predecessors as well. Where both forward labels and backward labels are extended to some time t_m (e.g., the middle of the planning horizon) but not further. At the end, complete paths are generated by merging the partial paths of forward and backward labels. All complete paths are evaluated and the path with the best objective function value is the optimal path. This bidirectional approach has shown great potential for improving the running time of related resource constrained shortest path problems (see, e.g. Righini and Salani [88] and Dabia et al. [25]).

The forward labeling algorithm is introduced in Section 2.4.1, followed by the backward labeling algorithm in section 2.4.2. If labels are dominated by the criterion introduced in Sections 2.4.1 and 2.4.2, they are removed from the list. Note that if in the procedure none of the labels are dominated, this algorithm is equal to the complete enumeration of all feasible paths. Therefore the dominance criterion is very important. Finally, we discuss the way to merge the partial paths of forward and backward labels in Section 2.4.3.

2.4.1 The forward labeling algorithm

In the forward labeling algorithm we start generating labels from the start depot. The definition of a forward label is discussed in Section 2.4.1. The labels are extended if the extension is feasible as discussed in Section 2.4.1. The dominance criterion for forward labels is discussed in Section 2.4.1.

Forward label

For each forward label L_f, we use the following notation:

Only the items marked with a * are stored in the label. The set $D(L_f)$ and $P(L_f)$ can be deduced from the sets $O(L_f)$ and $U(L_f)$. Furthermore, the partial path can be deduced from iteratively checking the last node visited in the parent label of which the this label was an extension.

Label extension

We extend a label $L_f^{'}$ along an arc $(v(L_f^{'}), j)$, only when the extension is feasible in terms of time windows and capacity. First, the following two conditions should be met:

$p(L_f)$		The partial path of label L_f.
$v(L_f)$	*	The last node visited on the partial path $p(L_f)$.
$L^{-1}(L_f)$	*	The parent label from which L_f originates by extending it with $v(L_f)$.
$O(L_f)$	*	The set of incomplete requests in $p(L_f)$, i.e., the pickup node is visited but not the delivery node.
$U(L_f)$	*	The set of requests for which the pickup nodes are already visited along the partial path $p(L_f)$. It contains both the complete and the incomplete requests. Therefore, $O(L_f) \subseteq U(L_f)$.
$P(L_f)$		The set of pickup nodes not visited in $p(L_f)$, i.e., $j \in N_P$ and $R_j \notin U(L_f)$.
$D(L_f)$		The set of delivery nodes of incomplete requests in $p(L_f)$, i.e., $j \in N_D$ and $R_{j-n} \in O(L_f)$.
$q(L_f)$	*	The load of the vehicle after visiting node $v(L_f)$.
$\delta_{L_f}(t)$	*	The piecewise linear function that represents the ready time at $v(L_f)$ if the vehicle departed at the origin depot at t and reached $v(L_f)$ through partial path $p(L_f)$. Moreover, $\delta_{L_f}(0)$ is the earliest ready time at $v(L_f)$ since the earliest departure time at the origin depot is 0.
$r(L_f)$	*	The overall profits collected by serving the requests visited on the partial path $p(L_f)$.

$$\delta_{L_f^{'}}(0) + \tau_{v(L_f^{'}),j}(\delta_{L_f^{'}}(0)) + s_j \leq \min\{t_m, l_j + s_j\} \quad \wedge \quad j \in N \setminus \{0\} \tag{2.20}$$

$$q(L_f) + q_j \leq Q \quad \wedge \quad j \in N \setminus \{0\} \tag{2.21}$$

Condition (2.20) ensures that an extension to node j can only be performed if node j can be reached within its time window and guarantees that the extension is stopped before t_m is exceeded. Condition (2.21) ensures that an extension to node j is only possible if there is enough capacity to deal with the load of node j.

Secondly, $L_f^{'}$ and j must also satisfy one of the following three conditions:

$$j \notin U(L_f^{'}) \quad \wedge \quad j \in N_P \tag{2.22}$$

$$j - n \in O(L_f^{'}) \quad \wedge \quad j \in N_D \tag{2.23}$$

$$O(L_f^{'}) = \emptyset \quad \wedge \quad j = 2n + 1 \tag{2.24}$$

Condition (2.22) states that j should not have been visited before, if it is a pickup node. Condition (2.23) indicates that if j is a delivery node, the corresponding pickup node should

have been visited already. The last condition, Condition (2.24), states that if j is the end depot then all visited requests should have been completed. In the presence of those conditions, only elementary paths that satisfy precedence constraint (2.11) are generated.

At last, we need to check that all delivery nodes of requests for which the pickup node is already visited in $p(L_f^{'})$ can still be reached. In case the triangle inequality holds, a node is unreachable if traversing the direct arc from j to this node is not possible by capacity, time window or precedence constraints. However, time-dependent travel times cannot guarantee the triangle inequality. Therefore, a node that is unreachable via the direct arc from node j by the time window constraints might still be reachable indirectly via a diverted route. First, we need to know the earliest ready time at node j after following partial path $p(L_f^{'})$ before visiting node j, which will be denoted by $t_r(L_f^{'}, j)$. It holds that $t_r(L_f^{'}, j) = \max\{e_j + s_j, \delta_{L_f^{'}}(0) + \tau_{L_f^{'},j}(\delta_{L_f^{'}}(0)) + s_j\}$. Then, we need for any unvisited node k the earliest arrival time given that the vehicle visits node j and k consecutively after partial path $p(L_f^{'})$, which can be computed by $t_r(L_f^{'}, j) + \tau_{jk}(t_r(L_f^{'}, j))$. Finally, the earliest possible time the vehicle could reach a node after node j is given by $t_e(L_f^{'}, j) = \min_{k \in P(L_f^{'}) \cup D(L_f^{'})}\{t_r(L_f^{'}, j) + \tau_{jk}(t_r(L_f^{'}, j))\}$. This means that any node k with the latest allowed arrival time l_k earlier than this time (i.e. $l_k < t_e(L_f^{'}, j)$) is unreachable from j, also in an indirect way as no node could be reached before $t_e(L_f^{'}, j)$. If a delivery node k is unreachable after j and its corresponding pickup node is already visited (i.e $k - n \in O(L_f')$), the extension to j is not feasible as the picked up item can not be delivered anymore. Therefore, as stated in condition (2.25), all delivery nodes of requests of which the pickup node is already visited in $p(L_f^{'})$ should still be reachable to make sure that extending label $L_f^{'}$ to j is feasible. Note that this test can be done quickly, but we might fail to find all unreachable delivery nodes.

$$l_k \geq t_e(L_f^{'}, j) \quad \forall k \in D(L_f^{'}) : k \neq j \tag{2.25}$$

If the extension along the arc $(v(L_f^{'}), j)$ is feasible according to all described conditions, then

a new label L_f is created. The information in label L_f is updated as follows:

$$L^{-1}(L_f) = L_f^{'} \tag{2.26}$$

$$v(L_f) = j \tag{2.27}$$

$$\delta_{L_f}(t) = \max\{e_j + s_j, \delta_{L_f^{'}}(t) + \tau_{L_f^{'},j}(\delta_{L_f^{'}}(t)) + s_j\} \tag{2.28}$$

$$q(L_f) = q(L_f^{'}) + q_j \tag{2.29}$$

$$r(L_f) = \begin{cases} r(L_f^{'}) + r_j & \text{if } j \in N_P, \\ r(L_f^{'}) & \text{otherwise.} \end{cases} \tag{2.30}$$

$$O(L_f) = \begin{cases} O(L_f^{'}) \cup \{j\} & \text{if } j \in N_P, \\ O(L_f^{'}) \setminus \{j-n\} & \text{if } j \in N_D. \end{cases} \tag{2.31}$$

$$U(L_f) = \begin{cases} U(L_f^{'}) \cup \{j\} & \text{if } j \in N_P, \\ U(L_f^{'}) & \text{otherwise.} \end{cases} \tag{2.32}$$

Equations (2.26)-(2.30) set the last visited node, the ready time function, the load, and the collected profits of the new label, respectively. Equation (2.31) updates the set of incomplete requests $O(L_f)$ and Equation (2.32) updates the set of visited pickup nodes $U(L_f)$.

Label dominance

Let dom(L_f) and img(L_f) be the domain and image of the ready time function $\delta_{L_f}(t)$ respectively. If the partial path is feasible, a departure at time 0 from the origin depot is always feasible. Therefore, dom(L_f) is always of the form $[0, t]$ for some $t \geq 0$. When $v(L_f) = 2n+1$, the objective function value of the path corresponding to L_f is:

$$obj(L_f) = r(L_f) - \min_{t \in Dom(L_f)} \{\delta_{L_f}(t) - t\} \tag{2.33}$$

In the labeling algorithm, all possible extensions are processed and stored for each label. However, the number of labels that can be processed is typically very large and computationally expensive. Therefore, a dominance test is established between pairs of labels that have the same last visited node. The number of labels is reduced by only storing the non-dominated labels. Before the dominance criterion is introduced some definitions need to be provided.

First, similar to the idea of Feillet et al. [38], we introduce for every label L_f the set

$\widetilde{U}(L_f)$ which extends $U(L_f)$ by adding requests of which the pickup node is unreachable from $v(L_f)$. Similar to the discussion in Section 2.4.1 it can be derived that the earliest possible time the vehicle could reach a pickup node after $v(L_f)$ is given by $t_e(L_f) = \min_{j \in P(L_f) \cup D(L_f)} \{\delta_{L_f}(0) + \tau_{v(L_f)j}(\delta_{L_f}(0))\}$. This means that any pickup node j with the latest allowed arrival time l_j earlier than this time (i.e. $l_j < t_e(L_f)$) is unreachable from $v(L_f)$. Therefore, the corresponding request cannot be served anymore and can be added to the set $\widetilde{U}(L_f)$. Note that the check $l_j < t_e(L_f)$ does not guarantee to find all unreachable pick up nodes.

Secondly, we define the interval

$$I \subseteq (-\infty, \max(dom(L_f^1)) - \max(dom(L_f^2))). \tag{2.34}$$

Based on I, we also define a real number $\phi(L_f^1, L_f^2)$,

$$\phi(L_f^1, L_f^2) = \max\{x \in I : \delta_{L_f^1}(\max\{0, t + x\}) \leq \delta_{L_f^2}(t), \forall t \in dom(L_f^2)\}. \tag{2.35}$$

When $\phi(L_f^1, L_f^2)$ is positive, it indicates that the vehicle can depart at maximum $\phi(L_f^1, L_f^2)$ time units later when traversing partial path $p(L_f^1)$ instead of traversing $p(L_f^2)$, to still reach node $v(L_f^1)$ earlier via path $p(L_f^1)$ than via partial path $p(L_f^2)$. When it is negative, it indicates that the vehicle should depart at least $\phi(L_f^1, L_f^2)$ time units earlier when traversing partial path $p(L_f^1)$ instead of traversing $p(L_f^2)$, to be able to reach node $v(L_f^1)$ earlier via path $p(L_f^1)$ then via partial path $p(L_f^2)$.

In Figure 2.3, we depict several simple examples: If there is no intersection between labels L_f^1 and L_f^2, $\phi(L_f^1, L_f^2)$ is positive when $\max(\text{dom}(L_f^1)) > \max(\text{dom}(L_f^2))$ (see Figure 2.3(a)), or negative when $\max(\text{dom}(L_f^1)) < \max(\text{dom}(L_f^2))$ (see Figure 2.3(b)). Otherwise, $\phi(L_f^1, L_f^2)$ can only be negative (see Figure 2.3(c)).

Finally, the dominance test is stated in Proposition 1 as follows:

Proposition 1. *Label L_f^2 is dominated by label L_f^1 if*
1. $v(L_f^1) = v(L_f^2)$
2. $U(L_f^1) \subseteq \widetilde{U}(L_f^2)$
3. $O(L_f^1) = O(L_f^2)$
4. $\delta_{L_f^1}(0) \leq \delta_{L_f^2}(0)$
5. $r(L_f^1) \geq r(L_f^2) - \phi(L_f^1, L_f^2)$

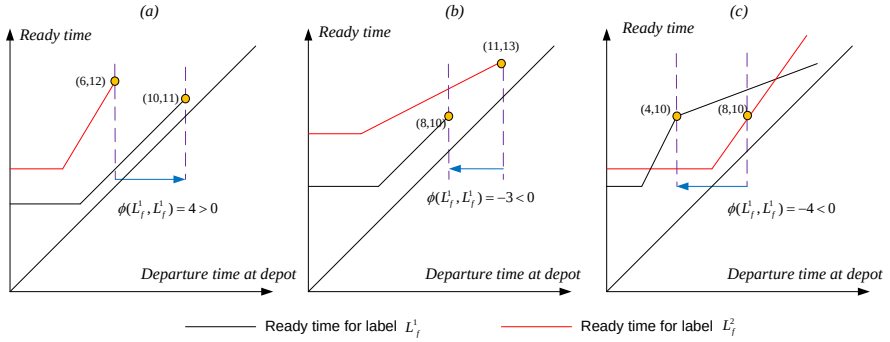

Figure 2.3: Illustration of $\phi(L_f^1, L_f^2)$.

6. $q(L_f^1) \leq q(L_f^2)$

PROOF OF PROPOSITION 2.1.

Consider two labels L_f^1 and L_f^2 that satisfy the six conditions in proposition 1. We need to show that (i) any feasible extension L that extends $p(L_f^2)$ to $2n+1$ is also a feasible extension for $p(L_f^1)$ to $2n+1$ and (ii) that for all these feasible extensions L it holds that $obj(L_f^1 \bigoplus L) \geq obj(L_f^2 \bigoplus L)$, where $L_f \bigoplus L$ is the label resulting from extending L_f with L.

With regards to point (i), first, capacity will not be violated along the path $p(L_f^1 \bigoplus L)$ as it was not violated on path $p(L_f^2 \bigoplus L)$ and by condition 6 it holds that $q(L_f^1) \leq q(L_f^2)$. Secondly, the path $p(L_f^1 \bigoplus L)$ is elementary. By conditions 2 and 3, all nodes visited in $p(L_f^1)$ are either nodes visited in $p(L_f^2)$ or nodes which could not be reached by any extension of label L_f^2 (i.e. the nodes of requests included in $\widetilde{U}(L_f^2) \setminus U(L_f^2)$). A feasible extension of L_f^2 cannot contain any node visited along path $p(L_f^2)$ or node which is unreachable from label L_f^2. Therefore, all nodes visited along path $p(L_f^1)$ are not visited in L, so path $p(L_f^1 \bigoplus L)$ is elementary as well. Third, there exists a departure time for path $p(L_f^1 \bigoplus L)$ which does not violate time windows. As L is a feasible extension of L_f^2 it means that there is a departure time at $v(L_f^1)$ after $\delta_{L_f^2}(0)$ making sure that all nodes in L are visited within their time windows. If condition 4 is met, the vehicle is via path $p(L_f^1)$ always able to reach $v(L_f^1)$ before this time. For example by departing at 0, the vehicle arrives at $v(L_f^1)$ at time $\delta_{L_f^1}(0)$ which is by condition 4 smaller than or equal to $\delta_{L_f^2}(0)$. Therefore, a departure at 0 over path $p(L_f^1 \bigoplus L)$ does not violate any time windows. In conclusion, any extension L of $p(L_f^2)$ to $2n+1$ will be a feasible extension of $p(L_f^1)$ to $2n+1$ as it results in an elementary path with does not violate time windows and capacity constraints.

Then for (ii), it still has to be proven that for all feasible extensions of L of $p(L_f^2)$ to $2n+1$ it holds that $obj(L_f^1 \bigoplus L) \geq obj(L_f^2 \bigoplus L)$. Let $L_f^{1*} = L_f^1 \bigoplus L$ and $L_f^{2*} = L_f^2 \bigoplus L$. We also denote $t_0^2 = argmin_{t \in dom(L_f^{2*})}\{\delta_{L_f^{2*}}(t) - t\}$ as the optimal departure time from the depot for path $p(L_f^{2*})$ and $r(L)$ as the sum of the profits associated with the nodes visited along path $p(L)$. The objective value of the path is:

$$\begin{aligned} obj(L_f^{2*}) &= r(L_f^{2*}) - (\delta_{L_f^{2*}}(t_0^2) - t_0^2) \\ &= r(L_f^2) + r(L) - (\delta_{L_f^{2*}}(t_0^2) - t_0^2) \end{aligned} \tag{2.36}$$

Now consider the path $p(L_f^{1*})$ resulting from extending L_f^1 by L. Moreover, consider a departure time at the depot of this path of $t_0^1 = \max\{0, t_0^2 + \phi(L_f^1, L_f^2)\}$. The time t_0^1 is a feasible departure time for label L_f^{1*} because a departure time of 0 is always possible (as the extension of L_f^1 by L is feasible) and by the definition of $\phi(L_f^1, L_f^2)$ in equation (2.35) $t_0^2 + \phi(L_f^1, L_f^2)$ belongs to $dom(L_f^1)$ if it is nonnegative. This departure time t_0^1 ensures that we reach node $v(L_f^1)$ at time $\delta_{L_f^2}(t_0^2)$ or earlier, meaning:

$$\delta_{L_f^1}(t_0^1) \leq \delta_{L_f^2}(t_0^2). \tag{2.37}$$

Moreover, as t_0^1 is a feasible departure time it can be used to compute a lower bound on $obj(L_f^{1*})$:

$$obj(L_f^{1*}) \geq r(L_f^{1*}) - (\delta_{L_f^{1*}}(t_0^1) - t_0^1) = r(L_f^1) + r(L) - (\delta_{L_f^{1*}}(t_0^1) - \max\{0, t_0^2 + \phi(L_f^1, L_f^2)\}) \tag{2.38}$$

$$\geq r(L_f^1) + r(L) - (\delta_{L_f^{2*}}(t_0^2) - \max\{0, t_0^2 + \phi(L_f^1, L_f^2)\}) \tag{2.39}$$

$$\geq r(L_f^1) + r(L) - (\delta_{L_f^{2*}}(t_0^2) - t_0^2 - \phi(L_f^1, L_f^2)) \tag{2.40}$$

$$\geq r(L_f^2) - \phi(L_f^1, L_f^2) + r(L) - (\delta_{L_f^{2*}}(t_0^2) - t_0^2 - \phi(L_f^1, L_f^2)) \tag{2.41}$$

$$\geq r(L_f^{2*}) + r(L) - (\delta_{L_f^{2*}}(t_0^2) - t_0^2) = obj(L_f^{2*}) \tag{2.42}$$

Note that in inequality (2.39) we use the property derived at (2.37). Inequality (2.40) is derived by the simple fact that $\forall x \in R : -\max\{0, x\} \leq -x$, and inequality (2.41) uses condition 5 of

Proposition 1. □

2.4.2 The backward labeling algorithm

In the backward labeling algorithm we start in the opposite direction and start generating labels from the end depot. The definition of a backward label is discussed in Section 2.4.2. The labels are extended as discussed in Section 2.4.2. The dominance criterion for backward labels is discussed in Section 2.4.2.

Backward label

In the backward labeling algorithm, labels are extended from the end depot $2n + 1$ to its predecessors. For a label L_b, we associate the following components:

$p(L_f)$		The partial path of label L_b.
$v(L_b)$	*	The first node visited on the partial path $p(L_b)$.
$L^{-1}(L_b)$	*	The parent label from which L_b originates by extending it with $v(L_b)$.
$O(L_b)$	*	The set of incomplete requests, i.e., the delivery is visited but not the pickup node.
$U(L_b)$	*	The set of requests for which the delivery nodes are already visited along the partial path $p(L_b)$. It contains both the complete and incomplete requests. Therefore, $O(L_b) \subseteq U(L_b)$.
$P(L_b)$		The set of pickup nodes of incomplete request in $p(L_b)$, i.e., $j \in N_P$ and $R_{j+n} \in O(L_b)$.
$D(L_b)$		The set of delivery nodes not visited in $p(L_b)$, .i.e., $j \in N_D$ and $R_{j-n} \notin U(L_b)$.
$q(L_b)$	*	The load of the tour after visiting node $v(L_b)$.
$\delta_{L_b}(t)$	*	The arrival time at the end node $2n + 1$ through the partial path represented by L_b when leaving node $v(L_b)$ at time t.
$r(L_b)$	*	The overall profits collected with the requests completed on the partial path $p(L_b)$.

Again, only the items marked with a * are stored in the label and the sets $D(L_b)$ and $P(L_b)$ can be deduced from the sets $O(L_b)$ and $U(L_b)$. Furthermore, the partial path can be deduced from iteratively checking the first node visited in the parent label of which the this label was an extension.

Label extension

Let dom(L_b) be the domain of the function $\delta_{L_b}(t)$ and let $t_l(L_b)$ denote the latest possible ready time at $v(L_b)$: $t_l(L_b) = \max(\text{dom}(L_b))$. We extend a label $L_b^{'}$ along an arc $(j, v(L_b^{'}))$ to create a new label L_b. To be a feasible extensions at least the following two conditions should

be met:

$$t_l(L_b) \geq \max\{t_m, e_j + s_j\} \quad \wedge \quad j \in N \setminus \{2n+1\} \tag{2.43}$$

$$q(L_b) + q_j \leq Q \quad \wedge \quad j \in N \setminus \{2n+1\} \tag{2.44}$$

Condition(2.43) ensures that node j can be reached within its time window and that the extension will be stopped before t_m is exceeded, while condition (2.44) ensures capacity feasibility. Furthermore, $L_b^{'}$ and j must satisfy one of the following three conditions:

$$j + n \in O(L_b^{'}) \quad \wedge \quad j \in N_P \tag{2.45}$$

$$j \notin U(L_b^{'}) \quad \wedge \quad j \in N_D \tag{2.46}$$

$$O(L_f^{'}) = \emptyset \quad \wedge \quad j = 0 \tag{2.47}$$

Condition (2.45) indicates that if j is a pickup node, the corresponding delivery node should have been visited already. Furthermore, condition (2.46) states that if j is a delivery node, it should not have been visited before. Finally, Condition (2.47), states that if j is the begin depot then all visited requests should have been completed. In the presence of those conditions, only elementary paths that satisfy precedence constraint (2.11) are generated.

At last, it needs to be checked that all pickup nodes of incomplete requests for which the delivery node is included in $p(L_b^{'})$ can be visited before node j. To do so, first the latest possible arrival time at node j which ensures a ready time of $t_l(L_b)$ at $v(L_b)$ should be determined. Let this be denoted by $t_r(j, L_b^{'})$, then it holds that $t_r(j, L_b^{'}) = \min\{l_j, \max\{t : t + s_j + \tau_{jv(L_b^{'})}(t) + s_{v(L_b^{'})} \leq t_l(L_b^{'})\}$. As shown $t_r(j, L_b^{'})$ is determined by the latest possible arrival time at j (i.e. l_j) or by the latest possibility departure time to reach $v(L_b^{'})$ on time.

Then, all nodes from which the vehicle cannot depart before time $t_r(j, L_b^{'})$ due to time window constraints, cannot be visited before node j and are defined as unreachable. This can be made stronger by considering the travel time to node j as well. However, again due to the absence of the triangle inequality and the time dependent travel times we cannot simply consider the travel time of the direct connection. Therefore, we need for any unvisited node k the latest possible departure time to arrive at j at time $t_r(j, L_b^{'})$, which can be computed by $\max\{t : t+\tau_{kj}(t) \leq t_r(j, L_b^{'}))\}$. Finally, the latest possible time the vehicle could depart from any node before node j is given by $t_d(j, L_f^{'}) = \max_{k \in P(L_f^{'}) \cup D(L_f^{'})}\{\max\{t : t + \tau_{kj}(t) \leq t_r(j, L_b^{'})\}\}$.

This means that any node k with the earliest allowed arrival time e_k later than this time (i.e.

$l_k > t_d(j, L_f^{'})$) cannot be a predecessor of node j, also not in an indirect way as no node could be left after $t_d(j, L_f^{'})$ and still reaching node j on time.

If a pickup node k cannot be a predecessor of node j and its corresponding delivery node is already visited (i.e $k + n \in O(L_f')$), the extension to j is not feasible as the item which should be delivered cannot be picked up anymore. Therefore, as stated in condition (2.48), all pickup nodes of requests of which the delivery node is already visited in $p(L_b^{'})$ should be a possible predecessor of j to make sure that extending label $L_b^{'}$ to j is feasible.

$$l_k \leq t_d(j, L_b^{'}) \quad \forall k \in P(L_b^{'}) : k \neq j \tag{2.48}$$

If the extension along the arc $(j, L_b^{'})$ is feasible according to the provided conditions, the information in label L_b is set as follows:

$$L^{-1}(L_b) = L_b^{'} \tag{2.49}$$

$$v(L_b) = j \tag{2.50}$$

$$\delta_{L_b}(t) = \delta_{L_b^{'}}(\max\{e_{v(L_b^{'})}, t + \tau_{jv(L_b^{'})}(t)\} + s_{v(L_b^{'})}) \tag{2.51}$$

$$q(L_b) = q(L_b^{'}) + q_j \tag{2.52}$$

$$r(L_b) = \begin{cases} r(L_b^{'}) + r_j & \text{if } j \in N_P, \\ r(L_b^{'}) & \text{otherwise.} \end{cases} \tag{2.53}$$

$$O(L_b) = \begin{cases} O(L_b^{'}) \setminus \{j\} & \text{if } j \in N_P, \\ O(L_b^{'}) \cup \{j - n\} & \text{if } j \in N_D. \end{cases} \tag{2.54}$$

$$U(L_b) = \begin{cases} U(L_b^{'}) \cup \{j - n\} & \text{if } j \in N_D, \\ U(L_b^{'}) & \text{otherwise.} \end{cases} \tag{2.55}$$

Label dominance

Dominance of the backward algorithm can be constructed in the same way as in the case of the forward algorithm, because the arrival time functions are non-decreasing and stepwise

linear as before.

Similar to the forward algorithm, in Proposition 2, we extend the set $U(L_b)$ to $\widetilde{U}(L_b)$ by including request of which the pickup or delivery node cannot be a predecessor of $v(L_b)$. Via the same reasoning as in Section 2.4.2 it can be derived that the latest possible time a vehicle could depart from a node to reach $v(L_b)$ on time is $t_d(L_b) = \max_{j \in P(L_b) \cup D(L_b)}\{\max\{t : t + \tau_{jv(L_b)}(t) \leq t_l(L_b) - s_{v(L_b)}\}\}$. This means that any node j with the earliest possible departure time $e_j + s_j$ later than this time (i.e. $e_j + s_j > t_d(L_b)$) cannot be a predecessor of $v(L_b)$. Therefore, the corresponding request cannot be served anymore and can be added to the set $\widetilde{U}(L_b)$.

Furthermore, we define $\phi(L_b^1, L_b^2)$ (see Figure 2.4) as:

$$\phi(L_b^1, L_b^2) = \max\{x \in R : \delta_{L_b^1}(t) + x \leq \delta_{L_b^2}(t), \quad \forall t \in dom(L_b^2)\} \tag{2.56}$$

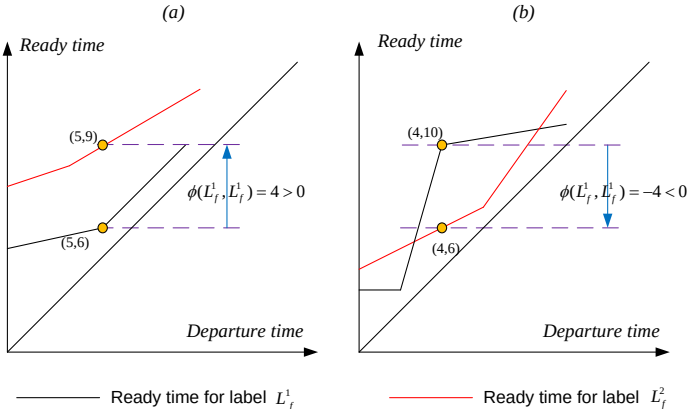

Figure 2.4: Illustration of $\phi(L_f^1, L_f^2)$.

Then the dominance criterion will become:

Proposition 2. *Label L_b^2 is dominated by label L_b^1 if*

1. $v(L_b^1) = v(L_b^2)$
2. $U(L_b^1) \subseteq \widetilde{U}(L_b^2)$
3. $O(L_b^1) = O(L_b^2)$
4. $t(L_b^1) \geq t(L_b^2)$

5. $r(L_b^1) \geq r(L_b^2) - \phi(L_b^1, L_b^2)$ 6. $q(L_b^1) \leq q(L_b^2)$

For the proof of proposition 2 the same reasoning as the proof of proposition 1 could be followed.

2.4.3 *Merging forward and backward labels*

When all forward and backward labels are generated, they are merged to construct feasible profitable tours. A forward label L_f and a backward label L_b can be merged if the following conditions are satisfied:

1. $v(L_f) = v(L_b)$
2. $O(L_f) \cap O(L_b) = \{v(L_f)\}$
3. $(U(L_f) \setminus O(L_f)) \cap (U(L_b) \setminus O(L_b)) = \emptyset$
4. $q(L_f) + q(L_b) = q_{v(L_f)}$
5. $Img(L_f) \cap dom(L_b) \neq \emptyset$

The resulting path $p(L) = (p(L_f) \bigoplus p(L_b))$ has the following attributes:

1. $v(L) = 2n + 1$
2. $r(L) = r(L_f) + r(L_b) - r_{v(L_f)}$
3. $O(L) = \emptyset$
4. $U(L) = U(L_f) \cup U(L_b)$
5. $q(L) = 0$
6. $\delta_L(t) = \delta_{L_b}(\delta_{L_f}(t)), \forall t \in dom(L_f)$ such that $\delta_{L_f}(t) \in dom(L_b)$

However, this bidirectional labeling algorithm can generate duplicate solutions. Consider a feasible solution p^* including nodes i, j and k in this order. Each node $x \in p^*$ is associated with a forward label $L_f(x)$ and a backward label $L_b(x)$ (i.e. $v(L_f(x)) = v(L_b(x)) = x$). Therefore, the path p^* can be obtained by merging $L_f(i)$ with $L_b(i)$ as well as merging by $L_f(j)$ with $L_b(j)$. To overcome this drawback, we devised an additional test: we accept a solution only when a further extension of the forward label is impossible. In our example (see Figure 2.5) the extension from $L_f(i)$ to node j is feasible and the extension from $L_f(j)$ to node k is infeasible by the predefined fixed time t_m. We generate solution p^* by merging $L_f(j)$ and $L_b(j)$ instead of $L_f(i)$ and $L_b(i)$. The test is performed for each candidate pair of labels and guarantees that each path is generated only once.

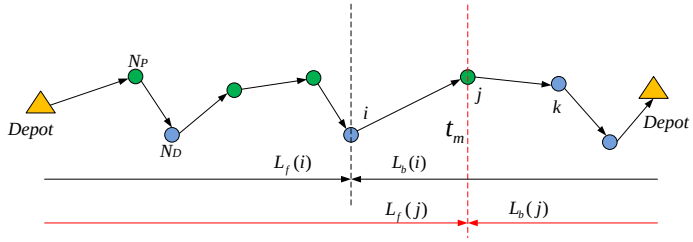

Figure 2.5: An Illustration of Preventing Duplicate Solutions.

2.5 Restricted dynamic programming heuristic algorithm

Although the tailored labeling algorithm returns the optimal solution of our proposed problem, it is not fast enough to solve problems of realistic size within reasonable computation time.

In order to avoid enormous computation times and to save memory usage, Malandraki and Dial [72] proposed a restricted dynamic programming heuristic algorithm for the TSP. The main concept is to limit in each stage the number of partial paths for further extension by a given parameter B. Furthermore, in each stage all partial paths should have the same number of visited nodes. Malandraki and Dial [72] show that increasing the value of B results in better solutions, but also in substantial higher computation times. Note that $B = 1$ results in a nearest neighborhood heuristic and $B = \infty$ results it an exact dynamic programming algorithm.

Gromicho et al. [49] propose a restricted dynamic programming algorithm for solving realistic VRPs and restrict the state space even further, by using a form of beam search (Bisiani. [13]), which means that each partial path is only expanded to E of its nearest feasible nodes.

The same principle of restricting the number of extension of partial paths can be applied to our labeling algorithm as well with some minor changes. Because of the precedence constraints, expanding a partial path to a pickup node may improve or deteriorate the objective function value depending on the profit of that node and the additional travel time to visit that node. However, expanding to a delivery node can only decrease the objective function value as additional travel time is necessary to visit that node while no profit is assigned to the node. Moreover, because of the time-dependent travel time, the objective function value of a partial path is determined by its optimal departure time at the origin depot, which might change after a certain extension.

Therefore, it is not sufficient to select the partial paths with the highest objective function value, and we introduce a new selection method for each stage taking global information into account. For every extension of a partial path the $E/2$ best expanded partial paths ending with a pickup node and the $E/2$ best expanded partial paths ending with a delivery node are selected. Furthermore, in each stage we select the best $B/2$ partial paths that expand to a pickup node and the best $B/2$ partial paths that expand to a delivery node for further expansion. Moreover, instead of just using the objective function value for the selection, we use the earliest arrival time to order the expanded partial paths. If two expanded partial paths have the same earliest arrival time, the objective function value will be used to order them.

The restricted dynamic programming heuristic is described in Algorithm 1. Herein, an empty labels set Z_k is created for each stage k for further extension (Step 2). Then the first label which represents the start from the depot is generated and put into the set Z_0 (Step 3 and Step 4). If the number of the labels stored in each stage k is larger than B, only the B best labels are selected for further extension (Step 7). Moreover, for each label in stage k, only E of its feasible extensions are made and put in Z_{k+1} (Step 9). In the end, the best route Rt^* will be found by checking the labels in Z_{2n+1} (Step 10).

Algorithm 1: The restricted dynamic programming heuristic

input : Request size n with two parameters B and E
output: The best route Rt^*

1 **for** *(each stage* $k = 0, 1, ..., 2n + 1$*)* **do**
2 $\quad$ $Label\ set\ Z_k = \emptyset$
3 $L_0 \leftarrow \{v(L_0), \delta_{L_0}(t), O(L_0), U(L_0), q(L_0), r(L_0)\}$
4 $Z_0 \leftarrow Z_0 \cup \{L_0\}$
5 **for** *(each stage* $k = 0, 1, ..., 2n$*)* **do**
6 $\quad$ **if** $(Size\ of\ Z_k \geq B)$ **then**
7 $\quad\quad$ $Keep\ the\ B\ best\ labels\ in\ Z_k$
8 $\quad$ **for** *(each Label* $L_k \in Z_k$*)* **do**
9 $\quad\quad$ $Do\ E\ of\ its\ feasible\ extensions.\ Put\ the\ non-dominated\ labels\ into\ Z_{k+1}$
10 $Find\ the\ best\ route\ Rt^*\ in\ set\ Z_{2n+1}$

Note that there is no optimality guarantee anymore with such a restricted dynamic programming heuristic algorithm.

2.6 Computational results

In this section the computational experiments are discussed. The algorithms are coded in JAVA and all computations are carried out on a single thread of a server with four CPU's

(2.4 GHz/6 cores) and 64GB RAM. Each run has a time limit of 2 weeks and a maximum memory allowance of 16GB RAM. For our numerical study we use an adapted version of the instances for the PDPTW in Røpke and Pisinger [93]. In these instances, the coordinates of each pickup and delivery location are both randomly and uniformly distributed over a $[0, 50] \times [0, 50]$ square and a single depot is located in this square. The load q_i for each request R_i is also randomly selected from the interval $[5, Q]$, where Q is the vehicle capacity. Table 2.2 summarizes the characteristic of these instances including the vehicle capacity Q and the width of the time windows W. For each group, there are 10 instances with the number of requests ranging from 30 to 75. For more details, we refer to Røpke and Pisinger [93].

Table 2.2: Characteristics of the TDPPDPTW and its variants instances

Group	Q	W
AA	15	60
BB	20	60
CC	15	120
DD	20	120

In addition to the instances proposed by Røpke and Pisinger [93], we created for each class four new small instances with the amount of requests ranging from 10 to 25. They are generated by only keeping the first α requests of the instance with 30 requests of each class, where $\alpha = 10$, 15, 20 and 25. For example, the instance AA10 is created by considering the first ten requests of AA30.

In our instances, the coordinates, time windows and loads of the depot, pickup nodes and delivery nodes are the same as in the instances of Røpke and Pisinger [93]. Furthermore, the vehicle capacity Q is also the same. In the instances of Røpke and Pisinger [93] the travel time is fixed and no profits were assigned to the requests.

To make the travel time time-dependent, road congestion is handled by a so-called speed model which consists of different speed profiles. It is used to determine the travel time between two nodes on a specific departure time. This speed model is based on the speed model of Verbeeck et al. [117] for the TDOP and Dabia et al. [25] for the TDVRPTW. Without loss of generality, we assume that breakpoints are the same for all speed profiles as congestion tends to happen around the same time regardless of the type of speed profile. The pickup and delivery node of each request is randomly assigned to one of the three predefined areas: morning and evening commuting area, city center and highways. Then, the speed profile of a link is assigned according to the type (e.g. depot or request node) and location of the tail node

and head node.

In our speed model, the planning horizon covers 14 working hours (840 minutes, from 7 am to 9 pm) and a minute is set to be one unit of time, while, in Røpke and Pisinger [93], the planning horizon of length 600 minutes is considered. Each speed profile has four non-overlapping time periods with constant speed, reflecting two congested periods and two periods with normal traffic conditions. Five speed profiles are included (see Figure 2.6 and Table 2.3):

- Slow speed (SS): these links represent a busy central business district (CBD) with a lot of traffic during the whole day.
- Normal speed with morning peak (NSMP): these links represent roads leading from a residential area to the CBD. These roads are in most cases congested in the morning.
- Normal speed with evening peak (NSEP): these links represent roads leading from a CBD to a residential zone. The roads typically encounter evening congestion.
- Fast speed with two peaks (FSTP): these links represent roads near the highway with a morning and evening peak in both directions.
- High speed (HS): these links connect the request nodes with the depot.

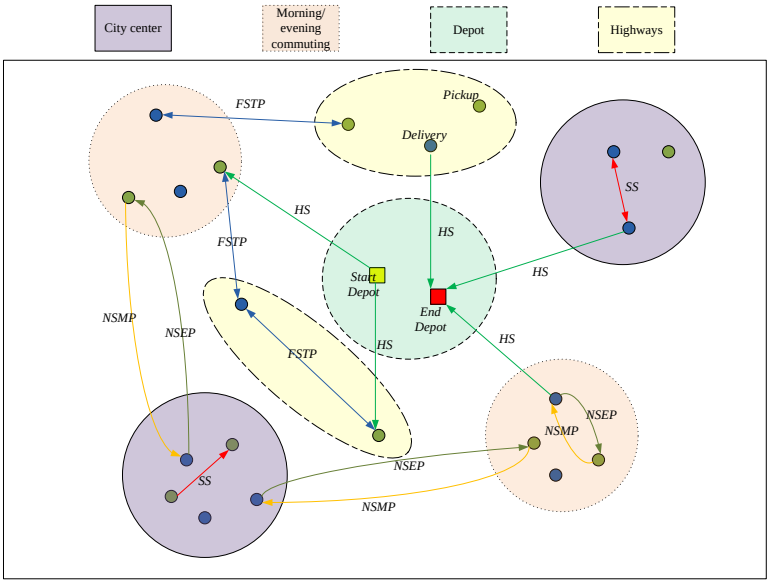

Figure 2.6: An Illustration of a Instance with Different Speed Profiles.

As in the PDPTW (Røpke and Pisinger [93]), all requests needed to be served, so no selection

Table 2.3: Speed Profiles

Congestion description Time periods	Morning peak 7 am-9 am	Normal 9 am-5 pm	Evening peak 5 pm-7 pm	Normal 7 pm-9 pm
1.SS	0.5	0.81	0.5	0.81
2.NSMP	0.67	1.33	0.88	1.33
3.NSEP	0.88	1.33	0.67	1.33
4.FSTP	0.85	1.5	0.85	1.5
5.HS	1.0	2.0	1.0	2.0

is required and no profits are considered. To come up with meaningful settings for the profits we did preliminary tests with 20, 40 and 80 units of profit assigned to each pickup node. As shown in Table 2.4, if the profit of the requests is 20, it will be too low. There are even routes in which it is best to do nothing. If the profit per request is 40, on average 56% of the requests are served and the total profit in the route is 52% higher than the traveling cost. When the profits increase to 80, the provider would like to serve more request (on average 74%) and the generated profits are way (on average 141%) higher than the traveling cost. To avoid bias on either profits or traveling cost, in the remainder of the computational experiments, we decided to assign a profit of 40 units to all pickup nodes, which makes a request only profitable if the additional travel time to serve the request (i.e. visiting both the pickup and delivery node) is less than 40 time units.

Table 2.4: Comparison of results for different profits settings

Instance	20 units profit for each request			40 units profit for each request			80 units profit for each request		
	Optimal value	Profits (Served)	Traveling cost	Optimal value	Profits (Served)	Traveling cost	Optimal value	Profits (Served)	Traveling cost
AA10	0	0 (0)	0	27.14	80 (2)	52.86	196.22	400 (5)	203.78
AA15	0	0 (0)	0	37.77	120 (3)	82.23	392.37	880 (11)	487.63
AA20	0	0 (0)	0	73.7	240 (6)	166.3	566.03	1200 (15)	633.97
AA25	9.06	60 (3)	50.94	203.01	560 (14)	356.99	884.41	1440 (18)	555.59
AA30	13.07	60 (3)	46.93	266.72	640 (16)	373.28	1020.86	1600 (20)	579.14
BB10	0	0 (0)	0	39.26	120 (3)	80.74	247.65	720 (9)	472.35
BB15	6.61	20 (1)	13.39	99.58	280 (7)	180.42	518.03	960 (12)	441.97
BB20	6.61	20 (1)	13.39	138.05	600 (15)	461.95	738.05	1200 (15)	461.95
BB25	6.25	40 (2)	33.75	147.88	520 (13)	372.12	787.64	1360 (17)	572.36
BB30	4.26	20 (1)	15.74	274.45	840 (21)	565.55	1114.45	1680 (21)	565.55
CC10	2.81	40 (2)	37.19	57.63	160 (4)	102.37	340.2	560 (7)	219.8
CC15	5.69	40 (2)	34.31	98.29	280 (7)	181.71	469.59	960 (12)	490.41
CC20	4.35	20 (1)	15.65	97.27	480 (12)	382.73	712.53	1280 (16)	567.47
CC25	9.42	60 (3)	50.58	226.23	720 (18)	493.77	984.3	1600 (20)	615.7
CC30	3.73	60 (3)	56.27	316.39	800 (20)	483.61	1167.47	1760 (22)	592.53
DD10	11.8	60 (3)	48.2	85.88	160 (4)	74.12	315.34	720 (9)	404.66
DD15	10.18	60 (3)	49.82	99.05	200 (5)	100.95	554.55	1040 (13)	485.45
DD20	14.44	80 (4)	65.56	182.14	640 (16)	457.86	832.05	1360 (17)	527.95
DD25	6.66	80 (4)	73.34	250.16	680 (17)	429.84	1025.75	1600 (20)	574.25
DD30	16.9	80 (4)	63.1	343.65	840 (21)	496.35	1247.22	1840 (23)	592.78

2.6.1 *Performance of the tailored labeling algorithm*

Table 2.5 gives the results of the experiments using the tailored labeling algorithm without dominance and feasibility check and the tailored labeling algorithm with dominance and feasibility check. In Table 2.5, we observe that the tailored labeling algorithm without dominance and feasibility check can only solve 11 instances (out of 20 instances) to optimality within the given memory limit (16GB) or time limit (1 day). In contrast, the tailored labeling algorithm with dominance and feasibility check is able to solve all of the instances within a reasonable running time (within 15 mins). Moreover, due to the dominance and feasibility check much less labels and routes (column 7 and 8) are generated compared to its counterpart (column 3 and 4). Therefore, it shows that the proposed dominance criterion and feasibility check both show their great potentials to get rid of the unpromising labels (column 9 and 10) which are proven not to be the part of the optimal solution. On average 84% of the labels are dominated and about 11% is proven to lead to an infeasible extension.

Table 2.5: Comparison of the tailored labeling algorithm without and with dominance and feasibility check

Instance	TDTL without dominance and feasibility check				TDTL with dominance and feasibility check					
	Optimal Value	Generated labels(#)	Generated routes(#)	Time(s)	Optimal Value	Generated labels(#)	Generated routes(#)	Dominated labels(#)	Infeasible labels(#)	Time(s)
AA10	27.14	1680	19	0.468	27.14	196	15	121	0	0.234
AA15	37.77	67201	57	0.998	37.77	471	26	329	0	0.374
AA20	73.7	1295435	658	12.044	73.7	1183	49	886	12	0.796
AA25	-	-	-	Out of memory	203.01	10174	80	8113	974	1.591
AA30	-	-	-	Out of memory	266.72	18719	77	15012	2247	2.792
BB10	39.26	3647	43	0.375	39.26	204	21	104	0	0.219
BB15	99.58	387803	5311	3.932	99.58	1729	52	1023	263	0.499
BB20	138.05	14067769	420587	30019.417	138.05	11572	89	9445	1565	1.95
BB25	-	-	-	Out of memory	147.88	13787	145	10660	1886	2.247
BB30	-	-	-	Out of memory	274.45	40335	152	31755	7019	3.167
CC10	57.63	4859	70	0.437	57.63	289	25	184	0	0.265
CC15	98.29	119687	1455	1.919	98.29	1799	43	1501	19	0.625
CC20	97.27	5763951	14981	47.207	97.27	5206	82	4337	215	1.248
CC25	-	-	-	86400	226.23	97782	278	81472	6626	11.951
CC30	-	-	-	Out of memory	316.39	347803	490	284760	42498	78.939
DD10	85.88	5567	55	0.608	85.88	222	20	129	0	0.25
DD15	99.05	1041425	2272	5.054	99.05	1471	53	850	267	0.609
DD20	-	-	-	Out of memory	182.14	11292	111	8473	1735	2.106
DD25	-	-	-	Out of memory	250.16	405617	630	324304	48709	199.735
DD30	-	-	-	Out of memory	343.65	1257385	547	1085122	128811	855.204

In Table 2.6, we present the results of the experiments using the mathematical model introduced in Section 2.3.2, solved by the optimization software Gurobi (version 5.6) with its default parameter settings and a computation time limit of 1 day. For each instance, we provide the lower bound (LB), best feasible solution found (Best), the processing time given in seconds (Time) and the gap (*Gap*) provided by Gurobi and the relative gap with respect to the optimal solution (Gap_{opt}). The relative gap is calculate by $Gap_{opt} = 100 \times \frac{Best-Opt}{Best}$ in which *Opt* is the value of the optimal solution for the instance derived by our tailored

labeling algorithm. As shown in Table 2.6, the proposed mathematical model is able to solve instances with up to 25 requests within the time limit. However, for larger instances, it takes already more than one day compared to the at maximum 15 minutes of our tailored labeling algorithm.

Table 2.6: Results of solving the mathematical model by using the Gurobi

Instances	LB	Best	Time(s)	Gap(%)	Gap_opt(%)
AA10	27.14	27.14	1.95	0	0
AA15	37.77	37.77	4.664	0	0
AA20	73.7	73.7	24.93	0	0
AA25	203.01	203.01	46344.063	0	0
AA30	216.34	543.475	86400	151	50.923
BB10	39.26	39.26	1.451	0	0
BB15	99.58	99.58	121.029	0	0
BB20	138.05	138.05	1669.737	0	0
BB25	147.88	147.88	48616.51	0	0
BB30	245.7	515.404	86400	110	46.751
CC10	57.63	57.63	3.292	0	0
CC15	98.29	98.29	44.337	0	0
CC20	97.27	97.27	7935.288	0	0
CC25	209.088	1316.688	86400	530	82.818
CC30	237.02	1674.739	86400	607	81.108
DD10	85.88	85.88	1.435	0	0
DD15	99.05	99.05	54.648	0	0
DD20	182.14	182.14	4959.477	0	0
DD25	221.391	1419.806	86400	541	82.381
DD30	279.51	1733.624	86400	520	80.177

2.6.2 Results of the tailored labeling algorithm on all instances

In Table 2.7, we report the results of our tailored labelling algorithm. We present the computation times for both the bidirectional search (i.e $t_m = 420$, the middle of the time horizon) and the mono-directional search (i.e $t_m = 840$, the end of the time horizon). Since both algorithms lead to the optimal value, this value is presented only once (column 2). The bidirectional search shows greater potential power in terms of computation time. For some instances (DD40 and DD50), the bidirectional search is even 10 times faster than the monodirectional search. Moreover, the bidirectional search is capable of finding the optimal value of more instances than the monodirectional search within our set computation time limit of 2 weeks. This is mainly because of the fact that the number of labels needed to be processed in the bidirectional search is considerably less compared to the mono-directional search. However, for several easy-to-solve instances (solved to optimality within 6 minutes) in AA group and BB group (AA35 and BB45), the mono-directional search outperforms the

bidirectional search. On the one hand, the total number of the generated labels for these instances are comparably limited. On the other hand, this phenomenon indicate that the merging forward and backward labels also takes a significant portion of processing times.

With the bidirectional search, 34 of the 40 instances could be solved within our time limit of 2 weeks. We did not find optimal solutions for the instances with more than 65 requests in the CC group and more than 55 requests in the DD group. This means that the algorithm was still generating labels after 2 weeks, and therefore also no upper bound is available.

Table 2.7: Monodirectional Algorithm vs. Bidirectional Algorithm

Instance	Optimal Value	Requests Served	Processing Time (s)	
			Monodirectional	Bidirectional
AA30	266.72	16	2.37	2.12
AA35	278.14	20	4.68	5.04
AA40	341.74	22	7.94	5.82
AA45	361.57	21	23.54	5.6
AA50	430.41	23	22.40	9.53
AA55	436.05	27	32.79	18.66
AA60	505.02	26	87.11	24.71
AA65	576.39	29	182.09	34.80
AA70	606.68	29	180.01	39.14
AA75	715.53	31	1038.67	257.60
BB30	274.45	21	2.36	2.78
BB35	337.55	22	1.30	1.03
BB40	326.82	23	8.30	9.95
BB45	378.90	23	34.88	27.94
BB50	432.10	25	31.81	37.41
BB55	530.84	27	40.14	59.02
BB60	558.02	28	164.38	119.2
BB65	547.84	25	353.15	332.13
BB70	558.29	27	225.79	351.57
BB75	613.35	27	775.58	511.37
CC30	316.39	20	32.51	27.64
CC35	386.26	24	133.15	104.18
CC40	464.28	25	2934.28	942.27
CC45	497.68	26	20641.34	3422.21
CC50	519.60	26	45532.40	13913.43
CC55	581.50	28	156770.32	28429.25
CC60	624.95	30	$\geq$ 2 weeks	52801.65
CC65	663.31	31	$\geq$ 2 weeks	699831.05
DD30	343.65	21	396.23	80.42
DD35	410.19	22	2815.31	338.16
DD40	490.92	25	53402.91	4618.11
DD45	540.17	27	67293.53	7667.97
DD50	610.07	30	186697.41	21889.86
DD55	639.75	29	$\geq$ 2 weeks	823755.21

It is not a surprise that the computation time increases with the number of requests. However, Table 2.7 also demonstrates that the computation time increases if the vehicle capacity (i.e BB compared with AA and DD compared with CC) or time window width (CC compared with AA and DD compared with BB) increases. This is mainly caused as a larger vehicle capacity or time window width will lead to a larger solution space.

In Table 2.8 we present a more detailed look at the solution of instance AA30. The solution only serves 16 requests in the reported sequence. Since the time dependent travel time instead of the traveled distance is considered as part of the objective, the vehicle's departure time becomes crucial. Therefore, delaying the departure time of the vehicle may lead to less traveling cost. We observe that the vehicle departs from the origin depot at 208.23 and arrives to the destination depot at 581.51. Other departure times will lead to higher travel cost.

Table 2.8: Solution of AA30

Start time	End time	Travel cost	Served request	Generated profit
208.23	581.51	373.28	1, 11, 3, 26, 22, 18, 28, 25, 19, 8, 5, 13, 15, 7, 4, 27	640

In Table 2.9, we also present the results on small instances if we double the capacity of the vehicles. On one hand, vehicles with more capacity can serve more requests, which returns a better objective than its less capacitate counterparts. On the other hand, for some groups, the instances with larger vehicle capacities need much more time to get the optimal solution (e.g. CC25 and DD25).

Table 2.9: Comparison of the different capacity settings

Instances	Original Ropke's capacity					Double Ropke's capacity				
	Optimal Value	Profits	Requests served	Travel cost	Time(s)	Optimal Value	Profits	Requests Served	Traveling cost	Time(s)
AA10	27.14	80	2	52.86	0.234	27.14	80	2	52.86	0.297
AA15	37.77	120	3	82.23	0.374	79.63	200	5	120.37	0.952
AA20	73.7	240	6	166.3	0.796	146.41	440	11	293.59	2.574
AA25	203.01	560	14	356.99	1.591	286.68	600	15	313.32	29.469
BB10	39.26	120	3	80.74	0.219	46.49	200	5	153.51	0.484
BB15	99.58	280	7	180.42	0.499	131.25	320	8	188.75	2.652
BB20	138.05	600	15	461.95	1.95	192.46	600	15	407.54	11.779
BB25	147.88	520	13	372.12	2.247	268.91	800	20	531.09	20.717
CC10	57.63	160	4	102.37	0.265	96.56	240	6	143.44	0.624
CC15	98.29	280	7	181.71	0.625	121.9	320	8	198.1	2.745
CC20	97.27	480	12	382.73	1.248	147.505	520	13	372.495	5.335
CC25	226.23	720	18	493.77	11.951	434.01	960	24	525.99	10808.19
DD10	85.88	160	4	74.12	0.25	99.13	160	4	60.87	0.312
DD15	99.05	200	5	100.95	0.609	108.55	200	5	91.45	1.373
DD20	182.14	640	16	457.86	2.106	233.61	640	16	406.39	19.298
DD25	250.16	680	17	429.84	199.735	452.94	960	24	507.06	82620.399

2.6.3 *Performance of the restricted dynamic programming heuristic*

We also have conducted computational experiments to analyze the solution quality produced by the restricted dynamic programming heuristic. In Table 2.7, we have seen that the exact procedure has enormous computation times. In Table 2.10, we show the results for the same instances but now with the restricted dynamic programming heuristic. The restricted dynamic programming heuristic algorithm has been run with different values for B. In these first tests we set $E = \infty$. For the instances of which the optimal solution is not known, we compare the solution to the best found solution (i.e. the solution found in case $B = 10000$) to have a lower bound on the gap.

With a value of $B = 1000$ the algorithm leads in on average 71.26 seconds to the optimal solution for 25 out of the 40 instances. For 1 instance it is not sure whether or not the optimal solution is reached. The average gap is at least 0.79%. With a value of $B = 2500$ the average computation time increased to 237.42 seconds. For one more instance the optimal solution is found, and again there is 1 instance for which it is not sure whether or not the optimal solution is reached. The average gap (at least 0.42%) is about half of the gap as in the case of $B = 1000$. With a value of $B = 5000$ the average gap is halved again to at least 0.22% and the number of instances for which the optimal solution is found increases considerably to 30. For 3 more instances the solution found could be the optimal one. However, the average computation time is also increased to 580.44 seconds. Increasing the value of B to 10000 ensures that 32 of the 34 instances of which we know the optimal solution are solved to optimality. The average computation time in case $B = 10000$ equals 1567.3 seconds. The two instances which were not solved to optimality have respectively a gap of 0.48% and 0.11%. This means that, compared to the tailored labeling algorithm, the average gap of the restricted dynamic programming heuristic algorithm over the 34 instances of which the optimal solution is known is just 0.02%.

Note that all the instances for which the exact labeling algorithm was not able to solve in 2 weeks (e.g. $DD75$ in Table 2.7), can be solved within 20 hours with this restricted dynamic programming algorithm.

In Table 2.11, we present the impact of different E values on the performances of the algorithm with $B = 10000$. We set the values of E to $0.5n$, $0.25n$ and $0.125n$ and fractional values are rounded to its closed integer.

The results indicate that the computation times decrease if the value for E decreases, but not that much. As can be seen in Table 2.11, when the value of E is equal to $0.5n$, the solution

Table 2.10: Impact of Different B Values on Instances in Table 2.7

		B=1000			B=2500			B=5000			B=10000		
Instance	Optimal	Value	Time(s)	Gap	Value	Time(s)	Gap	Value	Time(s)	Gap	Value	Time(s)	Gap
AA30	266.72	266.72	3.32	0.00	266.72	3.03	0.00	266.72	3.14	0.00	266.72	3.01	0.00
AA35	278.14	278.14	4.43	0.00	278.14	4.17	0.00	278.14	4.10	0.00	278.14	4.31	0.00
AA40	341.74	341.74	4.06	0.00	341.74	5.16	0.00	341.74	5.09	0.00	341.74	4.23	0.00
AA45	361.58	361.58	6.65	0.00	361.58	7.40	0.00	361.58	7.68	0.00	361.58	7.89	0.00
AA50	430.41	430.41	8.22	0.00	430.41	11.93	0.00	430.41	12.92	0.00	430.41	14.29	0.00
AA55	436.05	433.21	11.33	0.65	436.05	15.29	0.00	436.05	18.42	0.00	436.05	23.25	0.00
AA60	505.02	505.02	11.19	0.00	505.02	16.68	0.00	505.02	21.76	0.00	505.02	26.66	0.00
AA65	576.39	576.39	14.87	0.00	576.39	21.78	0.00	576.39	27.04	0.00	576.39	27.25	0.00
AA70	606.68	606.68	15.05	0.00	606.68	25.12	0.00	606.68	30.13	0.00	606.68	33.98	0.00
AA75	715.53	714.65	32.07	0.12	714.65	57.36	0.12	715.53	111.54	0.00	715.53	162.37	0.00
BB30	274.45	274.45	2.00	0.00	274.45	1.89	0.00	274.45	1.88	0.00	274.45	1.86	0.00
BB35	337.55	337.55	1.03	0.00	337.55	1.21	0.00	337.55	1.05	0.00	337.55	1.17	0.00
BB40	326.82	326.82	4.01	0.00	326.82	4.79	0.00	326.82	5.07	0.00	326.82	5.37	0.00
BB45	378.90	378.90	8.61	0.00	378.90	12.89	0.00	378.90	17.43	0.00	378.90	19.14	0.00
BB50	432.10	432.10	8.04	0.00	432.10	14.40	0.00	432.10	19.88	0.00	432.10	22.75	0.00
BB55	530.84	530.84	12.40	0.00	530.84	20.19	0.00	530.84	26.80	0.00	530.84	33.32	0.00
BB60	558.02	558.02	16.08	0.00	558.02	28.47	0.00	558.02	43.92	0.00	558.02	69.61	0.00
BB65	547.84	547.84	16.41	0.00	547.84	31.09	0.00	547.84	61.08	0.00	547.84	104.54	0.00
BB70	558.29	558.29	21.72	0.00	558.29	34.10	0.00	558.29	50.89	0.00	558.29	83.74	0.00
BB75	613.35	613.35	23.32	0.00	613.35	45.66	0.00	613.35	65.52	0.00	613.35	106.88	0.00
CC30	316.39	316.39	9.16	0.00	316.39	13.39	0.00	316.39	16.18	0.00	316.39	18.46	0.00
CC35	386.26	386.26	17.39	0.00	386.26	31.84	0.00	386.26	38.27	0.00	386.26	53.09	0.00
CC40	464.28	464.28	32.67	0.00	464.28	56.99	0.00	464.28	85.19	0.00	464.28	124.12	0.00
CC45	497.68	497.68	52.17	0.00	497.68	108.30	0.00	497.68	190.84	0.00	497.68	303.98	0.00
CC50	519.60	516.59	48.71	0.58	516.59	171.45	0.58	517.10	364.55	0.48	517.10	616.42	0.48
CC55	581.50	564.03	98.13	3.00	572.88	306.60	1.48	572.88	746.83	1.48	581.50	1416.82	0.00
CC60	624.95	593.46	114.55	5.04	606.11	327.94	3.01	624.95	911.43	0.00	624.95	1840.42	0.00
CC65	663.31	636.12	169.39	4.10	653.51	470.64	1.48	655.84	1295.07	1.13	662.61	3879.84	0.11
CC70	-	678.11	188.24	$\geq$ 0.55	680.30	585.96	$\geq$ 0.23	680.56	1589.12	$\geq$ 0.19	681.84	4858.95	$\geq$0.00
CC75	-	686.68	250.89	$\geq$ 0.00	686.68	634.38	$\geq$ 0.00	686.68	1745.03	$\geq$ 0.00	686.68	7159.92	$\geq$0.00
DD30	343.65	343.65	17.32	0.00	343.65	35.09	0.00	343.65	46.71	0.00	343.65	46.1	0.00
DD35	410.19	410.19	37.57	0.00	405.07	92.01	1.25	410.19	168.55	0.00	410.19	275.19	0.00
DD40	490.92	477.82	74.59	2.67	490.92	233.52	0.00	490.92	526.53	0.00	490.92	1026.12	0.00
DD45	540.17	540.17	78.77	0.00	540.17	210.87	0.00	540.17	415.69	0.00	540.17	953.24	0.00
DD50	610.07	589.67	104.49	3.34	599.02	264.49	1.81	610.07	725.41	0.00	610.07	1277.29	0.00
DD55	639.75	617.52	159.5	3.47	626.86	550.42	2.01	629.70	1132.39	1.57	639.75	2754.41	0.00
DD60	-	690.87	191.22	$\geq$ 3.16	695.74	742.63	$\geq$ 2.47	695.54	1785.51	$\geq$ 2.50	713.39	4054.79	$\geq$ 0.00
DD65	-	754.65	246.71	$\geq$ 0.75	758.57	963.77	$\geq$ 0.23	760.32	2212.53	$\geq$ 0.00	760.32	5853.15	$\geq$ 0.00
DD70	-	746.44	334.40	$\geq$ 2.31	751.56	1434.89	$\geq$ 1.64	753.58	3869.98	$\geq$ 1.38	764.10	9581.94	$\geq$ 0.00
DD75	-	830.50	399.73	$\geq$ 0.63	831.49	1899.06	$\geq$ 0.51	835.76	4816.39	$\geq$ 0.00	835.76	15842.28	$\geq$ 0.00
Average	515.18	510.34	71.26	$\geq$ 0.76	512.48	237.42	$\geq$ 0.42	513.72	580.44	$\geq$ 0.22	515.10	1567.30	$\geq$0.01

Table 2.11: Impact of Different E Values on Instances when B=10000

	E=0.5n			E=0.25n			E=0.125n		
Instance	Value	Time(s)	Gap	Value	Time(s)	Gap	Value	Time(s)	Gap
AA30	266.72	4.23	0.00	258.68	3.99	3.01	183.89	1.33	31.06
AA35	278.14	5.01	0.00	278.14	4.73	0.00	268.40	2.14	3.50
AA40	341.74	7.27	0.00	341.74	6.93	0.00	341.74	4.57	0.00
AA45	361.58	8.88	0.00	361.58	8.49	0.00	358.74	8.22	0.79
AA50	430.41	20.11	0.00	430.41	11.01	0.00	418.09	9.19	2.86
AA55	436.05	33.49	0.00	436.05	26.04	0.00	424.6	13.15	2.63
AA60	505.02	31.72	0.00	505.02	23.89	0.00	505.02	14.34	0.00
AA65	576.39	30.23	0.00	576.39	24.45	0.00	568.4	22.18	1.39
AA70	606.68	79.56	0.00	606.68	47.77	0.00	599.97	31.50	1.11
AA75	715.53	294.76	0.00	715.53	210.50	0.00	713.67	144.16	0.26
BB30	274.45	3.53	0.00	274.45	2.20	0.00	249.00	1.48	9.27
BB35	337.55	2.62	0.00	316.85	5.02	6.13	316.85	1.89	6.13
BB40	326.82	6.6	0.00	326.82	6.52	0.00	326.82	4.54	0.00
BB45	378.90	25	0.00	378.9	14.61	0.00	378.9	11.62	0.00
BB50	432.10	32.89	0.00	432.10	17.19	0.00	432.1	11.83	0.00
BB55	530.84	50.25	0.00	530.84	31.97	0.00	530.84	18.36	0.00
BB60	558.02	72.85	0.00	558.02	71.65	0.00	558.02	36.90	0.00
BB65	547.84	129.72	0.00	547.84	120.02	0.00	547.84	56.83	0.00
BB70	558.29	139.86	0.00	558.29	92.68	0.00	558.29	48.46	0.00
BB75	613.35	190.45	0.00	613.35	158.15	0.00	613.35	49.19	0.00
CC30	316.39	20.66	0.00	316.39	19.25	0.00	292.65	5.80	7.50
CC35	386.26	54.93	0.00	386.26	45.49	0.00	375.17	12.00	2.87
CC40	464.28	133.99	0.00	464.28	152.86	0.00	448	38.50	3.51
CC45	497.68	497.68	0.00	497.68	352.53	0.00	479.19	191.45	3.72
CC50	517.1	542.93	0.48	517.1	598.14	0.48	503.47	421.23	3.10
CC55	581.5	1920.55	0.00	581.5	1421.72	0.00	579.34	2338.11	0.37
CC60	624.95	1244.56	0.00	624.95	1579.68	0.00	624.95	2338.36	0.00
CC65	662.61	4098.04	0.11	662.61	2629.06	0.11	662.61	2203.57	0.11
CC70	681.84	4918.55	0.00	681.84	3110.90	0.00	681.84	2453.23	0.00
CC75	686.68	4290.45	0.00	686.68	5518.81	0.00	686.68	3183.47	0.00
DD30	343.65	68.19	0.00	343.65	63.74	0.00	330.79	17.25	3.74
DD35	410.19	350.53	0.00	410.19	399.22	0.00	385.85	84.60	5.93
DD40	490.92	1158.38	0.00	490.92	1133.28	0.00	483.48	1036.05	1.52
DD45	540.17	889.37	0.00	540.17	962.77	0.00	532.17	827.67	1.48
DD50	610.07	1919.31	0.00	610.07	1595.29	0.00	597.37	1352.25	2.08
DD55	639.75	2861.15	0.00	639.75	2733.71	0.00	638.36	4879.35	0.22
DD60	713.39	5432.74	0.00	713.39	4656.094	0.00	708.52	4264.71	0.68
DD65	760.32	6456.7	0.00	760.32	5255.72	0.00	758.57	4545.88	0.23
DD70	764.1	9593.25	0.00	764.1	8355.49	0.00	760.5	6054.51	0.47
DD75	835.76	11762.83	0.00	835.76	10770.09	0.00	835.76	8465.89	0.00
Average	515.10	1484.60	0.01	514.38	1306.04	0.15	506.50	1130.14	1.69

quality is the same as in case $E = \infty$ but the computation times are reduced by 5%. When we restrict E further the solution quality is going down drastically without gaining too much computation time. In comparison, a combination of $B = 1000$ and $E = \infty$ is much faster and has higher solution quality then the combination of $B = 10000$ and $E = 0.125n$

2.7 Conclusion

This chapter presents the time-dependent capacitated profitable tour problem with time windows and precedence constraints, which combines features of both the traveling salesman problem with pickup and delivery, and the traveling salesman problem with profits. Moreover, time-dependent traveling speeds are considered to capture road congestion, which increases the complexity of the problem.

We propose a tailored labeling algorithm to solve this problem enriched by new and strong dominance rules to discard 95% of the labels. Extensive computational results show that most instances with up to 75 requests can be solved to optimality within the given time limit of two weeks, but also show that some instances remain unsolved.

To reduce the computation time and memory usage, a restricted dynamic programming heuristic algorithm is implemented. The heuristic is able to find solutions for all instances with good qualities (on average 0.01% gap) and less computational time (on average 1567 seconds).

Obviously, the performance of our exact algorithm critically depends on the tailored dominance criterion that we propose. It shows great potential when there are relatively tight time windows attached to all the nodes. Further strengthening dominance is a promising direction for future research.

In addition, solving extensions of the time-dependent capacitated profitable tour problem with time windows and precedence constraints seems to be an attractive research direction. More specifically, the time-dependent variant of the pickup and delivery problem with time windows (TDPDPTW) and the time-dependent team orienteering problem (TDTOP) are very interesting as it aims to optimize the routes of a fleet of vehicles, instead of a single vehicle only. When column generation is utilized to solve those two problems in an exact way, our proposed model can be used as the pricing problem.

Chapter 3

The time-dependent pickup and delivery problem with time windows

"What is reasonable is real; that
which is real is reasonable."

Georg Wilhelm Friedrich Hegel

3.1 Introduction

The advent of E-commerce has led to increased shipping volumes both when the customer is a business (i.e. B2B) and when the customer is an individual (B2C). In either case, the trend in customer expectations is for shorter, and more precisely specified (i.e. just-in-time logistics), delivery times. To handle these trends at low cost and environmental impact, transportation companies need tools for effectively optimizing how they should respond to and serve transportation requests. In this chapter, we focus on transportation requests that indicate a pickup location for a shipment, a time window during which the shipment must be picked up, a delivery location for the shipment, and a time window during which the shipment must be delivered. In the literature [22, 82, 83], optimizing the service of such requests with a fixed fleet of capacitated vehicles has been termed the pickup and delivery problem with time windows (PDPTW)

The objective of the classical PDPTW is to minimize the transportation costs incurred when executing routes designed under the assumption that all transportation requests must be served. This assumption is particularly relevant to supply chains based upon transportation that is either owned or outsourced to a dedicated carrier. A growing trend in supply chain management is to instead outsource transportation needs on a request basis through the use of "load markets" or "boards". Some of the growing demand for transporting individual

requests can be attributed to the capabilities and ease-of-use of the software platforms supporting such boards. From the transportation supply side, the easy (sometimes mobile) access that these platforms provide also enables transportation service providers to pick and choose the requests that yield the greatest profits. As a result, in this chapter we study variants of the PDPTW wherein the transportation company can select a subset of the requests to serve based on the revenues it receives for serving those requests. In these variants, the objective is then to maximize profits.

The routing community has long studied problems (including PDPTWs) that assumed that travel times are constant, or, time-independent. However, both demographic trends towards larger populations in urban areas and improved technology for solving routing problems has led researchers [25, 56, 57, 114] to study those that consider time-dependent travel times (as reviewed in Gendreau et al. [46]). Such problems are particularly important as customer expectations for short delivery times reduce the flexibility transportation companies have for accommodating variations from "average" travel times. While some variation may be due to unforeseen events, some is due to traffic and congestion. As traffic patterns are somewhat predictable, embedding their impact on travel times in deterministic routing models is likely to yield plans that are more likely to meet customer expectations at low cost than those that ignore those impacts.

Thus, in all the variants of the PDPTW that we study in this chapter, we consider time-dependent travel times, meaning that the time it takes to travel between two locations depends on the time when travel begins. Like Ichoua et al. [56], we assume these travel times adhere to the "first in first out" (FIFO) property, meaning that the earlier the departure, the earlier the arrival. However, recognizing that travel times may vary throughout the day brings a new dimension to the PDPTW. Specifically, there can be value in optimizing the time at which a route begins. As providing a driver with a different start time each day may not be feasible in all labor/operational settings, we consider variants where the departure time is fixed as well as those where it can be optimized.

We present an optimization-based framework for variants of the PDPTW that can recognize time-dependent travel times as well as specify vehicle start times. To do so, and like Røpke and Cordeau [91], the framework models vehicle departure times with continuous variables. As a result, the framework optimizes vehicle routes based upon precise representations of the timings of vehicle departures and arrivals. Relatedly, continuous time variables make it easy to model travel times that adhere to the FIFO property. An alternative approach, and one

proposed by Mahmoudi and Zhou [70], is to optimize decisions on a network that represents both the physical and temporal attributes of decisions (i.e. a space-time network). While such a network can facilitate the representation of time-dependent travel times, it also necessitates discretizing time, which can introduce approximation errors into an optimization model.

In total, we study four variants of the time-dependent pickup and delivery problem with time windows (TDPDPTW), that differ with respect to two attributes: (1) whether the variant requires that all transportation requests be served, and, (2) whether the variant requires that all routes begin at a fixed departure time. For the sake of clarity, let Ω be our base problem, i.e. the time-dependent pickup and delivery problem with time windows. We denote the four variants then as follows: $\Omega(Fix, All)$, $\Omega(Fix, Prof)$, $\Omega(Flex, All)$ and $\Omega(Flex, Prof)$, where Fix versus $Flex$ refers to fixed versus flexible departure times, and All versus $Prof$ refers to handling all versus only the profitable requests. We present a solution framework based on branch-and-price that can solve each of these variants to optimality. Our framework adapts and extends the work presented in Røpke and Cordeau [91] and Røpke et al. [92] for solving the PDPTW and employs some of the techniques presented in Sun et al. [109] in an algorithm for solving single-vehicle variants of the problems we study.

As such, this chapter makes the following contributions to the literature:

- We present a general mathematical model for variants of the time-dependent pickup and delivery problems with time windows that are inspired by changing trends in the transportation industry.
- We present an exact algorithm for solving the time-dependent pickup and delivery problem with time windows and the variants we consider. To the best of our knowledge, this is the first algorithm that is able to solve all of these variants.
- We illustrate how to adapt known speed-up techniques from the literature on other, related, routing problems to the problems we seek to solve. An extended computational study demonstrates their effectiveness.

The remainder of this chapter is organized as follows. Section 3.2 presents a brief review of the existing work related to this chapter. Section 3.3 introduces the mathematical formulation for the variants of the time-dependent pickup and delivery problem with time windows we consider in this chapter. In Section 3.4, we describe our exact framework. Finally, computational results are reported in Section 3.5, followed by conclusions in Section 3.6.

3.2 Literature review

There are several problems in the literature that share characteristics with the time-dependent pickup and delivery problem with time windows. These include the pickup and delivery problem with time windows (PDPTW), the vehicle routing problem with profits (VRPP) and the time-dependent vehicle routing problem with time windows (TDVRPTW). We preview how the problems we study intersect with those problems in Figure 3.1 and will next discuss those intersections in detail.

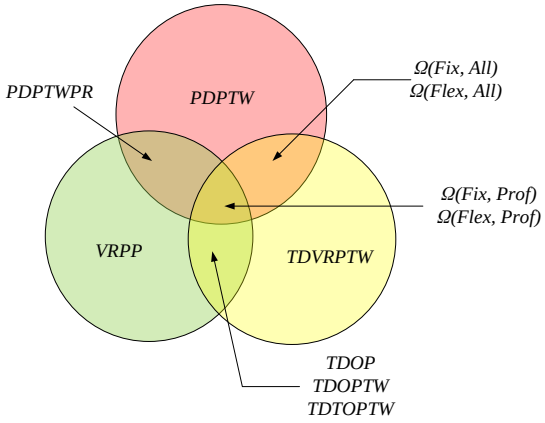

Figure 3.1: Classification of related problems

3.2.1 The pickup and delivery problem with time windows (PDPTW)

The PDPTW models a variety of operational planning problems in transportation and logistics. This results in a rich literature on PDPTW-related problems over the last few decades. In this problem, a set of homogeneous vehicles with limited capacity based at a depot is required to carry out a group of geographically scattered requests. Each request needs to be transported from a specified origin and delivered to a specified destination within the predefined time windows, which is the interval of time during which the service at a node must start. The interested readers are referred to Cordeau et al. [22] and Parragh et al. [82, 83] for recent surveys. Moreover, due to the difficulty of the problem, much of the existing literature concentrates on heuristics.

Less work has been done on exact algorithms for these types of problems. The pioneering work is done by Dumas et al. [34] who proposed the first branch-cut-price algorithm for the PDPTW. This method was able to solve tightly constrained instances with up to 55 requests. Recently, the exact approach was also used in Sigurd and Pisinger [102], Sol [103], Røpke

and Cordeau [91] and Røpke et al. [92]. The method of Røpke and Cordeau [91] is a branch-cut-and-price algorithm that uses different classes of valid inequalities to improve the lower bound and two different methods for solving the pricing problem. Baldacci et al. [6] have designed two exact algorithms relying on column generation and variable fixing based on reduced cost. Recently, Azadian et al. [4] proposed an exact method based on a decomposition approach for a variant of the PDPTW wherein requests are unpaired. In the problem they study, delivery costs are time-dependent, but travel times are time-independent.

Unlike the problems studied in this chapter, the majority of related publications consider a constant travel time environment. Moreover, unlike some of the problems we study, the majority of the literature studies problems wherein all requests must be served.

3.2.2 The vehicle routing problem with profits (VRPP)

The vehicle routing problem with profits (VRPP) also arises in a wide range of transportation and logistics applications. In these problems, along with $\Omega(Fix, Prof)$ and $\Omega(Flex, Prof)$, there is a request selection decision that must be made. Instead of maximizing the difference between total revenues and total costs, which defines the profitable tour problem (PTP), the orienteering problem (OP) aims to maximize the total collected profits subject to a maximum tour length. Conversely, the prize-collecting traveling salesman problem (PCTSP) aims to minimize the total traveling cost with the constraint that the total collected profits can not be less than a given amount. A recent survey (Vansteenwegen and Gunawan [115]) of routing problems with a single vehicle and profits indicates that most of existing literature focus on the OP with its variants. As an example, Righini and Salani [88] proposed an extended bidirectional dynamic programming algorithm to solve the orienteering problem with time windows (OPTW), which recognizes the presence of time window constraints. More recently, Duquei et al. [36] adapted the pulse algorithm to the OPTW by adding new problem-specific strategies. The resulting algorithm (computationally) outperforms the algorithm proposed by Righini and Salani [88]. Extending the orienteering problem to teams yields the team orienteering problem (TOP). Similarly, the OPTW has been extended to the team orienteering problem with time windows (TOPTW). Exact and heuristic solution strategies have been developed for both of these problems. An exact algorithm for the TOP can be found in Boussier et al. [15], whereas Labadie et al. [62] developed a granular variable neighborhood search for the TOPTW. Recently, Archetti et al. [3] proposed a branch-cut-and-price algorithm to solve a variant of the TOP known as the capacitated team orienteering problem (CTOP), which also models vehicle capacity. We refer the reader to the survey

presented by Vansteenwegen et al. [116] and Gunawan et al. [50] for a detailed review of the OP and its variants. For more details on VRPP, we refer the readers to the book written by Vansteenwegen and Gunawan [115]. However, ultimately, much of the literature on these problems assumes a constant travel time environment.

3.2.3 Time-dependent vehicle routing problems

Less research has been done on problems that allow travel times between origins and destinations to vary based on the time of departure from the origin. One problem that has been studied is the time-dependent vehicle routing problem with time windows (TDVRPTW). The objective of this problem is to find a set of routes that visit all customers with minimum traveling duration. Malandraki and Daskin [71] presented a mixed integer programming-based formulation of this problem, some construction heuristics, and a cutting plane-based solution method. Later, Ichoua et al. [56] proposed a parallel tabu search to solve the problem. However, they also made the important observation that previous models of time-dependent travel times did not adher to the "first in first out" (FIFO) property. Namely, that if two identical vehicles traverse the same arc, the one that leaves first will arrive first. As a result, they proposed a speed model that is a step-wise function and satisfies the FIFO principle. This speed model was also used in Donati et al. [33], which presented a multiple ant colony system (ACS) solution framework for the TDVRPTW. This ACS framework was then enhanced in Balseiro et al. [8]. Regarding exact approaches, Dabia et al. [25] developed a branch and price algorithm for TDVRPTW that employs a tailored labeling algorithm to solve a pricing problem that can be formulated as a time-dependent shortest path problem with resource constraints (TDSPPRC). The proposed algorithm is able to solve instances with up to 100 customers. Readers interested in the TDVRPTW and its variants should consult the recent survey by Gendreau et al. [46]. The TDVRPTW differs from the problems we study in this chapter as it does not consider the precedence relationships between locations (i.e. the pickup node of a request must be visited before the corresponding destination node).

3.2.4 Closely related problems

We have (briefly) reviewed three different classes of problems: (1) those that model pickups and deliveries that must occur with time windows, (2) those that model request selection, and, (3) those that model time-dependent travel times. Research has been done on problems that model attributes of two of these problem classes. For example, Li et al. [67] consider a PDPTW in a profitable context, which they refer to as the pickup and delivery problem with

time windows, profits, and reserved requests (PDPTWPR). In this problem, there is one type of request that must be served (called reserved requests), and another which may be rejected or outsourced. They propose an adaptive large neighborhood search to solve this problem.

Similarly, researchers have studied the Time-dependent orienteering problem (TDOP) and its variants. Fomin and Lingas [40] provide a $(2+\epsilon)-$ approximation algorithm for the TDOP which runs in polynomial time if the ratio between the minimum and maximum traveling time of any two sites is constant. Li [66] designed a dynamic labeling algorithm for the TDOP, where discrete time units are used. Verbeeck et al. [117] proposed an ant colony optimization-based heuristic for the TDOP. To the best of our knowledge, the only heuristic for the TDTOPTW is the iterated local search-based method proposed by Gavalas et al. [45].

Recently, Sun et al. [109] introduced the time-dependent capacitated profitable tour problem with time windows and precedence constraints. A tailored labeling algorithm is proposed that is able to solve instances with up to 75 requests to optimality. They also propose a dynamic programming-based heuristic for instances that cannot be solved by the exact method. However, unlike the problems considered in this chapter, it only considers a single vehicle.

Also related to the work presented in this chapter is that of Mahmoudi and Zhou [70], which studies a time-dependent PDPTW seen in ride-sharing settings. They propose a solution approach that couples a Lagrangian relaxation scheme for producing dual bounds with a dynamic programming-based method for producing primal solutions. Their approach is based on a state-space-time network representation of the problem, which is based on a discretization of time.

As discussed previously, the framework presented in this chapter and the method of Mahmoudi and Zhou [70] model the timing of decisions differently. However, they also consider profits in different ways. The variants considered in this chapter treat the profit earned from serving a customer request as a known and fixed value, and thus the framework we propose seeks to maximize the difference between the profits associated with serving customers and the costs incurred by doing so. The problem studied in Mahmoudi and Zhou [70] focuses on minimizing the total cost associated with transporting all passengers. However, the Lagrangian component of their method relaxes the requirement that each passenger is transported. Associated with the relaxed constraints are dual variables that are iteratively updated, and converge to values that can be interpreted as marginal profits.

Relatedly, the procedure for producing primal solutions allows for passengers to be assigned to virtual vehicles.

Finally, the framework in this chapter and the method of Mahmoudi and Zhou [70] differ in the known guarantees that can be made regarding their performance. The framework we present is based upon branch-and-price, an algorithmic strategy for solving integer programs that is known to be exact. Namely, that when run for sufficient time, it is guaranteed to produce both a primal solution that is optimal and a certificate of its optimality. This guarantee is based on appropriately-designed subroutines for generating new variables and partitioning the solution space. We propose such subroutines and thus exact algorithms for the variants we consider. As Mahmoudi and Zhou [70] do not include a proof that the method they propose is exact, such a guarantee is not yet known.

3.3 Problem description and mathematical formulation

In this section, we first present a mathematical programming formulation of $\Omega(Flex, Prof)$. From this formulation we show how to derive formulations of the other three variants by adapting its objective function and/or fixing certain variables. As a result, we refer to the $\Omega(Flex, Prof)$ as the *general* time-dependent pickup and delivery problem with time windows

The general time-dependent pickup and delivery problem with time windows considers a fleet of homogeneous vehicles that travel on routes constructed to serve customer pickup and delivery requests. To serve a request, which in turn generates profits, the vehicle must visit the pickup location during a given time window and then later visit the delivery location during another given time window. Visiting either a pickup or delivery location requires a service time. To model time-dependent travel times between a pair of locations, we divide the planning horizon into time zones for travel between those locations, wherein a different speed may be associated with each time zone. The objective of the problem is to maximize the difference between the profits earned by serving requests and costs associated with transportation. Transportation costs include costs associated with time spent traveling and a fixed cost incurred when a vehicle is used. The formulation we present for the $\Omega(Flex, Prof)$ is inspired by the PDPTW formulation presented in Røpke and Cordeau [91]. We define a directed graph $G = (N, A)$, wherein $N = 0, 1, ..., 2n + 1$ is the set of all nodes and A is the set of arcs. Nodes 0 and $2n + 1$ represent the origin and destination depot of the vehicles. The sets $N_P = 1, ..., n, \subseteq N$ and $N_D = n + 1, ..., 2n, \subseteq N$ represent the sets of pickup and delivery

locations, respectively. There is a set of n requests $R_1, ..., R_n$, wherein serving request i yields profit p_i. Associated with request R_i is a pickup node i, with service time s_i, and delivery node $n + i$, with service time s_{n+i}. A time window $[e_i, l_i]$ is associated with every node $i \in N_P \cup N_D$, wherein e_i and l_i represent the earliest and latest times, respectively, at which service may start at node i. If the vehicle arrives at i earlier than e_i, it waits; it can not arrive later than l_i. The depot nodes also have time windows $[e_0, l_0]$, $[e_{2n+1}, l_{2n+1}]$ representing the earliest and latest times, respectively, at which the vehicle may leave from and return to the depot.

Each request has a size, q_i. We associate with the pickup node i the quantity q_i and the delivery node $n + i$ the quantity $q_{n+i}(= -q_i)$. We presume there is no inventory at the depot, and thus $q_0 = q_{2n+1} = 0$. Similarly, we assume $s_0 = s_{2n+1} = 0$. To serve requests, there is a fleet of K identical vehicles, each with capacity Q and a fixed cost, Z, that is paid when the vehicle is used.

We let $\tau_{ij}(t_i^k)$ denote the travel time from node i to node j when vehicle k departs from node i at time t_i^k. We define the set of arcs as $A = \{(i, j) \in N \times N : i \neq j \text{ and } e_i + s_i + \tau_{ij}(e_i + s_i) \leq l_j\}$. In other words, an arc from node i to node j is included only if they can be visited in succession while respecting their time windows. Similar to Ichoua et al. [56], Jabali et al. [57], Dabia et al. [25] and Franceschetti et al. [41], associated with each $(i, j) \in A$ is a speed profile that can be represented by a step-wise function based upon a division of the planning horizon into time zones. In this profile, speeds are constant within a time zone but may vary from one zone to the next. With such a speed profile, we can generate travel times that satisfy the FIFO principle.

Specifically, we use a travel time function that is piece-wise linear and continuous (see Figure 3.2 for an example). This function is based on a set of breakpoints that denote when a change of speed (and hence travel time) occurs. Specifically, we let T_{ij} represent the set of time zones modeled by the travel time function $\tau_{ij}(\cdot)$. A time zone $T_{ij}^m \in T_{ij}$ can be defined by two consecutive travel time breakpoints, $T_{ij}^m = [w_m, w_{m+1}]$, wherein the travel time function is linear within those breakpoints. Thus, for time zone T_{ij}^m, given the values w_m, w_{m+1}, $\tau_{ij}(w_m)$, and $\tau_{ij}(w_{m+1})$, we calculate the slope, θ_m, of the function and its intersection, η_m, with the y-axis. From this slope and intercept, we compute the travel time when departure occurs at time $t_i^k \in T_{ij}^m$ as follows:

$$\tau_{ij}(t_i^k) = \theta_m t_i^k + \eta_m. \qquad \forall t_i^k \in T_{ij}^m \tag{3.1}$$

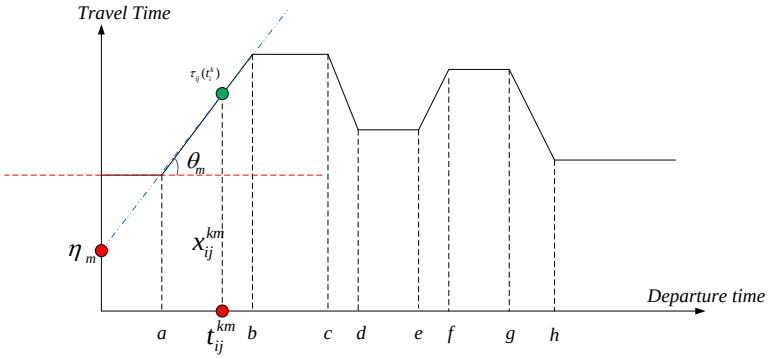

Figure 3.2: Travel time function

To model this piecewise-linear function within an integer program, we define the binary variable $x_{ij}^{km}, i, j \in N, k \in K, m = 1, \ldots, \mid T_{ij} \mid$, to represent whether vehicle k travels from node i to node j by departing from i in time zone T_{ij}^m. Relatedly, we let the continuous variable $t_{ij}^{km}, i, j \in N, k \in K$, represent the time when vehicle k departs from i to j during time zone T_{ij}^m. If vehicle k does not depart from i to j during time zone T_{ij}^m (i.e. $x_{ij}^{km} = 0$)), then $t_{ij}^{km} = 0$. With these variables, we can express the travel time function $\tau_{ij}(t_i^k)$ of arc (i, j):

$$\tau_{ij}(t_i^k) = \sum_{m=1}^{|T_{ij}|} (\theta_m t_{ij}^{km} + \eta_m x_{ij}^{km}) \tag{3.2}$$

Regarding request selection, we let $y_i^k, i \in N$, be a binary variable that represents whether vehicle k visits node i. Relatedly, we define $Q_i^k, i \in N, k \in K$ as a nonnegative integer variable that equals the load of vehicle k when leaving node i. Finally, we presume travel costs are a linear function of travel times, and represent this relationship with the cost per unit parameter, c_t.

With these variables and parameters, we formulate $\Omega(Flex, Prof)$ as the following mixed integer program:

$$Obj_{\Omega(Flex,Prof)} = \max \sum_{k \in K} [\sum_{i \in N_P} p_i y_i^k - c_t(t_{2n+1}^k - t_0^k) - Z y_0^k] \tag{3.3}$$

subject to

$$\sum_{j \in N_P} \sum_{m=1}^{|T_{0j}|} x_{0j}^{km} = y_0^k \qquad \forall k \in K, \tag{3.4}$$

$$\sum_{i \in N_D} \sum_{m=1}^{|T_{i,2n+1}|} x_{i,2n+1}^{km} = y_{2n+1}^k \qquad \forall k \in K, \tag{3.5}$$

$$\sum_{k \in K} y_j^k \leq 1 \qquad \forall j \in N_P \cup N_D, \tag{3.6}$$

$$\sum_{i \in N \setminus \{2n+1\}} \sum_{m=1}^{|T_{if}|} x_{if}^{km} - \sum_{j \in N \setminus \{0\}} \sum_{m=1}^{|T_{fj}|} x_{fj}^{km} = 0 \qquad \forall f, i, j \in N_P \cup N_D, \forall k \in K, \tag{3.7}$$

$$\sum_{i \in N \setminus \{2n+1\}} \sum_{m=1}^{|T_{ij}|} x_{ij}^{km} = y_j^k \qquad \forall j \in N_P \cup N_D, \tag{3.8}$$

$$\sum_{j \in N \setminus \{0\}} \sum_{m=1}^{|T_{ij}|} x_{ij}^{km} - \sum_{j \in N \setminus \{0\}} \sum_{m=1}^{|T_{n+i,j}|} x_{n+i,j}^{km} = 0 \qquad \forall i \in N_P, \forall k \in K, \tag{3.9}$$

$$t_i^k + \tau_{ij}(t_i^k) + s_j \leq t_j^k + M(1 - x_{ij}^{km}) \qquad \forall i, j \in N_P \cup N_D, m = 1, \ldots, \mid T_{ij} \mid, \forall k \in K, \tag{3.10}$$

$$Q_i^k + q_j \leq Q_j^k + M(1 - x_{ij}^{km}) \qquad \forall i, j \in N, m = 1, \ldots, \mid T_{ij} \mid, \forall k \in K, \tag{3.11}$$

$$t_i^k = \sum_{j \in N \setminus \{0\}} \sum_{m=1}^{|T_{ij}|} t_{ij}^{km} \qquad \forall i \in N \setminus \{2n+1\}, \forall k \in K, \tag{3.12}$$

$$t_{n+i}^k \geq t_i^k \qquad \forall i \in N_P, \forall k \in K, \tag{3.13}$$

$$w_m x_{ij}^{km} \leq t_{ij}^{km} \leq w_{m+1} x_{ij}^{km} \qquad \forall i, j \in N, m = 1, \ldots, \mid T_{ij} \mid -1, \forall k \in K, \tag{3.14}$$

$$(e_i + s_i) y_i^k \leq t_i^k \leq (l_i + s_i) y_i^k \qquad \forall i \in N, \forall k \in K, \tag{3.15}$$

$$\max\{0, q_i\} \leq Q_i^k \leq \min\{Q, Q + q_i\} \qquad \forall i \in N, \forall k \in K, \tag{3.16}$$

$$x_{ij}^{km}, y_i^k \in \{0, 1\} \qquad \forall i, j \in N, \forall m = 1, \ldots, \mid T_{ij} \mid, \forall k \in K. \tag{3.17}$$

The objective function (3.3) measures the difference between profits collected and costs incurred due to travel and vehicle use. Constraints (3.4) and (3.5) guarantee that if vehicle k is used, its route starts from and ends at the depot. Constraints (3.6) ensure that every request is served at most once. Constraints (3.7) are classical flow conservation constraints. Constraints

(3.8) and (3.9) ensure both the pickup and delivery nodes are visited when a request is served, and by the same vehicle. Given a vehicle's route, constraints (3.10) ensure its departure times from locations respect travel and service times. Constraints (3.11) compute the quantity carried by the vehicle throughout its route. Constraints (3.12) calculate the departure time for vehicle k from location i. Constraints (3.13) ensure that the vehicle visits the pickup node for a request before its delivery node. Constraints (3.14) ensure that the departure time for a vehicle from a location is associated with the correct time zone. Constraints (3.15) ensure that a vehicle departs at a location after starting service within the time window. Constraints (3.16) ensure that the vehicle capacity is respected. The last set of constraints define decision variables and their domains.

Formulation for $\Omega(Fix, All)$

This is the most restrictive variant, as it requires all customer requests to be served and each vehicle route to depart from the depot at time 0. We model these restrictions by adding constraints to the model presented above, which in turn, simplifies the objective function. The resulting integer program is as follows:

$$Obj_{\Omega(Fix,All)} = \max \sum_{i \in N_P} p_i - \sum_{k \in K} (c_t t^k_{2n+1} + Z y^k_0) \tag{3.18}$$

subject to (3.4)-(3.5), (3.7) - (3.17), with the following extra constraints:

$$\sum_{k \in K} y^k_j = 1 \qquad \forall j \in N_P \cup N_D \tag{3.19}$$

$$t^k_0 = 0 \qquad \forall k \in K \tag{3.20}$$

Constraints (3.19) ensure that each request is served, while constraints (3.20) ensure that each route departs from the depot at time 0. Due to the fixed variables the objective function reduces to (3.18).

Formulation for $\Omega(Flex, All)$

This variant requires all customer requests to be served, but allows for flexibility in when a route departs from the depot. As a result, we formulate the problem as follows:

$$Obj_{\Omega(Flex,All)} = \max \sum_{i \in N_P} p_i - \sum_{k \in K} [c_t(t^k_{2n+1} - t^k_0) + Z y^k_0] \tag{3.21}$$

subject to (3.4)-(3.5), (3.7) - (3.17) and (3.19).

Formulation for $\Omega(Fix, Prof)$

This variant does not require that each customer request be served, but does require that each route departs from the depot at time 0. The resulting integer program is as follows:

$$Obj_{\Omega(Fix,Prof)} = \max \sum_{k \in K} (\sum_{i \in N_P} p_i y^k_i - c_t t^k_{2n+1} - Z y^k_0) \tag{3.22}$$

subject to (3.4)-(3.17) and (3.20).

Relation between objective function values of the variants

As these four optimization problems are related, and can each be solved for the same instance, we can determine relationships between their optimal objective function values. For example, as $\Omega(Flex, Prof)$ is the least restricted variant, for a given instance, we have that $Obj_{\Omega(Flex,Prof)}$ will be the largest optimal objective function value. Conversely, as $\Omega(Fix, All)$ is the most restricted variant, we have that $Obj_{\Omega(Fix,All)}$ will be the smallest optimal objective function value for a given instance. Similarly, one can observe that $Obj_{\Omega(Fix,All)} \leq Obj_{\Omega(Fix,Prof)}$ and $Obj_{\Omega(Fix,All)} \leq Obj_{\Omega(Flex,All)}$. While statements can not be made regarding the relationship between $Obj_{\Omega(Fix,Prof)}$ and $Obj_{\Omega(Flex,All)}$ for general instances, we will analyze the relationship between these two quantities in our computational study.

3.4 Solution approach

It is well known that compact formulations of routing problems, such as the one presented above, have weaker linear relaxations than extended formulations, wherein each variable models an entire vehicle route. The stronger bounds that result from using an extended formulation can greatly speed up the solution of the integer program, but come at the expense

of a large number of variables. In particular, such extended formulations often involve variable sets that are either too large to enumerate in a reasonable run-time, or to fit in run-time memory, or both. However, using column generation, our algorithm begins with a model that includes a small portion of the variable set and then generates more variables (columns) when there is evidence they may appear in an optimal solution. We refer interested readers to [30] for a detailed description of column generation. Therefore, this algorithm can retain the benefits of using an extended formulation without the challenges presented by the ensuing variable set.

As a result, we propose a solution framework that is based on branch-and-price [9], a branch-and-bound-based technique for solving integer programs that relies on generating variables dynamically through column generation. We employ several techniques to improve the performance of the algorithm, including limited-memory subset-row cuts and route enumeration, which we will detail later. We next present the general structure of our framework.

- Step 1. Choose an unprocessed node from the branch-and-bound tree. If the difference between the upper bound associated with this node and the current best lower bound is below the predefined tolerance, fathom the node and return to Step 1. Otherwise, continue.
- Step 2. Solve the linear relaxation of the restricted master problem (RMP), (see Section Section 3.4.1).
- Step 3. Use the heuristic pricing procedure (see Section 3.4.3) to search for columns with positive reduced cost. If such columns are found, add them to RMP and go back to Step 2. Otherwise, continue.
- Step 4. Use the exact pricing procedure (see Sections 3.4.3 and 3.4.3) to search for columns with positive reduced cost. If new columns are found, add them to the RMP and go to Step 2. Otherwise, the optimal value of RMP is an upper bound on the optimal objective function value of the problem. If this upper bound is below the current best lower bound, fathom the node and return to Step 1. Otherwise, continue.
- Step 5. Generate $lm - SRCs$ cuts (see Section 3.4.2). If any violated cuts are found, add them to the RMP and go to Step 2. Otherwise, continue.
- Step 6. If the solution of the linear relaxation of the RMP is fractional, perform route enumeration (see Section 3.4.5) to generate extra columns. Update the lower bound by solving the RMP as an integer program.

- Step 7. Solve the linear relaxation of the RMP and branch (see Section 3.4.4) if the solution is fractional, adding the resulting child nodes to the set of unprocessed branch nodes. Mark the current node as processed and go back to Step 1.

We note that executing the exact pricing procedure (Step 4) when the heuristic procedure does not yield any columns with positive reduced cost ensures that the approach will solve the linear relaxation of the master problem at a node of the branch-and-bound tree. Much of this framework can be used, unchanged, to solve any of the four variants we consider. However, in the following sections, we will highlight the steps that need to be adapted when solving a specific variant.

3.4.1 Set packing formulation

In this section, we describe how we reformulate the integer program presented in Section 3.3 to a set packing problem. To do so, we let Ψ represent the set of feasible routes satisfying Constraints (3.7)-(3.17). We associate with each route $r \in \Psi$ the value v_r, which represents the profits generated by that route minus the costs it incurs:

$$v_r = \sum_{i \in r} p_i - c_t(\delta^r_{2n+1}(t^r_0) - t^r_0) - Z \tag{3.23}$$

Here, t^r_0 is the optimal departure time from the depot (0) on route r and $\delta^r_{2n+1}(t^r_0)$ is the resulting arrival time at the depot. For the variants $\Omega(Fix, All)$ and $\Omega(Fix, Prof)$, we set $t^r_0 = 0$.

For the two variants with flexible departure time, $\Omega(Flex, All)$ and $\Omega(Flex, Prof)$, we must solve for t^r_0. To do so, we let u_i denote the node at position i in route r (u_{i-1} is then the predecessor node of u_i on that route). We define $\delta^r_{u_i}(t)$ to be a "ready time" function, as it will indicate when a vehicle following route r can depart from u_i, assuming it departs from the depot at time t. Because travel times adhere to the FIFO property, this ready time function is nondecreasing in t, and can be calculated recursively for each node in the route as follows:

$$\delta^r_{u_i}(t) = \begin{cases} t & \text{if } i = 0, \\ \max\{e_{u_i} + s_{u_i}, \delta^r_{u_{i-1}}(t) + \tau_{u_{i-1},u_i}(\delta^r_{u_{i-1}}(t)) + s_{u_i}\} & \text{otherwise.} \end{cases} \tag{3.24}$$

The duration of the route, given that it departs from node 0 at time t can be calculated as $\delta^r_{2n+1}(t) - t$. As a result, the departure time from the depot that leads to the shortest route duration can be calculated as follows:

$$t^r_0 = argmin_{t \in T}\{\delta^r_{2n+1}(t) - t\} \tag{3.25}$$

We also associate with route $r \in \Psi$ the parameter $a_{ir}, i \in N_P$, which denotes whether route r visits node i. Finally, we let the binary variable w_r indicate whether route r is chosen. The resulting set packing formulation is as follows:

$$v(MP) = \max \sum_{r \in \Psi} v_r w_r \tag{3.26}$$

subject to

$$\sum_{r \in \Psi} a_{ir} w_r \leq 1 \qquad \forall i \in N_P \tag{3.27}$$

$$\sum_{r \in \Psi} w_r \leq K \tag{3.28}$$

$$w_r \in \{0, 1\} \qquad \forall r \in \Psi \tag{3.29}$$

Constraints (3.27) ensures that each request is visited at most once. For the $\Omega(Fix, All)$ and $\Omega(Flex, All)$ variants, this constraint is changed to an equality constraint. Constraint (3.28) limits the number of vehicles used to K.

As noted, because there may be a large number of routes (i.e. $|\Psi|$ is large) for many instances, we dynamically generate routes, r, to add to Ψ with column generation (Desaulniers et al. [30]). To do so, we repeatedly solve a restricted master problem, $RMP(\Psi^{'})$, based on a subset, $\Psi^{'} \subset \Psi$, of feasible routes. The subset $\Psi^{'}$ is initialized with a set of routes wherein each route visits a single pair of requests. At each iteration of the algorithm, the linear relaxation of $RMP(\Psi^{'})$ is solved and the corresponding dual variables are recorded. We let $\pi_i, i \in N_P$, denote the nonnegative dual variable associated with constraint (3.27) and π_0 the nonnegative

dual variable associated with constraint (3.28). The algorithm then solves a pricing problem to determine whether there are variables, $w_r, r \in \Psi/\Psi^{'}$, that are not yet considered in $RMP(\Psi^{'})$ and have positive reduced cost. In this context, the reduced cost of a variable r is calculated as:

$$\overline{v}_r = v_r - \sum_{i \in N_P \cup \{0\}} a_{ir}\pi_i = (\sum_{i \in r} p_i - c_t(\delta^r_{2n+1}(t^r_0) - t^r_0) - Z_r) - \sum_{i \in N_P \cup \{0\}} a_{ir}\pi_i \tag{3.30}$$

We note that whenever we have an upper bound, $\overline{\overline{v}}_r$ on the route with the largest reduced cost, we can derive an upper bound on the optimal objective function value of the original problem [69]. Specifically, we let $v(RMP)_{LPR}$ denote the optimal value of the linear relaxation of the RMP. $v(MP)_{LPR}$ is defined similarly. Given that at most k routes can be chosen, we have:

$$v(RMP)^{dual}_{LPR} = v(RMP)_{LPR} + k * \overline{\overline{v}} \geq v(MP)_{LPR} \tag{3.31}$$

This bound, $v(RMP)^{dual}_{LPR}$, provides a stopping criteria for column generation other than the absence of columns with positive reduced cost. Speficially, if $(v(RMP)^{dual}_{LPR} - v(RMP)_{LPR})/v(RMP)_{LPR}$ is within some tolerance (i.e. 1% in our experiments), we can terminate the column generation procedure.

3.4.2 Valid inequalities

Extended formulations of a problem can be further strengthened with the addition of valid inequalities, or, *cuts*. Jepsen et al. [58] applied the Chvatal-Gomory rounding procedure to Constraints (3.27) to develop the following family of valid inequalities for VRPTWs.

Specifically, given a base set $C \subseteq N_P$ and a multiplier γ, $0 < \gamma < 1$, we define the following $(C, \gamma)-$Subset Row Cut (SRC) as follows:

$$\sum_{r \in \Psi} \lfloor \gamma \sum_{i \in C} a_{ir} \rfloor w_r \leq \lfloor \gamma |C| \rfloor \tag{3.32}$$

While these cuts have been shown [58] to strengthen the linear relaxation, as they are defined on the route variables they can make the pricing subproblem harder to solve with dynamic programming. Specifically, each cut adds a dimension to the labels that must be stored. Recently, Pecin et al. [84] proposed limited-memory $SRCs$, that, while weaker than the $SRCs$, have a reduced impact on the pricing subproblem. As such, while the use of these limited-memory cuts, in comparison to those of Jepsen et al. [58], may lead to more time spent generating cuts, less time solving the pricing problem, and an overall reduction in computation time.

The definition of the limited memory $(C, M, \gamma)-$Subset Row Cut ($lm - SRC$) requires an additional set M, $C \subset M \subset N_P$. The resulting cut is:

$$\sum_{r \in \Psi} \alpha(C, M, \gamma, r) w_r \leq \lfloor \gamma |C| \rfloor \tag{3.33}$$

where α is a function of C, M, γ, r computed by the Algorithm 2.

Algorithm 2: Computing $\alpha(C, M, \gamma, r)$ of w_r in a $lm - SRC$

input : A feasible route r
output: A coefficient α

```
1  α ← 0 , state ← 0
2  for (each node i in N_P) do
3      if (i ∉ M) then
4          state ← 0
5      else
6          if (i ∈ C) then
7              state ← state + γ
8              if (state > 1) then
9                  α ← α + 1
10                 state ← state + 1
```

When $M = N_P$, the function α will return $\lfloor \gamma \sum_{i \in C} a_{ir} \rfloor$ and the $lm - SRC$ will be identical to a SRC. On the other hand, when M is not equal to N_P, the $lm - SRC$ may be a weakening of its corresponding SRC. This happens because the variable $state$ in the function is set to 0 once the route r leaves M.

The function $\alpha(C, M, \gamma, r)$ also impacts how the $lm - SRCs$ should be taken into account in the pricing problem discussed later. Specifically, for each $lm - SRC$, each label needs to store its state in the corresponding partial paths. However, the coefficient does not need to be stored in the label. Instead, whenever a label extension causes an increment of the coefficient of an

$lm-SRCs$ according to function α, the value of its dual variable is immediately added to the reduced cost of the generated label. Moreover, the number of possible states of a $lm-SRC$ depends on its γ. In this chapter, we only consider $lm-SRCs$ defined for subsets C with cardinality 3. In this case, $\gamma = 1/2$ and the state can be only 0 or $1/2$. Therefore, it can be represented by a single bit.

The $lm-SRCs$ have potential advantage over the classical $SRCs$ due to their reduced impact on the pricing problem when $|M| << |N_P|$. In the following, we introduce the strategy for separating $lm-SRCs$ such that small memory sets can be obtained. First, a violated $SRC(C,\gamma)$ need to be identified. Then, a minimal set M is determined such that $lm-SRC(C,M,\gamma)$ has the same violation. For example, take a given fractional solution with paths that visit $C=\{1,2,3\}$ at least twice. In this example they are: $r_1=(0-1-4-2-5-...-0)$ with $\lambda_1=0.5$, $r_2=(0-3-6-2-5-...-0)$ with $\lambda_2=0.5$ and $r_3=(0-3-7-1-4-...-0)$ with $\lambda_3=0.5$. The $SRC(C,1/2)$ $\lambda_1+\lambda_2+\lambda_3\leq 1$ has a violation of 0.5. A minimal set M that yields a $lm-SRC(C,M,1/2)$ with the same violation can be $M=C\cup\{4\}\cup\{6\}\cup\{7\}$. Moreover, like Cherkesly et al. [18] and Pecin et al. [84], we also limit the usage of $lm-SRCs$ to prevent running out of memory when solving the pricing problem.

3.4.3 The pricing problem

In this section, we present the algorithm used to solve the pricing problem, which aims to find a route, r that minimizes $\overline{v}_r$ of function (3.30) while taking into consideration the constraints that model feasibility. The pricing problem can be seen as a variant of the elementary shortest path problem, which is known to be NP-hard. We solve this problem with the tailored labeling algorithm proposed by Sun et al. [109].

In this algorithm, a label L is used to record the information associated with a path that starts at the depot and ends at node $v(L)$. The label includes information regarding the requests already visited, the cumulative reduced cost, and the ready time function. Starting from the initial label L_0 associated with the depot, the algorithm propagates labels toward the destination depot with a feasible extension process. Dominance tests are also employed to eliminate unpromising labels.

We next briefly present the tailored labeling algorithm introduced by Sun et al. [109], which is used to solve the pricing problem in our branch-and-price framework. Then, we explain how to incorporate the dual information associated with generated $lm-SRCs$ into this tailored

labeling algorithm. We finish with a discussion of a pricing heuristic we use to accelerate the branch-cut-and-price algorithm.

Pricing without lm-SRCs

In this section, we introduce the forward tailored labeling algorithm based on Sun et al. [109]. The interested reader is referred to Sun et al. [109] for the backward version and related bidirectional search techniques. For each forward label L_f, we use the same notation as introduced in Sun et al. [109], which is explained as the following:

$p(L_f)$	*	The partial path of label L_f.
$v(L_f)$	*	The last node of the partial path $p(L_f)$.
$L^{-1}(L_f)$	*	The parent label from which L_f originates by extending it with $v(L_f)$.
$O(L_f)$	*	The set of onboard requests in the partial path $p(L_f)$
$U(L_f)$	*	The set of unreachable requests in the partial path $p(L_f)$.
$q(L_f)$	*	The load of the vehicle after visiting the last node $v(L_f)$.
$\delta_{L_f}(t)$	*	The ready time function at $v(L_f)$
$r(L_f)$	*	The overall profits collected by serving the requests visited on the partial path $p(L_f)$.
$\pi(L_f)$	*	The sum of the dual values associated with the pickup nodes visited on the partial path $p(L_f)$ and dual value π_0 associated with constraint (3.28).

Only the items marked with a * are stored in the label. A request is said to be on-board if it has been started, but not completed, i.e., the request has been picked up but not delivered to its delivery mode. A request R_i is said to be unreachable if i has already been visited, or if traveling either directly or indirectly from $v(L_f)$ to i violates the time window at pickup node i. Therefore, $O(L_f) \subseteq U(L_f)$. The ready time function $\delta_{L_f}(t)$ is piece-wise linear and represents the ready time at $v(L_f)$ if the vehicle departed at the origin depot at t and reached $v(L_f)$ through partial path $p(L_f)$. For $\Omega(Flex, All)$ and $\Omega(Flex, Prof)$, we need to store this function for each generated label, while for $\Omega(Fix, All)$ and $\Omega(Fix, Prof)$ we only need to store $\delta_{L_f}(0)$, which is the earliest ready time at $v(L_f)$ as the departure time at the origin depot is imposed to be 0. The partial path can be deduced from iteratively checking the last node visited in the parent label of which the label was an extension.

Given a label $L_f^{'}$, its extension along an arc $(v(L_f^{'}), j)$ is feasible only if it satisfies the following conditions:

$$\delta_{L_f^{'}}(0) + \tau_{v(L_f^{'}),j}(\delta_{L_f^{'}}(0)) + s_j \leq \min\{t_m, l_j + s_j\} \quad \wedge \quad j \in N \setminus \{0\} \tag{3.34}$$

$$q(L_f) + q_j \leq Q \quad \wedge \quad j \in N \setminus \{0\} \tag{3.35}$$

Condition (3.34) stipulates that only if node j can be reached within its time window and the extension to node j takes place before t_m is exceeded, the extension to node j is allowed to be performed. Condition (3.35) ensures that the label can only be extended to node j if the remaining capacity is enough to deal with the load of node j. Besides, $L_f^{'}$ and j need satisfy one of the following three conditions:

$$j \notin U(L_f^{'}) \quad \wedge \quad j \in N_P \tag{3.36}$$

$$j - n \in O(L_f^{'}) \quad \wedge \quad j \in N_D \tag{3.37}$$

$$O(L_f^{'}) = \emptyset \quad \wedge \quad j = 2n + 1 \tag{3.38}$$

Condition (3.36) states that if node j is a pickup node, it should have never been visited before on the route. Condition (3.37) shows that if j is a delivery node then its corresponding pickup node $j - n$ should already been visited. Condition (3.38) states that if j is the end depot, then all visited requests should have been completed. In the presence of those conditions, we only keep elementary paths that satisfy precedence constraint (3.10). If the extension along the arc $(L_f^{'}, j)$ is feasible, then a new label L_f is created. The information in label L_f is updated as follows:

$$L^{-1}(L_f) = L_f^{'} \tag{3.39}$$

$$v(L_f) = j \tag{3.40}$$

$$\delta_{L_f}(t) = \max\{e_j + s_j, \delta_{L_f^{'}}(t) + \tau_{L_f^{'},j}(\delta_{L_f^{'}}(t)) + s_j\} \tag{3.41}$$

$$q(L_f) = q(L_f^{'}) + q_j \tag{3.42}$$

$$r(L_f) = \begin{cases} r(L_f^{'}) + r_j & \text{if } j \in N_P, \\ r(L_f^{'}) & \text{otherwise.} \end{cases} \tag{3.43}$$

$$\pi(L_f) = \begin{cases} \pi(L_f^{'}) + \pi_j & \text{if } j \in N_P, \\ \pi(L_f^{'}) & \text{otherwise.} \end{cases} \tag{3.44}$$

$$O(L_f) = \begin{cases} O(L_f^{'}) \cup \{j\} & \text{if } j \in N_P, \\ O(L_f^{'}) \setminus \{j - n\} & \text{if } j \in N_D. \end{cases} \tag{3.45}$$

$$U(L_f) = \begin{cases} U(L_f^{'}) \cup \{j\} & \text{if } j \in N_P, \\ U(L_f^{'}) & \text{otherwise.} \end{cases} \tag{3.46}$$

Equations (3.39)-(3.43) set the parent label, the last visited node, the ready time function, the load, and the collected profits of the new label, respectively. Equation (3.45) updates the set of onboard requests $O(L_f)$ and Equation (3.46) updates the set of unreachable requests $U(L_f)$. Similar to the idea of Feillet et al. [38] and Sun et al. [109], the unreachable request set $U(L_f)$ is further enriched by adding requests of which the pickup node is unreachable from $v(L_f)$.

Before the dominance criterion is introduced some definitions need to be provided. The first one is the definition of interval I as explained in Sun et al. [109].

$$I = (-\infty, \max(dom(L_f^1)) - \max(dom(L_f^2))) \tag{3.47}$$

Based on I, we define a real number $\phi(L_f^1, L_f^2)$,

$$\phi(L_f^1, L_f^2) = \max\{x \in I : \delta_{L_f^1}(\max\{0, t + x\}) \leq \delta_{L_f^2}(t), \forall t \in dom(L_f^2)\} \tag{3.48}$$

This function describes for the partial path represented by label L_f^1, how much the departure time can be postponed (when $\phi(L_f^1, L_f^2)$ is positive) or arranged in advance (if $\phi(L_f^1, L_f^2)$ is negative), compared to the departure time of partial path $p(L_f^2)$, such that $v(L_f^1)$ is reached at the same time or earlier via partial path $p(L_f^2)$ than via partial path $p(L_f^2)$.

A label L_f^1 dominates a label L_f^2 if every feasible extension of $p(L_f^2)$ yields a complete route with reduced cost smaller than the feasible route generated by using the same extension into $p(L_f^1)$. The following six conditions in Proposition 3, together, are sufficient to ensure such dominance:

Proposition 3. *Label L_f^2 is dominated by label L_f^1 if*
1. $v(L_f^1) = v(L_f^2)$
2. $U(L_f^1) \subseteq U(L_f^2)$
3. $O(L_f^1) = O(L_f^2)$
4. $\delta_{L_f^1}(0) \leq \delta_{L_f^2}(0)$
5. $r(L_f^1) - \pi(L_f^1) \geq r(L_f^2) - \pi(L_f^2) - \phi(L_f^1, L_f^2)$
6. $q(L_f^1) \leq q(L_f^2)$

Proof of Proposition 3. To prove this proposition, we consider two labels L_f^1 and L_f^2, each of which satisfies the six conditions of Proposition 3. We will prove two claims regarding these

labels, which will in turn prove this proposition:

1. Any feasible extension L of $p(L_f^2)$ that yields a complete route, $p(L_f^2 \bigoplus L)$, is also a feasible extension of $p(L_f^1)$, and yields a complete route $p(L_f^1 \bigoplus L)$, and,
2. For any feasible extension L that led to $p(L_f^1 \bigoplus L)$ and $p(L_f^2 \bigoplus L)$, the reduced cost $\overline{v}(L_f^1 \bigoplus L)$ is not smaller than the reduced cost $\overline{v}(L_f^2 \bigoplus L)$.

Regarding claim (1), we must show that $p(L_f^1 \bigoplus L)$ does not carry more load than the capacity of the vehicle, is elementary, and does not violate any time windows. Regarding vehicle capacity, we note that due to condition 6, the load associated with path $p(L_f^1 \bigoplus L)$ will not exceed vehicle capacity as it did not on path $p(L_f^2 \bigoplus L)$. We can conclude that $p(L_f^1 \bigoplus L)$ is elementary because conditions 2 and 3 ensure that all nodes visited in $p(L_f^1)$ are either nodes visited in $p(L_f^2)$ or nodes which could not be reached by any extension of label L_f^2. Thus, as $p(L_f^2 \bigoplus L)$ is elementary, so is $p(L_f^1 \bigoplus L)$. Turning to time windows, we observe that condition 4, coupled with the presumption that travel times adhere to the FIFO property implies that $p(L_f^1 \bigoplus L)$ will not violate any time windows. For example by departing at 0, the vehicle arrives at $v(L_f^1)$ at time $\delta_{L_f^1}(0)$ which is by condition 4 smaller than or equal to $\delta_{L_f^2}(0)$. Therefore, a departure at 0 over path $p(L_f^1)$ does not violate any time windows.

Regarding claim (2), consider an extension, L, of $p(L_f^2)$ that yields a complete route (i.e. $v(L_f^2 \bigoplus L) = 2n + 1$. Letting $L_f^{1*} = L_f^1 \bigoplus L$ and $L_f^{2*} = L_f^2 \bigoplus L$, we will show that $\overline{v}(L_f^{1*}) \geq \overline{v}(L_f^{2*})$. To do so, we let $t_0^2 = argmin_{t \in dom(L_f^{2*})}\{\delta_{L_f^{2*}}(t) - t\}$ represent the optimal departure time from the depot for path $p(L_f^{2*})$ and $r(L)$ the sum of the profits associated with the nodes visited along the path $p(L)$ associated with the extension, L. Given this, we have that the reduced cost of the path is:

$$\begin{aligned}\overline{v}(L_f^{2*}) &= r(L_f^{2*}) - (\delta_{L_f^{2*}}(t_0^2) - t_0^2) - \pi(L_f^{2*}) - Z \\ &= r(L_f^2) + r(L) - (\delta_{L_f^{2*}}(t_0^2) - t_0^2) - (\pi(L_f^2) + \pi(L)) - Z \end{aligned} \tag{3.49}$$

Next, consider the path $p(L_f^{1*})$, which departs from the depot at time $t_0^1 = \max\{0, t_0^2 + \phi(L_f^1, L_f^2)\}$. We note that the time t_0^1 is a feasible departure time for label L_f^{1*} because a departure time of 0 is always possible (as the extension of L_f^1 by L is feasible) and the definition of $\phi(L_f^1, L_f^2)$ in equation (3.48) implies that $t_0^2 + \phi(L_f^1, L_f^2)$ belongs to $dom(L_f^1)$ when

it is nonnegative. This departure time t_0^1 ensures that we reach node $v(L_f^1)$ at time $\delta_{L_f^2}(t_0^2)$ or earlier, meaning we have the following inequality:

$$\delta_{L_f^1}(t_0^1) \leq \delta_{L_f^2}(t_0^2). \tag{3.50}$$

Moreover, this value of t_0^1 enables us to show that $\overline{v}(L_f^{2*})$ is a lower bound on $\overline{v}(L_f^{1*})$:

$$\begin{aligned}
\overline{v}(L_f^{1*}) &\geq r(L_f^{1*}) - (\delta_{L_f^{1*}}(t_0^1) - t_0^1) - \pi(L_f^{1*}) - Z && (3.51)\\
&= r(L_f^1) + r(L) - (\delta_{L_f^{1*}}(t_0^1) - \max\{0, t_0^2 + \phi(L_f^1, L_f^2)\}) - (\pi(L_f^1) + \pi(L)) - Z && (3.52)\\
&\geq (r(L_f^1) - \pi(L_f^1)) + (r(L) - \pi(L)) + (\delta_{L_f^{2*}}(t_0^2) - \max\{0, t_0^2 + \phi(L_f^1, L_f^2)\}) - Z && (3.53)\\
&\geq (r(L_f^1) - \pi(L_f^1)) + (r(L) - \pi(L)) - (\delta_{L_f^{2*}}(t_0^2) - t_0^2 - \phi(L_f^1, L_f^2)) - Z && (3.54)\\
&\geq (r(L_f^2) - \pi(L_f^2) - \phi(L_f^1, L_f^2)) + (r(L) - \pi(L)) - (\delta_{L_f^{2*}}(t_0^2) - t_0^2 - \phi(L_f^1, L_f^2)) - Z && (3.55)\\
&= r(L_f^2) + r(L) - (\delta_{L_f^{2*}}(t_0^2) - t_0^2) - (\pi(L_f^2) + \pi(L)) - Z && (3.56)\\
&= r(L_f^{2*}) - (\delta_{L_f^{2*}}(t_0^2) - t_0^2) - \pi(L_f^{2*}) - Z = \overline{v}(L_f^{2*}) && (3.57)
\end{aligned}$$

We note that inequality (3.53) is derived in part from inequality (3.50). Inequality (3.54) is derived in part from the fact that $\forall x \in R : -\max\{0, x\} \leq -x$. Finally, inequality (3.55) relies upon condition 5 of Proposition 3. □ □

Pricing algorithm with lm-SRCs

When pricing in the presence of lm-SRCs, each label must have an additional dimension for each $lm - SRC$ with non-zero dual value in the current restricted master problem solution (hereafter called an active $lm - SRC$). To discuss this in more detail, as well as the corresponding dominance criteria, we define the following notation:

$E(L_f)$	*	The vector of states associated with the $lm - SRCs$.
σ_h	*	The dual variable associated with $lm - SRC$ h.

For an active $lm - SRC$ h, $E(L_f)[h]$ will first be assigned the value $E(L^{-1}(L_f))[h]$. However, if $v(L_f) \notin M(h)$, it is set to zero. Then, if $v(L_f) \notin C$, it is increased by γ. Finally, $E(L_f)[h]$ is decremented by 1 if it becomes larger than or equal to 1.

In addition, the calculation of $\pi(L_f)$ also sums over the duals associated with active $lm-SRC$s. We denote this quantity as $\sum_{h\in DEC_{L_f}}\sigma_h$, where DEC_{L_f} is subset of the active $lm-SRCs$ whose states were decremented by one when calculating L_f.

Given these definitions, the following six conditions, together, are sufficient to guarantee dominance:

Proposition 4. *Label L_f^2 is dominated by label L_f^1 if*
1. $v(L_f^1)=v(L_f^2)$
2. $U(L_f^1)\subseteq U(L_f^2)$
3. $O(L_f^1)=O(L_f^2)$
4. $\delta_{L_f^1}(0)\leq\delta_{L_f^2}(0)$
5. $r(L_f^1)-\pi(L_f^1)\geq r(L_f^2)-\pi(L_f^2)-\phi(L_f^1,L_f^2)-\sum_{h\in H:E(L_f^1)[h]>E(L_f^2)[h]}\sigma_h$
6. $q(L_f^1)\leq q(L_f^2)$

The conditions of Proposition 4 are analogous to those of Proposition 3. Specifically, only condition 5 differs, in that the expression on the right-hand-side reflects the duals associated with the currently active $lm-SRC$s. As such, the proof of Proposition 4 is nearly identical to that of Proposition 3.

Proof of Proposition 4. Similar to Proposition 3, and by using conditions 2, 3, 4, and 6, we can prove that any extension L of $p(L_f^2)$ to $2n+1$ will be a feasible extension of $p(L_f^1)$ to $2n+1$ as it results in an elementary path with does not violate time windows and capacity constraints.

As to point (2), compare to Proposition 3, we also need to consider $\sum_{h\in H:E(L_f^1)[h]>E(L_f^2)[h]}\sigma_h$ in condition 5, that is an upper bound on completion of route $p(L_f^1)$ can gain over the completion of route $p(L_f^2)$, by avoiding the penalizations of revisiting the limited-memory subset-row cuts in which $E(L_f^1)[h]>E(L_f^2)[h]$.□ □

We note that the traditional $SRCs$ have a high potential for disrupting the label dominance by making Condition 5 much less likely to be satisfied. The limited-memory mechanism mitigates that difficulty. As the state of given cut h is reset to 0 whenever a route visit a pickup node which is not included in $M(h)$, its dual variable σ_h is involved in fewer label comparisons.

Heuristic pricing

To accelerate the performance of the branch-and-price algorithm, a heuristic is used to quickly search for routes with positive reduced cost. The heuristic speeds up the search by using dominance criteria that are "too strong" and may lead to the pruning of optimal solutions to the pricing problem. Specifically, the heuristic replaces $O(L_f^1) = O(L_f^2)$ with $O(L_f^1) \subseteq O(L_f^2)$, which is less restrictive and may lead to the pruning of more partial paths. However, these criteria are invalid for the problems we study as they assume the triangle inequality holds, something that is not necessarily true in the problems we study. In summary, the dominance conditions considered by the heuristic are as follows:

Heuristic pricing rule. *Label L_f^2 is dominated by label L_f^1 if*
1. $v(L_f^1) = v(L_f^2)$
2. $U(L_f^1) \subseteq U(L_f^2)$
3. $O(L_f^1) \subseteq O(L_f^2)$
4. $r(L_f^1) - \pi(L_f^1) \geq r(L_f^2) - \pi(L_f^2)$
5. $q(L_f^1) \leq q(L_f^2)$

3.4.4 Branching rules

While column generation is a scheme for solving linear programs, by embedding it within a branch-and-bound scheme it can be used to solve an integer program. Such a scheme requires defining how to sub-divide the feasible region associated with a node in a branch-and-bound tree when a solution to the linear relaxation of the RMP associated with that node is not integral (and hence not a solution to MP). Such a sub-division yields "child nodes" of the current node. We next present, in the order in which they are applied, the branching rules used by the algorithm for creating these child nodes. We note that all rules create two such child nodes.

- The first branching rule concerns the value of $m = \sum_{r \in \Psi} w_r$, the number of vehicles used in the solution. If this value is fractional, we impose $\sum_{r \in \Psi} w_r \leq \lfloor m \rfloor$ and $\sum_{r \in \Psi} w_r \geq \lceil m \rceil$ on the other branch. In the pricing problem, a dual variable associated with this new constraint in the master is attached to the depot.
- The second branching rule concerns whether a request is served. If $\sum_{r \in \Psi} a_{ir} w_r$ is fractional for some $i \in N_P$, we impose on one branch $\sum_{r \in \Psi} a_{ir} w_r = 0$ and $\sum_{r \in \Psi} a_{ir} w_r = 1$ on the other branch. For branch $\sum_{r \in \Psi} a_{ir} w_r = 0$, the pickup node i will be set as unreachable in the pricing problem. For branch $\sum_{r \in \Psi} a_{ir} w_r = 1$, a dual

variable associated with this new constraint in the master will be attached to the pickup node i.

- The third branching rule considers an expression of the solution to the linear relaxation of RMP in terms of arc variables x_{ij}. Specifically, given the solution w^* to the linear relaxation of RMP, it calculates $x^*_{ij} = \sum_{r \in \Psi' : (i,j) \in r} w^*_r$. It then looks for pairs (i, j), $i, j \in N$ such that $x^*_{ij} + x^*_{ji}$ is close to 0.5. The algorithm then imposes two branches: (1) $x^*_{ij} + x^*_{ji} \leq \lfloor x^*_{ij} + x^*_{ji} \rfloor$, and, (2) $x^*_{ij} + x^*_{ji} \geq \lceil x^*_{ij} + x^*_{ji} \rceil$. For the node associated with $x^*_{ij} + x^*_{ji} \leq \lfloor x^*_{ij} + x^*_{ji} \rfloor$, the arc (i, j) will be removed from the graph constructed when solving the pricing problem. For the other branch, all arcs that emanate from i other than (i, j) are removed from the graph constructed when solving the pricing problem. In addition, $\sum_{r \in \Psi} a_{ir} w_r = 1$ is imposed on this branch. Thus, in the pricing problem, a dual variable will be attached to the pickup node i.

The proposed branching scheme is the same for both the $\Omega(Fix, Prof)$ and $\Omega(Flex, Prof)$, while, for $\Omega(Fix, All)$ and $\Omega(Flex, All)$, only the first rule and the last rule are applicable, since all the requests are required to be visited exactly once.

We note that branching constraints that restrict the number of vehicles used or whether a request is served can yield an infeasible RMP. To ensure the RMP remains feasible, we add to the formulation a variable κ with a large (in absolute value) and negative objective function coefficient. This variable is then included in the constraints representing a request that must be served and the number of vehicles used. The resulting constraints are $\sum_{r \in \Psi} a_{ir} w_r + \kappa = 1, \sum_{r \in \Psi} w_r - \kappa \leq \lfloor m \rfloor$, and $\sum_{r \in \Psi} w_r + \kappa \geq \lceil m \rceil$.

Finally, we note that the second and third rules may yield multiple candidates for branching. Namely, there may be multiple requests that are partially served (the second rule) and/or there may be multiple arcs that are fractionally chosen (the third rule). Given a rule that yields multiple candidates, we choose one by employing strong branching [90].

3.4.5 Route enumeration

For the algorithm to terminate with a provably (near-) optimal solution to the MP, it must find such a solution and produce a dual bound that guarantees its quality. The cuts presented above are used to reduce the time needed to produce such a dual bound. To reduce the time required to find a high-quality primal solution, we employ a route enumeration technique [7]. This technique generates routes, outside of the context of solving the pricing problem, that can

then be added to the RMP. With these extra routes, the algorithm solves the RMP (an integer program) to produce a potentially improving primal solution to the MP.

We employ the restricted dynamic programming heuristic proposed above to generate routes. However, as we do not want the algorithm spending a lot of time generating routes, this heuristic limits the number of partial paths that are extended. Specifically, at each stage of its execution, the heuristic only considers the best $B/2$ partial paths that expand to a pickup node and the best $B/2$ partial paths that expand to a delivery node for further expansion (B is an algorithm parameter that we set to 8000 in our experiments). To select these partial paths, the heuristic uses a mechanism similar to that presented in Sun et al. [109]. First, we order the expanded partial paths by the earliest arrival time. Then, if two expanded partial paths have the same earliest arrival time, we order them by their objective function value. The heuristic pricing rule is also applied to prune unpromising labels. After the heuristic has finished, it returns a set of potential routes, along with the reduced cost associated with each route.

However, we do not add all generated routes to the RMP. Instead, the algorithm filters the set of generated routes based on a prediction of whether a route will appear in an improving solution. To do so, we let z_{ub}^{bn} denote the upper bound associated with the node in the branch-and-bound tree that is currently being processed. Similarly, we let z_{lb}^{cbest} denote the objective function value of the best primal solution to the MP found. With these values, we calculate the relative gap $\varrho = (z_{ub}^{bn} - z_{lb}^{cbest})/z_{ub}^{bn} \times 100$. We observe that a route can only appear in an improving solution if its reduced cost is smaller than the gap ϱ.

3.5 Computational results

In this section, we present the results of a computational study on the effectiveness of the proposed algorithmic framework at solving the problems we are interested in. All algorithms are coded in Java, using Eclipse SDK version 4.2.0, and our experiments are performed on a single thread of a server with four CPU's (2.4 GHz/6 cores) and 64GB RAM. The LP relaxations are solved with Gurobi 5.6.3. A time limit of 10 hours and a maximum memory allowance of 16GB RAM has been imposed on each run.

For our numerical study, we generated a set of instances by adopting the instances presented in Sun et al. [109], which in turn were derived from those proposed by Røpke and Cordeau [91] for the PDPTW. Specifically, our instances are based on the same node coordinates, time windows, vehicle capacity, and load quantities as those in Røpke and Cordeau [91]. The

instances are divided into four groups, with each group containing nine instances. Table 3.1 presents the vehicle capacity, Q, and time window width, W, for each group. The profits were adapted in order to balance the fixed cost associated with using a vehicle (which was not included in Sun et al. [109]) and the profit associated with serving a request. Specifically, we set the fixed cost equal to 100, while randomly assigning to each request a profit in the interval $[50, 150]$.

Table 3.1: Characteristics of the instances used in computational study.

Group	Q	W
AA	15	60
BB	20	60
CC	15	120
DD	20	120

Regarding the time-dependent travel times, we use the same values as in Sun et al. [109]. There, road congestion is modeled by a so-called speed model which consists of different speed profiles. It is used to determine the travel time between two nodes on a specific departure time. As denoted in Table 3.2, five types of speed profiles are considered: slow speed (SS), normal speed with morning peak (NSMP), normal speed with evening peak (NSEP), fast speed with two peaks (FSTP) and high speed (HS). We refer the reader to Sun et al. [109] for detailed information regarding the explanation of the different speed profiles and how they are assigned to the arcs. Furthermore, without loss of generality, it is assumed that the breakpoints are the same for all speed profiles as congestion tends to happen around the same time regardless of the speed profiles' type. Each speed profile has four non-overlapping time periods with constant speed, reflecting two congested periods and two periods with normal traffic conditions. As a result, in terms of our model, the number of time zones m equals 7 for the instances we solve.

Table 3.2: Speed Profiles

Congestion description Time periods	Morning peak 7 am-9 am	Normal 9 am-5 pm	Evening peak 5 pm-7 pm	Normal 7 pm-9 pm
1.SS	0.5	0.81	0.5	0.81
2.NSMP	0.67	1.33	0.88	1.33
3.NSEP	0.88	1.33	0.67	1.33
4.FSTP	0.85	1.5	0.85	1.5
5.HS	1.0	2.0	1.0	2.0

Our computational experiments focus on understanding the following issues. First, we compare the performance of a commercial MIP solver (Gurobi) when solving the compact formulation of each variant with that of the pure branch and price framework presented in Section 3.4 solving the corresponding extended formulation. Second, we perform an in-depth analysis of the performance of the exact framework in order to determine the appropriate combination of speed-up techniques (i.e. $lm - SRCs$ and Route Enumeration) for each variant. Third, as each variant represents a different degree of operational flexibility, we analyze solutions to the different variants to quantify the value of that flexibility. We note that to test the correctness of our algorithm and implementation, we executed it on instances solved by the method presented in Røpke and Cordeau [91]. We observed that the framework we propose was able to generate the same results, however, as expected, in larger computation times. These larger computation times can be attributed to the fact that algorithm presented in Røpke and Cordeau [91] includes techniques that presume the triangle inequality is valid. Our algorithm is designed for problems with time-dependent travel times, wherein the triangle inequality does not necessarily hold, and thus does not employ those techniques. Detailed, instance-level results can be found in the Appendix in Table 9.

We note that while the following sections contain summary statistics derived from the computational results, detailed results can be found in 6.2.

3.5.1 Performance comparison of MIP solver and exact framework

We first, for each variant, compare the performance of a MIP solver, Gurobi 5.6.3, when solving the compact formulation with the proposed exact framework solving the corresponding extended formulation. We note that Gurobi was executed for ten hours and with all parameters set to their default values. In this analysis, we only consider four instances from each group (AA, BB, CC, and DD); those with 10,15, 20 or 25 requests, as Gurobi already could not solve instances with 20 or 25 requests.

We first present in Table 3.3 the percentage of instances solved by the MIP solver (Gurobi) and the exact framework (Exact), for each variant by the number of requests. We see that while the MIP solver is unable to solve many of the instances, including none with 20 or 25 requests, the proposed exact framework is able to solve almost all instances within 10 hours.

Continuing the comparison, in Table 3.4 we present averages of the time to termination for both Gurobi and the exact method. We see that in addition to solving all of the instances, the

Table 3.3: % instances solved by MIP solver and BP

	$\Omega(Fix, All)$		$\Omega(Fix, Prof)$		$\Omega(Flex, All)$		$\Omega(Flex, Prof)$	
# Requests	Gurobi	Exact	Gurobi	Exact	Gurobi	Exact	Gurobi	Exact
10	100.00%	100.00%	100.00%	100.00%	75.00%	100.00%	50.00%	100.00%
15	50.00%	100.00%	25.00%	100.00%	0.00%	100.00%	0.00%	100.00%
20	0.00%	100.00%	0.00%	100.00%	0.00%	100.00%	0.00%	100.00%
25	0.00%	50.00%	0.00%	100.00%	0.00%	100.00%	0.00%	100.00%
Average	37.50%	87.50%	31.25%	100.00%	18.75%	100.00%	12.50%	100.00%

exact framework was able to do so in relatively little time. Gurobi required on average the largest computation time for the most flexible variant, $\Omega(Flex, Prof)$ (i.e. the variant with the largest solution space), while our framework required on average the largest computation time for the least flexible variant, $\Omega(Fix, All)$ (i.e. the variant with the smallest solution space). Therefore, we did a more in depth analysis of the performance of our framework in the next section.

Table 3.4: Solve time (seconds) for MIP solver and the exact framework

	$\Omega(Fix, All)$		$\Omega(Fix, Prof)$		$\Omega(Flex, All)$		$\Omega(Flex, Prof)$	
# Requests	Gurobi	Exact	Gurobi	Exact	Gurobi	Exact	Gurobi	Exact
10	1,704.1	0.6	643.6	0.4	16,961.3	0.9	18,245.2	1.0
15	20,890.3	4.4	27,595.1	1.9	36,000.0	7.0	36,000.0	7.1
20	36,000.0	210.3	36,000.0	7.4	36,000.0	371.7	36,000.0	21.2
25	36,000.0	21,237.2	36,000.0	27.4	36,000.0	1,584.9	36,000.0	126.2
Average	23,648.6	5363.1	25,059.7	9.3	31,240.3	491.1	31,561.3	38.9

3.5.2 Analyzing the performance of the exact framework

Having established that the exact framework is more effective at solving instances of each variant, we next study its performance in greater detail. For each variant, we assess whether Limited-memory Subset-row cut and Route enumeration techniques improve the performance of the exact framework. To do so, we report in Table 3.5 how often the exact framework, when augmented with one or both of these techniques, performs better than when it does not use either. We note that in these results, we only consider instances that have 30 or fewer requests. We also do not consider instances that can be solved at the root node before any of the techniques would be employed.

How we compare the performance of two methods (the exact framework with techniques and without) on an instance depends on whether either method could solve the instance to optimality in the given time limit. If either could solve the instance, then we record the

Table 3.5: Performance of exact framework with and without enhancements

	$\Omega(Fix, All)$				$\Omega(Fix, Prof)$			
Options	$\#Better$	$\#Equal$	$\#Worse$	Overall	$\#Better$	$\#Equal$	$\#Worse$	Overall
BP+1	1	7	5	-4	0	1	4	-4
BP+2	4	5	4	0	0	0	5	-5
BP+1+2	5	5	3	2	0	0	5	-5
	$\Omega(Flex, All)$				$\Omega(Flex, Prof)$			
Options	$\#Better$	$\#Equal$	$\#Worse$	Overall	$\#Better$	$\#Equal$	$\#Worse$	Overall
BP+1	2	3	8	-6	1	4	6	-5
BP+2	6	3	4	2	2	4	5	-3
BP+1+2	9	3	1	8	4	3	4	0

1: Route Enumeration.
2: Limited-memory Subset-row Cuts.

one that could do so in less time as performing "better" on that instance, and the one that took more time as performing "worse." Note that if one method solves and the other does not, then the one that does is recorded as performing "better" and the one that does not is recorded as performing "worse." If neither could solve the instance, then we compare the methods in terms of the quality of the objective function values of the feasible solutions they produce. For either metric, there are also instances wherein the two methods perform equally well. We summarize these relative performance measures in the column "Overall", which we calculate as "# Better" - "# Worse." We see that for both $\Omega(-, All)$ variants, using both techniques ($BP+1+2$) leads to the best performance. However, the original branch and price algorithm (BP) performs much better than $BP + 1 + 2$ on $\Omega(Fix, Prof)$ and comparably on $\Omega(Flex, Prof)$. Therefore, in the remainder of this section, we will use the Branch-Cut-and-Price framework with route enumeration ($BP + 1 + 2$) for the $\Omega(-, All)$ variants and our regular Branch-and-Price framework (BP) for the $\Omega(-, Prof)$ variants.

Next, we study the performance of the selected exact frameworks on larger instances, with up to 45 requests. To that effect, we present in Table 3.6 three statistics measuring the performance of the algorithm when solving instances of each variant and by number of requests: (1) The percentage of instances solved (% Solved), (2) The average time to termination, in seconds, (Time), and, (3) The optimality gap reported at termination (Gap). Detailed, instance-level results can be found in the Appendix in Tables 4-7.

We see that for both $\Omega(-, Prof)$ variants, all instances up to 30 requests can be solved to optimality, while the same is not true for the $\Omega(-, All)$ instances (one instance (DD30) for $\Omega(Fix, All)$ and two instances (AA30 and DD30) for $\Omega(Flex, All)$). We also note that 30 requests is the point at which the exact framework begins to struggle to solve instances,

Table 3.6: Performance of exact framework on larger instances

	$\Omega(Fix, All)$			$\Omega(Fix, Prof)$			$\Omega(Flex, All)$			$\Omega(Flex, Prof)$		
# Requests	% Solved	Time	Gap	% Solved	Time	Gap	% Solved	Time	Gap	% Solved	Time	Gap
10	100%	0.7	0.0%	100%	0.4	0.4%	100%	1.0	0.0%	100%	1.0	0.0%
15	100%	2.8	0.0%	100%	1.9	0.0%	100%	5.9	0.0%	100%	7.1	0.0%
20	100%	1,015.3	0.0%	100%	7.4	0.0%	100%	317.4	0.0%	100%	21.2	0.0%
25	75%	11,809.4	1.9%	100%	27.4	0.0%	100%	2,138.0	0.0%	100%	126.2	0.0%
30	50%	18,276.2	8.4%	100%	475.0	0.0%	50%	24,340.3	4.7%	100%	4,405.3	0.1%
35	50%	18,584.4	20.4%	50%	18,104.0	1.1%	25%	27,006.9	3.1%	75%	19,506.2	0.1%
40	50%	18,417.9	12.5%	50%	18,191.0	1.0%	25%	27,996.8	0.1%	50%	18,239.1	0.6%
45	25%	27,876.3	14.5%	50%	18,084.6	5.0%	25%	27,898.3	1.9%	50%	18,357.0	0.8%
Average	68.8%	11,997.88	7.2%	81.3%	6,861.5	0.9%	65.6%	13,713.1	1.2%	84.4%	7,582.9	0.2%

with a dramatic increase in the solution time and optimality gap reported at termination. We conclude from this table that the exact framework is more effectiving at solving the $\Omega(-, Prof)$ variants, as it tends to solve more instances, take less time, and report a smaller optimality gap at termination.

To better understand when the exact framework is able to solve an instance, we present in Table 3.7 statistics related to the impact of valid inequalities and branching, averaged over the instances that are and are not solved. Specifically, we report the number of $lm - SRCs$ generated (# Cuts), the number of branches created (# Branches), and the improvement in the bound between when the root node linear programming relaxation is solved (but before cuts are added) and when the algorithm terminates (Bound gap). Specifically, for this last statistic, we let z_{LPR} represent the optimal value of the linear programming relaxation and z_{Final} represent the the final bound produced. With these statistics, we calculate the Bound gap as $(z_{Final} - z_{LPR})/z_{Final}$.

Table 3.7: Impact of cuts and branching

	$\Omega(Fix, All)$			$\Omega(Fix, Prof)$		$\Omega(Flex, All)$			$\Omega(Flex, Prof)$	
Instances	# Cuts	# Branches	Bound gap	# Branches	Bound gap	# Cuts	# Branches	Bound gap	# Branches	Bound gap
Unsolved	0.5	131.75	-3.49%	283.00	-2.70%	10.45	3.55	-0.71%	27.00	-1.27%
Solved	35.68	111.41	-2.87%	4.81	-0.34%	9.05	7.67	-0.90%	161.30	-1.29%
Average	24.69	116.83	-3.03%	49.67	-0.66%	9.53	6.25	-0.76%	140.31	-1.29%

From these results, we conclude that the $\Omega(Fix, -)$ instances are hard to solve when few $lm - SRCs$ are generated, as doing so leads to a large number of branches, with branching having a limited effect on the dual bound. On the other hand, we observe that for the $\Omega(Flex, -)$ instances, branching is more effective, as fewer branches lead to a bigger decrease in the dual bound.

3.5.3 *Value of flexibility*

Each variant studied in this chapter considers a different degree of flexibility in operations, wherein the $\Omega(Flex, Prof)$ variant is the most flexible and the $\Omega(Fix, All)$ is the least. As noted in Section 3.3, one can also derive relationships between objective function values of optimal solutions. Namely, that one must have $Obj_{\Omega(Fix,All)} \leq Obj_{\Omega(Fix,Prof)} \leq Obj_{\Omega(Flex,Prof)}$, and, $Obj_{\Omega(Fix,All)} \leq Obj_{\Omega(Flex,All)} \leq Obj_{\Omega(Flex,Prof)}$. We next quantify the relative differences in these objective function values. Specifically, for instances wherein $\Omega(Fix, All)$ could be solved to optimality, we calculate the percentage improvement in objective function value for a given variant with respect to $\Omega(Fix, All)$ with the expression $(Obj_{other} - Obj_{\Omega(Fix,All)})/(Obj_{\Omega(Fix,All)})$. Then, in Figure 3.3, we illustrate the averages of these differences for each variant and by number of requests.

Figure 3.3: Value of flexibility

We see in Figure 3.3 that the flexibility afforded by the $\Omega(Flex, Prof)$ variant leads to a significant increase in profits. This suggests that there is significant value for a carrier in moving away from publishing a fixed schedule to which they must adhere. We also see that no relationship can be derived between $\Omega(Fix, Prof)$ and $\Omega(Flex, All)$, as sometimes one yields a higher objective function value and sometimes the other. We also note that there seems to be a correlation between the number of requests and these differences. The larger the number of requests, the smaller the gains associated with adding flexibility.

3.6 Conclusions and future research opportunities

In this chapter, we introduced the time-dependent pickup and delivery problem with time windows, as well as three variants of the problem that model different levels of operational flexibility. We presented an exact branch-and-price-based algorithmic framework that can solve each of these variants. This framework incorporates a tailored labeling algorithm for solving the pricing problem, as well as speed-up techniques from the literature. The results of an extensive computational study indicate that the framework can solve to optimality instances with up to 45 pickup and delivery requests. We also assessed the impact of the speed-up techniques and the relative computational complexity of the four variants. Adding cuts to the framework only works out for the situation where all customers need to be served. For the more flexible variants, where it is allowed to skip some customers, the pure branch-and-price framework performs best. Surprisingly, these variants in which it is allowed to skip customers can be solved much faster than their respective counter parts in which all customers need to be served.

Regarding future research, we see multiple opportunities for adapting the framework we propose to solve related problems. For example, the framework could be adapted to solve problems wherein vehicles are heterogeneous. Similarly, it could be adapted to solve problems wherein not all vehicles are based at the same depot. We believe that both adaptations can be performed by changing the way the framework prices out new columns. Finally, we are exploring adapting the framework to a variant wherein travel times are both time dependent and stochastic. However, such an adaptation requires more algorithmic development.

Chapter 4

Adaptive large neighborhood search for the time-dependent profitable pickup and delivery problem with time windows

"The highest activity a human being
can attain is learning for
understanding, because to
understand is to be free"

Baruch Spinoza

4.1 Introduction

The advances in global digital technologies have accelerated the growth of the e-commerce market share in recent years. According to an Internet Retailer's analysis (Young [130]), e-commerce made up 14.3% of total retail sales in 2018 in the U.S. For customers, e-commerce provides an opportunity to purchase a wide range of products and services from local and/or foreign businesses. For logistics service providers e-commerce brings both opportunities and challenges. On the one hand, they face increased demands for their services, from which they can generate revenues. On the other hand, these demands involve narrow delivery time windows, which generally lower vehicle utilization and increase transportation costs.

One strategy a logistics provider can employ for meeting these increases in demands and expectations is to complement and coordinate its fleet operations with those of for-hire, third-party logistics providers. For instance, several e-commerce businesses such as Style Theory [85] and Vipshop.com ([128]) use a hybrid pickup and delivery model that consists of their own fleet and a partnership with third-party logistics providers such as Pickupp and SF Express. We study an optimization problem for coordinating those operations: the time-

dependent profitable pickup and delivery problem with time windows. In this problem, the logistics provider has the opportunity to use its fleet of capacitated vehicles to transport shipment requests, for a profit, from pickup to delivery locations. Owing to demographic and market trends, we focus on an urban setting, wherein road congestion is a factor. As a result, the problem explicitly recognizes that travel times may be time-dependent. The logistics provider seeks to maximize its profits from serving transportation requests, which we compute as the difference between the profits associated with transported requests and transportation costs.

In this study, we focus on the static version of the proposed problem, which assumes all orders are known before the operations start. This setting is used in many online shops. For example, VIP.com and Style Theory receive their order information before they construct delivery routes. Furthermore, often fixed scheduling deadlines are used in more instant delivey services. For example, in the Netherlands, the largest online retailer (Bol.com) enforces a timeline where orders received before 14:00 are delivered in the evening between 18:00 and 22:00. As a result, routes can be determined after 14:00 by which time all orders have arrived [14].

We propose an adaptive large neighborhood search (ALNS) approach for solving the routing problem of these companies. The proposed approach solves the time-dependent profitable pickup and delivery problem with time windows (TD-PPDP-TW). It decides which potential customers to serve and constructs cost-effective vehicle routes for doing so. This problem was addressed in the previous chapter as a variant of the time-dependent pickup and delivery problem with time windows, named as $\Omega(Flex, Prof)$.

ALNS consists of a group of removal and insertion operators, each of which modifies an incumbent solution to generate a number of neighboring solutions. To identify whether a neighboring solution is improving, ALNS evaluates its objective function value. In most applications of ALNS to routing problems, evaluating the objective function value of a solution is easy to do from a computational complexity perspective. However, for the problem we seek to solve, doing so involves solving an optimization problem, for which Vidal et al. [120] show that no efficient algorithm exists. Owing to the time-dependent travel times the triangle inequality is not guaranteed to hold anymore. In such cases, the travel time of an arc (i, j) is modeled as a piecewise linear function $\tau_{ij}(t)$, with departure time t at the tail node i of the arc (i, j). For a given route $h =< v_0, v_1, ...v_i, v_j, ..., v_m >$, the ready time function $\delta^v_{0,m}(t)$ is defined as the function that gives the ready time at node v_m, whereas the departure time

of v_0 is t. The ready time function can be derived by propagating the travel time functions $\tau_{ij}(t)$ of each arc (i, j) in route h. Then, the route duration can be calculated as $(\delta^v_{0,m}(t) - t)$. For instance, we can first calculate $\delta^v_{0,i}(t)$ and $\delta^v_{j,m}(t)$; then $\delta^v_{0,m}(t)$ can be derived by using the function composition $\delta^v_{j,m}(\delta^v_{0,i}(t) + \tau_{ij}(\delta^v_{0,i}(t)))$. However, extra memory is required to keep a binary tree-based data structure, which also needs to be re-built whenever a route is changed. In this chapter, we propose a method for approximating the objective function value of such a solution. Computational experiments indicate the method can provide quick and accurate estimates.

The contributions of this chapter are as follows:

- We adapt the classical ALNS heuristic to solve the time-dependent profitable pickup and delivery problem with time windows.
- To efficiently handle time-dependent travel times in ALNS, a simple and approximate route evaluation is employed, which performs well compared with the exact route evaluation.
- Compared with the existing literature, we show that large-sized instances can be solved within a relatively short CPU time with the proposed algorithm.
- The study results indicate that solutions obtained without considering time-dependent travel time will result in either sub-optimality or infeasibility.

The remainder of this chapter is organized as follows. Section 4.2 presents a brief review of the existing literature related to this study. Section 4.3 introduces a formal problem definition of the TD-PPDP-TW. In Section 4.4, we develop an ALNS to solve the proposed problem. Section 4.5 introduces the test instances. Section 4.6 first present the parameter tuning and relative performance of each operator used in ALNS. Then a series of experimental results and case studies are reported, followed by conclusions in Section 4.7.

4.2 Literature review

The problem addressed in this study is structurally similar to the PPDPTW, where serving all requests is not mandatory and a profit is associated with each request. To the best of our knowledge, only two similar problems have been studied by Li et al. [67] and Gansterer et al. [44]. Li et al. [67] presented the pickup and delivery problem with time windows, profits, and reserved requests (PDPTWPR). In this problem, the authors considered two types of requests:

reserved requests, which are required to be served, and selective requests, which may be rejected or outsourced to others. An ALNS approach is developed to solve this problem. Gansterer et al. [44] proposed and defined the multi-vehicle profitable pickup and delivery problem (MVPPDP), which does not consider the time window constraints. To solve this problem, the authors developed two variants of the general variable neighborhood search and tested them on self-created instances. At its core, the PPDPTW is also a generalization of the classical PDPTW, which models a variety of operational planning problems in transportation logistics and has been extensively studied in the literature. The interested readers are referred to Cordeau et al. [22] and Parragh et al. [82] for recent surveys.

Our problem involves an additional dimension of decision-making, as compared with the PPDPTW. In particular, because travel times are time-dependent, the departure time of a driver from the depot can also be optimized, whereas in the classical PPDPTW, the structure of the optimal solution is not affected by changes in the departure time. Excluding Sun et al. [110], no other study has considered time-dependent travel times, which can render their proposed solutions infeasible.

Another stream of research this work is rooted in is named the time-dependent vehicle routing problem with time windows (TDVRPTW). Compared with the PDPTW, considerably less research has been devoted to this area. Similar to our proposed problem, TDVRPTW assumes time-dependent travel times: a different travel time may be incurred if the departure time at a location is changes. Malandraki and Daskin [71] presented a mixed integer programming for this problem, and some simple constructive heuristics and a cutting plane method are employed. Later, Ichoua et al. [56] proposed a parallel tabu search. In their paper, they point out that previous models did not verify the "first in first out" (FIFO) property. This property states that if two identical vehicles traverse the same arc, the one that leaves first also should arrive first. Therefore, they proposed speed models, which are stepwise functions that satisfy the FIFO principle. An approximate evaluation procedure is used to reduce the complexity of the calculation. Then, the M best solutions according to this approximation are evaluated exactly, and the best solution is selected. Its objective is to find a set of minimum travel duration cost vehicle routes that serve every customer.

A similar objective function has also been used by Donati et al. [33] and Balseiro et al. [8]. Donati et al. [33] presented a multiple ant colony system (ACS) framework for the TDVRPTW. Then, Balseiro et al. [8] enhanced the ACS framework with an aggressive insertion heuristic relying on the minimum delay metric. Moreover, the evaluation of time feasibility can be

performed in O(1) time in Donati et al. [33] and Balseiro et al. [8], because the variables earliest departure and latest arrival are maintained for every customer in the solution. As to the exact approaches, Dabia et al. [25] developed a branch-and-price algorithm for TDVRPTW, where a tailored labeling algorithm is presented to solve the time-dependent shortest path problem with the resource constraint (TDSPPRC), which is the pricing problem of TDVRPTW. The proposed algorithm can solve instances up to 100 customers. The extension of the TDVRPTW to the case with a single vehicle with infinite capacity is known as the time-dependent traveling salesman problem TDTSP (with time windows TDTSPTW), which also received some attention in the past decade. Cordeau et al. [20] proposed a branch-and-cut algorithm to solve TDTSP. In this study, the travel speed functions are defined by employing the degradation of the congestion factors and the proposed algorithm can solve instances with up to 40 nodes. Most recently, Montero et al. [77] presented an integer linear programming model for the TDTSPTW and developed an exact algorithm. This approach is also able to solve instances with up to 40 customers. Vu et al. [122] proposed a dynamic discretization discovery framework for the TDTSPTW. Two objectives are considered in this problem. One seeks to make the driver return to the depot as early as possible, the other aims to minimize the travel duration. The interested readers are referred to Gendreau et al. [46] for recent surveys on TDVRPTW and its variants. Compared with our problem, none of these studies considered the precedence constraints for pickup and delivery services and possibility to serve a subset of requests. It is also difficult to directly apply methods from the time-dependent VRP to our problem given that it allows for customers to not be served.

Fu [42] has developed frameworks that explicitly incorporate a time-dependent, stochastic travel model for dial-a-ride applications subject to tight service time constraints, which is a variant of pickup and delivery service in passenger transportation. Later on, Schilde et al. [97] adapted four existing metaheuristics for the stochastic time-dependent dynamic dial-a-ride problem using statistical information available on historical accidents. The time-dependent travel time is included by modeling congestion as gradually expanding and shrinking circles. In Chapter 2, we introduced the time-dependent capacitated profitable tour problem with time windows and precedence constraints, which considers only a single vehicle. A tailored labeling algorithm is proposed to solve instances with up to 75 requests to optimality. Moreover, a dynamic programming heuristic is designed for difficult instances that cannot be solved by the exact method. The TD-PPDP-TW, was studied in Chapter 3 as one variant of the time-dependent pickup and delivery problem with time windows. It extended the PDPTW by considering time-dependent travel times and there is a possibility to skip requests that are not profitable. The authors provided a mix integer linear programming formulation for the

family of time-dependent pickup and delivery problems with time windows. A branch-cut-and-price framework is proposed to solve these problems. However, this method is able to solve instances up to 45 requests to optimality within a predefined time limit.

4.3 Problem description

Following the notation of Sun et al. [110], we define an undirected graph $G = (N, A)$, whose node set N consists of pickup nodes $N_P = \{1, 2, ..., n\}$, delivery nodes $N_D = \{n + 1, ..., 2n\}$, and depots $\{0, 2n + 1\}$. Each request is specified by its pickup node i and delivery node $i + n$. Moreover, all feasible routes need to start from depot 0 and to end at $2n + 1$ (which can be the same node as 0 in reality). For each pickup node $i \in N_P$, a nonnegative profit p_i and load q_i is assigned. It must hold that $q_i = -q_{n+i}$. Without loss of generality, $q_0 = q_{2n+1} = 0$. A time window $[e_i, l_i]$ is associated with every vertex $i \in N_P \cup N_D$, where e_i and l_i represent the earliest and latest times, respectively, at which service may start at node i. The vehicle waits until time e_i, if arriving at i before e_i; arriving later than l_i is not allowed. The service time of each node $i \in N_P \cup N_D$ is denoted by s_i. The depot nodes also have time windows $[e_0, l_0]$, $[e_{2n+1}, l_{2n+1}]$ representing the earliest and latest times, respectively, at which the vehicle may leave from and return to the depot. Without loss of generality, we assume that $s_0 = s_{2n+1} = 0$. Let K denote the set of vehicles to serve those requests. We assume that vehicles are identical and have capacity Q. Furthermore, a fixed operational cost, Z, is assigned per used vehicle.

In classical VRPs, the travel time between two locations is presumed to be known and constant. In particular, it is generally assumed to be a scalar transformation of distance. In the time-dependent VRP, the travel time between two nodes is a function of both the distance traveled and departure time, with speed modeled as a stepwise function of the departure time. This can lead to objectives that optimize the duration of a tour, in time, rather than its distance. Formally, we let the piecewise linear function $\tau_{ij}(t_i^k)$ denote the travel time on arc $(i, j) \in A$, as a function of departure time t_i^k of vehicle k at node i. As explained in Ichoua et al. [56], this function is based on a stepwise speed function, with fixed speeds per time zone. In this way, the FIFO property holds, i.e., vehicles that depart earlier from the departure node of an arc will not arrive later at the destination node of that arc.

The set of feasible arcs can be described as $A = \{(i, j) \in N \times N : i \neq j \text{ and } e_i + s_i + \tau_{ij}(e_i + s_i) \leq l_j\}$. Let $y_i^k, i \in N$ be a binary variable set to 1 if node i is visited by vehicle k, and 0 otherwise. The cost per unit of route duration is denoted as c_t.

The objective function includes three parts: (i) the profit obtained from served requests; (ii) cost related to travel duration; and (iii) total setup cost for the number of used vehicles. Therefore, the objective of the TD-PPDP-TW can be described as follows:

$$\max \sum_{k \in \mathcal{K}} [\sum_{i \in \mathcal{N}_\mathcal{P}} p_i y_i^k - c_t(t_{2n+1}^k - t_0^k) - Z y_0^k] \tag{4.1}$$

while satisfying the following constraints:

- The route of vehicle k starts from the origin depot and ends at the destination depot, if vehicle k is used.
- Every request is served at most once and its pickup and delivery nodes are visited by the same vehicle.
- For each request, its pickup node is required to be visited before the delivery node.
- The departure time at each node of the request should be within the given time window (if the request is served).
- Capacity constraints of vehicles.

The complete mathematical formulation of the constraints can be found in Apprendix A.

The TD-PPDP-TW is similar to the problems studied in Wang [125] and Cortés-Murcia et al. [23], which are also extensions of the VRP and which belong to the class of NP-hard problems. It means that an increase in the size of problem leads to exponential growth in the computational effort required to find the corresponding solution. To solve large instances in a reasonable times, we propose an efficient metaheuristics to handle the proposed problem in Section 4.4.

4.4 An adaptive large neighborhood search for the TD-PPDP-TW

In this section, we presents an ALNS heuristic for the TD-PPDP-TW. It is an extension of the LNS heuristic proposed by Shaw [99]. The ALNS metaheuristic was first developed by Røpke and Pisinger [93] and Pisinger and Røpke [86]. Unlike LNS, where only one removal and one insertion heuristic are used, the ALNS heuristic applies multiple removal and insertion

operators. In each iteration, a roulette wheel mechanism is used to choose one removal operator to partially deconstruct the current solution and one insertion operator to repair it in a different way. At the end of each iteration, a neighborhood of the current solution is obtained. Moreover, a weight is associated to each operator. The selection probability of an operator is related to its weight, which is adjusted dynamically and based on the historic and current performance of all operators. Furthermore, a simulated annealing algorithm (SA) is used to determine the acceptance of the incumbent solution.

In this study, we implement a similar framework as in Pisinger and Røpke [86], Røpke and Pisinger [93], and Demir et al. [29]. Moreover, to cope up with the peculiarities of our problem, we first introduce the route feasibility check and route evaluation during the removal or insertion in Section 4.4.1. The general structure of our ALNS algorithm with simulated annealing is provided in Algorithm 3. Its components are detailed in the following subsections. In Section 4.4.2, we describe how to construct the initial solution. The weight and score adjustments are introduced in Section 4.4.3. The various removal and insertion operators used in our ALNS are described in Section 4.4.4 and Section 4.4.5, respectively. The SA scheme is described in Section 4.4.6.

4.4.1 Route feasibility check and route evaluation

Due to the their dependency on departure time, travel times need to be evaluated through the end of a route in order to check the feasibility or to accurately measure the impact of a removal or insertion operator. Moreover, the triangle inequality property does not necessarily hold anymore, which for road networks means that, due to traffic jams on the direct road between vertices A and C, it might be faster to drive through vertex B. Therefore, the shortest path (in terms of duration) between vertices A and C varies over time and removing a request might cause an infeasibility (Figure 4.1) or increase the travel time of the solution (Figure 4.2(a)). Inserting a request might shorten the travel time of the solution (Figure 4.2(b)), where it normally, when the triangular inequality holds, always increases the travel time. For our solution without pre-calculating the shortest travel time between vertices A and C by given each possible departure time at vertex A, we prevent a full and time-consuming feasibility check and evaluation of the solution after every removal and insertion attempt, by storing for each node of a given route, the earliest departure and latest arrival times. This reduces the computational complexity of the feasibility check and evaluation of a solution from $O(n)$ time to $O(1)$ time, which enables an efficient checking and updating mechanism.

Algorithm 3: Pseudo-code of the ALNS with SA

input : Removal operators D, insertion operators Ψ, initial temperature T, cooling rate κ
output: A feasible solution X_{best}

Generate an initial solution by using the Greedy insertion algorithm (Section 4.4.2)
Initialize probability for each destroy operator $d \in D$ and each insertion operator $\psi \in \Psi$ (Section 4.4.3)
Let j be the iteration counter with initial value of $j \leftarrow 1$
$X_{current} \leftarrow X_{best} \leftarrow X_{init}$
repeat
  Choose a removal operator $d^* \in D$ with probability $\omega_d / \sum_{i=1}^{|D|} \omega_i$ (Section 4.4.3)
  Let X_{new}^{**} be the solution generated after using operator d^* to $X_{current}$ (Section 4.4.4)
  Let X_{new}^{*} be the solution generated after the removal feasibility check of solution X_{new}^{**} (Section 4.4.1)
  Choose an insertion operator $\psi^* \in \Psi$ with probability $\omega_\psi / \sum_{i=1}^{|\Psi|} \omega_i$ (Section 4.4.3)
  Let X_{new} be the newly-generated solution after applying operator ψ^* to X_{new}^{*} (Section 4.4.5)
  if *obj(X_{new}) > obj($X_{current}$) (Section 4.4.6)* **then**
    $X_{current} \leftarrow X_{new}$
    if *obj($X_{current}$) > obj(X_{best})* **then**
      $X_{best} \leftarrow X_{current}$
  else
    Let $\nu \leftarrow e^{-(obj(X_{new}) - obj(X_{current}))/T}$
    Generate a random number $\epsilon \in [0, 1]$
    if $\epsilon < \nu$ **then**
      $X_{current} \leftarrow X_{new}$
  $T \leftarrow \kappa T$
  Update probabilities using the adaptive probability adjustment procedure
  $j \leftarrow j + 1$
until *the maximum number of iterations is reached or no improvement is found to X_{best} after a predefined number of iterations*

For a route $h =< v_0, v_1, v_2, ..., v_m >$, the earliest departure ed_{v_i} records the earliest time the vehicle can depart from v_i after serving all the previous nodes in the route including v_i. If $A_{ij}(t)$ is the arrival time to node j when departing from i at time t, and given that the departure time at the depot equals 0 these values can be computed as follows:

$$ed_{v_0} = 0 \tag{4.2}$$

$$ed_{v_i} = max(A_{i-1,i}(ed_{v_{i-1}}) + s_{v_i}, e_{v_i} + s_{v_i}) \qquad \forall i = 1, ..., m \tag{4.3}$$

Analogously, the latest arrival la_{v_i} is the latest time the vehicle can arrive at v_i, so that it can service all the subsequent nodes in the route including v_i without violating the time window constraints and capacity constraints. It is computed as

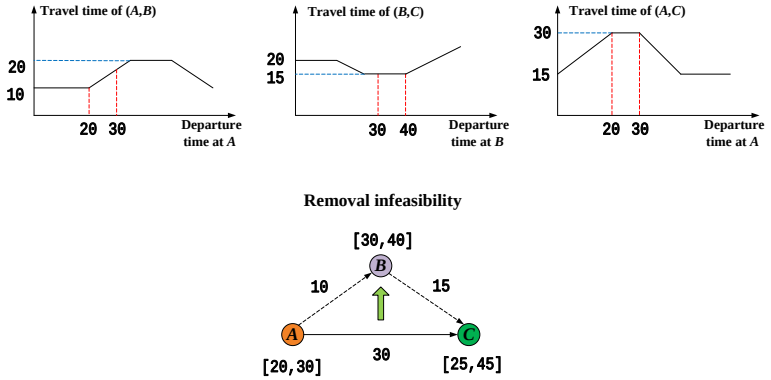

Figure 4.1: Examples of removal infeasibility

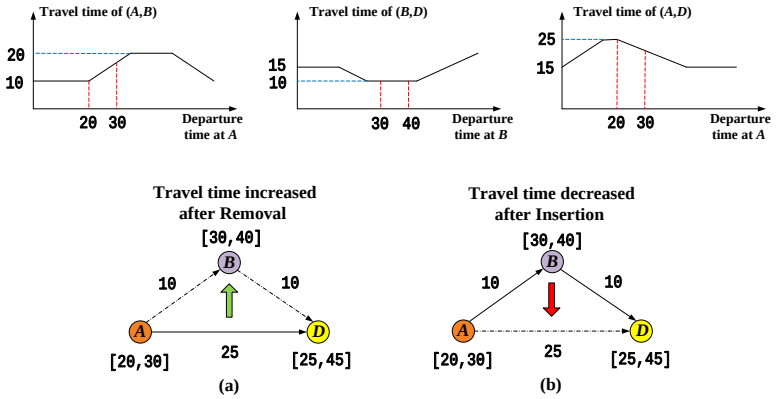

Figure 4.2: Examples of time-dependent removal and insertion

$$la_{v_m} = l_{v_m} \tag{4.4}$$

$$la_{v_i} = min(A^{-1}_{i,i+1}(la_{v_{i+1}}) - s_{v_i}, l_{v_i}) \qquad \forall i = 0, ..., m-1 \tag{4.5}$$

where $A^{-1}_{i,j}(t)$ is the departure time from i to reach j by time t.

According to Visser and Spliet [121], each node v_i has an arrival time function, which is nondecreasing piecewise linear function with a group of breaking points. Therefore, whenever a request is removed from or inserted into route h, the optimal departure time at the origin depot, 0, needs to be recalculated, because the arrival time functions for all nodes

visited after the removal or insertion position needs to be updated. Therefore, a new arrival time function of end depot $2n + 1$ is derived by propagating the arrival time function to the end of the route. This propagation needs to consider all the possible break points in the travel and arrival time functions, which is computationally expensive. For instance, when node i is inserted into a partial route $< 0, j >$ and a new partial route is generated as $< 0, i, j >$. Let $[3, 8]$ and $[7, 9]$ be the time windows of nodes i and j, respectively. We assume that the service time of both nodes i and j equals 1 ($s_i = s_j = 1$). In Figures 4.3 and Figure 4.4, we show how to derive the ready time function at node j. We first derive the arrival time function at node i (Figure 4.3(b)) by propagating the given travel time function of arc $(0, i)$ such that all the possible break points in Figure 4.3(a) need to be checked. Then the ready (Figure 4.3(c)) and departure time functions (Figure 4.3(d)) of node i can be easily derived by checking its time window and service time, respectively. Furthermore, we consider all the break points in both the departure time function of node i (Figure 4.3(d)) and the travel time function of arc (i, j) (Figure 4.4(a)) to derive the arrival time function of node j (Figure 4.4(b)). After checking the time window and service time of node j, its ready (Figure 4.4(c)) and departure time functions (Figure 4.4(d)) can also be easily derived.

Next, we set the latest departure time to not cause an infeasibility at depot 0 equal to 5 as the estimate for the optimal departure time. It could be an efficient approximation for route evaluation but cannot guarantee optimality. For instance, we first obtain the latest possible departure time of 7 in the travel time function of arc (i, j) (Figure 4.4(a)) because $l_j = 8$. The latest ready time at node i is 7, by deducting its service time of 1. Thus, the latest departure time at node 0 can be calculated to be 5 by checking the travel time function of arc (i, j) (Figure 4.3(a)). In conclusion, using two methods, we derive the optimal departure time at node 0 to be 5 with minimum travel duration value 5. Of course, the earliest departure, latest arrival, and real departure times of each node on route h need not be fully recomputed every time. Because information is not changed before inserting or deleting a position, recalculation is required only after that position.

In some cases, the approximate method might have a result different from that of the exact method. Generally, the performance of the approximate method mainly depends on the travel time function of each arc and number of visited nodes in the route. In the worst case scenario, the approximate method is worse in determining the route duration than the exact method with factor χ, which is equal to the largest ratio amongst all visited arcs of the longest travel time and the shortest travel time in the travel time function (which is also equal to the fastest speed in the speed profile divided by the slowest speed in the speed profile). This can be

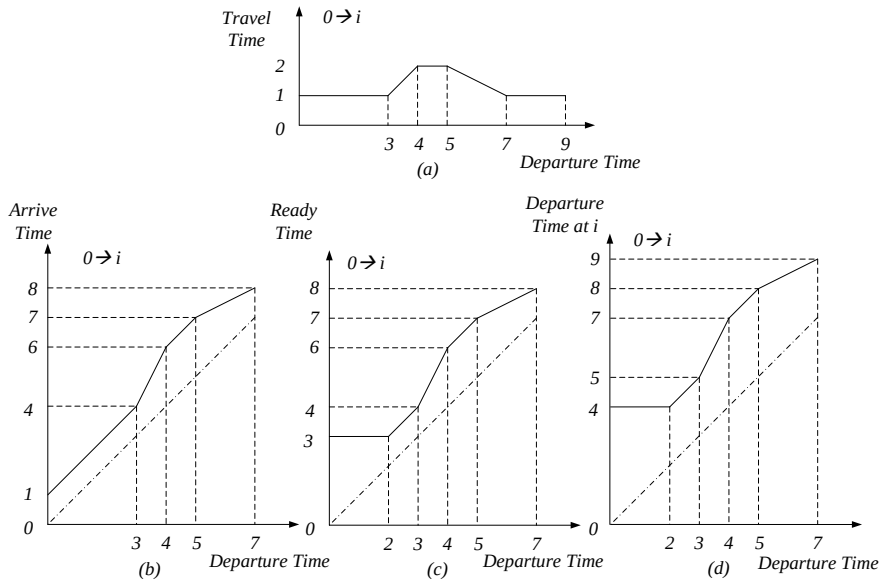

Figure 4.3: Examples of travel time functions propagating with route $0 \rightarrow i$

reasoned from the fact that, in the worst case, departure time is selected where all arcs are traversed at times within a minimum speed time zone. However, in the optimal departure time, all arcs might be traversed during the maximum speed time zones.

For instance, in Figure 4.5(a), one can see that the longest and shortest travel times have values 3 and 1, respectively. Therefore, factor χ is equal to 3. In the ready time function (see Figure 4.5(c)), it shows that the optimal departure time is 2, which leads to a travel duration of 1. When using the latest departure time of 5, the travel duration equals 3, which is three times worse than the travel duration given by optimal departure time 2. Moreover, it is likely that the side effect of the approximate method will be leveraged out once more nodes are visited in the route. For instance, Figure 4.6 shows that the travel duration equals 5.5 by using the latest departure time 3.5. It is only 0.5 worse than the optimal travel duration of 5.

4.4.2 Initial solution construction

We developed a modified version of the sequential insertion heuristic (SIH), first introduced by Solomon [104], to generate an initial feasible solution. All requests are initially stored in a list $\mathcal{L}$. The SIH begins by creating one route for each individual request $R_i \in \mathcal{L}$. These routes are sorted in the decreasing order of their objective values. Then, to diversify the search space, the method randomly chooses K routes as the "seed" routes for further insertion in

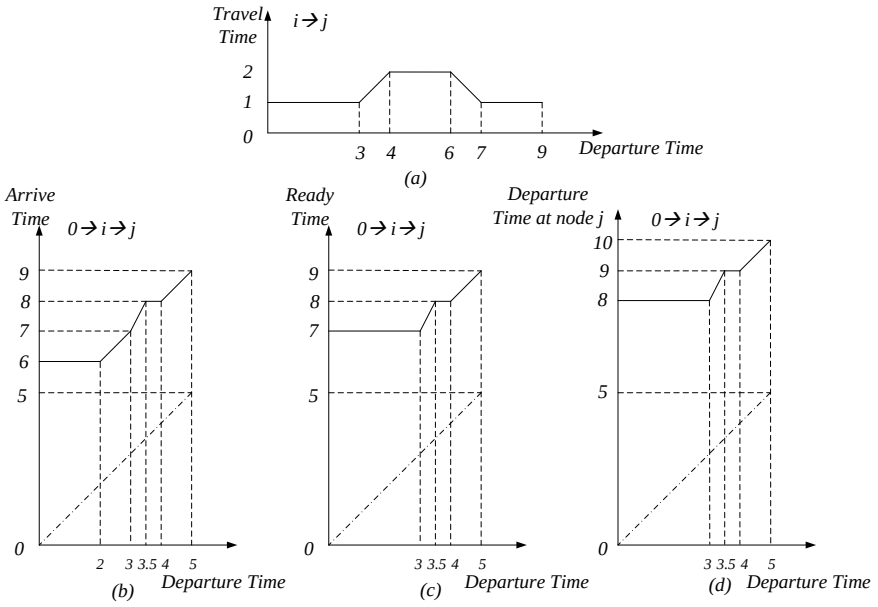

Figure 4.4: Examples of travel time functions propagating with route $0 \rightarrow i \rightarrow j$

each run. The related requests are removed from $\mathcal{L}$. In each subsequent step, SIH tries to add one of the currently unserved requests in $\mathcal{L}$ into one of the existing routes, and its pickup and delivery nodes are inserted at their most profitable positions (for more details, see Section 4.4.5). However, those requests that cannot be feasibly inserted in any of the routes or those where the insertion deteriorates the objective value of the solution remain stored in list $\mathcal{L}$. Moreover, whenever a request $R_i = (i, i+n)$ is added to route h, the earliest departure time of each node after the insertion of node i, latest arrival time of each node before the insertion of node $i+n$, and real departure time of each node in the route h need to be updated (see Section 4.4.1). This process is repeated until no request can be inserted to improve the objective value of the solution.

4.4.3 Adaptive weight and score adjustment

To select the removal and insertion operators more effectively, as in the implementation of Ropke and Pisinger [93], a roulette-wheel mechanism is used in our algorithm. Let D and Ψ denote the sets of removal and insertion operators, respectively, and let $d \in D$ be the removal operator and $\psi \in \Psi$ be the insertion operator. At the beginning, the weight of each removal ω_d and insertion operator ω_ψ is set equal to 1. In each iteration, the probability of choosing a removal or insertion operator is calculated as $\omega_d / \sum_{i=1}^{|D|} \omega_i$ and $\omega_\psi / \sum_{i=1}^{|\Psi|} \omega_i$, respectively. Let

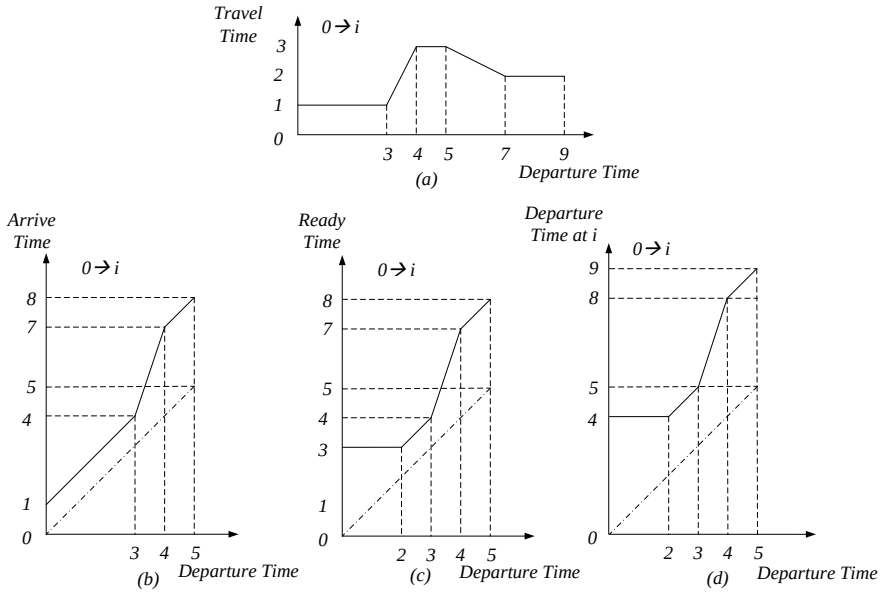

Figure 4.5: Example showing approximate method is worse than exact method with route $0 \rightarrow i$

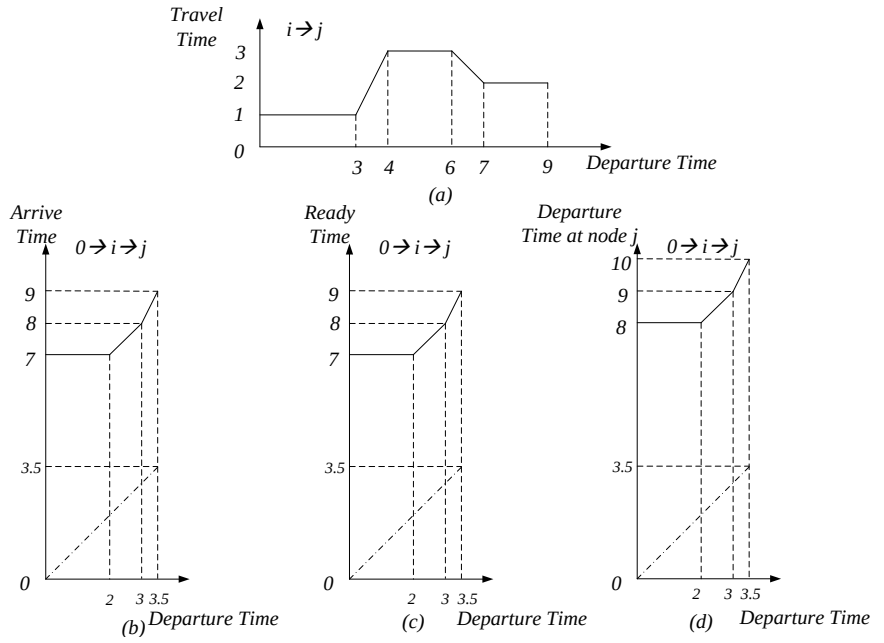

Figure 4.6: Example showing approximate method is worse than exact method with route $0 \rightarrow i \rightarrow j$

sg denote a segment that includes a set of 100 iterations. During the iterations of a segment, the weights of all removal and insertion operators remain fixed; however, they are updated at the end of each segment as follows:

$$\omega_d^{sg+1} = \omega_d^{sg}(1-\xi) + \xi\frac{\pi_d}{\beta_d} \tag{4.6}$$

$$\omega_\psi^{sg+1} = \omega_\psi^{sg}(1-\xi) + \xi\frac{\pi_\psi}{\beta_\psi} \tag{4.7}$$

where ξ is the adjustment parameter; π_d and π_ψ are the resulting scores of operators d and ψ; and β_d and β_ψ represent the number of times the operator has been called in segment sg. To let operators that are seldom selected in the previous segment still have a chance to be called in the current segment, π_d and π_ψ are set to 0 for all $d \in D$ and $\psi \in \Psi$ at the beginning of each segment. Moreover, if a new solution is accepted at the end of each iteration, the scores π_d and π_ψ of the chosen removal and insertion operators d and ψ are updated by adding either σ_1, σ_2, or σ_3 according to the three different situations in Table 4.1.

Table 4.1: Adaptive adjustment of operator scores

Increment	Conditions of the solution obtained by the chosen operators
σ_1	A new best solution is obtained
σ_2	A newly generated solution improves the current solution
σ_3	A solution worse than the current solution is accepted

4.4.4 Removal stage

Seven removal operators are used in our ALNS heuristic algorithm. All operators are adapted from or inspired by Shaw [100] or Røpke and Pisinger [93]. The removal stage mainly consists of removing ϕ requests from the current solution and adding them to the so-called removal list $\mathcal{L}$, where ϕ is a random integer number between 1 and $\rho * n$ (ρ is the destroy rate and n is the number of served requests in the solution). The general structure of the removal operators is provided in Algorithm 4.

The removal operators used in our implementation are introduced below.

- **Random Removal (RR)**: This operator randomly removes ϕ requests from the current solution. The idea of randomly selecting requests helps to diversify the search space.

Algorithm 4: Overall structure of removal operators

input : A feasible solution X and the number of requests to be removed ϕ
output: A partially destroyed solution X_p

```
Initialize removal list (L ← ∅)
for φ iterations do
    Apply removal operator to remove a request r (includes two nodes; pickup and delivery)
    L ← L ∪ r
```

- **Worst Travel Duration Removal (WTR)**: This operator repeatedly removes ϕ requests. The cost is considered to be the total travel duration between the pickup and delivery nodes of a request with their prior and succeeding nodes within the given routes, i.e., it removes request $(i, i+n)^* = argmax_{h \in \mathcal{N}_\mathcal{P}}\{(t_{i+1} - t_{i-1}) + (t_{j+1} - t_{j-1})\}$, where $i-1$ and $j-1$ are predecessors and $i+1$ and $j+1$ are successors of the pickup and delivery nodes, respectively. Note that a large value of $(t_{i+1} - t_{i-1}) + (t_{j+1} - t_{j-1})$ could also be caused by significant waiting times at $i+1$ or $j+1$; therefore, this difference does not automatically represent the expected savings in duration.

- **Early Departure Removal (EDR)**: For each request $(i, i+n)$, this operator calculates the deviation of the departure time t_i and t_{i+n} from the latest possible arrival time, and then removes the request with the largest deviation (i.e., $(i, i+n)^* = argmax_{i \in N_p}\{|[la_i - t_i] + [la_{i+n} - t_{i+n}]|\}$, where la_x is the latest possible arrival time at node x, so that it can service all the subsequent nodes in the route including x (for more details on la_x-see Section 4.4.1). The idea is to prevent long waits or delayed service start times.

- **Least profit removal (LPR)**: Considering that profit is an essential part of the objective function value of a solution, this operator aims to remove the ϕ requests with the least assigned profits.

- **Worst Removal (WR)**: This operator removes the ϕ requests one at a time by determining which removal has the largest positive effect γ on the objective function value. Note that, all removals could have a negative effect on the objective. In that case the removal with the smallest negative effect is removed. The improvement or deterioration γ in this case is computed as follows: given a solution, the γ_i of a request $(i, i+n)$ is the difference in the objective function between the current solution (with $(i, i+n)$) and the same solution without serving $(i, i+n)$.

- **Route Removal (ROR)**: This operator removes a full route from the solution. It randomly selects a route from the set of routes in the solution. It removes all nodes from the selected route except 0 and $2n+1$.

- **Shaw Removal (SR)**: The objective of the SR operator is to remove requests that are similar in terms of certain aspects. The algorithm randomly selects a request r_1 and adds it to $\mathcal{L}$. Let $l_{r_1,r_2} = -1$ if two nodes, one related to r_1 (i.e., i or $i+n$) and the other to r_2 (i.e., j or $j+n$), are in the same vehicle route, and $l_{r_1,r_2} = 1$ otherwise. This operator selects the request $r^* = argmin_{j \in \mathcal{N}_\mathcal{P}} \{\Pi_1[d_{i,j} + d_{i+n,j+n}] + \Pi_2[|t_i - t_j| + |t_{i+n} - t_{j+n}|] + \Pi_3 l_{r_1,r_2} + \Pi_4|q_i - q_j| + \Pi_5|p_i - p_j|\}$, where Π_1–Π_5 are normalized weights. The operator is applied ϕ times by selecting a request that is not yet added in $\mathcal{L}$ and is the most similar to the last added request.

As mentioned in Section 4.4.1, the removal operator might make the partial solution infeasible. Therefore, Algorithm 5 is used to check the feasibility of each route in the partial solution. Moreover, if the partial solution is feasible, Algorithm 5 returns the earliest departure time of each node in the partial solution. Otherwise, more requests are removed from the current solution to make the solution feasible. Particularly, when route h leads to node i that has a time window violation, node i and each node $j \in N_P \cup N_D$ after node i will be aborted, which lead to a new route h^*. Moreover, if route h^* contains incomplete served requests, which means that only the pickup node is visited in route h^*, those pickup nodes are also removed from route h^*.

Algorithm 5: Generic structure of route feasibility check and its earliest departure time calculation

input : A route h, earliest departure time $ed_{v_0} = 0$ at depot 0
output: Boolean value B: $TRUE$ if feasible, $FALSE$ otherwise

```
B ← TRUE
for each node v_i and v_{i+1} in route h do
    if B then
        if (Q_{v_i} + q_{i+1} ≤ Q) then
            if (A_{v_i,v_{i+1}}(ed_{v_i}) < l_i) then
                ed_{v_{i+1}} ← max{A_{v_i,v_{i+1}}(ed_{v_i}) + s_{i+1}, e_{i+1} + s_{i+1}}
            else
                B ← FALSE
        else
            B ← FALSE
    else
        Break
Return B
```

4.4.5 *Insertion stage*

We used seven insertion operators in our ALNS algorithm. Similar to removal operators, all operators are adapted from or inspired by Shaw [100] and Røpke and Pisinger [93]. With the help of those insertion operators, the unserved requests in $\mathcal{L}$ can be inserted into the existing routes if the insertion is feasible and improves the objective function value of the solution. Moreover, to reduce the size of the neighborhood considered at every iteration of the insertion stage, we adapt the evaluation scheme introduced by Cordeau and Laporte [21] to evaluate the impact of inserting request R_i in one route. First, the best insertion position is determined for the pickup node. Because adding a pickup node increases the profit and changes the travel duration of a route, the difference in the objective function value is evaluated. Next, holding the pickup node in its best position, the best insertion position is determined for the delivery node. Because adding a delivery node only changes the travel duration of a route, the insertion position of the delivery node that leads to minimal travel duration is selected. This rule has a striking effect on computing times because it reduces the maximum number of possible insertions involving request R_i from $O(\theta^2)$ to $O(\theta)$, where θ is the number of nodes in route h.

- **Sequential Insertion (SI)**: This operator is based on the position of each unserved request in the removal list $\mathcal{L}$. It sequentially removes the unserved requests one by one from the removal list $\mathcal{L}$ and inserts them into the best position of the existing routes. Therefore, this operator first tries to insert the requests that cannot be served in the solution in the previous iteration, and then it aims to insert the requests removed by the removal operation in the current iteration.
- **Greedy Insertion (GI)**: This operator repeatedly inserts a request (both pickup and delivery nodes) in the best possible position of the routes. In each iteration, the unserved request that best improves the objective function value is inserted at its best insertion position. Let $\Delta(i, h, X)$ denote the difference in objective function value if request R_i is inserted at its best position into route h of solution X, which is evaluated by the procedure of Section 4.4.1. Now assume that $\Delta(i, 1, X)$ is associated with the route where R_i can be inserted best. Then the request that will be inserted is $R_i^* = argmax_{i \in N_p}\{\Delta(i, 1, X)\}$. Subsequently, the next iteration starts, and $\Delta(i, h, X)$ value are recomputed for all the remaining unserved requests.
- **Duration Greedy Insertion (DGI)**: The DGI operator works in a similar way as the GI operator, except that only the insertion cost, which is measured as the travel duration, is considered. Let $f(X)$ be the total travel duration cost of solution X with request R_i

served, and $f(X^{'})$ represent the total traveling duration cost of solution $X^{'}$ without request R_i. Therefore, $f(X) - f(X^{'})$ is denoted as the difference between travel duration costs with and without request R_i. Similar to the GI operator, this operator also accepts the request that might deteriorate the objective function value of the incumbent solution as long as the insertion is feasible. Compared with the GI operator that considers the difference of the whole objective function, DGI only considers the change in the travel duration.

- **Regret Insertion (RI)**: The RI operator first calculates the best insertion $\Delta(i, 1, X)$ and second best insertion $\Delta(i, 2, X)$ for each unserved request R_i. Next, a regret value for each request R_i is measured as $\Delta(i, 1, X) - \Delta(i, 2, X)$, which is the difference between the best and second best insertions. The request with the maximum regret value is then removed from $\mathcal{L}$ and inserted first at the position where it generates the highest objective value. Note that regrets of each request need to be recalculated after each insertion because some insertion positions are no longer available.
- **Sequential Insertion with Noise function (SIN)**: This operator is similar to the Sequential Insertion but uses a degree of freedom in selecting the best place for a node. This degree of freedom is achieved by modifying the profit of request R_i: $Obj_{new} = Obj_{actual} + p_i\mu\epsilon$, where p_i is the profit of request R_i, μ is a noise parameter used for diversification and ϵ is a random number between $[-1, 1]$. Obj_{new} is calculated for each request in $\mathcal{L}$.
- **Greedy Insertion with Noise function (GIN)**: This operator is an extension of the GI operator and considers the same noise function as the SIN operator.
- **Regret Insertion with Noise function (RIN)**: This operator is similar to the SI operator and uses the same noise function as the SIN and GIN operators.

4.4.6 Acceptance and stopping criteria

To avoid the algorithm becoming trapped in a local optimum, SA is embedded in our ALNS framework. In the algorithm, X_{best} indicates the best solution found during the search, $X_{current}$ denotes the current solution obtained at the beginning of an iteration, and X_{new} represents a temporary solution found at the end of the iteration that can be discarded or become the current solution. The objective function value of solution X is denoted as $obj(X)$. In addition, a tabu list is used for $X_{current}$ to prevent cycling in the proposed algorithm. This list prohibits the use of a specific set of solutions for several iterations if one of them is used in recent iterations. More specifically, the new generated solution X_{new} is retained

in the tabu list at the end of each iteration. Then, X_{new} is forbidden to be accepted as $X_{current}$ in the following iterations. Therefore, the search does not travel around the local optimum solution space. A solution X_{new} is accepted if X_{new} is a non-tabu solution and $obj(X_{new}) > obj(X_{current})$, and accepted with probability $e^{(obj(X_{new})-obj(X_{current}))/T}$ if $obj(X_{new}) < obj(X_{current})$, where T is the temperature. The initial temperature is set at T_{init}, where T_{init} is an initialization constant. The current temperature gradually decreases during the algorithm as κT, where $0 < \kappa < 1$ is a fixed parameter and generally close to 1 to achieve slow cooling. The main idea of the SA is to accept worse solutions at the beginning to diversify the search space, whereas the algorithm only accepts the improved solutions after several iterations when the temperature is decreased. Our stopping criterion is 10,000 iterations in total or 3,000 consecutive iterations without further improvement.

4.5 Test instances

Two classes of instances are used in this study. Both are adapted from the instances proposed by Røpke and Cordeau [91]. Each class has four groups of instances (AA, BB, CC, and DD) with different vehicle capacities and time windows. The single-vehicle instances are the instances for the time-dependent capacitated profitable tour problem with time windows and precedence constraints introduced by Sun et al. [109]. The multi-vehicle instances are based on the instances for the time-dependent pickup and delivery problem with time windows proposed by Sun et al. [110]. Because the data set in Sun et al. [110] only considers the instances with up to 45 requests, we enrich this data set by creating six new instances for each group having the number of requests ranging from 50 to 75. Moreover, to differentiate these two classes of instances, each instance name has a format V_tp_n, where tp is the group type and n is the number of considered request; if $V = S$, it belongs to the class of single-vehicle instances, otherwise, when $V = M$, it represents the class of multi-vehicle instances. The extra instances of this class are obtained by using the same mechanism as introduced in Sun et al. [110]. Particularly, these two classes of instances are based on the same node coordinates, vehicle capacity, time windows, and load quantities as the original instances in Røpke and Cordeau [91]. In addition, for each single-vehicle instance, a profit of 40 units is assigned to each request. For each multi-vehicle instance, a randomly generated profit in the interval $[50, 150]$ interval is assigned to each request.

To model the time-dependent travel times, as in Sun et al. [109], road congestion is handled by a *speed model* consisting of different speed profiles. It is used to determine the travel time between two nodes at different departure times. In our study, five types of speed profiles

are considered (see Table 4.2): slow speed (SS), normal speed with morning peak (NSMP), normal speed with evening peak (NSEP), fast speed with two peaks (FSTP) and high speed (HS). To generate an instance, we randomly assign one of these speed profiles to each arc. We refer the reader to Sun et al. [109] for detailed information regarding the explanation of the different speed profiles. Furthermore, without loss of generality, it is assumed that breakpoints are the same for all speed profiles because congestion tends to occur around the same time regardless of the speed profile type. Each speed profile has four non-overlapping time periods with constant speed, reflecting two congested periods and two periods with normal traffic conditions.

Table 4.2: Speed Profiles

Congestion description Time periods	Morning peak 7 am-9 am	Normal 9 am-5 pm	Evening peak 5 pm-7 pm	Normal 7 pm-9 pm
1. SS	0.5	0.81	0.5	0.81
2. NSMP	0.67	1.33	0.88	1.33
3. NSEP	0.88	1.33	0.67	1.33
4. FSTP	0.85	1.5	0.85	1.5
5. HS	1.0	2.0	1.0	2.0

4.6 Performance of proposed ALNS

The algorithm described in this chapter is implemented in Java, using Eclipse SDK version 4.2.0. We performed experiments on a single thread of a server with four CPU's (2.4 GHz/6 cores) and 64GB RAM. The following sections first present the parameters of the proposed algorithm, followed by a series of experiments to determine the best set of parameters, and then an assessment of the performance of the proposed algorithm.

4.6.1 Parameter tuning

Table 4.3 lists 15 parameters. As in Demir et al. [29] and Ghilas et al. [48], we group them into three categories. The parameters to control the simulated annealing algorithm in Group I. Group II includes the parameters that are related to the roulette wheel mechanism that decides which operators to use in each iteration. Group III parameters are used in the removal and insertion operators.

To identify a value for these parameters to enable the ALNS to find high-quality solutions,

Table 4.3: Adaptive adjustment of operator scores

Group	Notation	Description	Tuned Value
(I)	Υ	The total number of iterations	10000
	T	Initial temperature	200
	κ	Cooling rate	0.9995
(II)	Υ_{sg}	The number of iterations per segment	100
	ξ	Roulette-wheel parameter	0.1
	σ_1	A new best solution is obtained	5
	σ_2	A newly generated solution improves the current solution	3
	σ_3	A solution worse than the current solution is accepted	1
(III)	ρ	Destroy rate for removal operators	0.45
	Π_1	First Shaw parameter	0.2
	Π_2	Second Shaw parameter	0.25
	Π_3	Third Shaw parameter	0.1
	Π_4	Fourth Shaw parameter	0.2
	Π_5	Fifth Shaw parameter	0.25
	μ	Noise parameter	0.1

some parameter-tuning experiments are performed. All tests are performed on a tuning set consisting of the following 23 instances: 14 instances from the single-vehicle instance class, which belong to the BB group with the number of requests ranging from 10 to 75 (S_BB10-S_BB75), and nine instances from the multiple-vehicle instance class, which also belong to the BB group with the number of requests ranging from 10 to 50 (M_BB10-M_BB50). Moreover, for each given combination of parameters, all instances from the tuning set are solved ten times.

To tune σ_1, σ_2, and σ_3 introduced in Section 4.4.3, which are the control parameters used in the roulette wheel mechanism, we run numerical tests on all instances from the tuning set. There are four value combinations considered in this test: $(5, 1, 3)$, $(1, 3, 5)$, $(1, 1, 1)$, and $(5, 3, 1)$. The comparison of these four different combinations is presented in Table 4.4, which gives Gap (%), which is the gap between the best solution found by ALNS out of ten runs and the best solution found by either the tailored labeling algorithm (Sun et al. [109]) or the branch-and-price-based framework (Sun et al. [110]) (which is in most situations the optimal solution). Moreover, as the proposed algorithm has randomized components, we also show the standard deviations of the solutions with ten runs (STD) to show the robustness of the proposed approaches.

The results presented in Table 4.4 show that even though the difference across all combinations is relatively small, $(5, 3, 1)$ performs the best overall in terms of both the solution quality

and robustness of the algorithm. In contrast, $(1, 3, 5)$ performs the worst in terms of the quality of the found solutions and $(5, 1, 3)$ performs the worst in terms of the robustness of the algorithm. Therefore, based on this analysis, we chose to use (σ_1, σ_2, σ_3)=$(5, 3, 1)$ in the rest of the numerical experiments. In this combination set, it rewards most when a new best solution is found, and rewards least when an incumbent solution that is worse than the current solution is accepted. Detailed instance-level results can be found in Table 10 in the Appendix C.

Table 4.4: Tuning of roulette wheel mechanism parameters (σ_1, σ_2, σ_3)

Problem type	# Instances	(σ_1, σ_2, σ_3)	Gap (%)	STD	Time (s)
BB group in single-vehicle instances	14	(5,1,3)	1.04	1.40	21.2
		(1,3,5)	1.02	1.58	22.2
		(1,1,1)	1.11	1.57	23.0
		(5,3,1)	0.62	1.17	21.7
BB group in multi-vehicle instances	9	(5,1,3)	0.00	0.31	6.5
		(1,3,5)	-0.06	0.16	6.7
		(1,1,1)	-0.06	0.31	7.1
		(5,3,1)	-0.18	0.33	6.5
Average over 23 instances in BB group	23	(5,1,3)	0.63	0.97	15.4
		(1,3,5)	0.60	1.02	16.1
		(1,1,1)	0.65	1.08	16.8
		(5,3,1)	0.34	0.84	15.8

To tune the cooling rate parameter κ, which controls the decrease in the speed of the temperature in the simulated annealing algorithm, we test all instances from the tuning set using three different values for κ: 0.95, 0.995, and 0.9995. A higher value of κ means that the temperature is decreasing slower, resulting in a higher probability of accepting a worse incumbent solution in an iteration. In Table 4.5, the comparison of the results obtained with the three different cooling rate values is presented. The values of Gap (%) and STD (%) are the same as explained in Table 4.4. Moreover, we also report Time(s), which describes the average processing time in seconds of the 10 runs. Our results suggest that setting κ to 0.9995 performs the best in terms of both the objective function value and the robustness of the algorithm. Therefore, in the following computational experiments, we chose to set the value of κ equal to 0.9995. Detailed instance-level results can be found in Table 11 in the Appendix C.

To tune the destroy rate parameter ρ, which controls the removal fraction of the removal operators in each iteration, we run tests on all instances from the tuning set using three

Table 4.5: Tuning of cooling rate parameter

Problem Type	# Instance	Cooling Rate (κ)	Gap (%)	STD (%)	Time (s)
BB group in single-vehicle instances	14	0.95	1.42	2.59	17.5
		0.995	1.35	2.19	18.
		0.9995	0.62	1.17	23.2
BB group in multiple-vehicle instances	9	0.95	-0.14	0.56	5.9
		0.995	-0.06	0.33	6.1
		0.9995	-0.18	0.33	6.6
Average over 23 instances in BB group	23	0.95	0.81	1.80	13.0
		0.995	0.80	1.46	13.9
		0.9995	0.31	0.84	15.8

different destroy rates: 0.25, 0.35, and 0.45. A higher value of ρ means that more requests are removed from the current solution in each iteration. The comparison is reported in Table 4.6. The meanings of Gap (%), STD, and Time(s) are the same as in Table 4.4. Our results suggest that it is preferable to use a large value for ρ because ρ =0.45 generally leads to better performance both in terms of solution quality and robustness of the algorithm. However, this requires more processing time, particularly in the insertion stage as more requests were removed and need to be inserted again. Based on this analysis, we decided to set $\rho = 0.45$ in our computational experiments.

Table 4.6: Tuning of destroy rate parameter

Problem Type	# Instance	Destroy Rate (ρ)	Gap (%)	STD (%)	Time (s)
BB group in single-vehicle instances	14	0.25	0.80	1.59	15.1
		0.35	0.97	1.42	18.4
		0.45	0.62	1.17	21.7
BB group in multiple-vehicle instances	9	0.25	-0.05	0.28	3.7
		0.35	-0.09	0.28	5.4
		0.45	-0.18	0.33	6.5
Average over 23 instances in BB group	23	0.25	0.47	1.08	10.7
		0.35	0.56	0.97	13.3
		0.45	0.31	0.84	15.8

4.6.2 *Relative performances of operators*

Figure 4.7 and 4.8 present the relative performances of the removal and insertion operators in terms of the speed and solution quality, respectively. To this end, we solve all instances from the tuning set. For each instance, we run our algorithm ten times. In Figure 4.7, Total time(s) is the average time spent to run each operator, and Usage (%) reports the percentage of the total iterations in which the operator is used. The results in Figure 4.7(a) show that all the removal operators can be processed within 0.1 s and WR is the slowest removal operator owing to its computational complexity. Figure 4.7(b) indicates that the frequencies of using different removal operators do not significantly vary from one another. Compared with the removal operators, the running times of the insertion operators (except SI) are significantly longer. Moreover, Figure 4.7(c) shows that the times consumed by GI, DGI, and RI are higher than those of the remaining insertion operators owing to the high usage of these three operators (see Figure 4.7(d)).

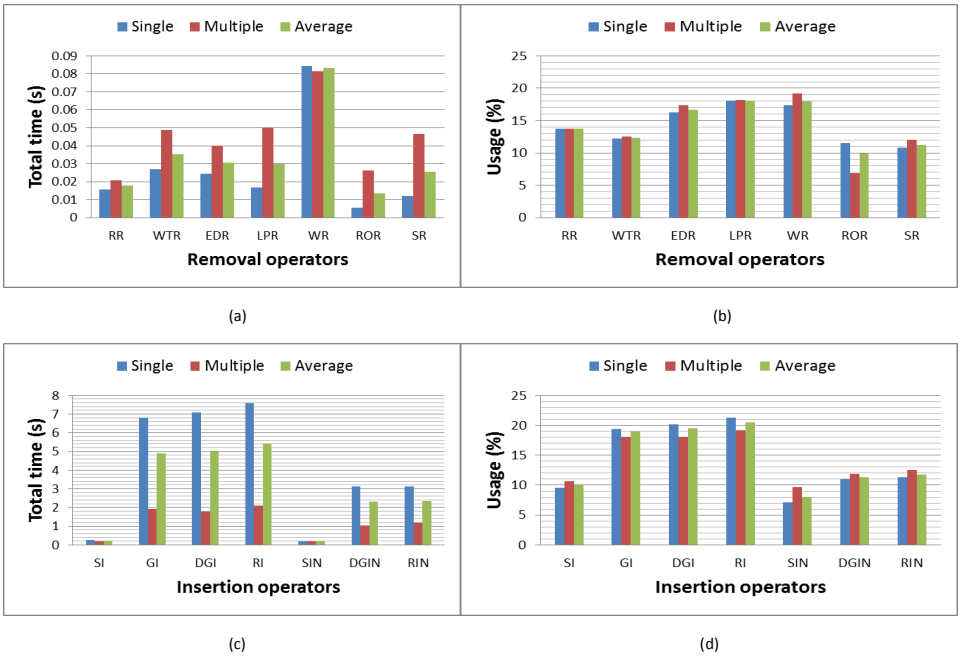

Figure 4.7: Running time and percentage of usage of operators

Figure 4.8 exhibits the percentage of iterations for which insertion and removal operators yielded a solution that was better than the best found solution and current solution for which that particular operator was used. Figure 4.8(a) shows that RR is the best performing removal operator to improve the best-found solution, whereas EDR, LPR, and WR perform better than

the rest of the removal operators in terms of improving the current solution (see Figure 4.8(b)). According to Figure 4.8(c) and (d), the best performing insertion operators are GI, DGI, and RI. Note that some operators (viz., SIN and GIN) scarcely improve the best-found solution or current solution. However, these operators are required to diversify the search and achieve a better overall performance of the algorithm.

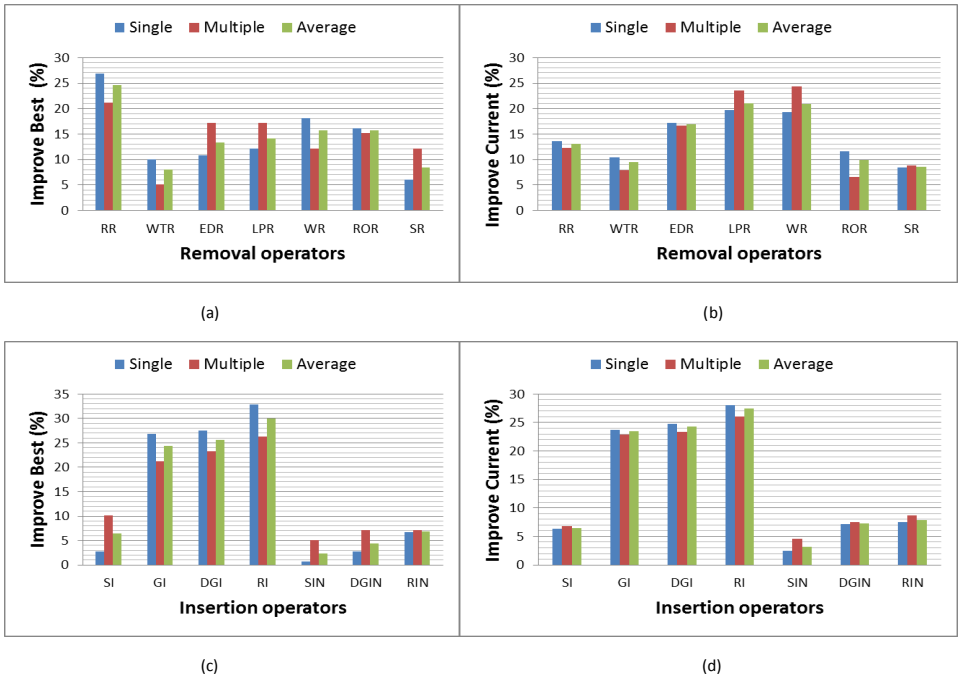

Figure 4.8: Percentage improvement over best solutions found and current solutions achieved by operators

4.6.3 Exact evaluation versus approximation

As mentioned in Section 4.4.1, an approximate route evaluation is proposed. To test the effectiveness and efficiency of this proposed approximation, we implement an exact route evaluation for our proposed ALNS. In Table 4.7, we compare the average performance of our proposed ALNS using the exact route evaluation with the ALNS using the approximate route evaluation. As discussed in Section 4.4.1, the approximate route evaluation can make an error in estimating the optimal departure time leading to, at the most, two times longer route duration for our specific instances (note that the largest relative difference in the speed profiles of Table 4.2 is 200% for both profiles, NSMP and HS). However, the results show that the solution quality is almost the same for both evaluation methods. For single-vehicle

instances, the ALNS using the approximate route evaluation is, however, 2.5 times faster than the ALNS using the exact route evaluation. For multi-vehicle instances, the ALNS using approximate route evaluation can also save 50% of the processing time compared with the ALNS using exact evaluation. Detailed instance-level results can be found in Table 13 and 14 in the Appendix C.

Table 4.7: Exact evaluation versus approximation

Instance Type	Obj_{EE}	$Time_{EE}$	Obj_{AE}	$Time_{AE}$
Single-Vehicle instances	398.58	138.70	397.21	54.61
Multi-Vehicle instances	2267.16	49.2	2266.83	31.6

Obj_{EE}: Objective value of ALNS with Exact Evaluation.
Obj_{AE}: Objective value of ALNS with Approximate Evaluation.
$Time_{EE}$ (s): Processing Time of ALNS with Exact Evaluation.
$Time_{AE}$ (s): Processing Time of ALNS with Approximate Evaluation.

4.6.4 Performance of proposed ALNS under various settings

We next present in Table 4.8 the average performance of our proposed ALNS when configured to use different sets of operators. The results show that the ALNS configured to use all operators has the best performance. Furthermore, all the proposed removal operators help in improving the performance of the proposed ALNS. The ALNS with full machinery also outperforms the traditional LNS without adaptive mechanism.

Table 4.8: Average performance of proposed ALNS under various settings

Settings	Best	Aver	Time
LNS	1435.02	1415.70	67.61
ALNS without SIN, GIN and RIN	1438.25	1420.39	72.66
ALNS without GI	1436.32	1417.23	53.23
ALNS without RI	1433.61	1413.27	56.37
ALNS without SR	1432.53	1415.38	57.14
ALNS without WR	1440.05	1419.44	60.16
ALNS without RR	1433.14	1410.37	55.52
ALNS without WDR	1435.01	1418.64	63.18
ALNS without ROR	1438.45	1420.20	65.91
ALNS without EDR	1437.06	1419.07	65.41
ALNS without LPR	1439.32	1420.62	62.19
ALNS	1441.63	1418.73	65.55

4.6.5 *Impact of vehicle costs*

In this section, we evaluate the performance of the proposed ALNS for various vehicle costs. As can be seen in Figure 4.9, high vehicle costs lead to both fewer requests being served and fewer vehicles used. This is due to the total collected profits not being able to compensate for the high vehicle cost. In some special cases (e.g., M_BB10), high vehicle costs lead to no vehicles being used to serve requests.

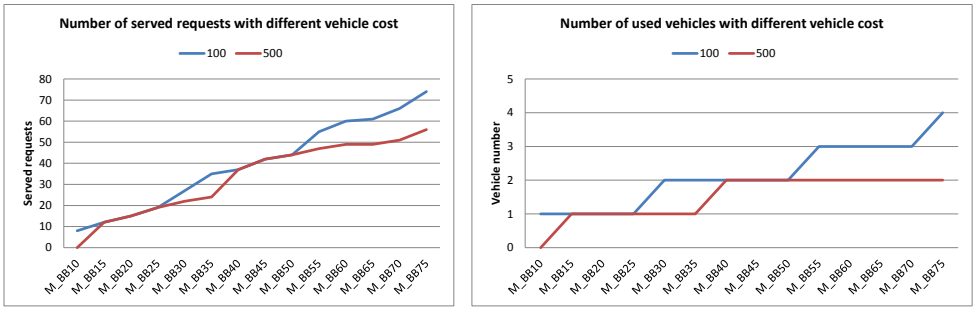

Figure 4.9: Impact of vehicle costs

4.6.6 *Performance on TD-PPDP-TW instances*

In this section, we discuss the results on the two generated sets of TD-PPDP-TW instances described in Section 4.5. All instances are solved ten times using the proposed ALNS framework. The detailed results of these experiments are presented in Tables 4.9-4.12.

In Tables 4.9-4.10, we compare the results of the single-vehicle instances using our proposed ALNS framework with the results obtained by Sun et al. [109]. In particular, Table 4.9 shows the statistics of the single-vehicle instances that can be solved to optimality within 10 h by using the tailored labeling algorithm in Sun et al. [109]; Table 4.10 presents the statistics of the challenging instances that are solved by dynamic programming heuristic in Sun et al. [109], because they are too difficult to be solved to optimality within the predefined time limit. Column Best reports the optimal solution derived by the tailored labeling algorithm and column Best(DPH) shows the best solution calculated by the dynamic programming heuristic as in Sun et al. [109]. Columns Gap (%) and STD are the same as explained for Table 4.4. We also present the processing times of the tailored labeling algorithm ($Time_{TL}$ (s)), dynamic programming heuristic $Time_{DPH}$ (s), and proposed ALNS algorithm ($Time_{ALNS}$ (s)) in seconds. The results given in Table 4.9 show that our ALNS heuristic is able to compete with the exact method proposed in Sun et al. [109]. Although the exact method outperforms the ALNS for instances up to 35 requests, the proposed ALNS needs significantly less processing

time for instances with more than 35 requests. The average gap is less than 0.60%. The data in Table 4.10 indicate that, for those challenging instances with up to 75 requests, the average gap is not greater than 2.00% for the proposed ALNS heuristic. In particular, the performance of the ALNS is better than the dynamic programming heuristics (see Table 4.10) for one instance (S_DD75). Our ALNS heuristic also needs significantly less processing time than the dynamic programming heuristic.

Table 4.11 provides the results obtained for multi-vehicle instances considered in Sun et al. [110]. For the exact method proposed by Sun et al. [110], the lower bound, best upper bound, and the processing time of the exact method are reported in columns LB, BestUB, and $Time_{exact}$(s), respectively. As for our proposed ALNS heuristic, the values of Best, STD, $Time_{ALNS}$(s), and Gap% are as described in Table 4.10. The results show that the ALNS heuristic is able to compete with the exact branch-and-price-based algorithm proposed in Sun et al. [110], because it obtained the optimal solutions in at least one out of the ten runs for 22 of the 27 optimally solved instances by Sun et al. [110]. For instances that could not find any solution or obtained sub-optimal solutions in Sun et al. [110], the proposed ALNS is able to find better solutions within the imposed time limit, which is calculated as $100 * (Best - BestUB)/BestUB$. The average CPU time needed to solve those instances is 10.3 s, which is significantly shorter CPU time compared with the exact method proposed in Sun et al. [110].

We present computational results of the multi-vehicle instances with 50 to 75 requests in Table 4.12. These instances are not considered in Sun et al. [110] due to the limited scalability of the exact algorithm. Columns $Best$, STD, and $Time_{ALNS}(s)$ are self-explanatory. For each instance, we also present the average objective function value of the ten runs (Avg.), number of served requests (Served), and number of used vehicles (# Vehicle) in the best-found solution. Based on the obtained results, the proposed ALNS is stable and computationally efficient. All these instances are solved with a relatively small standard deviation (0.80% on average), and the average processing time of these instances is less than 1 min (59.9 s).

Furthermore, the data in Table 4.9 and 4.10 show several single-vehicle instances with a relatively high gap. One of the possible reasons for this is the profit distribution among the requests. Compared with the multiple-vehicle instances with randomly generated profit in interval [50, 150], a profit of 40 units is assigned to each request in the single-vehicle instances. Therefore, the proposed algorithm gets easily stuck into local optima for several single vehicle instances. We added two local search components to our proposed ALNS algorithm: one is

Table 4.9: Computational results for challenge single-vehicle instances solved to optimality within 10 h in Sun et al. [109]

Instance	Best	Gap (%)	STD (%)	$Time_{TL}$ (s)	$Time_{ALNS}$ (s)
S_AA10	27.14	0.00	0.00	0.2	0.6
S_BB10	39.26	0.00	0.00	0.2	0.6
S_CC10	57.63	0.00	0.00	0.3	0.7
S_DD10	85.88	0.00	0.00	0.3	0.7
S_AA15	37.77	0.00	0.00	0.4	1.2
S_BB15	99.58	0.00	0.00	0.5	1.6
S_CC15	98.29	0.00	0.00	0.6	1.7
S_DD15	99.05	0.00	0.00	0.6	1.7
S_AA20	73.70	0.00	0.00	0.8	1.9
S_BB20	138.05	0.00	0.00	2.0	2.6
S_CC20	97.27	0.00	0.00	1.3	2.4
S_DD20	182.14	0.00	0.00	2.1	4.1
S_AA25	203.01	0.00	0.00	1.6	4.7
S_BB25	147.88	0.00	0.00	2.3	4.8
S_CC25	226.23	0.60	0.00	12.0	9.9
S_DD25	250.16	0.00	2.37	199.7	13.9
S_AA30	266.72	0.00	0.00	2.1	8.6
S_BB30	274.45	0.00	3.73	2.8	7.4
S_CC30	316.39	0.00	1.65	27.6	18.5
S_DD30	343.65	0.00	1.04	80.4	24.6
S_AA35	278.14	0.00	0.00	5.0	10.9
S_BB35	337.55	3.09	0.00	1.0	7.1
S_CC35	386.26	0.00	4.65	104.2	23.2
S_DD35	410.19	0.00	3.29	338.2	28.9
S_AA40	341.74	0.00	0.77	5.8	14.1
S_BB40	326.82	0.00	0.00	10.0	10.5
S_CC40	464.28	0.00	1.10	942.3	55.2
S_DD40	490.92	4.16	1.08	4618.1	42.2
S_AA45	361.57	0.00	0.33	5.6	26.0
S_BB45	378.90	0.62	0.13	27.9	20.6
S_CC45	497.68	0.00	1.42	3422.2	50.8
S_DD45	540.17	0.78	3.87	7668.0	62.3
S_AA50	430.41	0.00	3.44	9.5	31.1
S_BB50	432.10	2.30	2.82	37.4	22.6
S_CC50	519.60	0.68	2.64	13913.4	98.0
S_DD50	610.07	0.00	3.61	21889.9	79.9
S_AA55	436.05	0.00	2.02	18.7	52.0
S_BB55	530.84	0.00	2.48	59.0	33.8
S_CC55	581.50	0.00	1.87	28429.3	90.7
S_AA60	505.02	1.00	0.48	31.7	43.3
S_BB60	558.02	0.36	0.04	72.9	39.8
S_AA65	576.39	1.43	0.84	30.2	103.6
S_BB65	547.84	0.00	4.90	129.7	41.4
S_AA70	606.68	0.00	1.19	39.1	86.3
S_BB70	558.29	4.54	0.93	351.6	45.4
S_AA75	715.53	3.19	0.85	257.6	161.9
S_BB75	613.35	3.07	2.44	511.4	65.8
Average	342.56	0.55	1.19	1771.6	31.1

Table 4.10: Computational results for challenge single-vehicle instances solved by dynamic programming heuristic (Sun et al. [109])

Instance	Best (DPH)	Gap (%)	STD (%)	$Time_{DPH}$ (s)	$Time_{ALNS}$ (s)
S_DD55	639.75	1.68	3.78	2754.4	94.2
S_CC60	624.95	2.49	1.49	1840.4	165.5
S_DD60	713.39	4.52	0.84	3879.8	147.6
S_CC65	662.61	0.01	3.83	4054.8	142.9
S_DD65	760.32	2.13	4.94	5853.2	170.5
S_CC70	681.84	1.82	2.70	4859.0	195.6
S_DD70	764.10	0.75	2.13	9581.9	238.5
S_CC75	686.68	2.14	0.89	7159.9	153.2
S_DD75	835.76	-1.21	3.26	15842.3	290.9
Average	707.71	1.59	2.65	6202.9	177.7

the well-known 2-opt mechanism, which considers exchanging the sequence of two nodes already linked by a direct arc; the other is called replace, which removes one served request from the route and replaces it by non-included requests if this replacement yields a higher objective value. To keep the computational effort at a reasonable level, both of them are called only at the end of each segment. Moreover, we also check all the feasible moves in the incumbent neighborhood and keep the best one. One local search operator continues to be used as long as an improvement is obtained. If there is no improvement, the unused components are explored. The local search stops when the last local search operator yields no improvement. In Table 1 in Appendix B, we compare the performances of the proposed ALNS algorithm with and without these two local search components. We perform experiments on those single-vehicle instances with gaps between 2% and 4.6%. It shows that additional local search components barely have an impact on improving the performance of the proposed ALNS algorithm.

4.6.7 Impact of time windows

For both classes of instances, the time windows of instances in the DD group are twice as wide as those in the BB group. Because time windows affects the performance of the proposed ALNS, we present the average performance of each removal and insertion operators for all instances in the BB and DD groups of both classes. In particular, for each instance, we list the total running time of each operator (Total time), the number of times that an operator is used in the algorithm (# Number), the number of times an operator has improved the best solution (# ImproveBest), the number of times an operator has improved the current solution

(# ImproveCurrent), and the number of times an incumbent solution worse than the current solution is accepted (# AcceptWorse). Note that these are average values calculated across all BB and DD group instances up to 50 requests in Table 4.13 and 4.14, respectively. We report the average performance of each operator for BB group and DD group instances with 50-75 requests in Table 4.15 and 4.16, respectively. The results show that the performances of removal operators are almost the same for BB group instances (Table 4.13 and 4.15) and DD group instances (Table 4.14 and 4.16). In contrast, compared with the instances of the BB group, the insertion operators need more computation time for instances of the DD group. Because the time windows considered in instances of the DD group are twice as wide as their counterpart in instances of the BB group, the selected insertion operator needs to evaluate more possible insert positions to get the best potential improvement during each iteration.

4.6.8 Significance of time-dependent travel times

We perform several experiments to demonstrate the advantages of considering time-dependent travel time. Particularly, we generated three different time-independent cases based on data given in Table 4.2, which are the slow- (TID-Slow), average- (TID-Average), and fast-speed cases (TID-Fast) by using the lowest, average, and the fastest speeds of their originally assigned speed profiles, respectively. For each setting, we present in Table 4.17 the objective function value of the best-found solution (Value), objective value of the best-found solution under the time-dependent settings (TDobj), number of nodes in the best found solution in which the time window constraints are violated under time-dependent settings (# VioNum), the percentage of violation (VioRate (%)) (i.e., let # totalVisit be the total number of pickup and delivery nodes visited in the best-found solution. Then $VioRate(\%) = 100 * \#VioNum/\#totalVisit$), lateness of the best-found solution (Lateness), the traveling cost of the best-found solution (TravelCost) and total profits collected by the best-found solution (Profits), assuming that the profit is still collected if the time window is violated. We list the statistics related to the comparative results under the different speed profile settings in Table 4.17. Note that these are average values calculated across the single-vehicle instances, multi-vehicle instances, and all instances, respectively. Detailed, instance-level results can be found in Table 15-20 in Apprendix C. We see that the best-found solution in the fast-speed case (where it is assumed that the vehicle can always run on the fastest possible speed) is clearly over-optimistic, which also meets the high risk of violating the time window constraints. In contrary, the best-found solution in the slow-speed case is too conservative. Even though no violation and penalty costs are generated, the quality of the best-found solution is significantly worse than that of its counterpart under the time-dependent setting. The data in table also

show that assuming a time-independent relation with the average speed as approximation for the time-dependent variant is not working out. It leads to 3% of violated time windows and more than 10% higher routing cost.

4.6.9 Alternative objective functions

In the preceding sections, we have only used the duration to represent the travel cost. This follows the majority of the literature and makes it possible to compare against the optimal solutions provided in Sun et al. [109] and Sun et al. [110]. We note that travel cost is dependent on both duration (e.g., wage of driver) and distance (e.g., fuel consumption). We have used duration only because distance is also highly correlated to this. However, we realize that different companies have different weights for the variable cost in terms of distance and duration costs. Therefore, in this section, we consider an alternative objective function wherein the cost of a route is a weighted combination of its distance and duration. We report in Tables 4.18 and 4.19 the performance of the ALNS when optimizing this objective function under different weights (α is the weight given to duration, β is the weight given to distance) on two different sets of instances. We note that we still use a fixed travel cost of 100 for using a vehicle in the multiple vehicle setting.

Tables 4.18 and 4.19 show that the ALNS produces high-quality solutions in the same amount of time regardless of the weights. As expected an objective that heavily weights distance traveled leads to shorter routes with longer durations. In the original objective (alpha=1) the vehicles have an average speed of 1074/732 = 1.47, whereas in the situation of a very low weight for duration (alpha =0.1) this average speed goes dowen by more than 20% to 1.13.

Table 4.11: Computational results for multi-vehicle instance considered in Sun et al. [110]

	Exact			ALNS			
Instance	LB	BestUB	$Time_{exact}$ (s)	Best	STD (%)	$Time_{ALNS}$ (s)	Gap (%)
M_AA10	262.66	262.66	0.3	262.66	0.00	0.9	0.0
M_BB10	222.60	222.60	0.3	222.60	0.00	1.1	0.0
M_CC10	372.37	372.37	1.5	372.37	0.00	0.6	0.0
M_DD10	271.96	271.96	1.9	271.96	0.00	1.0	0.0
M_AA15	520.81	520.81	3.4	520.80	0.00	0.9	0.0
M_BB15	554.24	554.24	0.9	554.24	0.00	1.6	0.0
M_CC15	684.31	684.31	14.3	684.31	0.00	1.9	0.0
M_DD15	606.48	606.48	8.6	606.48	0.00	2.1	0.0
M_AA20	706.36	706.97	47.0	706.36	0.24	3.3	0.1
M_BB20	843.94	843.94	3.4	843.94	0.00	2.9	0.0
M_CC20	964.99	964.99	23.5	965.00	0.00	4.5	0.0
M_DD20	942.38	942.38	9.2	942.38	0.00	7.0	0.0
M_AA25	1051.88	1052.68	65.8	1051.88	0.00	3.4	0.1
M_BB25	1184.67	1184.67	14.3	1184.67	0.00	4.0	0.0
M_CC25	1355.72	1355.72	214.6	1355.72	0.00	8.4	0.0
M_DD25	1303.20	1303.20	210.0	1303.20	0.08	17.1	0.0
M_AA30	1284.11	1284.11	4512.3	1279.92	1.13	5.6	0.3
M_BB30	1506.89	1508.29	267.3	1506.89	0.37	6.3	0.1
M_CC30	1689.39	1689.77	3544.1	1687.69	0.61	11.0	0.1
M_DD30	1627.76	1629.33	9158.2	1627.75	0.35	24.4	0.1
M_AA35	1656.81	1658.46	15804.2	1656.81	0.43	5.8	0.1
M_BB35	1898.97	1898.97	138.5	1898.97	0.00	5.9	0.0
M_CC35	2028.82	2030.72	26082.1	2027.52	0.18	17.2	0.2
M_DD35	1960.66	2021.95	36000.0	1999.89	0.49	22.4	1.1
M_AA40	1887.74	1889.57	433.4	1877.99	0.28	8.2	0.6
M_BB40	2150.70	2150.70	523.1	2150.70	0.13	7.9	0.0
M_CC40	2317.31	2356.45	36000.0	2337.87	0.34	25.2	0.8
M_DD40	-	-	36000.0	2288.32	0.63	29.7	-
M_AA45	2259.00	2259.00	1170.0	2259.00	0.82	13.3	0.0
M_BB45	2525.08	2525.08	257.9	2525.08	0.77	17.9	0.0
M_CC45	-	-	36000.0	2617.54	0.85	39.7	-
M_DD45	2425.56	2703.40	36000.0	2644.57	0.61	29.0	2.2
Average	1302.25	1315.19	7578.4	1310.97	0.26	10.3	0.2

Table 4.12: Computational results on multi-vehicle instances with 50-75 requests

Instance	Best	Avg.	STD	Time (s)	Served	# Vehicle
M_AA50	2382.86	2368.04	0.76	19.1	43	2
M_BB50	2811.01	2755.63	1.66	19.8	44	2
M_CC50	2900.53	2887.15	0.44	53.2	49	3
M_DD50	2981.62	2972.07	0.24	38.8	50	2
M_AA55	2749.24	2661.57	1.82	25.4	47	2
M_BB55	3022.66	3002.14	0.55	15.4	55	3
M_CC55	3219.65	3176.46	1.10	65.7	54	3
M_DD55	3307.22	3236.89	1.55	68.6	53	2
M_AA60	2869.56	2870.09	0.18	32.5	57	3
M_BB50	3316.43	3289.98	0.77	24.0	60	3
M_CC60	3481.61	3446.79	0.54	81.7	60	3
M_DD60	3510.05	3479.57	0.78	86.7	59	3
M_AA65	3195.79	3168.67	0.29	35.0	61	4
M_BB65	3463.66	3396.69	1.48	27.0	61	3
M_CC65	3743.19	3716.17	0.62	85.9	63	3
M_DD65	3738.84	3702.24	0.53	83.8	65	3
M_AA70	3545.38	3540.92	0.22	45.9	67	3
M_BB70	3765.20	3670.26	1.43	36.2	66	3
M_CC70	4022.77	3957.91	1.36	117.6	70	3
M_DD70	4091.04	4057.52	0.42	136.4	69	3
M_AA75	3959.73	3937.51	0.29	50.4	72	3
M_BB75	4049.40	3995.48	0.67	34.1	74	4
M_CC75	4249.11	4228.19	0.54	134.5	73	4
M_DD75	4330.78	4274.36	0.94	119.7	74	3
Average	3446.14	3408.01	0.80	59.9	60.25	2.92

Table 4.13: Average performance of each operator for all BB group instances (time window width 60) of both classes with up to 50 requests

Removal	RR	WTR	EDR	LPR	WR	ROR	SR
Total time	0.013	0.035	0.029	0.032	0.069	0.014	0.028
# Number	624.8	564.2	731.8	775.8	802.7	420.7	538.2
# ImproveBest	2.2	0.8	1.4	1.3	1.4	1.6	1.0
# ImproveCurrent	124.2	88.2	151.2	189.3	197.6	88.8	84.4
# AcceptWorse	237.1	218.7	344.1	327.9	348.3	129.8	174.9
Insertion	SI	GI	DGI	RI	SIN	DGIN	RIN
Total time	0.215	2.595	2.619	2.827	0.171	1.447	1.488
# Number	478.6	818.8	829.4	868.6	389.3	527.6	546
# ImproveBest	0.7	2.4	2.3	2.9	0.3	0.5	0.7
# ImproveCurrent	68.4	213.4	218.7	242.4	32.1	71.0	77.6
# AcceptWorse	162.6	345.2	345.3	380.7	134.6	203.2	209.2

Table 4.14: Average performance of each operator for all DD group (time window width 120) instances of both classes with up to 50 requests

Removal	RR	WTR	EDR	LPR	WR	ROR	SR
Total time	0.011	0.036	0.028	0.022	0.08	0.008	0.016
# Number	713.5	719.2	772.9	776.1	791.7	435.4	626.6
# ImproveBest	3.1	1.8	1.4	1.4	2.6	1.7	1.7
# ImproveCurrent	142.7	159.9	157.9	172.6	183.1	78.4	108.1
# AcceptWorse	206.1	210.3	271.1	271.9	258.1	106.4	158.6
Insertion	SI	GI	DGI	RI	SIN	DGIN	RIN
Total time	0.437	6.987	6.927	7.190	0.188	2.018	2.221
# Number	574.8	1003.9	1019.3	1056.5	324.4	421.6	434.7
# ImproveBest	0.9	3.3	4.3	4.7	0.1	0.2	0.1
# ImproveCurrent	89.3	263.8	266.7	290.5	21.2	34.6	36.7
# AcceptWorse	159.3	322.9	332.2	345.4	80.3	118.7	123.7

Table 4.15: Average performance of each operator for all BB group (time window width 60) instances of both classes with 55 requests to 75 requests

Removal	RR	WTR	EDR	LPR	WR	ROR	SR
Total time	0.025	0.056	0.048	0.036	0.172	0.009	0.031
# Number	1010.8	1051.8	1338.8	1354.4	1300.3	456.1	816.6
# ImproveBest	6	2.2	2.2	4	4.3	1.9	3.6
# ImproveCurrent	107.7	105.4	176.5	206.7	188.8	52.8	72.5
# AcceptWorse	195.2	251.1	384.4	335.9	305	62.3	143.2
Insertion	SI	GI	DGI	RI	SIN	DGIN	RIN
Total time	0.599	10.440	10.314	12.550	0.483	4.346	4.658
# Number	657	1453.6	1463.2	1804.8	486.5	713.7	750
# ImproveBest	1	5.1	6	10.6	0.2	0.6	0.7
#ImproveCurrent	40.7	218.4	212.9	310.9	22.2	49.3	56
# AcceptWorse	95.1	356.9	368.9	539.8	60.1	121.1	135.2

Table 4.16: Average performance of each operator for all DD group (time window width 120) instances of both classes with 55 requests to 75 requests

Removal	RR	WTR	EDR	LPR	WR	ROR	SR
Total time	0.032	0.058	0.049	0.045	0.185	0.002	0.036
# Number	1159.4	1140.8	1287.4	1337.4	1261.2	564.0	877.9
# ImproveBest	8.6	3.1	2.0	4.6	4.1	2.1	4.4
# ImproveCurrent	134.1	135.9	162.5	197.5	171.5	58.7	109.3
# AcceptWorse	192.9	196.1	246.6	263.8	245.3	64.2	155.9
Insertion	SI	GI	DGI	RI	SIN	DGIN	RIN
Total time	1.253	45.310	45.089	49.813	0.552	9.917	10.731
# Number	662.3	1760.4	1764.5	1970.1	324.9	571.7	574.2
# ImproveBest	1.4	7.2	7.1	12.6	0.1	0.3	0.2
# ImproveCurrent	36.3	262.9	267.9	336.5	9.7	27.5	28.7
# AcceptWorse	76.9	338.3	346.3	416.3	32.2	82.3	72.5

Table 4.17: Results under various speed settings

Speed Settings	Value	TDobj	# VioNum	VioRate (%)	Lateness	TravelCost	Profits
Single-vehicle instances							
TID-Slow	113.63	189.21	0	0.00	0.00	338.76	526
TID-Average	399.42	399.02	2	3.15	13.96	600.97	1001
TID-Fast	593.30	499.36	24	39.00	609.93	708.14	1210
TD	510.93	510.93	0	0.00	0.00	539.07	1050
Multi-vehicle instances							
TID-Slow	2002.03	2252.67	0	0.00	0.00	1495.28	4065
TID-Average	2764.03	2728.21	3	2.74	31.35	1357.78	4344
TID-Fast	3033.26	2852.55	19	17.78	333.21	1281.00	4372
TD	2868.93	2868.93	0	0.00	0.00	1195.60	4320
All Instances							
TID-Slow	1057.83	1220.94	0	0.00	0.00	917.02	2296
TID-Average	1581.73	1563.62	2	2.95	22.66	979.38	2673
TID-Fast	1813.28	1675.96	22	28.39	471.57	994.57	2791
TD	1689.93	1689.93	0	0.00	0.00	867.34	2685

Table 4.18: Average performance of various alternative objective functions for all BB group instances

Optimized with (α, β)	Obj.	Dur.	Dist.	Profit	Vehicle cost	Comp. Time (s)	Objective if solution is evaluated with different (α, β) (1.0,0.0)	(0.9,0.1)	(0.8,0.2)	(0.7,0.3)	(0.6,0.4)	(0.5,0.5)	(0.4,0.6)	(0.3,0.7)	(0.2,0.8)	(0.1,0.9)	(1.0,1.0)
(1.0, 0.0)	1297.2	732.4	1074.1	2129.5	100.0	18.3	**1297.2**	1263.0	1228.8	1194.6	1160.5	1126.3	1092.1	1057.9	1023.8	989.6	223.1
(0.9, 0.1)	1263.2	730.5	1065.2	2127.2	100.0	19.8	1296.7	**1263.2**	1229.8	1196.3	1162.8	1129.4	1095.9	1062.4	1029.0	995.5	231.5
(0.8, 0.2)	1230.0	731.8	1059.0	2127.2	100.0	20.3	1295.5	1262.7	**1230.0**	1197.3	1164.6	1131.9	1099.1	1066.4	1033.7	1001.0	236.5
(0.7, 0.3)	1197.3	723.2	1032.4	2109.7	96.4	20.4	1290.1	1259.1	1228.2	**1197.3**	1166.4	1135.5	1104.6	1073.7	1042.8	1011.8	257.7
(0.6, 0.4)	1168.0	728.4	1006.4	2104.0	96.4	18.8	1279.2	1251.4	1223.6	1195.8	**1168.0**	1140.2	1112.4	1084.6	1056.7	1028.9	272.8
(0.5, 0.5)	1140.6	736.4	1000.4	2105.4	96.4	20.0	1272.7	1246.2	1219.8	1193.4	1167.0	**1140.6**	1114.2	1087.8	1061.4	1035.0	272.2
(0.4, 0.6)	1114.5	742.6	987.5	2100.5	96.4	19.2	1261.5	1237.0	1212.5	1188.0	1163.5	1139.0	**1114.5**	1090.1	1065.6	1041.1	274.0
(0.3, 0.7)	1090.3	765.1	988.8	2111.9	100.0	19.0	1246.8	1224.4	1202.1	1179.7	1157.4	1135.0	1112.6	**1090.3**	1067.9	1045.5	258.0
(0.2, 0.8)	1071.6	781.0	961.8	2097.3	100.0	19.3	1216.3	1198.2	1180.1	1162.1	1144.0	1125.9	1107.8	1089.7	**1071.6**	1053.6	254.5
(0.1, 0.9)	1056.1	834.0	938.4	2091.1	107.1	21.1	1150.0	1139.6	1129.1	1118.7	1108.3	1097.8	1087.4	1077.0	1066.5	**1056.1**	211.6
(1.0, 1.0)	516.3	320.8	466.9	1375.4	71.4	21.6	983.2	968.6	953.9	939.3	924.7	910.1	895.5	880.9	866.3	851.7	**516.3**

Table 4.19: Average performance of various alternative objective functions for all DD group instances

Optimized with (α, β)	Obj.	Dur.	Dist.	Profit	Vehicle cost	Comp. Time (s)	Objective if solution is evaluated with different (α, β) (1.0,0.0)	(0.9,0.1)	(0.8,0.2)	(0.7,0.3)	(0.6,0.4)	(0.5,0.5)	(0.4,0.6)	(0.3,0.7)	(0.2,0.8)	(0.1,0.9)	(1.0,1.0)
(1.0, 0.0)	1447.0	669.8	1055.4	2206.1	89.3	64.6	**1447.0**	1408.5	1369.9	1331.3	1292.8	1254.2	1215.7	1177.1	1138.5	1100.0	391.6
(0.9, 0.1)	1409.0	669.2	1041.8	2204.7	89.3	70.6	1446.3	**1409.0**	1371.7	1334.5	1297.2	1259.9	1222.7	1185.4	1148.1	1110.9	404.5
(0.8, 0.2)	1372.1	674.0	1036.0	2207.8	89.3	70.2	1444.6	1408.3	**1372.1**	1335.9	1299.7	1263.25	1227.3	1191.1	1154.9	1118.7	408.5
(0.7, 0.3)	1336.7	666.2	1005.1	2193.8	89.3	68.4	1438.3	1404.4	1370.6	**1336.7**	1302.8	1268.9	1235.0	1201.1	1167.2	1133.3	433.2
(0.6, 0.4)	1303.1	673.7	1002.9	2197.7	89.3	65.2	1434.8	1401.8	1368.9	1336.0	**1303.1**	1270.1	1237.2	1204.3	1171.4	1138.5	431.9
(0.5, 0.5)	1270.9	679.8	993.1	2196.6	89.3	66.5	1427.5	1396.2	1364.9	1333.5	1302.2	**1270.9**	1239.6	1208.2	1176.9	1145.6	434.4
(0.4, 0.6)	1241.2	684.7	975.1	2189.4	89.3	65.1	1415.5	1386.4	1357.4	1328.3	1299.3	1270.3	**1241.2**	1212.2	1183.2	1154.1	440.4
(0.3, 0.7)	1213.7	710.8	967.0	2193.1	89.3	65.1	1393.0	1367.4	1341.8	1316.1	1290.5	1264.9	1239.3	**1213.7**	1188.0	1162.4	426.0
(0.2, 0.8)	1189.5	726.4	963.0	2194.5	89.3	62.4	1378.9	1355.2	1331.5	1307.8	1284.2	1260.5	1236.8	1213.2	**1189.5**	1165.8	415.8
(0.1, 0.9)	1166.9	729.7	951.6	2185.6	89.3	61.4	1366.6	1344.4	1322.2	1300.0	1277.8	1255.6	1233.4	1211.2	1189.0	**1166.8**	414.9
(1.0, 1.0)	626.2	340.6	509.5	1551.4	75.0	62.8	1135.8	1118.9	1102.0	1085.1	1068.2	1051.3	1034.4	1017.5	1000.6	983.8	**626.3**

4.7 Conclusions

We proposed a metaheuristic for the time-dependent profitable pickup and delivery problem with time windows (TD-PPDP-TW). To derive high-quality solutions within a reasonable computation time, we proposed techniques, such as approximate evaluation of objective function values, tabu lists, and hybrid stopping criterion, to improve the basic ALNS method. Moreover, seven tailored removal operators and 7 tailored insertion operators were designed to cope up with the characteristics of the proposed problem. To the best of our knowledge, this is the first time an ALNS has been proposed for a time-dependent pickup and delivery problem.

To evaluate its performance, we applied the proposed ALNS to the TD-PPDP-TW. The evaluation was based on solution quality and computation time on two classes of TD-PPDP-TW instances from the literature. Compared with the solutions provided by exact and/or heuristic methods in the literature, the proposed algorithm can generate promising results in a reasonable amount of computation time. Regarding future work, Turkeš et al. [112] reviewed the ALNS literature and carry out a meta-analysis to gain insights into the importance of the adaptive layer in ALNS, recently. They conclude that, on average, the addition of an adaptive layer in an ALNS has a very little positive impact. It would also be interesting to evaluate the contribution of each components of our proposed ALNS. Moreover, additional aspects may be considered such as a heterogeneous vehicle fleet, multiple vehicle depots, and stochastic aspects (demands, customers and travel time). Investigation of a fair profit reallocation scheme could also be an interesting research direction.

Chapter 5

A scenario-based approach for the time-dependent laundry routing problem with stochastic pickup demands

"Always bear in mind that your own
resolution to succeed is more
important than any other."

Abraham Lincoln

5.1 Introduction

Managing a reliable laundry program, to ensure full availability of clean textiles such as bed sheets, linens, blankets and curtains, has always been a complicated, time consuming and costly task for most hotels and restaurants. Most hospitality firms outsource this process to a commercial laundry provider. In addition to cleaning linens, the laundry business has to arrange the logistical processes (e.g., the collection of dirty laundry and the delivery of clean linen to the customers) as efficiently as possible.

Many laundry service providers work with weekly (or twice a week) schedules. As an example, in week A, the dirty laundry of that week is collected. Then it is cleaned during the week, and during week B, the clean laundry is brought back to the customer (e.g., hotel) and at that same time the dirty laundry of week B is collected. In these systems, the amount of clean linen to be delivered is known. However, most of the time it is not known beforehand how much dirty linen has to be picked up as this varies depending on, for example, the number of reservations the hotel or restaurant had during the past week. However, the laundry service provider can make predictions based on historic pick up volumes. Another interesting feature is that while the number of items picked up and delivered is, on average,

the same, the volume of dirty linen is typically greater than that of clean, folded linen.

In this chapter, we examine the operations of a laundry service provider. Its main objective is to deliver the known amount of clean textiles to hotels and restaurants while also collecting the unknown amount of used textiles from the same customers in the most efficient way. The pickup quantity of each hotel or restaurant is considered stochastic, but the laundry service provider can make a reasonable prediction based on each restaurant's or hotel's historical performance. Each restaurant or hotel indicates a time window in which they want to be served, which is dependent on when their (cleaning) staff is present to handle the exchange of textiles. It can happen that, when a vehicle of the laundry service arrives at one customer's location, the actual quantity of textile to pick up minus its delivery volume exceeds the remaining vehicle capacity. In that case, certain types of recourse actions must be taken. In this chapter, the effectiveness of multiple recourse actions is investigated. For example, the vehicle can return to the depot to unload all dirty textiles before resuming service. Due to the additional travel times generated by these recourse actions, time windows of the remaining customers along the planned delivery route may be violated.

At the same time, with the increasing volume of freight transport and limited road capacity, road congestion has become a big challenge in almost all modern cities. Therefore, travel times are fluctuating over the course of the day. In this study, we also intend to improve the pickup and delivery efficiency of a laundry service provider while considering these varying travel times on the road network. In order to capture the variation of the travel time during a day, we model it as a time-dependent travel time with a step-wise speed function assigned to each edge linking two nodes.

The scientific contributions of this chapter are summarized as follows:

- We consider a laundry delivery service by taking stochastic pickup demands into account and introduce a two-stage stochastic programming formulation to the time-dependent laundry (pickup and delivery) routing problem considering stochastic pickup demands.
- To handle the uncertainty in the pickup demands of each request, we propose a sample average approximation method together with an adaptive large neighborhood search (ALNS) algorithm. Moreover, four different types of recourse policies are considered to reduce the number of failures caused by pickup demand uncertainty.
- A number of computational experiments are performed, and a comparison with a pure

deterministic solution approach shows that on average more than 50 % cost savings can be achieved by using a stochastic solution approach.

The remainder of this chapter is organized as follows. The relevant literature of the existing work is presented in Section 5.2. We provide a detailed problem description and describe four recourse actions in Section 5.3. A two-staged mathematical formulation for our problem is introduced in Section 5.4, followed by the solution methods described in Section 5.5. In Section 5.6, we report the test instances and the corresponding computational results, respectively. Finally, Section 5.7 concludes the chapter with the main findings and brief discussion of future research questions.

5.2 Literature review

The time-dependent laundry routing problem with stochastic pickup demands (TDLRPSPD) studied in this chapter generalizes the vehicle routing problem with simultaneous pickup and delivery (VRPSPD) introduced by Min [73] by extending it with time windows, time-dependent travel time, and stochastic demands. It also has its roots in the stochastic routing and time-dependent routing literature. In this section, we briefly review the related literature.

Many different heuristic methods have been proposed to solve the VRPSPD. Dethloff [31] proposed an extension of the cheapest insertion heuristic to the VRPSPD. Several tabu search algorithms for VRPSPD were proposed in Montané and Galvao [76], Chen and Wu [17], Bianchessi and Righini [12], and Crispim and Brandão [24]. Later on, Ai and Kachitvichyanukul [1], Gajpal and Abad [43], and Subramanian et al. [106] proposed several population-search-based heuristics to solve VRPSPD. Besides heuristic solution methods, the VRPSPD has also been solved by exact methods. A branch-and-price method for the VRPSPD was designed by Dell'Amico et al. [28] to solve instances with up to 40 customers. Subramanian et al. [107] and Subramanian et al. [108] proposed a branch-and-cut algorithm and a branch- cut-and-price method, respectively, which are both capable of solving instances with up to 100 customers.

Only a few researchers have considered time window constraints in the VRPSPD. Angelelli and Mansini [2] introduced this problem and developed a branch-and-price algorithm to solve it. This approach was able to obtain optimal solutions on small-sized instances with up to 20 customers. Mingyong and Erbao [74] and Wang and Chen [124] proposed genetic algorithms for the vehicle routing problem with simultaneous pickup and delivery and time windows

(VRP-SPD-TW). A parallel simulated annealing method was introduced for VRP-SPD-TW by Wang et al. [123]. Liu et al. [68] developed a metaheuristic based on iterated local search for VRP-SPD-TW. To strengthen the search, they applied an adaptive neighborhood selection mechanism embedded into the improvement steps and the perturbation steps of iterated local search, respectively. Their results showed that the proposed approach outperformed the previous methods. Interested readers are referred to Battarra et al. [11] and Koç et al. [61] for more details.

In the real world, routing problems usually include one or several uncertain elements. According to Oyola et al. [80], demand, the presence of customers, travel times, service time and profit are sometimes modeled as stochastic in the routing literature. The problem discussed in this chapter is one variant of the vehicle routing problem with stochastic demands, which has been well studied in the literature. Overviews of the research in this area can be found in Gendreau et al. [47] and Henchiri et al. [53]. Only a few articles have considered routing problems with both time windows and stochastic demands. Chang [16] proposed a two-stage stochastic programming with recourse model for the VRPTWSD. In the first stage, a set of a priori delivery routes is planned. The actual customer demands are then revealed in the second stage. A detour-to-depot recourse policy is deployed in cases of route failures. The objective is to plan a set of a priori delivery routes that minimize the carrier's expected total cost, which is composed of the deterministic cost of the first-stage solution, the expected cost of recourse actions in the second stage, and the expected penalty cost for time window violations. Lei et al. [65] also formulated a two-stage stochastic programming with recourse model for the VRPTWSD to minimize the carrier's expected total cost. Recently, Zhang et al. [131] studied the VRPTWSD by taking account of on-time delivery issue and different recourse policies. Three probabilistic models have been proposed for the VRPTWSD to address on-time delivery from different perspectives.

Recent literature offers a few contributions to the VRPSPD with stochastic demands. Dimitrakos and Kyriakidis [32], Minis and Tatarakis [75], and Pandelis et al. [81] studied the single-vehicle case of the VRPSPD with stochastic demands, such that the vehicle followed a predefined customer sequence and returned to the depot whenever failure occurred. Wollenberg et al. [126] presented a two-stage stochastic programming model with recourse as well as an L-shaped based algorithm to the single vehicle case of the VRPSPD with stochastic demands, while the visiting sequence of a customer needs to be decided. For the multiple-vehicles case, Zhu and Sheu [133] investigated the generation of a priori routes for vehicles that simultaneously pick up and deliver items with stochastic demands. A failure-

specific cooperative recourse strategy is proposed to tackle the potential failures. Under this strategy, a pair of vehicles is scheduled to serve customers sequentially: a lead vehicle and a partner vehicle. If the lead vehicle fails, it resumes its route to serve its remaining assigned customers. The partner vehicle assists the lead vehicle in completing its service after finishing its own assigned customer service. Therefore, the traveling cost can be reduced by vehicle cooperation. But this strategy is not applicable to our problem. In the time-dependent laundry routing with stochastic pickup demands, each customer has a chosen time window to be served. Untimely response to the failure influences customer service satisfaction level and consequently reduces market share. Furthermore, the existing literature considers only the single commodity situation. In contrast, our problem considers to pickup and delivery of multiple commodities because most restaurants and hotels prefer to use textiles with their own logos (such as uniforms). A logo provides a unique identity, which plays a vital role in the amplification of a hotel or restaurant's brand-recognition value.

Besides, the literature related to VRPSPD with other uncertainties is also limited. Zhang et al. [132] studied the VRPSPD with stochastic travel time. However, they did not take into account the time window constraints, and a new scatter search approach was proposed for it by incorporating a new chance-constrained programming method. Shi et al. [101] modeled a home health care routing problem as VRP-SPD-TW with stochastic travel and service times. A stochastic simulation method and a simulated annealing algorithm were integrated to solve this problem. In Table 5.1, there is a summary of surveyed papers dealing with the VRPTWSD and the VRPSPD with uncertainties.

Table 5.1: Summary of papers dealing with the VRPTWSD and the VRPSPD with uncertainties

Author	Time Windows	Pickup and Delivery	Stochastic/TD Travel time	Recourse Action	Probability Distribution	Evaluation
Chang [16]	✓	✗	✗/✗	DTD	Discrete uniform	Analytically
Lei et al. [65]	✓	✗	✗/✗	DTD	Poisson	Analytically
Zhang et al. [131]	✓	✗	✗/✗	DTD, PR	Discrete uniform	Analytically
Dimitrakos and Kyriakidis [32]	✗	✓	✗/✗	DTD	Normal	Analytically
Minis and Tatarakis [75]	✗	✓	✗/✗	DTD	Normal	Analytically
Pandelis et al. [81]	✗	✓	✗/✗	DTD	Normal	Analytically
Wollenberg et al. [126]	✗	✓	✗/✗	DTD	Normal	Analytically
Zhu and Sheu [133]	✗	✓	✗/✗	CP	Discrete uniform	Analytically
Zhang et al. [132]	✗	✓	✓/✗	DTD	Discrete uniform	Analytically
Shi et al. [101]	✓	✓	✓/✗	CCP	Discrete uniform	Simulation
This chapter	✓	✓	✗/✓	OS, DTD, RO, PR	Poisson	Simulation

Additionally, limited literature has addressed time-dependent routing with uncertainty. Most of it considers time constraints in the context of stochastic time-dependent travel times. However, several assumptions are needed to make the analytical calculation computationally tractable. For instance, the distribution of the arrival time is assumed to be known before

the planning (e.g., gamma distribution by Taş et al. [111]), which may not hold in real life. Therefore, in this chapter, we consider only time-dependent travel time. Lecluyse et al. [64] applied queueing theory to consider a vehicle routing problem with stochastic time-dependent travel times (VRP-STT). A tabu search procedure was developed to minimize the weighted sum of the mean and the standard deviation of the total travel time. Nahum and Hadas [78] developed a chance-constrained model for the VRP-STT. They designed an efficient savings algorithm. In this procedure, simulation was used to estimate each route's probability of being the quickest. The objective function values of the solutions yielded by this algorithm were on average 10% higher than optimal solutions. Taş et al. [111] studied a vehicle routing problem with stochastic time-dependent travel times and soft time windows. A mathematical model was developed to consider both efficiencies for service and reliability for customers. A tabu search and an ALNS were both derived to solve the problem. Verbeeck et al. [118] considered the stochastic time-dependent orienteering problem with time windows (S-TDOP-TW), a stochastic version of an ant colony optimization algorithm was specifically developed for this problem. Moreover, three prominent complications and an estimation algorithm were also introduced to calculate the departure and arrival time distribution. Xiang et al. [127] studied a dynamic dial-a-ride problem considering stochastic and time-dependent travel speed regarding the transport of the elderly, disabled people or patients other than freight requests (DARP-STT). A fast heuristic was introduced, and several intensive computational simulations were used to evaluate the solutions. Another similar work was presented by Schilde et al. [98], and historical accident data were used to deduce the stochastic deviations of the time-dependent travel speeds. Moreover, a dynamic stochastic variable neighborhood search and a multiple scenario approach were introduced. To the best of our knowledge, this chapter is the first attempt to consider the stochastic demands under the context of a deterministic time-dependent routing problem. In Table 5.2, there is a summary of all the problem variants discussed above.

Table 5.2: Summary of related problems

Problem type	Time Windows	Pickup and Delivery	Stochastic Travel time	Stochastic Demand	Time-dependent	Recourse Action
VRPSD				✓		DTD, PR
VRPTWSD	✓			✓		DTD
VRPSPD		✓				
VRP-SPD-TW	✓	✓				
VRP-STT			✓		✓	
DARP-STT		✓	✓		✓	
S-TDOP-TW	✓		✓		✓	
TDLRPSPD	✓	✓		✓	✓	OS, DTD, RO, PR

5.3 Problem description

The problem discussed in this chapter is defined on a graph $G = (V, A)$, where $V = N \cup \{0, n+1\}$ is the set of nodes and A is the set of all feasible arcs. Nodes 0 and $n+1$ denote the origin and destination depot of the vehicle, and other nodes $N = \{1, ..., n\}$ represent the requests. These requests are served by a fleet of homogeneous vehicles with capacity Q. A service time s_i is incurred when visiting a request $i \in N$. Each request $i \in N$ is associated with a time window $[e_i, l_i]$ during which the delivery and pickup should take place. The vehicle has to wait until time e_i but is allowed to arrive later than time l_i. If a driver arrives at node i later than its latest service start time l_i, a proportional penalty cost must be paid and the parameter β is the cost for one unit of time window violation. Moreover, each request has two types of demands: a non-negative and deterministic volume d_i to be delivered and a non-negative and stochastic volume $\tilde{p}_i$ to be collected. $\tilde{p}_i$ is assumed as an independent random variable with a Poisson distribution. The actual pickup demand of each request i is revealed only when a vehicle arrives. For each arc $(i, j) \in A$, a speed profile is associated, that divides the planning horizon into time zones, and each time zone has a constant speed. We assume that these time-dependent travel speeds are known from historical data. Let $\tau_{ij}(t)$ represent the time-dependent travel time for each arc $(i, j) \in A$, which depends on the departure time t at node i. The set of feasible arcs can be described as $A = \{(i, j) \in V \times V : i \neq j$ and $e_i + s_i + \tau_{ij}(e_i + s_i) \leq l_j\}$.

In this problem, a failure may occur when a vehicle arrives at a request location and the revealed pickup demand minus its delivery demand exceeds the remaining vehicle capacity. This causes a failure that requires a recourse action. In this paper, we consider the following four recourse opportunities:

- Outsourcing (R_0): In this case, a vehicle starts from the origin depot 0, serves its assigned requests following a predefined sequence, and returns to the destination depot $n+1$. Whenever, this vehicle is out of its capacity to perform the pickup operation, it will only perform the delivery. The unmet pickup demand is assumed to be served by an outsourced service at a certain cost, which depends on the amount of unmet pickup demand (see Figure 5.1(a));
- Detour-to-depot (R_1): Assume that if a vehicle arrives at a request n_3, it first satisfies delivery demand $D3$. If its remaining capacity is not enough to pick up the revealed pickup demand $P3$ of request n_3, the vehicle returns to the depot to unload its collected pickup quantities. After unloading at the depot, the vehicle resumes service at the

request n_3 where the failure occurred (see Figure 5.1(b));

- Reschedule the customers (R_2): Similar to R_0, requests are visited following the predefined sequence. If, when serving a request, its revealed pickup demand exceeds the remaining capacity of the vehicle after satisfying the delivery demand of this request, then the pickup demands will be completely skipped. Once all the requests are visited by following the predefined sequence, the requests with unmet pickup demands will be re-visited (see Figure 5.1(c)). More specifically, those requests are inserted after the given route in a greedy order by favoring customers that increase the sum of the routing cost and recourse cost the least;

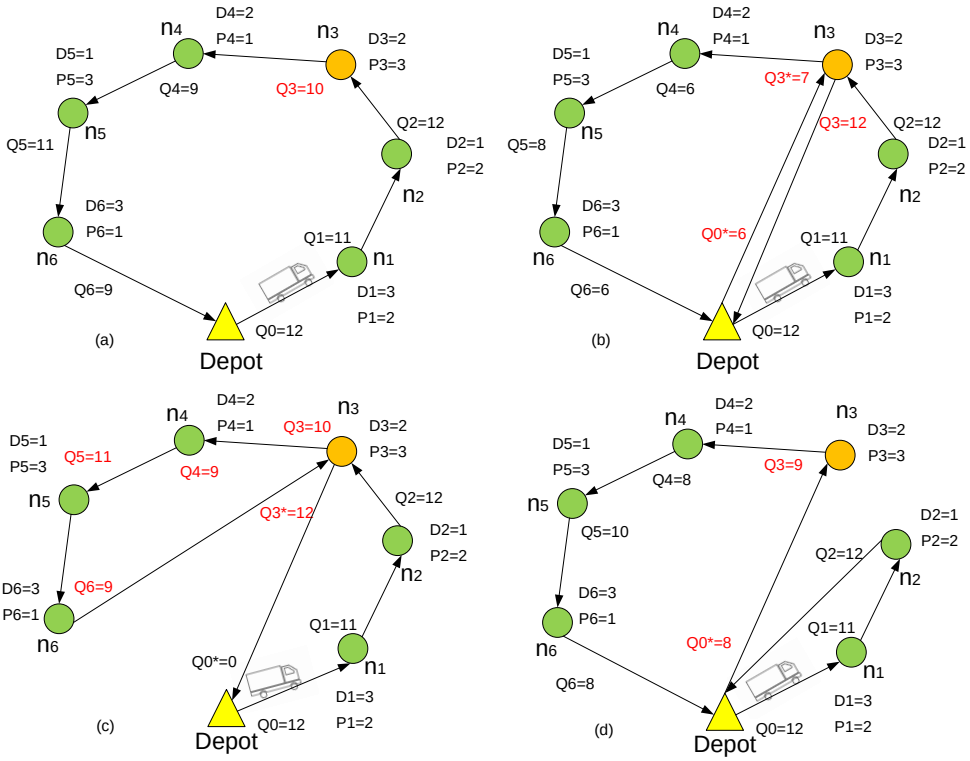

Figure 5.1: An example of different recourse actions

- Preventive policy (R_3): Inspired by Yang et al. [129], a preventive replenish policy is also developed for our problem. Under the preventive replenish policy, a vehicle may return to the depot to unload before a failure occurs. After serving one customer (see Figure 5.1(d)), the vehicle decides whether or not to return to the depot to unload based on the amount of its remaining capacity and the estimated pickup demands of the next

customer. If this remaining capacity is smaller than a certain value (e.g., 25^{th} percentile of the pickup demands distribution in a scenario sample set.), then the vehicle returns to the depot to unload; otherwise it proceeds directly to the next customer along the planned route. Similar to R_1, the vehicle also returns to the depot to unload its collected pickup quantities, if a failure occurs after the pickup demand of a request is observed.

5.4 A two-stage stochastic model

In this section, we will introduce a two-stage stochastic model to capture the stochastic elements of the TDLRPSPD. Several assumptions are made in this section: (1) requests' pickup demands are the only stochastic elements, and other elements such as requests' delivery demands and locations of requests are deterministic; (2) a request's pickup demand can be split if a failure happens; (3) a request's actual pickup demand is only revealed when the vehicle arrives at that request's location.

Conceptually, planning decisions are made in two stages. The first stage decides upon the serving sequence of each route by its own transport resource. Its objective is to plan a set of a priori pickup and delivery routes which minimize the expected routing cost. In the second-stage, when realizations of the random variables are revealed, recourse actions are used to compute the additional cost that must be added to the set of routes derived in the first stage. Moreover, in this problem, each request $i \in N$ must be visited exactly once in the intended routing plan. However, if unexpected high pickup loads are observed, recourse actions need to take place and a second visit to a customer is allowed.

First, let $\boldsymbol{\omega} = (p_i^{\omega})_{i \in N}$ be a random variable vector, where p_i^{ω} is the stochastic pickup demand for request i. p_i^{ω} is defined with a probability distribution function and the real demand value is only known when the vehicle arrives at the customer. Let $\overline{p}_i$ denote the expected pickup demands of request i. As a result, a *route failure* may occur along a route if the total collected pickup demands plus remaining delivery demands exceed the vehicle capacity.

In addition, throughout this chapter, the following variables are used:

- t_i^k : If vehicle k serves request node i, the departure time of vehicle k at node i;
- y_i^k : If vehicle k serves request node i, the value is 1; otherwise, it is equal to 0;
- x_{ij}^k: If vehicle k travels directly from node i to node j, the value is 1; otherwise, it is equal to 0;

- D_i^k: If vehicle k serves request node i, the amount of remaining delivery volume carried by vehicle k when departing from node i;

- P_i^k: If vehicle k serves request node i, the amount of the collected pickup volume carried by vehicle k when departing from node i.

Then the two-stage stochastic model can be formulated as follows:

$$\min_{\boldsymbol{x}}\{\sum_{k\in K}[c_t(t_{n+1}^k - t_0^k) + Zy_0^k] + \beta\sum_{k\in K}\sum_{i\in N}[(t_i^k - (l_i+s_i)y_i^k]^+ + E[T(\boldsymbol{x},\boldsymbol{\omega})]\} \tag{5.1}$$

subject to

$$\sum_{j\in N} x_{0j}^k = y_0^k \qquad \forall k \in K, \tag{5.2}$$

$$\sum_{i\in N} x_{i,n+1}^k = y_{n+1}^k \qquad \forall k \in K, \tag{5.3}$$

$$y_0^k \le 1 \qquad \forall k \in K, \tag{5.4}$$

$$y_0^k = y_{n+1}^k \qquad \forall k \in K, \tag{5.5}$$

$$\sum_{k\in K}\sum_{i\in N\cup\{0\}} x_{ij}^k = \sum_{k\in K} y_j^k \qquad \forall j \in N, \forall k \in K, \tag{5.6}$$

$$\sum_{k\in K} y_j^k = 1 \qquad \forall j \in N, \forall k \in K, \tag{5.7}$$

$$\sum_{i\in N\cup\{0\}} x_{if}^k - \sum_{j\in N\cup\{n+1\}} x_{fj}^k = 0 \qquad \forall f \in N, \forall k \in K, \tag{5.8}$$

$$D_i^k + P_i^k \le Q \qquad \forall i \in V, \forall k \in K, \tag{5.9}$$

$$P_0^k = D_{n+1}^k = 0 \qquad \forall k \in K, \tag{5.10}$$

$$D_0^k = \sum_{i\in N} y_i^k d_i \qquad \forall k \in K, \tag{5.11}$$

$$P_{n+1}^k = \sum_{i\in N} y_i^k \overline{p}_i \qquad \forall k \in K, \tag{5.12}$$

$$D_i^k - d_j \le D_j^k + M(1 - x_{ij}^k) \qquad \forall i,j \in V, \forall k \in K, \tag{5.13}$$

$$P_i^k + \overline{p}_j \le P_j^k + M(1 - x_{ij}^k) \qquad \forall i,j \in V, \forall k \in K, \tag{5.14}$$

$$t_i^k + \tau_{ij}(t_i^k) + s_j \le t_j^k + M(1 - x_{ij}^k) \qquad \forall i,j \in V, \forall k \in K, \tag{5.15}$$

$$t_i^k \ge (e_i + s_i)y_i^k \qquad \forall i \in V, \forall k \in K, \tag{5.16}$$

$$D_i^k, P_i^k \geq 0 \qquad \forall i \in V, \forall k \in K, \tag{5.17}$$

$$x_{ij}^k, y_i^k \in \{0,1\} \qquad \forall i,j \in V, \forall k \in K \tag{5.18}$$

The objective function (5.1) minimizes the sum of the total expected routing cost and the recourse costs. In the recourse function, $\boldsymbol{x}$ is the vector of routing decisions and the realization of random pickup demands is described by $\boldsymbol{\omega}$. Constraints (5.2)-(5.5) guarantee that the route of each vehicle k starts from the origin depot and ends at the destination depot if vehicle k is used. Constraints (5.6)-(5.7) ensure that every request is served at most once in the first stage. Constraints (5.8) are classical flow conservation constraints. Constraints (5.9) ensure that the sum of the remaining delivery load plus the expected collected pickup load of vehicle k is lower than the vehicle capacity, when departing from node i. Constraints (5.10) ensure each vehicle leaves the depot when the pick-up load is null and returns to the depot with all their deliveries distributed. Constraints (5.11) guarantee that each vehicle leaves the depot fully loaded with the products to be distributed. Constraints (5.12) guarantee that when vehicles return to the depot they are fully loaded with the expected pickup quantities. Constraints (5.13)-(5.14) establish that if arc (i,j) is visited by vehicle k, then the quantity to be delivered by the vehicle has to decrease by d_j while the quantity pickup has to increase by $\overline{p}_j$. Constraints (5.15) ensure its departure times from locations respect travel and service times. Constraints (5.16) ensure that the departure time at each node of the request i is larger than its earliest service start time e_i plus its service time s_i. Finally, Constraints (5.17) are non-negative conditions, while Constraints (5.18) are binary constraints.

However, the two-stage stochastic model with the preventive policy (R_3) is slightly different from its counterparts with other recourse actions. This is because, for each request i, we use an estimate of the pickup demands $\hat{p}_i$ rather than using $\overline{p}_i$. Under the preventive policy (R_3), a vehicle can return to the depot to unload, if the amount of its remaining capacity is smaller than the estimated pickup demands of the next customer. Let $D_0^k(i)$ be the amount of remaining delivery volume carried by vehicle k when vehicle k departs from request i and returns to the depot 0 to unload. As a result, we formulate the problem as follows:

$$\min_{\boldsymbol{x}}\{\sum_{k\in K}[c_t(t_{n+1}^k - t_0^k) + Zy_0^k] + \beta \sum_{k\in K}\sum_{i\in N}[(t_i^k - (l_i + s_i)y_i^k]^+ + E[T(\boldsymbol{x},\boldsymbol{\omega})]\}$$

subject to (5.2)-(5.3), (5.6)-(5.18), with the following extra constraints:

$$y_0^k \leq 2 \qquad \forall k \in K, \tag{5.19}$$

$$y_{n+1}^k \leq 1 \qquad \forall k \in K, \tag{5.20}$$

$$D_i^k \leq D_0^k(i) + M(1 - x_{i0}^k) \qquad \forall i \in V, \forall k \in K, \tag{5.21}$$

$$Q - (D_i^k + P_i^k) \geq d_j + \hat{q}_j + M(1 - x_{ij}^k) \qquad \forall i, j \in V, \forall k \in K, \tag{5.22}$$

$$y_0^k \in \{0, 2\} \qquad \forall i \in V, \forall k \in K. \tag{5.23}$$

Constraints (5.19)-(5.20) ensure that, in the first stage, origin depot 0 is not allowed to be visited more than twice but the destination depot $n+1$ can only be visited at most once. Constraints (5.21) guarantee that, if vehicle k travels back to the origin depot 0, the amount of its delivery volume when departing from 0 equals its counterpart at its previous visited node i. Constraints (5.22) ensure that, after serving node i, the remaining capacity of vehicle k is larger than the sum of delivery demands d_j and estimated pickup demands $\hat{p}_j$.

The second-stage recourse cost $E[T(\boldsymbol{x}, \boldsymbol{\omega})]$ depends on the considered recourse action. If a failure occurs, skipping some requests' pickup demands is allowed by using outsourcing policy (R_0). We introduce a new decision variable λ_i^{ω}, which equals 1 if the pickup demands of request i are skipped and equals 0 if the pickup demands of request i are collected in the second stage solution. Let α be the penalty per unit of pickup demands that are not collected. The second stage cost of using R_0 can be expressed as:

$$T(\boldsymbol{x}, \boldsymbol{\omega}) = \alpha \sum_{i \in N} p_i \lambda_i^{\boldsymbol{\omega}} \tag{5.24}$$

While, for the rest of recourse actions $(R_1, R_2,$ and $R_3)$, a request i served by vehicle k is allowed to be visited twice, if failures occur. A real departure time $t_i^{k\boldsymbol{\omega}}$ is derived in the second stage. Moreover, for each vehicle k, the real departure time from the depot $t_0^{k\boldsymbol{\omega}}$ in the second stage is equal to the expected departure time t_0^k. Let $R(\boldsymbol{x}, \boldsymbol{\omega})$ be the travel cost increase on $\sum_{k \in K} c_t(t_{n+1}^k - t_0^k)$, which can be expressed as:

$$
\begin{aligned}
R(\boldsymbol{x}, \boldsymbol{\omega}) &= \sum_{k \in K} c_t((t_{n+1}^{k\boldsymbol{\omega}} - t_0^{k\boldsymbol{\omega}}) - (t_{n+1}^{k} - t_0^{k})) \\
&= \sum_{k \in K} c_t((t_{n+1}^{k\boldsymbol{\omega}} - t_0^{k}) - (t_{n+1}^{k} - t_0^{k})) \\
&= \sum_{k \in K} c_t(t_{n+1}^{k\boldsymbol{\omega}} - t_{n+1}^{k}) \qquad (5.25)
\end{aligned}
$$

Similarly, the total penalty cost for time window violations for all requests in the second stage, can be expressed as $\beta \sum_{k \in K} \sum_{i \in N} [(t_i^{k\boldsymbol{\omega}} - (l_i + s_i) y_i^k]^+$. Let $W(\boldsymbol{x}, \boldsymbol{\omega})$ be the increase on the total penalty cost for time window violations, which can be expressed as:

$$
W(\boldsymbol{x}, \boldsymbol{\omega}) = \beta \sum_{k \in K} \sum_{i \in N} \{[(t_i^{k\boldsymbol{\omega}} - (l_i + s_i) y_i^k]^+ - [(t_i^k - (l_i + s_i) y_i^k]^+\} \qquad (5.26)
$$

Therefore, the second stage cost of using R_1, R_2 or R_3 is then:

$$
T(\boldsymbol{x}, \boldsymbol{\omega}) = \min\{R(\boldsymbol{x}, \boldsymbol{\omega}) + W(\boldsymbol{x}, \boldsymbol{\omega})\} \qquad (5.27)
$$

5.5 Solution methodology

In this section, we describe our solution methodology for the time-dependent laundry routing with stochastic pickup demands.

5.5.1 The sample average approximation method

The sample average approximation method (SAA) is a well-known sampling-based technique to solve stochastic discrete optimization problems. Instead of summing over all possible scenarios, we use Monte Carlo Sampling to sample a subset of scenarios according to the given probabilities. Following the procedure of Kleywegt et al. [60], the proposed problem is repeatedly solved M times by the SAA method in order to obtain statistical estimates of upper and lower bounds of the objective value and estimates of the variances of these bounds. More specifically, in each iteration, a set of scenarios Ω is introduced to estimate

$E[T(\boldsymbol{x}, \boldsymbol{\omega})]$ and to approximate the optimal solution by the average objective value over those scenarios $\frac{1}{|\Omega|}\sum_{\omega\in\Omega} T(\boldsymbol{x}, \omega)$. Then, a new set $\Omega^{'}$ of scenarios is used to evaluate the optimality gap between the solution found before and the true optimal solution of the associated SAA problem. Normally, the sample size $\Omega^{'}$ is chosen much larger than Ω ($\Omega \ll \Omega^{'}$). Since Ω is used in the optimization to find the candidate solution, having a large set slows down the procedure a lot, and $\Omega^{'}$ is just used to evaluate the objective function value of the candidate solution, which is in general much faster.

After solving these associated M separate SAA problems, M objective function values, say $v^1, ..., v^M$ and M candidate solutions $\hat{x}^1 ... \hat{x}^M$ are provided. Therefore, the average of the optimal objective values obtained in M iterations, described as $\hat{x}_\Omega$, and its corresponding variance is shown as follows:

$$\hat{v}_\Omega^M = \frac{1}{M}\sum_{m=1}^{M} v_\Omega^m \tag{5.28}$$

$$\sigma^2_{\hat{v}_\Omega^M} = \frac{1}{M(M-1)}\sum_{m=1}^{M}(v_\Omega^m - \hat{v}_\Omega)^2 \tag{5.29}$$

It is well-known that $\hat{v}_\Omega^M$ provides a statistical estimate for a lower bound on the optimal value of the original problem. Moreover, for any feasible solution $\hat{x} \in X$, its objective function value provides an upper bound that can be estimated using the set $\Omega^{'}$ of newly generated scenarios. This estimated upper bound is denoted as $\hat{v}_{\Omega'}(\hat{x})$. Thus, its value and corresponding variance are shown as below:

$$\hat{v}_{\Omega'}(\hat{x}) = \frac{1}{|\Omega'|}\sum_{\omega\in\Omega'} v_\omega(\hat{x}) \tag{5.30}$$

$$\sigma^2_{\Omega'}(\hat{x}) = \frac{1}{|\Omega'|(|\Omega'-1|)}\sum_{\omega\in\Omega'}(v_\omega(\hat{x}) - \hat{v}_{\Omega'}(\hat{x}))^2 \tag{5.31}$$

The quality of $\hat{x}$ can be evaluated by calculating the SAA gap $\epsilon_{SAA}(\Omega, \Omega^{'})$ and its associated variance $\sigma^2_{\epsilon_{SAA}(\Omega,\Omega')}$ as follows:

$$\epsilon_{SAA}(\Omega, \Omega^{'}) = \hat{v}_{\Omega} - \hat{v}_{\Omega'}(\hat{x}) \tag{5.32}$$

$$\sigma^2_{\epsilon_{SAA}(\Omega,\Omega')} = \sigma^2_{\hat{v}_{\Omega}} + \sigma^2_{\Omega'}(\hat{x}) \tag{5.33}$$

If the optimality gap and the variance of the gap estimator are sufficiently small, the generated solution is accepted. The detailed SAA procedure is described as follows (Algorithm 6):

Algorithm 6: The SAA procedure

input : Number of iterations M, initial sample size for solution generation $|\Omega|$ and sample size solution evaluation for $|\Omega^{'}|$ such that $|\Omega^{'}| \gg |\Omega|$
output: Candidate solution $\hat{x}^*$

1 *Generate a sample set* $\Omega^{'}$.
2 **for** *Iteration* $m = 1 \rightarrow M$ **do**
3 *Generate a sample set* Ω *and apply ALNS to solve the SAA problem by considering all the scenarios* $\omega \in \Omega$. *The objective value* v^m_{Ω} *and the solution* $\hat{x}^m$ *are obtained.*
4 *Determine the upper bound* $\hat{v}^m_{\Omega}$ *of the solution* $\hat{x}^m$ *and its associated variance* $\sigma^2_{\hat{v}^m_{\Omega}}$ *by (5.28) and (5.29) respectively.*
5 *Update the solution* $\hat{x}^*$ *that has the best value of the upper bound* $\hat{v}_{\Omega}$ and variance $\sigma^2_{\hat{v}_{\Omega}}$ after m iterations.
6 *Update the lower bound* $\hat{v}_{\Omega'}(\hat{x}^*)$ *and its associated variance* $\sigma^2_{\Omega'}(\hat{x}^*)$ *using (5.30) and (5.31) respectively.*
7 *Calculate the SAA gap* $\epsilon_{SAA}(\Omega, \Omega^{'})$ *and its corresponding variance* $\sigma^2_{\epsilon_{SAA}(\Omega,\Omega')}$ by (5.32) and (5.33).
8 *Return the best solution* $\hat{x}^*$

5.5.2 The ALNS heuristic

We tailored the ALNS algorithm described in Chapter 4 to solve the SAA problems of the time-dependent laundry routing problem with stochastic pickup demands, as discussed in Section 5.5.1 (for more details, see Algorithm 7). In particular, most of the effort in tailoring the algorithm is the objective function evaluation. The performance of the algorithm is enhanced by adapting several operators in a stochastic setting.

We introduced seven removal operators in Chapter 4. Except for Least Profit Removal (LPR) and Worst Travel Duration Removal (WTR), the rest of them can be used for this problem. These five removal operators can be categorized as deterministic removal operators and stochastic removal operators. The deterministic removal operators include Random Removal (RR), Early Departure Removal (EDR), and Route Removal (ROR). For more details, the reader is referred to Chapter 4. The stochastic removal operators consist of Shaw Removal (SR) and

Worst Removal (WR).

- **Worst Removal (WR)**: This operator removes several requests one at a time by determining which removal has the largest positive effect on the expected objective function value. Given a solution, the cost of a node i is the difference in the objective function between the current solution with i and the same solution without serving i.
- **Shaw Removal (SR)**: The objective of the SR operator is to remove nodes that are similar in terms of certain aspects. The algorithm randomly selects a node i and adds it to $\mathcal{L}$. Let $l_{ij} = -1$ if node i and node j are in the same vehicle route, and $l_{ij} = 1$ otherwise. This operator selects the node $j^* = argmin_{j \in \mathcal{N}}\{\Pi_1 d_{ij} + \Pi_2 |t_i - t_j| + \Pi_3 l_{ij} + \Pi_4 |d_i - d_j| + \Pi_5 |\bar{p}_i - \bar{p}_j|\}$, where Π_1–Π_5 are normalized weights, $\bar{p}_i$ and $\bar{p}_j$ is the expected pickup demands of node i and j given the probability distribution, respectively.

Algorithm 7: Pseudo-code of the ALNS with SA

input : Removal operators D, insertion operators Ψ, initial temperature T, cooling rate κ
output: A feasible solution X_{best}

Generate an initial solution by using the Greedy insertion algorithm
Initialize probability for each destroy operator $d \in D$ and each insertion operator $\psi \in \Psi$
$X_{current} \leftarrow X_{best} \leftarrow X_{init}$
repeat
  Choose a removal operator $d^* \in D$ with probability $\xi_d / \sum_{i=1}^{|D|} \xi_i$ and apply it to $X_{current}$ to get the partially destroyed solution X_d
  Choose an insertion operator $\psi^* \in \Psi$ with probability $\xi_\psi / \sum_{i=1}^{|\Psi|} \xi_i$ and apply it to X_d to get a new solution X_{new}
  if *obj(X_{new}) > obj($X_{current}$)* **then**
    $X_{current} \leftarrow X_{new}$
    if *obj($X_{current}$) > obj(X_{best})* **then**
      $X_{best} \leftarrow X_{current}$
  else
    Let $\nu \leftarrow e^{(obj(X_{new}) - obj(X_{current}))/T}$
    Generate a random number $\epsilon \in [0, 1]$
    if $\epsilon < \nu$ **then**
      $X_{current} \leftarrow X_{new}$
  $T \leftarrow \kappa T$
  Update probabilities using the adaptive probability adjustment procedure
  $j \leftarrow j + 1$
until *the maximum number of iterations is reached or no improvement is found after a predefined number of iterations*

Instead of evaluating the objective function value, all the insertion operators, presented in Chapter 4, are naturally adapted to compute the changes in routing and recourse costs over $|\Omega|$ scenarios at every iteration. In particular, for a given routing solution, we calculate the change in the routing and recourse cost of the modified solution include i and its counterpart

where i is not served (removed) on the sample set Ω.

5.6 Computational results

This section presents the results of computational experiments performed to assess the performance of the proposed methodology. We first describe the instance characteristics and the parameters used in Section 5.6.1. In Section 5.6.2, we test the preventive policy with different expected pickup demands. In Section 5.6.3, we assess the value of stochastic information and compare different recourse policies. The impact of different outsourcing cost is presented in Section 5.6.4.

5.6.1 Data and experimental setting

To the best of our knowledge, this chapter is the first study of this specific vehicle routing problem in the laundry business. There are no benchmark instances to evaluate the performances of our SAA-based approaches. Therefore, we adapt the existing benchmark instances from Wang and Chen [124] to consider stochastic pickup demands and time-dependent travel times. These instances are generated by revising the benchmark problems of the vehicle routing problem with time windows from Solomon [104]. They are divided into six classes that differ by the geographical distribution of the customers (R1, R2, C1, C2, RC1 and RC2). The customers are clustered in the C type instances, and randomly distributed in the R type instances. In RC type instances, the customers are partly clustered and partly randomly distributed. Moreover, Type 1 instances have narrow time windows and Type 2 instances have large time windows. All instances have 100 customer vertices and we also generate 25 and 50 customer instances by just considering the first 25 and 50 vertices in the benchmark instances. In total, we have 54 instances for experiments. Moreover, the coordinates and delivery demand of each vertex are the same as in the benchmark instances from Solomon [104] and Wang and Chen [124]. The pickup demand of each vertex is generated according to a Poisson distribution having a mean value equal to the pickup demand from Wang and Chen [124]. However, the vehicle capacity in the instances from Wang and Chen [124] was too large for our problem, and the failure probability was too small so that failure would rarely occur. Suppose, for each vertex i, the mean value of its pickup demand is denoted as $\bar{p}_i$. The realization of its pickup demand is rarely over $2\bar{p}_i$ in the generated scenarios set, because it follows the Poisson distribution. Therefore, we have reduced the capacity of the vehicle while ensuring that $\max_{i \in N}\{2\bar{p}_i\} \leq Q$.

Road congestion is handled by a so-called speed model which consists of different speed profiles. It is used to determine the travel time between two nodes at a specific departure time. This speed model for the laundry routing problem with stochastic pickup demands is based on the speed model of Verbeeck et al. [117] for the TDOP and Dabia et al. [25] for the TDVRPTW. Furthermore, without loss of generality, we assume that breakpoints are the same for all speed profiles as congestion tends to happen around the same time regardless of the speed profiles' type. Each speed profile has four non-overlapping time periods with constant speed, reflecting two congested periods and two periods with normal traffic conditions. To generate an instance, we consider five speed profiles (see Table 5.3) and randomly assign one of these speed profiles to each arc.

Table 5.3: Speed Profiles

Congestion description	Morning peak	Normal	Evening peak	Normal
Time periods	7 am-9 am	9 am-5 pm	5 pm-7 pm	7 pm-9 pm
1.SS	0.5	0.81	0.5	0.81
2.NSMP	0.67	1.33	0.88	1.33
3.NSEP	0.88	1.33	0.67	1.33
4.FSTP	0.85	1.5	0.85	1.5
5.HS	1.0	2.0	1.0	2.0

The parameters used in the computational experiments are given in Table 5.4. The ALNS-specific parameters are assigned the same values as in Chapter 4 for solving the deterministic TDPPDPTW. The settings used also worked well on the laundry routing problem with stochastic pickup demands. We assume a driving time unit equals 1 unit of cost. Moreover, we assume that every unit of outsourced pickup demand corresponds to 100 units of violation cost. Moreover, we consider 40 units of cost per unit of time window violation.

Table 5.4: Parameters used in the methodology

Notation	Definition	Value
$\|\Omega^{'}\|$	The size of the large set of scenarios	10000
$\|\Omega\|$	The size of the small set of scenarios	60
α	Cost for outsourcing	100
β	Cost for time window violation	40
Z	fix cost of vehicle	100
M	Number of SAA iterations	10

Kleywegt et al. [60] mentioned that both, $|\Omega^{'}|$ and $|\Omega|$, need to be chosen in a way to consider the trade-off between solution quality and computational complexity of the SAA problems.

Hence, to evaluate the SAA gap in our computational experiments, we used $|\Omega^{'}|$ = 10,000. This value proved to be a good estimate for a related routing problem (i.e., Production Routing Problem under Demand Uncertainty) (Adulyasak et al., 2015). In addition, we consider $|\Omega|$ = 60. Li et al. [67] shows that this value is a good trade-off between solution quality and CPU time of the SAA problem, considering a Dial-a-Ride Problem with Stochastic Travel Times and Stochastic Customers.

5.6.2 The test of preventive policy using different expected pickup demands

Under the preventive policy, the vehicle returns to the depot if the probability for a failure at the next customer is higher than a certain percentage. For each customer i, we take the 5^{th} percentile, 10^{th} percentile, 25^{th} percentile, 50^{th} percentile and 75^{th} percentile of the pickup demand distribution in the scenario sample set $|\Omega^{'}|$ as the expected pickup demands, respectively. If the remaining capacity is below the difference of the expected pickup demands and delivery demands, the driver returns directly to the depot and then resumes the service along the planned route. The comparative results are presented in Table 5.5. The columns "Obj." describe the costs of the objective function values. We also report the CPU time in seconds in the $Time(s)$ columns. It shows that using the 25^{th} percentile of the pickup demand distribution (i.e. the probability of failure should be larger than 75%) performs the best in most of the 25 requests instances, except instances cdp102-25, cdp202-25 and rcdp202-25. In contrast the 75^{th} percentile of the pickup demands value performs the worst. This indicates that the preventive policy becomes ineffective, if the expected pickup demands are too conservative (in the 75th percentile you return to the depot if there is a 25% probability of failure). The driver then would return to the depot more often than usual, which would be unnecessary in most of the scenarios. In addition, preventive policy also becomes ineffective, if the expected pickup demands underestimated the real pickup demands (in the 5th percentile you return to the depot only if there is a 95% probability of failure).

Table 5.5: Comparison of using different expected pickup demands in preventive policy

	5^{th}		10^{th}		25^{th}		50^{th}		75^{th}	
Instances	Obj.	Time (s)	Obj.	Time (s)	Obj.	Time (s)	Obj.	Time (s)	Obj.	Time (s)
cdp101-25	2831.5	35.2	2831.8	34.9	2831.8	33.6	2833.3	41.3	2839.2	47.5
cdp102-25	2753.7	77.9	2753.9	86.3	2745.6	84.5	2757.9	84.0	2761.0	92.9
cdp103-25	2751.0	218.9	2750.4	216.4	2747.3	217.9	2770.6	238.0	2766.7	231.4
cdp201-25	3044.6	52.5	3043.6	22.4	3043.6	54.6	3047.3	27.2	3045.7	26.4
cdp202-25	2828.3	173.9	2828.3	167.5	2828.3	156.0	2828.0	200.6	2827.4	176.3
cdp203-25	2827.9	209.4	2827.9	225.2	2827.9	225.4	2833.8	203.7	2828.2	280.6
rcdp101-25	1498.3	55.4	1490.7	49.1	1490.7	48.9	1513.3	76.7	1547.1	51.4
rcdp102-25	1442.7	71.4	1442.7	66.1	1450.3	68.3	1473.5	104.3	1490.7	100.1
rcdp103-25	1494.2	98.1	1442.3	101.4	1424.7	94.4	1480.6	125.7	1489.8	122.9
rcdp201-25	1705.3	114.2	1705.3	125.1	1705.3	119.7	1761.5	147.4	1778.6	160.6
rcdp202-25	1531.9	149.5	1536.8	185.1	1278.2	167.9	1259.1	87.3	1271.5	100.7
rcdp203-25	1346.2	198.0	1344.8	193.9	1360.7	182.1	1397.3	189.1	1388.5	236.1
rdp101-25	1708.3	28.9	1708.3	29.2	1708.3	24.5	1708.3	34.0	1708.3	34.4
rdp102-25	1413.8	40.5	1413.8	43.9	1413.8	46.5	1416.6	50.3	1421.8	54.3
rdp103-25	1283.5	65.0	1248.4	61.5	1242.7	65.5	1243.1	82.7	1243.9	80.9
rdp201-25	1382.6	85.4	1402.6	74.9	1362.6	87.5	1402.6	90.9	1482.1	84.5
rdp202-25	1245.1	210.3	1245.1	196.4	1245.1	201.1	1265.3	240.5	1271.5	240.5
rdp203-25	1213.6	295.1	1210.4	319.7	1211.3	326.8	1238.5	311.5	1235.7	315.0
Average	1905.7	121.1	1901.5	122.2	1884.4	122.5	1901.7	129.7	1911.0	135.4

5.6.3 The value of a stochastic solution and recourse policy comparison

To assess the value of stochastic information, we obtain a stochastic solution X^s using our SAA framework, as well as a deterministic solution X^d, in which all uncertain pickup demands are replaced by their expected values. The average results over three runs of the SAA algorithm are presented. Detailed, instance-level results can be found in the Appendix in Tables 21-24. Next, we compare the routing costs (first stage), as well as the recourse costs (second stage) for both solutions. The expected recourse cost for a solution is obtained through Algorithm 2, using a large scenario sample size $|\Omega^{'}|$ =10000. We first compare X^d with X^s using recourse $R_0 - R_3$ on instances with different size. The average performance of all the instances with 25, 50 and 100 requests is summarized in Table 5.6. Then, we compare the average performance of different recourse actions on instances with different types (cdp, rcdp and rdp) in Table 5.7. The columns "Deterministic" summarize the results obtained from the feasible and deterministic solutions found by the ALNS proposed in Chapter 4. The columns "Stochastic" summarize the results obtained by applying the proposed SAA algorithm to the TDLRPSPD. We also show the percentage improvement in columns $Imp(\%)$ obtained by our SAA over the deterministic ALNS.

As can be observed in Table 5.6 and Table 5.7, the total costs of the deterministic solutions are significantly higher than the costs of the stochastic solutions. By using our proposed SAA approach, compared to the deterministic solutions, the overall average cost reduction of 54.7%, 54.1%, 54.2% and 50.9% can be realized by using R_0, R_1, R_2 and R_3, respectively. The differences in first-stage routing costs are small but the second-stage routing costs make a significant difference. The main changes stem from vehicle-customer assignment decisions and the sequence of service. Moreover, note that the first stage cost of the deterministic solution, x^d, is independent of the used recourse action. However, its second-stage cost changes due to its dependency on different recourse policies.

Table 5.6 shows that preventive policy (R_3) outperforms the other three recourse policies among the instances within 50 requests since the objective value of using R_3 is the lowest. In contrast, R_0 and R_1 are the worst policies for instances with 25 requests and 50 requests. However, for instances with 100 requests, the customer reschedule policy (R_2) outperforms the other three recourse policies. R_3 is the worst policy among the instances with 100 requests, which is an interesting finding.

Table 5.6: Average comparison of the results on the instances with different size

	Instances	Deterministic			Stochastic				
	Size	1^{st}	2^{nd}	Obj.	1^{st}	2^{nd}	Obj.	Time (s)	Imp (%)
R_0	25	1853.6	2750.1	4603.7	1911.7	25.5	1937.1	110.2	48.8
	50	3673.6	8132.6	11806.2	3882.2	75.3	3957.5	418.8	57.5
	100	6888.8	11802.8	18691.6	7283.2	172.9	7456	1679.8	57.7
	Aver.	4138.7	7561.8	11700.5	4359	91.2	4450.2	736.3	54.7
R_1	25	1853.6	2462.0	4315.6	1885.4	31.4	1916.8	42.2	40.2
	50	3673.6	11304.0	14977.6	3853.9	114.2	3968.2	528.5	58.6
	100	6888.8	16591.0	23479.8	7264.7	237.4	7502.0	2085.3	63.3
	Aver.	4138.7	10119.0	14257.7	4334.7	127.7	4462.3	885.3	54.1
R_2	25	1853.6	2148.6	4002.1	1898.0	31.7	1929.6	139.5	39.3
	50	3673.6	10002.0	13675.6	3851.8	94.7	3946.5	600.6	60.4
	100	6888.8	21862.1	28751.0	7190.4	252.4	7442.8	2157.7	62.8
	Aver.	4138.7	11337.6	15476.2	4313.4	126.2	4439.6	965.9	54.2
R_3	25	1853.6	2006.2	3859.8	1868.5	15.8	1884.4	122.5	37.2
	50	3673.6	10283.7	13957.3	3823.1	77.8	3901.0	559.3	55.6
	100	6888.8	14102.3	20991.1	7189.6	324.8	7514.3	2135.2	59.8
	Aver.	4138.7	8797.4	12936.0	4293.7	139.5	4433.2	939.0	50.9

Table 5.7 shows that R_1, R_2 and R_3 outperform the other three recourse policies among the C type instances, the RC type instances and R type instances, respectively. In contrast, R_2 performs the worst in C type instances and R type instances. For RC type instances, R_0 is the worst recourse policy.

We also confirm that preventive policy R_3 outperforms the policy R_1 and R_2, if the selected expected pickup demands are less conservative. In the extreme case, policy R_3 is degenerated to policy R_1, when the expected pickup demands are set as 0. Moreover, policy R_2 is more sensitive to time window violation cost. If the failure happens too early, instead of re-visiting after serving the rest of the customers, it is better to re-visit as soon as possible to decrease the time window violation cost.

Table 5.7: Average comparison of the results on the instances with different type

	Instances	Deterministic			Stochastic				
	Type	1^{st}	2^{nd}	Obj.	1^{st}	2^{nd}	Obj.	Time (s)	Imp (%)
R_0	C	6591.6	6525.4	13117.0	6741.3	39.6	6780.9	860.0	40.6
	RC	2997.0	10845	13842.0	3392.5	151.5	3544	662.1	70.7
	R	2827.4	5315.1	8142.5	2943.4	82.5	3025.8	686.2	52.7
	Aver.	4138.7	7561.8	11700.5	4359	91.2	4450.2	736.3	54.7
R_1	C	6591.6	8521.3	15112.9	6707.8	44.7	6752.5	1024.4	40.9
	RC	2997.0	15903.9	18900.9	3373.3	228.6	3601.9	847.8	70.1
	R	2827.4	5931.8	8759.2	2922.9	109.7	3032.6	783.8	51.3
	Aver.	4138.7	10119.0	14257.7	4334.7	127.7	4462.3	885.3	54.1
R_2	C	6591.6	21856.7	28448.3	6742.9	113.5	6856.3	1076.7	57.0
	RC	2997.0	7511.6	10508.7	3264.3	165.4	3429.7	877.0	58.0
	R	2827.4	4644.4	7471.8	2933.0	99.9	3032.9	944.1	47.5
	Aver.	4138.7	11337.6	15476.2	4313.4	126.2	4439.6	965.9	54.2
R_3	C	6591.6	6471.6	13063.2	6696.2	158.8	6855.0	1106.7	34.7
	RC	2997.0	14748.4	17745.4	3287.8	182.2	3470.0	851.2	68.2
	R	2827.4	5172.1	7999.6	2912.5	81.6	2994.1	859.1	49.1
	Aver.	4138.7	8797.4	12936.0	4298.8	140.9	4439.7	939.0	50.6

5.6.4 The impact of different outsourcing costs

To show the impact of different outsourcing costs, we also conducted experiments on all the 54 instances by using R_0 with $\alpha = 100$. A comparison between R_0 with two different α and the other three recourse policies with $\beta = 40$ is provided in Table 5.8. We provide the total number of instances in which the objective function value of the stochastic solution derived by policy R_0 is better than the objective function values of the other recourse policies in column "#better(R_0)". We also count the number of instances such that using R_0 is outperformed by the other policy in column "#worse(R_0)". The results show that the outsourcing unit

cost ($\alpha = 500$) is relatively higher than the time window violation cost ($\beta = 100$). If we decrease the outsourcing unit cost to 100 ($\alpha = 100$), the outsourcing policy R_0 becomes the most attractive policy compared to other three policies.

Table 5.8: The comparison between R_0 with different α and other policies with $\beta = 40$

	Comparison	#better(R_0)	#worse(R_0)
$\alpha = 500$	R_0 vs. R_1	23	30
$\beta = 40$	R_0 vs. R_2	28	25
	R_0 vs. R_3	20	33
$\alpha = 100$	R_0 vs. R_1	39	14
$\beta = 40$	R_0 vs. R_2	47	6
	R_0 vs. R_3	32	21

5.7 Conclusions and future work

We addressed a practical transportation problem in the laundry business, where pickup demand uncertainty is also taken into consideration. We modeled the problem as a two-stage stochastic problem and implemented a hierarchical, scenario-based sample approximation method, in combination with the ALNS heuristic algorithm, to take travel time uncertainty into account. Moreover, four types of recourse policies are considered. The computational results on instances with up to 100 requests reveal that, compared to a pure deterministic solution approach using expected pickup demands, cost savings of on average more than 50% can be achieved, which indicates significant reduction in expected operational costs.

Although some interesting results were obtained, ample opportunities exist for future work. One extension could involve modeling the uncertainty of the customer presence and customer service time into this problem. Besides, our approach assumes that the underlying probability distributions of data are precisely known beforehand. In reality, the exact probability distribution of historical data is not known. Another direction could be using a robust scenario approach to solve this problem.

Chapter 6

Conclusion

"Therefore, just as water retains no constant shape, so in warfare there are no constant conditions."

Sun Tzu

.

At the moment of writing this chapter, the whole world is under the pressure of the global prevalence of the COVID-19 epidemic. It fundamentally affects global logistics on a scale unseen in recent times. On the one hand, more restricted customs regulations result in longer waiting times at the border, and movement restrictions lead to lack of capacity for last-mile fulfillment, which brings in more challenges for logistics organizations. On the other hand, amid COVID-19 impact, many countries are experiencing supply pressure due to panic buying by consumers. E-commerce companies also continue to meet the increasing demand for daily supplies by the consumers. For instance, the Walmart Grocery application hit all-time-high downloads in the United States (Reportlinker [87]). As a result, logistics providers need to look for innovative and effective transportation solutions to respond to new conditions and sustain their business operations. We all believe the COVID-19 emergency will pass eventually. However, finding a balance between consumers' delivery expectations and maintaining profitability will still be one of the biggest challenges for most e-commerce companies and logistical retailers in the pro-COVID era.

In addition, in classical route planning the travel time between two customers is assumed to be a constant value, such as a distance metric or an average travel-time estimate throughout a day. This assumption is too simplistic to accurately model travel times because planners have to take into account congestion issues. The travel time between two customers can often vary significantly during the day due to external factors such as traffic congestion or rush hours.

Therefore, this motivates modeling time-dependent travel times, which means that the travel time between two customers depends on the departure time at the first customer. This leads to a more realistic vehicle trip planning process.

Therefore, in this thesis, we investigated two variants of the vehicle routing problem with time windows that consider time-dependent travel times. One is the time-dependent pickup and delivery problem with time windows, which also considers different degrees of flexibility of the customers selection and departure times. The other one is the time-dependent laundry routing problem, which also models the uncertainty of pickup demands. Throughout this thesis we assumed that each link in the network is assigned with a time-dependent travel time function and we have full knowledge of this function which results from the predictable traffic congestion. We further assumed that the first-in-first-out property is respected, which means, for every link, a vehicle that departs earlier must arrive earlier than a vehicle that departs at a later moment on the same link.

The remainder of the chapter is organized as follows. In Section 6.1, we respond to the research questions we proposed in Chapter 1 by discussing the main conclusions from Chapters 2-5. In Section 6.2, we highlight the general directions for future research.

6.1 Discussion

Research question 1. Which exact and heuristic methods are effective to manage pickup and delivery service under time-dependent environment?

In Chapters 2 and 3, different models and exact algorithms were proposed to solve the problem. More specifically, an arc-based mixed-integer program (MIP) was first presented for a single-vehicle problem in Chapter 2, which was extended to multiple-vehicle problems in Chapter 3. Then, a tailored labeling algorithm and a branch-and-price based framework were proposed in Chapters 2 and 3, respectively. As expected, these two exact algorithms performed significantly better than did directly solving the proposed MIP with optimization software (i.e., Gurobi 5.6). The tailored labeling algorithm was capable of solving single-vehicle problem instances with up to 150 locations (75 pickup and delivery requests) to optimality. The branch-and-price-based framework was able to solve multiple-vehicle problem instances with up to 45 freight requests (90 locations). However, the arc-based formulation could determine optimal solutions for single-vehicle instances with up to 30 requests and could not find optimal solutions for multiple-vehicle instances with 20 or 25

requests.

In Chapter 2, we also developed a restricted dynamic programming heuristic to reduce the computation time needed to solve large-size instances. According to the obtained results, 32 instances out of the 34 instances could be solved to optimality in short computation time. Moreover, in Chapter 4, we developed an adaptive large neighborhood search (ALNS) heuristic for one variant of the multiple-vehicle problem studied in Chapter 3. We made ten removal and five insertion operators, which were adapted or inspired from the literature. Instances with up to 75 requests (150 locations) were solved quickly.

Research question 2. What are the benefits and advantages of taking fluctuating travel times into account while planning the routes in pickup and delivery services?

We performed several experiments to demonstrate the advantages of considering time-dependent travel time in Chapter 4. Particularly, we generated three different time-independent solution approaches, the slow- (TID-Slow), average- (TID-Average), and fast-speed (TID-Fast) cases by using the lowest, average, and fastest speeds of their originally assigned speed profiles, respectively, as the estimated fixed speed during the day. We saw that the best-found solution in the fast-speed case (where it is assumed that the vehicle can always run on the fastest possible speed) was clearly over-optimistic, which also met the high risk of violating the time window constraints. On the contrary, the best-found solution in the slow-speed case was too conservative. Even though no violation and penalty costs were generated, the quality of the best-found solution was significantly worse than that of its counterpart under the time-dependent setting. The results also showed that assuming a constant travel-time relation with the average speed as approximation of the fixed speed during the day did not work out. It led to 3% of violated time windows and more than 10% higher routing cost than in cases in which time-dependent travel times were considered in the route optimization phase.

Research question 3. What is the value of different degrees of flexibility in pickup and delivery service when fluctuating travel times are considered?

In Chapter 2, we studied four variants of the time-dependent pickup and delivery problem with time windows and considered different degrees of flexibility in driver start time and requests selection. Moreover, we derived relationships between objective function values of optimal solutions, such that the objective function values of the variant with the least

flexibility were smaller than the values of its counterpart with more flexibility. We next quantified the relative differences in these objective function values. Specifically, for instances wherein the variant with the least flexibility could be solved to optimality, we calculated the percentage improvement in objective function value for a given variant with respect to the variant with the least flexibility. Moreover, we showed that the variant with the most flexibility led to a significant increase in profits. This suggests that there is significant value for a carrier in moving away from publishing a fixed schedule to which they must adhere. We also saw that no relationship can be derived between variants with different types of flexibility, as sometimes one yields a higher objective function value and sometimes the other. We also noted that there seems to be a correlation between the number of requests and these differences. The larger the number of requests, the smaller the gains associated with adding flexibility.

Research question 4. Is a pickup and delivery service still efficient and effective when information regarding pickup demand is uncertain during the planning process?

Finally, in Chapter 5, we investigated an important process for the commercial laundry business, where the pickup demands are uncertain at the moment of planning, given corresponding distributions. The pickup demands depend on how many customers a hotel or restaurant had in the previous days, and the amount is not known to the laundry provider untill the moment it arrives at the hotel or restaurant. We developed a two-stage stochastic programming with recourse model for this problem. We also extended the ALNS algorithm proposed in Chapter 4 of this thesis and embedded it into a sample average approximation method to obtain a heuristic solution. The ALNS operators were updated to account for stochastic elements of the problem. We solved instances with up to 100 requests. Moreover, four types of recourse policies were proposed. Compared to a pure deterministic solution approach using expected pickup demands, the computational results showed that cost savings of on average more than 50% can be achieved.

6.2 Further research directions

The research presented in this thesis can be extended in multiple directions. In this thesis, we made several simplifying assumptions because of the enormous complexity of real-life systems and limited computing resources available to tackle real-life problems. For instance, we considered time-dependent travel times, which excels at capturing recurrent traffic situations such as daily rush hours. However, we did not model rare events such as

large unexpected traffic jams. In general, the empirical data are not sufficient to measure the real probability distribution of these events. Thus, robust optimization or distributionally robust optimization can be used to consider both uncertain and time-dependent travel times and the development of appropriate algorithms could be an interesting direction for further research.

In three of the four chapters we focused on operational decision support tools for static problems, in which all the required information, such as available vehicles and customers' location and demands, are entirely known before carrying out service. However, in the actual service process, many unexpected events (e.g., presence of the customers or drivers) may lead to difficulties in executing a static plan. Thus, considering dynamic aspects of the problem and developing fast and efficient algorithms could be an interesting research question. In addition, such dynamic algorithms may be used as recourse actions within methodologies focused on stochastic variants of the problem.

From a methodological point of view, the exact algorithm presented in Chapters 2 and 3 can be improved by adding appropriate valid inequalities to tighten the lower bound. In addition, because most of the processing effort is spent in solving the pricing problem, stronger valid dominance criteria need to be determined to speed-up the algorithm.

Moreover, recent years have seen a huge surge in information and data collection while logistics operations are carried out. This data-rich operating environment calls for data-driven approaches while conducting pickup and delivery services. New transport planning and scheduling systems will be developed using big data to forecast delivery routes (for tactical planning) and using real-time traffic information and availability of unloading zones (for operational planning) for planning and scheduling.

In addition, to make urban transportation more efficient and sustainable, investing in more environmentally friendly and safe modes of transportation such as electric vehicles (EVs) is becoming a necessity. In fact, EVs now represent a credible alternative to more conventional engines. However, EVs are currently facing several weaknesses related to limited driving range, long charging times, the availability of a charging infrastructure, and high purchasing costs. Therefore, the models given in Chapter 2, 3 and 5 can be extended by integrating operational considerations associated with EVs.

Appendix A

Appendix A: Mathematical formulation

Similar to the mathematical formulation proposed in Sun et al. [110], $y_i^k, i \in N$, is denoted as a binary variable set to 1 if node i is visited by vehicle k; 0 otherwise. $x_{ij}^k, i \in N$, is a binary variable set to 1 if arc (i, j) is traversed by vehicle k; 0 otherwise. The cost per unit of route duration is denoted c_t. We also define $Q_i^k, i \in N, k \in K$ as a non-negative integer that is an upper bound on the amount of capacity at node i. Therefore, we formulate the TD-PPDP-TW as the following mixed-integer program:

$$\max \sum_{k \in \mathcal{K}} [\sum_{i \in \mathcal{N}_\mathcal{P}} p_i y_i^k - c_t(t_{2n+1}^k - t_0^k) - Z y_0^k] \tag{1}$$

subject to

$$\sum_{j \in N_P} x_{0j}^k = y_0^k \qquad \forall k \in K, \tag{2}$$

$$\sum_{i \in N_D} x_{i,2n+1}^k = y_{2n+1}^k \qquad \forall k \in K, \tag{3}$$

$$\sum_{k \in K} y_j^k \leq 1 \qquad \forall j \in N_P \cup N_D, \tag{4}$$

$$\sum_{j \in N \setminus \{0\}} x_{ij}^k - \sum_{j \in N \setminus \{0\}} x_{n+i,j}^k = 0 \qquad \forall i \in N_P, \forall k \in K, \tag{5}$$

$$\sum_{i \in N \setminus \{2n+1\}} x_{ij}^k = y_j^k \qquad \forall j \in N_P \cup N_D, \tag{6}$$

$$\sum_{i \in N \setminus \{2n+1\}} x_{if}^k - \sum_{j \in N \setminus \{0\}} x_{fj}^k = 0 \qquad \forall f, i, j \in N_P \cup N_D, \forall k \in K, \tag{7}$$

$$t_i^k + \tau_{ij}(t_i^k) + s_j \leq t_j^k + M(1 - x_{ij}^k) \qquad \forall i, j \in N_P \cup N_D, \forall k \in K, \tag{8}$$

$$t_{n+i}^k \geq t_i^k \qquad \forall i \in N_P, \forall k \in K, \tag{9}$$

$$(e_i + s_i) y_i^k \leq t_i^k \leq (l_i + s_i) y_i^k \qquad \forall i \in N, \forall k \in K, \tag{10}$$

$$Q_i^k + q_j \leq Q_j^k + M(1 - x_{ij}^k) \qquad \forall i, j \in N, \forall k \in K, \tag{11}$$

$$\max\{0, q_i\} \leq Q_i^k \leq \min\{Q, Q + q_i\} \qquad \forall i \in N, \forall k \in K, \tag{12}$$

$$x_{ij}^k, y_i^k \in \{0, 1\} \qquad \forall i, j \in N, \forall k \in K. \tag{13}$$

Constraints (2) and (3) guarantee that the route of vehicle k starts from the origin depot and ends at the destination depot if vehicle k is used. Constraints (4)-(6) ensure that every request

is served at most once, and its pickup and delivery nodes are visited by the same vehicle. respectively. Flow conservation is confirmed by Constraints (7). Constraints (8) ensure the departure time from each location respects travel time and service time. Constraints (9) ensure that the pickup node is visited before the delivery node for each request. Constraints (10) ensure that the departure time at each node of the request should be within the given time window (if the request is served). Constraints (11) and (12) are capacity constraints of vehicles.

Appendix B

Appendix B: Potential improvement by local search

Table 1: Improvement by local search

		Without Local search		With Local search	
Instance	Best	Gap(%)	Time (s)	Gap(%)	Time (s)
S_BB35	337.55	3.09	7.1	3.09	7.3
S_DD40	490.92	4.16	42.2	2.62	46.6
S_BB50	432.1	2.30	22.6	2.30	22.9
S_DD65	760.32	2.13	170.5	2.61	173.9
S_BB70	558.29	4.54	45.4	4.76	45.9
S_CC70	681.84	1.82	195.6	3.45	197.7
S_AA75	715.53	3.19	161.9	3.19	165.4
S_BB75	613.35	3.07	65.8	3.24	68.8
S_CC75	686.68	2.14	153.2	0.80	163.0
Average	586.29	2.94	96.0	2.90	99.1

Appendix C

Table 2: Results of solving the $\Omega(Fix, All)$ and $\Omega(Fix, Prof)$ using Gurobi

	$\Omega(Fix, All)$				$\Omega(Fix, Prof)$			
Instances	LB	UB	Time	Gap(%)	LB	UB	Time	Gap(%)
AA10	110.92	110.92	194.5	0	110.92	110.92	203.1	0
BB10	156.61	156.61	18.0	0	156.61	156.61	108.9	0
CC10	157.36	157.36	527.2	0	224.62	224.62	950.4	0
DD10	497.70	497.70	6076.7	0	206.39	206.39	1311.9	0
AA15	315.31	1076.49	36000.0	241	421.03	1018.00	36000.0	142
BB15	369.74	369.74	6042.0	0	516.61	516.61	2380.3	0
CC15	563.36	1118.00	36000.0	98.5	563.36	1157.14	36000.0	105
DD15	497.70	497.70	5519.1	0	541.46	1156.00	36000.0	113
AA20	331.09	1458.00	36000.0	340	659.21	1458.00	36000.0	121
BB20	580.75	884.61	36000.0	52.3	789.17	879.15	36000.0	11.4
CC20	658.96	1640.00	36000.0	149	810.96	1640.00	36000.0	102
DD20	905.48	1640.00	36000.0	81.1	871.48	1640.00	36000.0	88.2
AA25	776.93	1922.00	36000.0	147	1023.01	1922.00	36000.0	87.9
BB25	885.49	1959.02	36000.0	121	1169.59	1940.17	36000.0	65.9
CC25	-	2122.00	36000.0	-	1210.71	2122.00	36000.0	75.3
DD25	-	2122.00	36000.0	-	1165.66	2122.00	36000.0	82
Average			23648.6	87.9			25059.7	62.1

Table 3: Results of solving the $\Omega(Flex, All)$ and $\Omega(Flex, Prof)$ using Gurobi

	$\Omega(Flex, All)$				$\Omega(Flex, Prof)$			
Instances	LB	UB	Time	Gap(%)	LB	UB	Time	Gap(%)
AA10	256.92	256.92	390.9	0	262.66	262.66	938.8	0
BB10	211.66	211.66	51.8	0	222.60	222.60	42.0	0
CC10	372.37	522.43	36000.0	40.3	372.37	574.00	36000.0	54.1
DD10	197.04	197.04	31402.6	0	271.96	378.92	36000.0	39.3
AA15	517.09	1021.84	36000.0	97.6	520.81	1018.00	36000.0	95.5
BB15	420.42	834.57	36000.0	98.5	554.24	918.00	36000.0	65.6
CC15	593.08	1118.00	36000.0	88.5	684.31	1157.30	36000.0	69.1
DD15	467.80	1112.44	36000.0	138	606.48	1156.00	36000.0	90.6
AA20	668.68	1458.00	36000.0	118	706.36	1458.00	36000.0	106
BB20	703.46	1350.95	36000.0	92	843.94	1445.59	36000.0	71.3
CC20	856.99	1640.00	36000.0	91.4	964.99	1640.00	36000.0	69.9
DD20	-	1640.00	36000.0	-	870.12	1640.00	36000.0	88.5
AA25	992.54	1922.00	36000.0	93.6	1051.75	1922.00	36000.0	82.7
BB25	918.17	2022.00	36000.0	120	1184.67	2023.66	36000.0	70.8
CC25	738.25	2122.00	36000.0	187	1269.14	2122.00	36000.0	67.2
DD25	992.48	2122.00	36000.0	114	1129.80	2122.00	36000.0	87.8
Average			31240.3	85.26			31561.3	66.15

Table 4: Detailed computational results for the $\Omega(Fix, All)$

Instance	LB	RootUB	BestUB	Time	Gap(%)	Branches	Cuts	Served
AA10	110.92	110.92	110.92	0.6	0.00	1	0	10
BB10	156.61	156.61	156.61	0.4	0.00	1	0	10
CC10	157.36	157.36	157.36	0.7	0.00	1	0	10
DD10	91.70	91.70	91.70	1.0	0.00	1	0	10
AA15	315.31	315.31	315.31	2.3	0.00	1	5	15
BB15	369.73	369.73	369.73	1.7	0.00	1	5	15
CC15	563.36	563.36	563.36	4.7	0.00	1	0	15
DD15	497.70	497.70	497.70	2.5	0.00	1	0	15
AA20	331.09	331.09	331.09	390.9	0.00	1	55	20
BB20	580.74	580.74	580.74	201.8	0.00	2	75	20
CC20	687.57	862.89	688.14	3457.6	0.08	126	30	20
DD20	905.48	905.48	905.48	10.8	0.00	1	0	20
AA25	776.93	776.93	776.93	7.6	0.00	1	90	25
BB25	1018.25	1023.33	1018.25	557.7	0.00	3	55	25
CC25	1079.42	1233.22	1080.46	10672.4	0.10	1101	15	25
DD25	992.71	1138.53	1075.18	36000.0	7.67	561	5	25
AA30	966.29	1096.22	966.29	1021.3	0.00	19	35	30
BB30	1393.12	1393.12	1393.12	83.7	0.00	1	120	30
CC30	1450.51	1622.67	1455.38	36000.0	0.33	39	0	30
DD30	1005.95	1599.88	1503.48	36000.0	33.09	42	0	30
AA35	1419.48	1510.69	1419.48	2304.1	0.00	1139	50	35
BB35	1843.79	1843.79	1843.79	33.6	0.00	1	10	35
CC35	1635.02	2008.12	1941.69	36000.0	15.79	51	0	35
DD35	673.79	1975.92	1975.92	36000.0	65.90	1	0	35
AA40	1685.98	1687.32	1687.32	1034.5	0.08	1	15	40
BB40	2000.09	2020.92	2001.84	637.0	0.09	4	225	40
CC40	1446.21	2305.85	2305.85	36000.0	37.28	1	0	40
DD40	-	2276.72	-	36000.0	-	-	0	-
AA45	1927.92	2088.80	2074.39	36000.0	7.06	358	0	45
BB45	2391.44	2448.23	2393.71	3505.3	0.09	43	0	45
CC45	1656.90	2606.69	2606.69	36000.0	36.44	1	0	45
DD45	-	2564.44	-	36000.0	-	-	0	-

* LB: the lower bound, RootUB: the root upper bound, BestUB: the best upper bound within the time limit, Time: the processing time in seconds, Gap%: gap, Branches: number of branches considered in the algorithm and Served: the number of served requests in the lower bound solution.

Table 5: Detailed computational results for the $\Omega(Fix, Prof)$

Instance	LB	RootUB	BestUB	Time	Gap(%)	Branches	Served
AA10	110.92	110.92	110.92	0.2	0.00	1	10
BB10	156.61	156.61	156.61	0.2	0.00	1	10
CC10	224.62	224.62	224.62	0.4	0.00	1	7
DD10	206.39	206.39	206.39	0.6	0.00	1	7
AA15	421.03	421.03	421.03	1.0	0.00	1	14
BB15	516.61	516.61	516.61	1.2	0.00	1	13
CC15	563.36	563.36	563.36	3.7	0.00	1	15
DD15	621.36	621.36	621.36	1.8	0.00	1	14
AA20	659.21	659.21	659.21	3.9	0.00	1	16
BB20	789.17	789.17	789.17	2.6	0.00	1	17
CC20	888.96	888.96	888.96	12.5	0.00	1	19
DD20	905.48	905.48	905.48	10.5	0.00	1	20
AA25	1023.01	1023.01	1023.01	4.2	0.00	2	22
BB25	1169.59	1169.59	1169.59	4.3	0.00	1	19
CC25	1296.93	1296.93	1296.93	26.9	0.00	1	23
DD25	1284.99	1284.99	1284.99	74.2	0.00	1	22
AA30	1196.72	1204.55	1196.72	263.2	0.00	2	23
BB30	1471.75	1471.75	1471.75	23.1	0.00	1	22
CC30	1615.59	1650.49	1615.59	409.9	0.00	1	26
DD30	1591.88	1613.11	1591.88	1203.9	0.00	1	27
AA35	1569.12	1587.34	1569.12	120.2	0.00	27	33
BB35	1843.79	1891.99	1843.79	295.7	0.00	36	35
CC35	1903.91	2019.04	1945.01	36000.0	2.11	187	34
DD35	1880.32	1988.88	1920.44	36000.0	2.09	11	27
AA40	1811.67	1811.67	1811.67	525.5	0.00	22	37
BB40	2110.83	2133.10	2110.83	238.5	0.00	13	37
CC40	2120.81	2258.60	2183.52	36000.0	2.87	1215	36
DD40	-	2279.46	-	36000.0	-	-	-
AA45	2193.08	2195.61	2193.08	265.3	0.00	4	41
BB45	2509.95	2509.95	2509.95	73.0	0.00	1	42
CC45	2228.02	2618.63	2618.63	36000.0	14.92	1	38
DD45	-	2567.93	-	36000.0	-	1	-

Table 6: Detailed computational results for the $\Omega(Flex, All)$

Instance	LB	RootUB	BestUB	Time	Gap(%)	Branches	Cuts	Served
AA10	256.92	256.92	256.92	0.5	0.00	1	0	10
BB10	211.66	211.66	211.66	0.5	0.00	1	0	10
CC10	372.37	372.37	372.37	1.0	0.00	1	0	10
DD10	197.04	197.04	197.04	1.8	0.00	1	0	10
AA15	517.09	517.09	517.09	3.1	0.00	1	5	15
BB15	420.41	420.42	420.42	2.6	0.00	1	5	15
CC15	684.31	684.31	684.31	7.9	0.00	1	0	15
DD15	540.76	540.76	540.76	10.2	0.00	1	0	15
AA20	668.68	668.68	668.68	6.7	0.00	1	10	20
BB20	705.68	705.68	705.68	44.9	0.00	1	60	20
CC20	900.45	961.90	901.24	1204.4	0.07	73	0	20
DD20	932.60	932.60	932.60	13.6	0.00	1	0	20
AA25	1032.34	1032.34	1032.34	12.7	0.00	1	10	25
BB25	1071.91	1085.12	1071.91	571.1	0.00	3	65	25
CC25	1298.40	1344.70	1299.63	5409.3	0.03	19	0	25
DD25	1235.69	1296.89	1236.73	2559.1	0.07	36	0	25
AA30	1175.56	1178.74	1178.74	36000.0	0.27	1	75	30
BB30	1483.97	1483.97	1483.97	41.2	0.00	1	25	30
CC30	1689.39	1711.53	1689.78	25103.9	0.00	4	0	30
DD30	1361.70	1669.71	1669.71	36000.0	18.45	1	0	30
AA35	1571.87	1610.55	1610.55	36000.0	2.40	1	40	35
BB35	1897.79	1897.79	1897.79	27.5	0.00	1	5	35
CC35	1925.26	2048.17	1945.01	36000.0	1.02	1	0	35
DD35	1893.00	2082.59	2082.59	36000.0	9.10	1	0	35
AA40	1791.66	1810.23	1810.23	36000.0	0.10	1	0	40
BB40	2058.63	2072.87	2059.81	3987.1	0.06	9	5	40
CC40	-	2347.11	2347.11	36000.0	-	1	0	-
DD40	2585.33	2589.86	2588.20	36000.0	0.11	1	0	45
AA45	2101.25	2190.45	2182.90	36000.0	3.74	29	0	45
BB45	2455.72	2474.87	2457.44	3593.0	0.07	3	0	45
CC45	-	2664.00	2664.00	36000.0	-	1	0	-
DD45	-	2698.68	2698.68	36000.0	-	1	0	-

Table 7: Detailed computational results for the $\Omega(Flex, Prof)$

Instance	LB	RootUB	BestUB	Time	Gap(%)	Branches	Served
AA10	262.66	262.66	262.66	0.3	0.00	1	9
BB10	222.60	222.60	222.60	0.3	0.00	1	8
CC10	372.37	372.37	372.37	1.5	0.00	1	10
DD10	271.96	271.96	271.96	1.8	0.00	1	6
AA15	520.81	547.43	520.81	3.4	0.00	2	14
BB15	554.24	554.24	554.24	1.0	0.00	1	12
CC15	684.31	684.31	684.31	14.5	0.00	1	15
DD15	606.48	606.48	606.48	9.6	0.00	1	12
AA20	706.36	743.31	706.36	47.0	0.09	57	18
BB20	843.94	848.79	843.94	3.4	0.00	3	15
CC20	964.99	964.99	964.99	24.6	0.00	1	18
DD20	942.38	942.38	942.38	9.6	0.00	1	17
AA25	1051.88	1074.07	1051.88	65.8	0.08	19	22
BB25	1184.67	1203.89	1184.67	14.3	0.00	2	19
CC25	1355.72	1362.03	1355.72	214.6	0.00	3	22
DD25	1303.20	1324.21	1303.20	210.0	0.00	2	21
AA30	1284.11	1331.86	1284.11	4651.8	0.00	1905	28
BB30	1506.89	1569.83	1506.89	267.3	0.09	59	27
CC30	1689.39	1722.56	1689.44	3544.1	0.02	8	30
DD30	1627.76	1686.20	1629.26	9158.2	0.10	36	24
AA35	1656.81	1684.16	1658.46	15804.2	0.10	2175	33
BB35	1897.79	1925.87	1897.79	138.5	0.00	13	35
CC35	2028.82	2050.13	2030.72	26082.1	0.09	36	34
DD35	1960.66	2099.28	2021.95	36000.0	3.03	131	31
AA40	1887.74	1892.81	1889.57	433.4	0.10	12	37
BB40	2150.70	2154.69	2150.70	523.1	0.00	9	37
CC40	2317.31	2356.45	2356.45	36000.0	1.66	1	37
DD40	-	2370.77	-	36000.0	-	1	-
AA45	2259.00	2268.92	2259.00	1170.0	0.01	4	41
BB45	2525.08	2525.08	2525.08	257.9	0.00	1	42
CC45	-	2664.79	-	36000.0	-	1	-
DD45	2425.56	2703.40	2703.40	36000.0	10.28	1	40

Table 8: The objective improvement of other 3 variants compare to $\Omega(Fix, All)$

Instance	$\Omega(Fix, All)$(Value)	$\Omega(Fix, Prof)$(%)	$\Omega(Flex, All)$(%)	$\Omega(Flex, Prof)$(%)
AA10	110.919	0.00	131.63	136.81
BB10	156.608	0.00	35.15	42.14
CC10	157.355	42.75	136.64	136.64
DD10	91.695	125.08	114.88	196.59
AA15	315.306	33.53	64.00	65.17
BB15	369.735	39.72	13.71	49.90
CC15	563.355	0.00	21.47	21.47
DD15	497.695	24.85	8.65	21.86
AA20	331.091	99.10	101.96	113.34
BB20	580.745	35.89	21.51	45.32
CC20	687.57	29.29	30.96	40.35
DD20	905.483	0.00	3.00	4.08
AA25	776.928	31.67	32.87	35.39
BB25	1018.25	14.86	5.27	16.34
CC25	1079.42	20.15	20.29	25.60
DD25	992.712	29.44	24.48	31.28
AA30	966.289	23.85	21.66	32.07
BB30	1393.12	5.64	6.52	8.17
CC30	1450.51	11.38	16.47	16.47
DD30	1005.95	58.25	35.36	61.81
AA35	1419.48	10.54	10.74	16.72
BB35	1843.79	0.00	2.93	2.99
CC35	1635.02	16.45	17.75	24.09
DD35	673.79	179.07	180.95	190.99
AA40	1685.98	7.45	6.27	11.97
BB40	2000.09	5.54	2.93	7.53
CC40	1446.21	46.65	-	60.23
DD40	1927.92	-	34.10	-
AA45	1928.64	13.71	8.95	17.13
BB45	2391.44	4.96	2.69	5.59
CC45	1656.90	34.47	-	-
DD45	-	-	-	-

Table 9: Performance of the exact framework on instances of Røpke and Cordeau [91]. *Gap* represents the % difference of our optimal (or best found, in case the time limit of 36000 seconds is exceeded) solution compared to the optimal solution of Røpke and Cordeau [91].

	Røpke and Cordeau [91]	Our Framework	
Instance	Time (s)	Gap (%)	Time (s)
AA30	7	0	593
BB30	6	0	342
CC30	11	0	415
DD30	25	0	7178
AA35	16	0	1058
BB35	18	0	2559
CC35	59	0	2208
DD35	765	0	17243
AA40	13	0	11723
BB40	19	0	2617
CC40	255	0	17414
DD40	161	*	36000
AA45	16	0	11060
BB45	46	0	35077
CC45	176	32	36000
DD45	525	*	36000
Average	132		13593

* = no feasible solution found within 36000 seconds.

Table 10: Detailed results for tuning of the roulette wheel mechanism parameters ($\sigma_1, \sigma_2, \sigma_3$)

		(5,1,3)			(1,3,5)			(1,1,1)			(5,3,1)		
Instance	Best	G (%)	S (%)	T (s)	G (%)	S (%)	T (s)	G (%)	S (%)	T (s)	G (%)	S (%)	T (s)
S_BB10	39.26	0.01	0.00	0.6	0.01	0.00	0.6	0.01	0.00	0.6	0.01	0.00	0.6
S_BB15	99.58	0.00	0.00	1.3	0.00	0.00	1.4	0.00	0.00	1.4	0.00	0.00	1.6
S_BB20	138.05	0.00	0.00	2.6	0.00	0.00	2.6	0.00	2.30	2.4	0.00	0.00	2.6
S_BB25	147.88	0.00	0.00	4.7	0.00	3.12	4.5	0.00	3.05	6.2	0.00	0.00	4.8
S_BB30	274.45	0.00	3.37	6.6	0.00	2.67	6.2	0.00	3.37	6.7	0.00	1.77	7.4
S_BB35	337.55	3.09	0.94	7.2	3.09	2.74	8.2	3.09	0.00	7.4	3.09	1.32	7.1
S_BB40	326.82	0.00	0.00	9.6	0.00	0.00	10.9	0.00	0.00	11.6	0.00	0.00	10.5
S_BB45	378.90	0.62	0.24	18.3	0.62	0.09	19.1	0.62	0.03	19.1	0.62	0.11	20.6
S_BB50	432.10	2.30	3.44	25.3	2.30	2.83	23.0	2.30	3.22	28.0	**0.00**	3.04	22.6
S_BB55	530.84	0.00	2.78	27.5	0.00	2.78	26.9	0.00	2.51	35.1	0.00	2.39	33.8
S_BB60	558.02	0.36	1.43	34.2	0.36	1.91	32.9	0.36	1.43	42.2	**0.00**	0.75	39.8
S_BB65	547.84	0.17	5.32	45.6	0.00	2.10	52.7	0.00	2.75	60.9	0.00	3.90	41.4
S_BB70	558.29	4.76	0.42	49.4	4.76	0.72	44.4	5.31	1.77	44.1	**1.70**	1.36	45.4
S_BB75	613.35	3.20	1.65	63.4	3.20	3.21	77.1	3.80	1.58	56.6	3.20	1.69	65.8
M_BB10	222.60	0.00	0.00	0.7	0.00	0.00	0.6	0.00	0.00	0.7	0.00	0.00	0.6
M_BB15	554.24	0.00	0.00	1.6	0.00	0.00	1.4	0.00	0.00	1.4	0.00	0.00	1.5
M_BB20	843.94	0.00	0.00	2.7	0.00	0.00	2.6	0.00	0.00	2.9	0.00	0.00	3.2
M_BB25	1184.67	0.00	0.00	4.3	0.00	0.00	4.0	0.00	0.00	4.1	0.00	0.00	4.1
M_BB30	1506.89	0.75	0.00	5.7	0.75	0.00	6.0	0.75	0.00	6.8	**0.00**	0.37	6.4
M_BB35	1898.97	0.00	0.36	5.2	0.00	0.24	4.8	0.00	0.31	5.1	0.00	0.00	4.2
M_BB40	2151.24	0.00	0.00	7.0	0.00	0.00	7.5	0.00	0.11	7.4	0.00	0.13	7.7
M_BB45	2525.08	0.00	0.48	15.7	0.00	0.49	14.1	0.00	0.78	14.7	0.00	0.77	15.5
M_BB50	2767.25	-0.74	1.97	15.8	-1.25	0.68	19.5	-1.25	1.54	20.5	**-1.58**	1.66	15.4

G (%) : Gap (%)
S (%) : STD (%)
T (s): Time (s)

Table 11: Detailed results for tuning of cooling rate parameter

		$\kappa = 0.95$			$\kappa = 0.995$			$\kappa = 0.9995$		
Instance	Best	Gap (%)	STD (%)	Time (s)	Gap (%)	STD (%)	Time (s)	Gap (%)	STD (%)	Time (s)
S_BB10	39.26	0.01	0.00	1.0	0.01	0.00	0.7	0.01	0.00	0.6
S_BB15	99.58	0.00	0.00	1.7	0.00	0.00	1.8	0.00	0.00	1.6
S_BB20	138.05	0.00	7.62	3.2	0.00	6.62	2.6	0.00	0.00	2.6
S_BB25	147.88	0.00	6.03	4.6	0.00	5.19	4.5	0.00	0.00	4.8
S_BB30	274.45	0.00	3.37	5.9	0.00	4.56	6.5	0.00	1.77	7.4
S_BB35	337.55	3.09	2.74	6.0	3.09	0.00	7.0	3.09	1.32	7.1
S_BB40	326.82	0.00	0.00	8.8	0.00	0.00	9.6	0.00	0.00	10.5
S_BB45	378.90	0.62	0.26	16.6	0.62	0.02	16.4	0.62	0.11	20.6
S_BB50	432.10	2.30	3.29	18.2	2.30	3.72	19.3	**0.00**	3.04	22.6
S_BB55	530.84	5.05	4.87	26.5	5.05	0.87	27.5	**0.00**	2.39	33.8
S_BB60	558.02	0.00	0.15	29.9	0.00	1.47	33.2	0.00	0.75	39.8
S_BB65	547.84	0.00	5.46	35.3	0.00	5.26	42.4	0.00	3.90	41.4
S_BB70	558.29	4.76	0.75	34.7	4.03	1.40	40.8	**1.70**	1.36	45.4
S_BB75	613.35	4.03	1.78	53.1	3.80	1.60	52.8	**3.20**	1.69	65.8
M_BB10	222.60	0.00	0.00	0.7	0.00	0.00	0.6	0.00	0.00	0.6
M_BB15	554.24	0.00	0.00	1.8	0.00	0.00	1.7	0.00	0.00	1.5
M_BB20	843.94	0.00	0.00	3.2	0.00	0.00	3.3	0.00	0.00	3.2
M_BB25	1184.67	0.00	0.87	3.6	0.00	0.00	3.8	0.00	0.00	4.1
M_BB30	1506.89	0.00	0.91	4.1	0.75	0.00	6.1	0.00	0.37	6.4
M_BB35	1898.97	0.00	0.48	4.0	0.00	0.39	4.0	0.00	0.00	4.2
M_BB40	2151.24	0.00	0.00	6.1	0.00	0.55	5.7	0.00	0.13	7.7
M_BB45	2525.08	0.00	0.84	13.9	0.00	0.93	11.3	0.00	0.77	15.5
M_BB50	2767.25	-1.25	1.97	15.9	-1.25	1.09	17.9	**-1.58**	1.66	15.4

Table 12: Detailed results for tuning of destroy rate parameter

Instance	Best	0.25			0.35			0.45		
		Gap (%)	STD (%)	Time (s)	Gap (%)	STD (%)	Time (s)	Gap (%)	STD (%)	Time (s)
S_BB10	39.26	0.01	0.00	0.6	0.01	0.00	0.6	0.01	0.00	0.6
S_BB15	99.58	0.00	0.00	1.2	0.00	0.00	1.4	0.00	0.00	1.6
S_BB20	138.05	0.00	0.00	1.8	0.00	0.00	2.4	0.00	0.00	2.6
S_BB25	147.88	0.00	2.57	4.2	0.00	0.00	4.7	0.00	0.00	4.8
S_BB30	274.45	0.00	2.95	5.3	0.00	3.18	6.1	0.00	1.77	7.4
S_BB35	337.55	3.09	0.94	5.7	3.09	0.94	6.9	3.09	1.32	7.1
S_BB40	326.82	0.00	0.00	7.7	0.00	0.00	8.6	0.00	0.00	10.5
S_BB45	378.90	0.62	0.18	13.6	0.62	0.10	18.5	0.62	0.11	20.6
S_BB50	432.10	2.30	3.28	13.2	2.30	3.28	17.5	**0.00**	3.04	22.6
S_BB55	530.84	0.00	3.63	20.1	0.00	3.02	28.9	0.00	2.39	33.8
S_BB60	558.02	0.00	1.47	21.4	0.36	2.71	25.4	0.00	0.75	39.8
S_BB65	547.84	0.00	4.18	35.0	0.00	4.03	39.8	0.00	3.90	41.4
S_BB70	558.29	1.83	1.50	31.9	4.07	1.15	41.5	**1.70**	1.36	45.4
S_BB75	613.35	3.38	1.60	49.9	**3.17**	1.42	55.8	3.20	1.69	65.8
M_BB10	222.60	0.00	0.00	0.6	0.00	0.00	0.6	0.00	0.00	0.6
M_BB15	554.24	0.00	0.00	1.2	0.00	0.00	1.4	0.00	0.00	1.5
M_BB20	843.94	0.00	0.00	2.3	0.00	0.00	2.7	0.00	0.00	3.2
M_BB25	1184.67	0.00	0.00	2.8	0.00	0.00	3.3	0.00	0.00	4.1
M_BB30	1506.89	0.75	0.00	4.4	0.75	0.00	5.3	**0.00**	0.37	6.4
M_BB35	1898.97	0.00	0.42	2.1	0.00	0.39	3.9	0.00	0.00	4.2
M_BB40	2151.24	0.00	0.00	3.6	0.00	0.00	6.4	0.00	0.13	7.7
M_BB45	2525.08	0.04	0.79	8.1	0.00	0.60	10.9	0.00	0.77	15.5
M_BB50	2767.25	-1.25	1.32	8.2	-1.58	1.48	14.3	-1.58	1.66	15.4

Table 13: Exact evaluation versus approximation for single-vehicle instances

Instance	Obj_{EE}	$Time_{EE}$	Obj_{AE}	$Time_{AE}$	Instance	Obj_{EE}	$Time_{EE}$	Obj_{AE}	$Time_{AE}$
S_AA10	27.14	1.8	27.14	0.6	S_AA45	361.58	65.3	361.58	26.0
S_BB10	39.26	2.1	39.26	0.6	S_BB45	376.57	48.0	376.57	20.6
S_CC10	57.63	1.8	57.63	0.7	S_CC45	497.69	112.0	497.69	50.8
S_DD10	85.88	2.3	85.88	0.7	S_DD45	535.98	151.3	535.98	62.3
S_AA15	37.77	3.7	37.77	1.2	S_AA50	430.41	87.5	430.41	31.1
S_BB15	99.58	4.3	99.58	1.6	S_BB50	422.15	65.7	422.15	22.6
S_CC15	98.29	4.3	98.29	1.7	S_CC50	516.08	203.0	516.08	98.0
S_DD15	99.05	5.4	99.05	1.7	S_DD50	614.09	208.2	614.09	79.9
S_AA20	73.70	7.9	73.70	1.9	S_AA55	436.05	129.9	436.05	52.0
S_BB20	138.05	7.1	138.05	2.6	S_BB55	530.84	106.0	530.84	33.8
S_CC20	97.27	7.9	97.27	2.4	S_CC55	581.50	213.9	581.50	90.7
S_DD20	182.14	7.9	182.14	4.1	S_DD55	629.01	221.7	629.01	94.2
S_AA25	203.01	13.7	203.01	4.7	S_AA60	499.97	127.1	499.97	43.3
S_BB25	147.88	15.2	147.88	4.8	S_BB60	555.99	112.0	555.99	39.8
S_CC25	**226.23**	23.3	224.88	9.9	S_CC60	609.38	278.7	609.38	165.5
S_DD25	250.16	25.1	250.16	13.9	S_DD60	713.29	418.6	713.29	142.9
S_AA30	266.72	25.1	266.72	8.6	S_AA65	568.15	189.8	568.15	103.6
S_BB30	274.45	19.6	274.45	7.4	S_BB65	547.84	128.0	547.84	41.4
S_CC30	316.39	39.9	316.39	18.5	S_CC65	632.63	340.3	632.63	147.6
S_DD30	343.65	49.0	343.65	24.6	S_DD65	744.15	319.8	744.15	170.5
S_AA35	278.14	29.1	278.14	10.9	S_AA70	606.68	295.4	606.68	86.3
S_BB35	**337.55**	35.5	327.11	7.1	S_BB70	532.96	367.5	532.96	45.4
S_CC35	386.26	76.7	386.26	23.2	S_CC70	669.44	522.4	669.44	195.6
S_DD35	410.19	91.3	410.19	28.9	S_DD70	758.39	493.8	758.39	238.5
S_AA40	341.74	54.1	341.74	14.1	S_AA75	**710.35**	338.3	692.68	161.9
S_BB40	326.82	39.6	326.82	10.5	S_BB75	**611.39**	200.1	594.49	65.8
S_CC40	464.29	118.8	464.29	55.2	S_CC75	**681.60**	505.4	671.98	153.2
S_DD40	**490.92**	130.2	470.52	42.2	S_DD75	845.91	674.8	845.91	290.9
					Average	**398.58**	138.7	397.21	54.61

Obj_{EE}: Objective value of ALNS with Exact Evaluation.
Obj_{AE}: Objective value of ALNS with Approximate Evaluation.
$Time_{EE}$ (s): Processing Time of ALNS with Exact Evaluation.
$Time_{AE}$ (s): Processing Time of ALNS with Approximate Evaluation.

Table 14: Exact route evaluation versus approximate route evaluation for multi-vehicle instances

Instance	Obj_{EE}	$Time_{EE}$	Obj_{AE}	$Time_{AE}$	Instance	Obj_{EE}	$Time_{EE}$	Obj_{AE}	$Time_{AE}$
M_AA10	262.66	1.3	262.66	0.9	M_AA45	2259.00	31.4	2259.00	13.3
M_BB10	222.60	1.6	222.60	1.1	M_BB45	2525.08	31.1	2525.08	17.9
M_CC10	372.37	1.9	372.37	0.6	M_CC45	**2626.80**	58.7	2617.54	39.7
M_DD10	271.96	3.8	271.96	1.0	M_DD45	2644.57	53.1	2644.57	29.0
M_AA15	520.80	2.4	520.80	0.9	M_AA50	**2385.28**	38.1	2382.86	19.1
M_BB15	554.24	4.2	554.24	1.6	M_BB50	2811.01	27.9	2811.01	19.8
M_CC15	684.31	4.6	684.31	1.9	M_CC50	**2905.82**	68.9	2900.53	53.2
M_DD15	606.48	7.1	606.48	2.1	M_DD50	2981.62	52.7	2981.62	38.8
M_AA20	706.36	9.2	706.36	3.3	M_AA55	2749.24	57.0	2749.24	25.4
M_BB20	843.94	9.1	843.94	2.9	M_BB55	3022.66	33.1	3022.66	15.4
M_CC20	965.00	14.1	965.00	4.5	M_CC55	3219.65	99.1	3219.65	65.7
M_DD20	942.38	15.2	942.38	7.0	M_DD55	3307.22	124.1	3307.22	68.6
M_AA25	1051.88	8.3	1051.88	3.4	M_AA60	2869.56	42.6	2869.56	32.5
M_BB25	1184.67	11.4	1184.67	4.0	M_BB60	3316.43	46.7	3316.43	24.0
M_CC25	1355.72	22.1	1355.72	8.4	M_CC60	3481.61	84.6	3481.61	81.7
M_DD25	1303.20	53.6	1303.20	17.1	M_DD60	3510.05	134.0	3510.05	86.7
M_AA30	1279.92	13.8	1279.92	5.6	M_AA65	3195.79	53.0	3195.79	35.0
M_BB30	1506.89	22.7	1506.89	6.3	M_BB65	3463.66	56.5	3463.66	27.0
M_CC30	**1689.40**	18.9	1687.69	11.0	M_CC65	3743.19	111.4	3743.19	85.9
M_DD30	1627.75	54.3	1627.75	24.4	M_DD65	3738.84	105.3	3738.84	83.8
M_AA35	1656.81	16.3	1656.81	5.8	M_AA70	3545.38	64.1	3545.38	45.9
M_BB35	1898.97	11.1	1898.97	5.9	M_BB70	3765.20	66.6	3765.20	36.2
M_CC35	2027.52	27.9	2027.52	17.2	M_CC70	4022.77	157.1	4022.77	117.6
M_DD35	**2000.10**	39.2	1999.89	22.4	M_DD70	4091.04	141.9	4091.04	136.4
M_AA40	1877.99	17.9	1877.99	8.2	M_AA75	3959.73	67.7	3959.73	50.4
M_BB40	2150.70	18.9	2150.70	7.9	M_BB75	4049.40	57.0	4049.40	34.1
M_CC40	2337.87	60.8	2337.87	25.2	M_CC75	4249.11	207.7	4249.11	134.5
M_DD40	2288.32	41.2	2288.32	29.7	M_DD75	4330.78	172.3	4330.78	119.7
						2267.16	49.2	2266.83	31.6

Obj_{EE}: Objective value of ALNS with Exact Evaluation.
Obj_{AE}: Objective value of ALNS with Approximate Evaluation.
$Time_{EE}$ (s): Processing Time of ALNS with Exact Evaluation.
$Time_{AE}$ (s): Processing Time of ALNS with Approximate Evaluation.

Table 15: Slow-speed case for single-vehicle instances

Instance	TIDobj	TDobj	Travel Cost	Profit	VioCost	VioRate	# Vio	Total	Time (s)
S_AA30	27.35	73.07	46.93	120	0.00	0.00	0	6	2.28
S_BB30	23.84	53.04	106.96	160	0.00	0.00	0	8	2.06
S_CC30	12.16	62.64	57.36	120	0.00	0.00	0	6	3.42
S_DD30	40.72	96.10	63.90	160	0.00	0.00	0	8	3.94
S_AA35	72.20	109.63	370.37	480	0.00	0.00	0	24	3.98
S_BB35	47.82	105.09	374.91	480	0.00	0.00	0	24	3.43
S_CC35	39.28	115.26	84.74	200	0.00	0.00	0	10	4.60
S_DD35	44.37	135.87	384.13	520	0.00	0.00	0	26	4.64
S_AA40	46.36	112.42	407.58	520	0.00	0.00	0	26	6.18
S_BB40	49.16	139.50	99.50	160	0.00	0.00	0	8	4.32
S_CC40	49.75	142.61	337.40	480	0.00	0.00	0	24	9.51
S_DD40	93.72	192.04	287.96	480	0.00	0.00	0	24	9.56
S_AA45	73.22	135.87	384.13	520	0.00	0.00	0	26	5.59
S_BB45	63.76	102.14	297.86	400	0.00	0.00	0	20	6.32
S_CC45	84.69	180.62	99.38	280	0.00	0.00	0	14	13.12
S_DD45	85.52	196.16	323.84	520	0.00	0.00	0	26	14.86
S_AA50	119.12	168.92	391.09	560	0.00	0.00	0	28	10.31
S_BB50	62.512	111.26	328.74	440	0.00	0.00	0	22	6.27
S_CC50	132.79	200.19	519.81	720	0.00	0.00	0	36	19.53
S_DD50	165.87	274.49	325.51	600	0.00	0.00	0	30	15.01
S_AA55	111.18	193.46	406.54	600	0.00	0.00	0	30	12.91
S_BB55	111.87	177.10	302.90	480	0.00	0.00	0	24	9.25
S_CC55	132.53	232.73	487.27	720	0.00	0.00	0	36	18.25
S_DD55	135.06	233.67	326.33	560	0.00	0.00	0	28	21.42
S_AA60	105.80	157.50	402.50	560	0.00	0.00	0	28	11.29
S_BB60	125.41	184.04	335.96	520	0.00	0.00	0	26	11.28
S_CC60	176.45	282.07	477.93	760	0.00	0.00	0	38	24.43
S_DD60	189.39	305.40	414.60	720	0.00	0.00	0	36	35.73
S_AA65	161.25	210.58	389.42	600	0.00	0.00	0	30	12.73
S_BB65	106.75	166.91	313.09	480	0.00	0.00	0	24	10.37
S_CC65	185.53	307.72	492.28	800	0.00	0.00	0	40	33.11
S_DD65	212.71	333.17	346.83	680	0.00	0.00	0	34	37.81
S_AA70	184.51	255.57	384.43	640	0.00	0.00	0	32	19.69
S_BB70	83.85	136.03	503.97	640	0.00	0.00	0	32	14.85
S_CC70	178.33	248.27	551.73	800	0.00	0.00	0	40	39.67
S_DD70	248.57	393.76	326.24	720	0.00	0.00	0	48	46.28
S_AA75	175.74	248.50	391.50	640	0.00	0.00	0	32	20.75
S_BB75	126.79	166.17	473.83	640	0.00	0.00	0	32	14.80
S_CC75	179.30	252.12	547.88	800	0.00	0.00	0	40	33.48
S_DD75	280.09	376.74	383.26	760	0.00	0.00	0	38	40.30
Average	113.63	189.21	338.76	526	0.00	0.00	0.00	26.60	15.43

Table 16: Average-speed case for single-vehicle instances

Instance	TIDobj	TDobj	Travel Cost	Profit	VioCost	VioRate	# Vio	Total	Time (s)
S_AA30	165.39	186.72	373.28	560	0.00	0.00	0	28	9.95
S_BB30	231.66	243.73	516.27	760	23.78	10.53	4	38	5.46
S_CC30	204.23	273.12	486.88	760	0.00	0.00	0	38	11.13
S_DD30	254.05	281.01	518.99	800	0.00	0.00	0	40	12.18
S_AA35	234.15	246.05	553.95	800	0.00	0.00	0	40	9.06
S_BB35	245.11	253.23	506.77	760	28.57	5.26	2	38	9.09
S_CC35	297.52	302.95	657.06	960	0.00	0.00	0	48	15.30
S_DD35	310.30	343.86	536.14	880	0.00	0.00	0	44	18.45
S_AA40	267.46	242.09	597.92	880	0.00	0.00	0	44	10.51
S_BB40	247.57	241.34	518.66	760	32.52	5.26	2	38	11.08
S_CC40	364.91	349.23	690.78	1040	0.00	0.00	0	52	27.27
S_DD40	351.18	396.27	523.73	920	0.00	0.00	0	46	40.30
S_AA45	261.42	302.78	657.22	960	0.00	0.00	0	48	18.56
S_BB45	318.38	307.69	572.31	880	14.59	11.36	5	44	13.81
S_CC45	394.12	371.31	668.69	1040	3.19	3.85	2	52	41.85
S_DD45	385.90	378.79	581.22	960	0.00	0.00	0	48	44.77
S_AA50	312.99	312.61	607.39	920	4.30	4.35	2	46	28.67
S_BB50	304.46	307.92	572.08	880	48.23	6.82	3	44	20.23
S_CC50	395.01	456.75	543.25	1000	0.00	0.00	0	50	51.67
S_DD50	458.96	440.42	639.58	1080	0.00	0.00	0	54	71.92
S_AA55	328.56	311.37	608.64	920	2.97	4.35	2	46	32.49
S_BB55	393.77	401.14	518.86	920	6.66	4.35	2	46	24.36
S_CC55	464.49	444.17	675.83	1120	36.32	7.14	4	56	61.16
S_DD55	456.20	470.63	569.37	1040	0.00	0.00	0	52	92.50
S_AA60	426.49	429.51	610.49	1040	41.47	7.69	4	52	30.41
S_BB60	404.30	414.52	585.48	1000	28.99	4.00	2	50	27.25
S_CC60	483.38	471.43	688.57	1160	22.18	3.45	2	58	62.39
S_DD60	544.17	533.22	666.78	1200	0.00	0.00	0	60	86.89
S_AA65	474.49	452.13	627.87	1080	0.00	0.00	0	54	42.31
S_BB65	406.72	385.10	574.90	960	19.94	4.17	2	48	27.08
S_CC65	549.37	503.70	656.27	1160	1.85	1.72	1	58	113.61
S_DD65	600.72	559.33	720.67	1280	28.00	3.13	2	64	124.07
S_AA70	496.28	503.10	576.90	1080	40.70	7.41	4	54	74.95
S_BB70	440.08	417.37	622.63	1040	46.48	9.62	5	52	34.75
S_CC70	572.02	530.37	709.63	1240	16.99	4.84	3	62	138.22
S_DD70	613.68	579.54	700.46	1280	13.17	1.56	1	64	141.01
S_AA75	534.31	528.40	631.60	1160	25.96	5.17	3	58	83.05
S_BB75	512.12	522.41	597.59	1120	11.18	3.57	2	56	50.87
S_CC75	577.63	570.34	709.66	1280	60.36	6.25	4	64	116.81
S_DD75	693.23	695.50	664.50	1360	0.00	0.00	0	68	206.60
Average	399.42	399.03	600.97	1001	13.96	3.15	1.58	50.05	51.05

Table 17: Fast-speed case for single-vehicle instances

Instance	TIDobj	TDobj	Travel Cost	Profit	VioCost	VioRate	# Vio	Total	Time (s)
S_AA30	289.39	274.86	525.14	800	32.658	20	8	40	8.84
S_BB30	343.43	282.34	637.66	920	336.303	45.652	21	46	9.76
S_CC30	336.13	156.62	723.38	880	16.305	4.545	2	44	19.54
S_DD30	383.33	279.62	760.38	1040	236.087	17.308	9	52	15.81
S_AA35	328.11	264.39	615.61	880	9.801	2.273	1	44	13.57
S_BB35	390.98	318.44	641.56	960	610.639	66.667	32	48	10.89
S_CC35	467.00	370.92	709.08	1080	127.74	20.37	11	54	26.25
S_DD35	355.70	348.52	731.48	1080	252.31	20.37	11	54	22.24
S_AA40	377.38	302.78	657.22	960	0.00	0	0	48	18.49
S_BB40	408.97	376.82	543.18	920	602.903	56.522	26	46	16.86
S_CC40	545.58	446.33	753.67	1200	235.197	18.333	11	60	51.93
S_DD40	540.32	431.01	728.99	1160	54.171	3.45	2	58	49.78
S_AA45	408.23	367.87	592.14	960	191.025	33.333	16	48	33.54
S_BB45	449.33	397.16	562.84	960	243.443	31.25	15	48	17.12
S_CC45	594.72	482.30	757.70	1240	336.518	24.194	15	62	49.83
S_DD45	577.65	497.76	662.24	1160	0	0	0	58	56.40
S_AA50	492.87	390.43	729.57	1120	907.049	60.714	34	56	31.14
S_BB50	515.30	359.04	620.96	1080	690.957	62.963	34	54	32.15
S_CC50	614.53	514.02	765.98	1280	488.48	34.375	22	64	99.73
S_DD50	702.26	602.83	757.17	1360	602.456	33.824	23	68	91.03
S_AA55	524.00	426.26	733.74	1160	892.799	58.621	34	58	48.43
S_BB55	604.99	549.23	650.77	1200	215.616	31.667	19	60	39.94
S_CC55	666.69	560.51	759.49	1320	663.48	39.394	26	66	107.10
S_DD55	741.57	632.57	807.43	1440	1339.48	45.833	33	72	114.91
S_AA60	586.28	487.04	712.96	1200	351.002	43.333	26	60	51.11
S_BB60	625.00	563.43	636.57	1200	459.565	51.667	31	60	34.53
S_CC60	717.75	616.76	783.24	1400	804.671	40	28	70	113.95
S_DD60	793.13	684.03	795.97	1480	1233.6	43.243	32	74	143.98
S_AA65	674.70	568.79	711.21	1280	756.415	59.375	38	64	73.52
S_BB65	627.91	553.94	646.06	1200	713.036	58.333	35	60	37.54
S_CC65	773.64	665.18	774.82	1440	456.123	25	18	72	116.13
S_DD65	904.52	800.84	799.16	1600	1212.63	42.5	34	80	167.50
S_AA70	688.11	612.83	707.17	1320	436.245	50	33	66	91.07
S_BB70	641.38	541.41	698.59	1240	688.515	50	31	62	61.36
S_CC70	820.10	716.18	803.82	1520	1537.35	63.158	48	76	211.48
S_DD70	879.60	623.15	736.85	1360	1150.89	70.588	48	68	188.76
S_AA75	799.34	695.15	744.85	1440	1692.00	80.555	58	72	142.87
S_BB75	728.73	623.15	736.85	1360	1150.89	70.588	48	68	88.67
S_CC75	842.20	729.81	790.19	1520	966.33	39.473	30	76	166.76
S_DD75	971.12	860.21	819.79	1680	1702.49	40.476	34	84	249.10
Average	593.30	499.36	708.14	1210	609.93	39.00	24.43	60.50	73.09

Table 18: Slow-speed case for multi-vehicle instances

Instance	TIDobj	TDobj	Travel Cost	Profit	VioCost	VioRate	# Vio	# Vehicle	Time (s)
M_AA30	913.82	1064.52	1065.48	2330	0.00	0.00	0	2	2.45
M_BB30	1084.27	1187.94	986.06	2374	0.00	0.00	0	2	2.58
M_CC30	1078.09	1322.74	923.26	2446	0.00	0.00	0	2	4.40
M_DD30	1094.40	1203.06	936.95	2340	0.00	0.00	0	2	5.37
M_AA35	1170.41	1335.06	1050.94	2586	0.00	0.00	0	2	3.47
M_BB35	1284.00	1409.99	934.01	2544	0.00	0.00	0	2	3.52
M_CC35	1362.11	1532.81	1065.19	2798	0.00	0.00	0	2	5.19
M_DD35	1379.91	1556.71	899.29	2656	0.00	0.00	0	2	7.45
M_AA40	1302.93	1451.34	1334.66	3086	0.00	0.00	0	3	6.35
M_BB40	1480.23	1603.69	1498.31	3402	0.00	0.00	0	3	3.84
M_CC40	1604.67	1924.07	1095.94	3320	0.00	0.00	0	3	12.24
M_DD40	1538.77	1690.47	1061.53	2952	0.00	0.00	0	2	9.33
M_AA45	1604.85	1728.93	1413.07	3442	0.00	0.00	0	3	5.44
M_BB45	1698.30	1841.30	1382.70	3524	0.00	0.00	0	3	7.10
M_CC45	1840.54	2188.09	1207.91	3696	0.00	0.00	0	3	13.74
M_DD45	1846.18	2006.99	1127.01	3334	0.00	0.00	0	2	14.26
M_AA50	1723.29	1865.52	1716.48	3882	0.00	0.00	0	3	7.28
M_BB50	1954.60	2077.66	1482.34	3860	0.00	0.00	0	3	7.31
M_CC50	1967.79	2279.75	1346.25	3926	0.00	0.00	0	3	13.62
M_DD50	2103.49	2432.29	1519.71	4352	0.00	0.00	0		17.84
M_AA55	1962.41	2157.16	1716.84	4174	0.00	0.00	0	3	9.46
M_BB55	2148.61	2252.67	1481.33	4034	0.00	0.00	0	3	9.46
M_CC55	2190.90	2724.70	1357.30	4482	0.00	0.00	0	4	22.71
M_DD55	2322.67	2599.76	1400.24	4300	0.00	0.00	0	3	31.02
M_AA60	2128.77	2332.19	1695.82	4328	0.00	0.00	0	3	11.79
M_BB60	2270.92	2478.17	1743.83	4622	0.00	0.00	0	4	12.76
M_CC60	2380.84	2830.19	1689.81	4920	0.00	0.00	0	4	23.36
M_DD60	2451.71	2866.04	1455.97	4622	0.00	0.00	0	3	23.65
M_AA65	2362.42	2511.32	2186.69	5098	0.00	0.00	0	4	15.78
M_BB65	2257.14	2390.30	1935.71	4726	0.00	0.00	0	4	14.96
M_CC65	2517.84	2951.70	1652.30	5004	0.00	0.00	0	4	29.31
M_DD65	2650.74	3112.04	1763.96	5276	0.00	0.00	0	4	30.90
M_AA70	2621.73	2799.95	2194.05	5394	0.00	0.00	0	4	18.42
M_BB70	2427.93	2594.72	1889.28	4884	0.00	0.00	0	4	16.43
M_CC70	2730.36	3311.02	1616.98	5328	0.00	0.00	0	4	31.18
M_DD70	2919.48	3353.15	1742.85	5496	0.00	0.00	0	4	40.55
M_AA75	2920.93	3161.56	2240.44	5802	0.00	0.00	0	4	24.42
M_BB75	2707.49	2985.15	1932.85	5318	0.00	0.00	0	4	20.41
M_CC75	2987.55	3406.59	2129.41	6036	0.00	0.00	0	5	45.33
M_DD75	3088.30	3585.34	1938.66	5924	0.00	0.00	0	4	32.17
Average	2002.03	2252.67	1495.28	4065	0.00	0.00	0.00	3.15	15.42

Table 19: Average-speed case for multi-vehicle instances

Instance	TIDobj	TDobj	Travel Cost	Profit	VioCost	VioRate	# Vio	# Vehicle	Time (s)
M_AA30	1325.96	1274.58	927.42	2502	24.12	6.90	4	3	3.59
M_BB30	1430.83	1409.47	902.53	2512	0.00	0.00	0	2	3.89
M_CC30	1586.90	1625.03	720.97	2546	0.00	0.00	0	2	9.59
M_DD30	1521.90	1497.23	680.77	2278	0.00	0.00	0	1	19.80
M_AA35	1695.80	1672.84	1165.16	3038	14.24	4.41	3	2	4.53
M_BB35	1580.09	1620.88	1043.12	2864	0.00	0.00	0	2	3.99
M_CC35	1952.04	1926.17	893.83	3020	0.00	0.00	0	2	13.41
M_DD35	1878.16	1886.46	975.55	3062	0.00	0.00	0	2	12.00
M_AA40	1862.18	1779.77	1256.23	3236	13.85	1.39	1	2	9.26
M_BB40	2092.19	2049.55	1092.45	3342	20.49	5.41	4	2	8.54
M_CC40	2232.45	2224.43	947.57	3372	0.00	0.00	0	2	22.71
M_DD40	2193.58	2188.86	1047.14	3436	0.00	0.00	0	2	19.48
M_AA45	2183.60	2077.04	1280.96	3558	4.27	2.50	2	2	10.91
M_BB45	2368.30	2368.92	1025.08	3594	112.21	12.50	10	2	14.31
M_CC45	2526.68	2420.04	1187.96	3808	26.39	2.27	2	2	27.72
M_DD45	2514.14	2539.16	1110.84	3850	0.00	0.00	0	2	20.24
M_AA50	2386.74	2336.33	1509.67	4146	18.68	5.21	5	3	12.92
M_BB50	2649.81	2598.14	1345.86	4244	72.25	7.14	7	3	13.27
M_CC50	2487.20	2888.31	1287.23	4213	0.00	0.00	0	3	53.70
M_DD50	2894.99	2877.78	1178.22	4256	0.00	0.00	0	2	45.65
M_AA55	2715.24	2628.63	1583.37	4512	110.23	7.69	8	3	18.80
M_BB55	2935.70	2869.30	1436.70	4606	53.39	5.66	6	3	14.54
M_CC55	3054.51	2937.97	1338.03	4576	3.21	1.92	2	3	45.30
M_DD55	3108.87	2968.36	1367.64	4536	28.89	1.89	2	2	47.97
M_AA60	2955.86	2831.07	1860.93	4992	64.75	6.03	7	3	26.37
M_BB60	3206.82	3162.21	1491.79	4954	62.46	3.51	4	3	25.83
M_CC60	3291.42	3377.33	1246.67	4924	0.00	0.00	0	3	45.33
M_DD60	3347.42	3362.66	1399.34	5062	0.00	0.00	0	3	54.37
M_AA65	3289.72	3224.84	1839.16	5364	124.09	5.47	7	3	32.03
M_BB65	3268.49	3241.58	1754.42	5396	50.91	3.85	5	4	25.58
M_CC65	3555.25	3434.19	1619.81	5354	34.95	3.13	4	3	72.67
M_DD65	3560.63	3400.23	1673.77	5374	18.36	0.78	1	3	75.90
M_AA70	3637.35	3586.97	1825.03	5712	37.14	4.41	6	3	33.06
M_BB70	3517.17	3422.56	1821.44	5644	70.22	2.99	4	4	36.83
M_CC70	3818.56	3858.81	1551.19	5710	0.00	0.00	0	3	95.02
M_DD70	3889.32	3917.36	1556.64	5774	0.00	0.00	0	3	70.59
M_AA75	3994.22	3930.98	1857.02	6088	165.10	5.48	8	3	40.30
M_BB75	3847.42	3750.28	1887.72	6038	107.07	7.75	11	4	36.40
M_CC75	4042.21	3813.29	1926.71	6140	16.60	1.37	2	4	111.91
M_DD75	4161.67	4148.74	1695.26	6144	0.00	0.00	0	3	77.46
Average	2764.03	2728.21	1357.78	4344	31.35	2.74	2.88	2.65	32.89

Table 20: Fast-Speed case for multi-vehicle instances

Instance	TIDobj	TDobj	Travel Cost	Profit	VioCost	VioRate	# Vio	# Vehicle	Time (s)
M_AA30	1428.60	1297.47	904.53	2502	120.79	15.52	9	3	6.47
M_BB30	1614.41	1532.91	643.09	2276	475.11	58.00	29	1	8.37
M_CC30	1712.09	1623.33	660.68	2384	0.00	0.00	0	1	18.73
M_DD30	1769.13	1650.92	761.08	2512	142.28	12.07	7	1	16.19
M_AA35	1793.67	1624.82	1137.18	3062	13.87	5.71	4	3	6.17
M_BB35	1977.19	1889.29	972.71	3062	8.42	2.86	2	2	5.86
M_CC35	2107.42	2014.59	847.41	3062	0.00	0.00	0	2	17.72
M_DD35	2093.21	1867.75	994.25	3062	103.97	7.14	5	2	17.98
M_AA40	2053.81	1928.32	1307.68	3436	181.93	17.50	14	2	8.93
M_BB40	2283.68	2175.31	1060.69	3436	404.53	41.25	33	2	8.96
M_CC40	2429.10	2214.50	1021.50	3436	56.81	2.50	2	2	24.84
M_DD40	2369.63	2192.79	1043.21	3436	419.27	21.25	17	2	32.31
M_AA45	2456.77	2318.50	1299.50	3818	60.88	6.82	6	2	11.76
M_BB45	2685.81	2570.70	1045.30	3816	320.78	32.95	29	2	14.20
M_CC45	2774.93	2620.23	1029.77	3850	320.26	14.44	13	2	41.28
M_DD45	2719.30	2563.60	1086.40	3850	110.52	4.44	4	2	35.60
M_AA50	2655.40	2510.81	1373.19	4084	326.41	23.40	22	2	22.62
M_BB50	2935.83	2817.76	1112.24	4130	448.84	37.23	35	2	22.15
M_CC50	2995.87	2835.74	1050.26	4086	264.90	13.54	13	2	29.32
M_DD50	3078.65	2961.31	1116.69	4278	0.00	0.00	0	2	52.70
M_AA55	3108.05	2941.80	1398.20	4540	812.67	33.96	36	2	30.93
M_BB55	3179.00	3032.31	1259.69	4592	578.09	41.51	44	3	24.55
M_CC55	3289.18	3028.50	1347.51	4676	31.58	1.82	2	3	60.82
M_DD55	3404.65	3188.51	1287.49	4676	86.30	2.73	3	2	69.35
M_AA60	3240.73	3017.67	1682.33	5000	119.31	10.35	12	3	32.25
M_BB60	3482.70	3312.55	1415.45	5028	264.80	17.80	21	3	31.13
M_CC60	3600.29	3263.30	1498.70	5062	64.46	1.67	2	3	73.56
M_DD60	3684.32	3525.28	1336.72	5062	453.77	11.67	14	2	100.74
M_AA65	3488.66	3310.81	1621.19	5232	665.85	35.48	44	3	37.57
M_BB65	3661.17	3444.13	1523.87	5268	1126.93	48.39	60	3	35.46
M_CC65	3845.22	3625.95	1448.05	5374	170.56	3.13	4	3	92.97
M_DD65	3950.15	3705.71	1444.29	5350	450.43	19.05	24	2	88.42
M_AA70	3809.69	3611.81	1862.19	5774	491.17	19.29	27	3	50.49
M_BB70	3907.50	3731.73	1576.27	5608	973.67	43.94	58	3	43.33
M_CC70	4153.91	3942.42	1531.58	5774	304.87	6.43	9	3	126.65
M_DD70	4195.36	3858.55	1615.45	5774	86.38	3.57	5	3	106.67
M_AA75	4219.23	3937.34	1936.66	6204	1077.59	30.67	46	3	57.55
M_BB75	4217.97	4022.65	1611.35	5934	865.78	37.68	52	3	61.47
M_CC75	4470.72	4182.44	1679.56	6162	282.08	6.08	9	3	113.90
M_DD75	4487.25	4207.98	1696.02	6204	642.42	19.33	29	3	123.14
Average	3033.26	2852.55	1281.00	4372	333.21	17.78	18.63	2.38	44.08

Table 21: Comparison with a Deterministic Method by using R_0

Instances	Deterministic			Stochastic				
	1^{st}	2^{nd}	Obj.	1^{st}	2^{nd}	Obj.	Time (s)	Imp (%)
cdp101-25	2811.6	555.7	3367.3	2867.4	5.8	2873.1	41.4	14.7
cdp102-25	2749.9	281.9	3031.8	2769.8	3.2	2773.0	78.1	8.5
cdp103-25	2734.3	2927.8	5662.1	2841.6	25.8	2867.4	168.8	49.4
cdp201-25	3035.6	240.6	3276.3	3040.4	0.2	3040.6	41.6	7.2
cdp202-25	2812.1	4020.4	6832.5	2834.3	0.4	2834.7	130.0	58.5
cdp203-25	2789.3	991.5	3780.8	2819.1	8.8	2827.8	225.8	25.2
rcdp101-25	1355.3	3500.3	4855.5	1499.8	102.6	1602.5	51.5	67.0
rcdp102-25	1318.1	6139.1	7457.1	1494.8	11.2	1506.0	82.1	79.8
rcdp103-25	1330.4	4437.1	5767.5	1474.3	45.3	1519.6	119.2	73.7
rcdp201-25	1644.8	3818.4	5463.2	1769.2	37.6	1806.8	126.8	66.9
rcdp202-25	1445.9	4458.4	5904.3	1301.2	5.6	1306.8	74.7	77.9
rcdp203-25	1296.7	2537.9	3834.6	1402.2	77.9	1480.0	208.5	61.4
rdp101-25	1708.3	0.0	1708.3	1708.3	0.0	1708.3	39.6	0.0
rdp102-25	1406.3	17.3	1423.6	1406.3	17.3	1423.6	62.3	0.0
rdp103-25	1210.5	2733.4	3943.9	1223.1	43.0	1266.1	69.1	67.9
rdp201-25	1337.6	2382.1	3719.7	1415.6	4.2	1419.8	80.5	61.8
rdp202-25	1232.7	4486.9	5719.7	1299.1	4.8	1303.9	168.5	77.2
rdp203-25	1144.9	5973.7	7118.6	1243.9	64.9	1308.8	215.5	81.6
cdp101-50	5644.5	4705.4	10349.9	5703.0	13.3	5716.3	127.3	44.8
cdp102-50	5585.1	3633.1	9218.2	5696.0	28.8	5724.8	308.7	37.9
cdp103-50	5548.1	1711.7	7259.8	5685.4	7.3	5692.7	560.5	21.6
cdp201-50	5807.3	5555.5	11362.8	5947.2	3.4	5950.6	241.6	47.6
cdp202-50	5605.3	10181.9	15787.2	5753.1	11.9	5765.0	601.3	63.5
cdp203-50	5591.3	8992.5	14583.8	5700.0	6.2	5706.1	907.1	60.9
rcdp101-50	2793.1	14887.4	17680.5	3278.9	138.2	3417.1	139.1	80.7
rcdp102-50	2705.8	14095.5	16801.3	3185.7	212.9	3398.5	209.8	79.8
rcdp103-50	2683.7	19016.8	21700.4	3308.5	133.2	3441.7	309.5	84.1
rcdp201-50	3164.8	10452.2	13617.0	3625.8	76.0	3701.8	413.8	72.8
rcdp202-50	2846.0	17333.5	20179.5	3354.7	205.9	3560.6	693.6	82.4
rcdp203-50	2786.8	17458.7	20245.5	3171.5	267.9	3439.4	812.1	83.0
rdp101-50	3054.0	226.2	3280.2	3082.9	5.4	3088.3	82.7	5.9
rdp102-50	2549.5	967.8	3517.3	2571.3	23.6	2594.9	140.0	26.2
rdp103-50	2821.7	2150.5	4972.2	2361.9	31.4	2393.3	173.1	51.9
rdp201-50	2497.8	3161.8	5659.6	2616.5	10.4	2626.9	321.4	53.6
rdp202-50	2284.4	5681.1	7965.5	2451.1	74.5	2525.6	616.5	68.3
rdp203-50	2155.8	6174.7	8330.5	2386.4	105.4	2491.8	881.1	70.1
cdp101-100	11432.0	8288.1	19720.1	12051.56	85.7	12137.26	447.9	38.5
cdp102-100	11332.8	8355.7	19688.5	11665.86	81.4	11747.25	1137.7	40.3
cdp103-100	11257.7	13277.0	24534.6	11375.03	142.4	11517.48	2157.9	53.1
cdp201-100	11500.4	7714.2	19214.6	11617.83	61.9	11679.69	1158.8	39.2
cdp202-100	11235.1	17086.9	28322.1	11545.95	106.5	11652.47	2771.9	58.9
cdp203-100	11175.8	18937.5	30113.3	11429.31	120.7	11549.98	4383.1	61.6
rcdp101-100	4330.1	708.5	5038.6	4347.98	13.7	4361.68	448.41	13.4
rcdp102-100	3959.4	16650.6	20610.0	5492.51	281.5	5773.97	650.3	72.0
rcdp103-100	4977.1	15603.0	20580.1	5543.96	225.1	5769.01	1118.9	72.0
rcdp201-100	5532.6	9054.9	14587.6	5971.84	132.6	6104.47	1485.2	58.2
rcdp202-100	5026.8	18732.0	23758.8	5608.89	370.8	5979.64	2260.7	74.8
rcdp203-100	4749.3	16325.3	21074.6	5232.3	389.4	5621.68	2713.9	73.3
rdp101-100	5294.4	4394.2	9688.5	5383.81	65.0	5448.82	436.67	43.8
rdp102-100	4651.6	10494.6	15146.2	4977.5	172.8	5150.26	526.7	66.0
rdp103-100	4350.1	8943.2	13293.3	4459.31	258.4	4717.71	821.1	64.5
rdp201-100	4640.5	13411.9	18052.3	5016.54	172.8	5189.34	1464.6	71.3
rdp202-100	4289.5	8967.5	13256.9	4660.1	144.8	4804.92	2304.1	63.8
rdp203-100	4263.9	15504.6	19768.6	4716.91	286.2	5003.1	3948.8	74.7
Average	4138.7	7561.8	11700.5	4359.0	91.2	4450.2	736.3	54.7

Table 22: Comparison with a Deterministic Method by using R_1

Instances	Deterministic			Stochastic				
	1^{st}	2^{nd}	Obj.	1^{st}	2^{nd}	Obj.	Time (s)	Imp (%)
cdp101-25	2811.6	20.7	2832.3	2811.6	20.7	2832.3	14.8	0.0
cdp102-25	2749.9	272.6	3022.5	2742.4	28.9	2771.4	31.2	8.3
cdp103-25	2734.3	479.7	3214.0	2745.4	62.9	2808.3	66.0	12.6
cdp201-25	3035.6	15.9	3051.6	3040.4	5.3	3045.7	8.0	0.2
cdp202-25	2812.1	10542.0	13354.1	2828.2	0.2	2828.4	52.7	78.8
cdp203-25	2789.3	6581.0	9370.2	2793.8	35.5	2829.3	74.0	69.8
rcdp101-25	1355.3	6065.5	7420.8	1499.8	81.0	1580.8	20.2	78.7
rcdp102-25	1318.1	4851.7	6169.8	1491.7	31.8	1523.5	34.3	75.3
rcdp103-25	1330.4	6646.7	7977.1	1457.0	59.6	1516.6	40.2	81.0
rcdp201-25	1644.8	1559.7	3204.5	1660.5	106.2	1766.7	50.7	44.9
rcdp202-25	1445.9	1253.7	2699.6	1279.1	13.0	1292.1	34.1	52.1
rcdp203-25	1296.7	968.4	2265.2	1407.4	10.9	1418.3	64.6	37.4
rdp101-25	1708.3	0.0	1708.3	1708.3	0.0	1708.3	10.4	0.0
rdp102-25	1406.3	20.7	1427.0	1404.3	21.0	1425.2	18.7	0.1
rdp103-25	1210.5	2058.5	3269.0	1232.2	11.6	1243.8	24.3	62.0
rdp201-25	1337.6	1196.4	2533.9	1361.3	3.6	1364.9	29.2	46.1
rdp202-25	1232.7	1049.0	2281.8	1238.6	63.7	1302.3	73.1	42.9
rdp203-25	1144.9	734.6	1879.5	1235.8	8.7	1244.4	113.3	33.8
cdp101-50	5644.5	3084.7	8729.1	5652.3	89.9	5742.3	109.6	34.2
cdp102-50	5585.1	3886.0	9471.1	5606.7	63.2	5669.9	304.1	40.1
cdp103-50	5548.1	2799.6	8347.7	5629.3	9.3	5638.6	614.3	32.5
cdp201-50	5807.3	1755.8	7563.1	5892.0	45.9	5937.9	267.1	21.5
cdp202-50	5605.3	12196.8	17802.2	5745.5	17.8	5763.3	851.1	67.6
cdp203-50	5591.3	2311.3	7902.6	5699.9	1.5	5701.4	1333.4	27.9
rcdp101-50	2793.1	39582.4	42375.6	3306.4	286.0	3592.4	167.4	91.5
rcdp102-50	2705.8	36273.2	38979.0	3269.5	349.5	3618.9	258.9	90.7
rcdp103-50	2683.7	32025.4	34709.1	3292.1	273.8	3565.8	441.3	89.7
rcdp201-50	3164.8	4572.1	7736.9	3446.7	266.9	3713.6	563.8	52.0
rcdp202-50	2846.0	17563.3	20409.2	3130.3	229.3	3359.6	858.2	83.5
rcdp203-50	2786.8	18130.9	20917.8	3267.1	132.9	3400.0	1296.6	83.7
rdp101-50	3054.0	139.4	3193.4	3090.0	62.4	3152.4	73.7	1.3
rdp102-50	2549.5	3565.8	6115.3	2582.1	24.3	2606.4	126.8	57.4
rdp103-50	2821.7	8711.2	11533.0	2381.7	33.5	2415.2	179.6	79.1
rdp201-50	2497.8	7435.6	9933.4	2604.0	39.5	2643.5	338.8	73.4
rdp202-50	2284.4	4879.3	7163.7	2410.7	99.6	2510.3	752.8	65.0
rdp203-50	2155.8	4559.0	6714.8	2364.6	30.8	2395.3	975.2	64.3
cdp101-100	11432.0	23287.5	34719.5	11807.3	46.0	11853.3	430.9	65.9
cdp102-100	11332.8	18464.8	29797.5	11539.2	55.2	11594.4	1252.6	61.1
cdp103-100	11257.7	4691.7	15949.4	11441.1	58.6	11499.7	2053.6	27.9
cdp201-100	11500.4	27033.4	38533.7	11835.0	59.6	11894.6	1542.5	69.1
cdp202-100	11235.1	12524.0	23759.2	11460.8	66.0	11526.8	3480.3	51.5
cdp203-100	11175.8	23436.7	34612.6	11470.1	137.5	11607.7	5953.1	66.5
rcdp101-100	4330.1	1505.0	5835.1	4319.8	9.0	4328.8	378.9	25.8
rcdp102-100	3959.4	9473.2	13432.6	5683.2	663.0	6346.3	779.5	52.8
rcdp103-100	4977.1	32729.6	37706.7	5425.8	319.4	5745.2	1340.1	84.8
rcdp201-100	5532.6	31982.1	37514.8	6043.4	364.9	6408.4	1957.1	82.9
rcdp202-100	5026.8	21963.1	26990.0	5679.6	349.6	6029.2	3010.2	77.7
rcdp203-100	4749.3	19123.2	23872.5	5060.4	567.8	5628.2	3965.3	76.4
rdp101-100	5294.4	6565.2	11859.6	5330.1	244.0	5574.2	298.3	53.0
rdp102-100	4651.6	22036.2	26687.8	4966.3	231.3	5197.6	507.1	80.5
rdp103-100	4350.1	17349.8	21699.9	4685.8	282.4	4968.2	858.6	77.1
rdp201-100	4640.5	8397.9	13038.3	4979.2	320.0	5299.2	1844.6	59.4
rdp202-100	4289.5	7681.5	11971.0	4441.2	348.2	4789.4	2822.2	60.0
rdp203-100	4263.9	10392.6	14656.5	4595.6	150.0	4745.6	5061.2	67.6
Average	4138.7	10119.0	14257.7	4334.7	127.7	4462.3	885.3	54.1

Table 23: Comparison with a Deterministic Method by using R_2

Instances	Deterministic			Stochastic				
	1^{st}	2^{nd}	Obj.	1^{st}	2^{nd}	Obj.	Time (s)	Imp (%)
cdp101-25	2811.6	2790.7	5602.2	2867.4	25.3	2892.6	43.9	48.4
cdp102-25	2749.9	2064.9	4814.9	2769.4	8.7	2778.1	105.1	42.3
cdp103-25	2734.3	449.0	3183.3	2763.3	94.2	2857.5	301.8	10.2
cdp201-25	3035.6	73.7	3109.3	3040.4	9.4	3049.9	25.8	1.9
cdp202-25	2812.1	13933.3	16745.4	2834.3	5.9	2840.2	128.6	83.0
cdp203-25	2789.3	1182.6	3971.9	2793.8	30.8	2824.6	242.7	28.9
rcdp101-25	1355.3	3164.0	4519.3	1506.7	94.2	1600.9	63.7	64.6
rcdp102-25	1318.1	1817.7	3135.8	1480.7	37.5	1518.2	100.7	51.6
rcdp103-25	1330.4	2863.4	4193.8	1454.8	44.3	1499.1	148.9	64.3
rcdp201-25	1644.8	4109.7	5754.5	1761.1	34.2	1795.3	186.2	68.8
rcdp202-25	1445.9	348.1	1794.0	1243.9	37.6	1281.5	96.4	28.6
rcdp203-25	1296.7	1093.7	2390.4	1408.9	23.0	1431.9	229.2	40.1
rdp101-25	1708.3	0.0	1708.3	1708.3	0.0	1708.3	40.9	0.0
rdp102-25	1406.3	20.7	1427.0	1404.3	20.7	1425.0	65.9	0.1
rdp103-25	1210.5	879.8	2090.3	1227.0	9.9	1236.9	79.5	40.8
rdp201-25	1337.6	2133.5	3471.1	1394.9	40.6	1435.5	94.5	58.6
rdp202-25	1232.7	822.7	2055.4	1271.7	23.0	1294.8	211.1	37.0
rdp203-25	1144.9	926.5	2071.44	1232.6	30.6	1263.2	346.5	39.0
cdp101-50	5644.5	6910.7	12555.2	5755.1	8.0	5763.1	159.8	54.1
cdp102-50	5585.1	4451.3	10036.4	5635.7	102.3	5738.0	380.3	42.8
cdp103-50	5548.1	12726.2	18274.3	5635.3	28.5	5663.8	751.4	69.0
cdp201-50	5807.3	29848.5	35655.9	5910.0	35.9	5945.9	317.3	83.3
cdp202-50	5605.3	34968.6	40573.9	5788.2	8.5	5796.7	723.3	85.7
cdp203-50	5591.3	1297.9	6889.2	5695.0	30.3	5725.3	1120.1	16.9
rcdp101-50	2793.1	19019.8	21812.9	3279.1	193.9	3473.0	218.1	84.1
rcdp102-50	2705.8	11634.4	14340.2	3339.8	143.6	3483.4	291.3	75.7
rcdp103-50	2683.7	10655.2	13338.8	3292.9	158.4	3451.2	516.3	74.1
rcdp201-50	3164.8	2449.9	5614.6	3506.7	136.8	3643.5	717.3	35.1
rcdp202-50	2846.0	5506.4	8352.4	3081.0	266.8	3347.8	1202.6	59.9
rcdp203-50	2786.8	9128.6	11915.4	3042.4	268.8	3311.2	1535.2	72.2
rdp101-50	3054.0	111.3	3165.4	3064.3	41.0	3105.3	110.3	1.9
rdp102-50	2549.5	2085.1	4634.6	2579.0	14.4	2593.4	171.2	44.0
rdp103-50	2821.7	4760.4	7582.2	2329.8	56.8	2386.7	228.0	68.5
rdp201-50	2497.8	13484.6	15982.5	2606.1	43.5	2649.6	542.9	83.4
rdp202-50	2284.4	5265.3	7549.7	2469.4	48.6	2518.0	919.9	66.6
rdp203-50	2155.8	5732.5	7888.2	2322.7	119.3	2441.9	905.5	69.0
cdp101-100	11432.0	63259.7	74691.6	11842.0	215.8	12057.8	551.2	83.9
cdp102-100	11332.8	35467.3	46800.1	11753.9	180.1	11934.0	1508.6	74.5
cdp103-100	11257.7	26952.6	38210.3	11517.1	63.4	11580.4	2315.7	69.7
cdp201-100	11500.4	72271.0	83771.3	11754.5	247.6	12002.0	1552.8	85.7
cdp202-100	11235.1	65057.3	76292.4	11649.8	608.7	12258.5	3381.7	83.9
cdp203-100	11175.8	19715.4	30891.2	11366.6	339.0	11705.6	5770.1	62.1
rcdp101-100	4330.1	669.0	4999.1	4337.2	28.6	4365.8	150.6	12.7
rcdp102-100	3959.4	1938.7	5898.1	3924.1	39.6	3963.6	939.3	32.8
rcdp103-100	4977.1	16084.7	21061.8	5511.9	486.5	5998.3	1457.5	71.5
rcdp201-100	5532.6	21379.7	26912.3	6097.7	325.6	6423.2	1941.6	76.1
rcdp202-100	5026.8	12160.2	17187.0	5354.3	399.4	5753.7	3073.4	66.5
rcdp203-100	4749.3	11186.5	15935.8	5134.6	258.9	5393.5	2918.3	66.2
rdp101-100	5294.4	6666.1	11960.5	5518.1	34.0	5552.1	150.6	53.6
rdp102-100	4651.6	12453.8	17105.4	5131.4	183.9	5315.3	749.8	68.9
rdp103-100	4350.1	10699.8	15049.9	4660.3	119.5	4779.8	917.8	68.2
rdp201-100	4640.5	6109.7	10750.2	4894.6	251.4	5146.1	2350.8	52.1
rdp202-100	4289.5	5583.8	9873.2	4544.1	153.3	4697.3	2922.8	52.4
rdp203-100	4263.9	5863.3	10127.2	4435.2	607.5	5042.7	6185.4	50.2
Average	4138.7	11337.6	15476.2	4313.4	126.2	4439.6	965.9	54.2

Table 24: Comparison with a Deterministic Method by using R_3

Instances	Deterministic			Stochastic				
	1^{st}	2^{nd}	Obj.	1^{st}	2^{nd}	Obj.	Time (s)	Imp (%)
cdp101-25	2811.6	20.2	2831.8	2811.6	20.2	2831.8	33.6	0.0
cdp102-25	2749.9	155.3	2905.2	2742.7	2.9	2745.6	84.5	5.5
cdp103-25	2734.3	253.2	2987.5	2736.8	10.5	2747.3	217.9	8.0
cdp201-25	3035.6	9.0	3044.6	3035.6	8.0	3043.6	54.6	0.0
cdp202-25	2812.1	8726.1	11538.2	2823.3	5.0	2828.3	156.0	75.5
cdp203-25	2789.3	4080.0	6869.3	2793.8	34.1	2827.9	225.4	58.8
rcdp101-25	1355.3	5023.0	6378.2	1488.4	2.3	1490.7	48.9	76.6
rcdp102-25	1318.1	4142.4	5460.5	1444.9	5.4	1450.3	68.3	73.4
rcdp103-25	1330.4	6265.8	7596.2	1371.0	53.7	1424.7	94.4	81.2
rcdp201-25	1644.8	1385.1	3029.8	1678.1	27.2	1705.3	119.7	43.7
rcdp202-25	1445.9	1136.3	2582.2	1263.5	14.8	1278.2	167.9	50.5
rcdp203-25	1296.7	446.3	1743.0	1329.1	31.6	1360.7	182.1	21.9
rdp101-25	1708.3	0.0	1708.3	1708.3	0.0	1708.3	24.5	0.0
rdp102-25	1406.3	10.5	1416.8	1404.3	9.5	1413.8	46.5	0.2
rdp103-25	1210.5	1849.4	3059.9	1227.0	15.7	1242.7	65.5	59.4
rdp201-25	1337.6	1114.0	2451.5	1361.3	1.3	1362.6	87.5	44.4
rdp202-25	1232.7	993.0	2225.7	1244.3	0.8	1245.1	201.1	44.1
rdp203-25	1144.9	502.1	1647.0	1169.7	41.6	1211.3	326.8	26.5
cdp101-50	5644.5	1806.8	7451.3	5708.9	5.8	5714.8	114.1	23.3
cdp102-50	5585.1	2718.3	8303.4	5611.2	19.0	5630.2	306.7	32.2
cdp103-50	5548.1	1426.1	6974.2	5601.6	13.3	5614.9	555.4	19.5
cdp201-50	5807.3	1643.9	7451.3	5911.9	18.8	5930.8	259.9	20.4
cdp202-50	5605.3	11743.7	17349.0	5736.4	70.7	5807.1	872.3	66.5
cdp203-50	5591.3	2116.3	7707.6	5622.1	52.2	5674.3	1365.1	26.4
rcdp101-50	2793.1	35753.9	38547.0	3238.9	198.6	3437.5	146.5	91.1
rcdp102-50	2705.8	35027.3	37733.1	3171.9	189.5	3361.4	225.1	91.1
rcdp103-50	2683.7	30727.4	33411.1	3176.1	199.5	3375.6	415.2	89.9
rcdp201-50	3164.8	3889.4	7054.2	3512.7	175.4	3688.1	605.7	47.7
rcdp202-50	2846.0	16701.4	19547.4	3199.2	71.6	3270.8	1003.8	83.3
rcdp203-50	2786.8	15962.2	18749.0	3066.0	165.2	3231.2	1358.7	82.8
rdp101-50	3054.0	89.3	3143.4	3050.8	62.7	3113.5	76.6	0.9
rdp102-50	2549.5	3216.8	5766.4	2590.0	17.9	2607.9	162.1	54.8
rdp103-50	2821.7	7815.5	10637.2	2330.7	92.8	2423.5	178.1	77.2
rdp201-50	2497.8	6739.8	9237.6	2561.0	27.1	2588.1	365.5	72.0
rdp202-50	2284.4	3796.7	6081.1	2387.9	9.6	2397.6	874.5	60.6
rdp203-50	2155.8	3931.4	6087.2	2338.6	11.6	2350.2	1182.6	61.4
cdp101-100	11432.0	16921.3	28353.3	11706.9	395.1	12102.0	546.3	57.3
cdp102-100	11332.8	10554.3	21887.1	11653.5	46.7	11700.2	1251.3	46.5
cdp103-100	11257.7	2794.2	14051.8	11425.3	135.8	11561.1	2113.9	17.7
cdp201-100	11500.4	21278.8	32779.2	11606.0	1479.5	13085.4	1685.7	60.1
cdp202-100	11235.1	9883.8	21118.9	11528.8	204.9	11733.7	4124.8	44.4
cdp203-100	11175.8	20357.5	31533.3	11471.2	314.1	11785.3	5942.9	62.6
rcdp101-100	4330.1	1419.4	5749.4	4363.0	92.4	4455.4	161.6	22.5
rcdp102-100	3959.4	8585.1	12544.4	3916.4	103.1	4019.5	319.0	68.0
rcdp103-100	4977.1	31167.5	36144.6	5651.5	286.5	5938.0	1428.1	83.6
rcdp201-100	5532.6	30037.7	35570.3	6154.4	522.3	6676.7	2190.2	81.2
rcdp202-100	5026.8	20979.8	26006.7	5606.2	735.5	6341.7	3254.5	75.6
rcdp203-100	4749.3	16821.2	21570.5	5277.5	362.1	5639.6	3502.6	73.9
rdp101-100	5294.4	6058.3	11352.7	5379.9	188.5	5568.4	199.6	51.0
rdp102-100	4651.6	18601.1	23252.7	5016.3	319.6	5336.0	574.1	77.1
rdp103-100	4350.1	15619.3	19969.4	4633.7	252.8	4886.5	949.6	75.5
rdp201-100	4640.5	6862.2	11502.7	4919.6	93.0	5012.6	2342.0	56.4
rdp202-100	4289.5	6900.9	11190.4	4616.2	128.1	4744.3	2668.6	57.6
rdp203-100	4263.9	8998.4	13262.4	4485.8	185.6	4671.3	5178.3	64.8
Average	4138.7	8797.4	12936.0	4293.7	139.5	4433.2	939.0	50.9

Table 25: Comparison with a Deterministic Method by using R_0 with α =100

Instances	Deterministic			Stochastic				
	1^{st}	2^{nd}	Obj.	1^{st}	2^{nd}	Obj.	Time (s)	Imp (%)
cdp101-25	2811.6	124.2	2935.7	2867.4	1.4	2868.8	38.6	2.3
cdp102-25	2749.9	611.0	3360.9	2766.5	1.2	2767.8	83.8	17.6
cdp103-25	2734.3	722.5	3456.8	2842.1	5.8	2847.9	179.4	17.6
cdp201-25	3035.6	53.5	3089.1	3040.4	0.1	3040.5	25.8	1.6
cdp202-25	2812.1	885.9	3698.0	2834.3	0.1	2834.4	126.0	23.4
cdp203-25	2789.3	451.5	3240.7	2793.8	8.5	2802.3	201.9	13.5
rcdp101-25	1355.3	799.6	2154.8	1492.1	30.4	1522.5	48.0	29.3
rcdp102-25	1318.1	1037.4	2355.5	1474.6	7.1	1481.7	69.1	37.1
rcdp103-25	1330.4	1547.7	2878.2	1453.7	12.0	1465.7	94.3	49.1
rcdp201-25	1644.8	1203.9	2848.6	1730.8	38.4	1769.2	136.8	37.9
rcdp202-25	1445.9	975.2	2421.1	1235.2	62.8	1298.0	64.7	46.4
rcdp203-25	1296.7	555.9	1852.7	1414.6	28.8	1443.4	170.7	22.1
rdp101-25	1708.3	0.0	1708.3	1708.3	0.0	1708.3	32.9	0.0
rdp102-25	1406.3	4.2	1410.5	1406.3	4.2	1410.5	57.3	0.0
rdp103-25	1210.5	695.9	1906.4	1223.1	10.6	1233.6	77.3	35.3
rdp201-25	1337.6	273.0	1610.6	1350.0	12.6	1362.6	96.2	15.4
rdp202-25	1232.7	802.5	2035.2	1298.6	1.3	1299.9	185.8	36.1
rdp203-25	1144.9	1067.8	2212.7	1228.9	21.7	1250.6	211.8	43.5
cdp101-50	5644.5	715.4	6359.9	5774.8	0.8	5775.5	144.8	9.2
cdp102-50	5585.1	1444.3	7029.4	5650.7	52.0	5702.7	263.0	18.9
cdp103-50	5548.1	1541.8	7089.8	5573.9	32.1	5606.0	730.6	20.9
cdp201-50	5807.3	1424.3	7231.6	5912.2	9.4	5921.6	276.8	18.1
cdp202-50	5605.3	1988.1	7593.4	5736.3	15.8	5752.0	626.6	24.2
cdp203-50	5591.3	1392.7	6984.0	5721.0	7.8	5728.8	843.5	18.0
rcdp101-50	2793.1	4010.3	6803.5	3170.8	192.1	3362.9	166.7	50.6
rcdp102-50	2705.8	3472.7	6178.5	3176.3	65.0	3241.3	258.6	47.5
rcdp103-50	2683.7	3752.1	6435.8	3218.9	67.1	3286.0	336.5	48.9
rcdp201-50	3164.8	1849.9	5014.7	3459.6	83.5	3543.1	458.6	29.3
rcdp202-50	2846.0	3499.7	6345.7	3267.9	57.0	3324.9	645.5	47.6
rcdp203-50	2786.8	4176.2	6963.0	3177.3	64.7	3242.0	964.0	53.4
rdp101-50	3054.0	16.8	3070.8	3064.0	5.9	3069.9	78.4	0.0
rdp102-50	2549.5	446.0	2995.5	2571.9	15.4	2587.3	171.6	13.6
rdp103-50	2821.7	741.2	3562.9	2313.8	40.8	2354.5	192.7	33.9
rdp201-50	2497.8	1877.7	4375.5	2570.7	41.8	2612.4	344.0	40.3
rdp202-50	2284.4	1233.6	3518.0	2423.0	37.5	2460.5	606.4	30.1
rdp203-50	2155.8	1914.1	4069.9	2368.1	38.4	2406.5	842.6	40.9
cdp101-100	11432.0	3652.1	15084.0	11863.0	28.9	11891.9	459.2	21.2
cdp102-100	11332.8	3098.5	14431.3	11494.5	31.8	11526.3	1235.8	20.1
cdp103-100	11257.7	3229.0	14486.7	11354.9	75.9	11430.7	2101.2	21.1
cdp201-100	11500.4	4044.7	15545.0	11840.3	27.7	11867.9	1090.1	23.7
cdp202-100	11235.1	3009.2	14244.3	11450.4	39.3	11489.7	2735.9	19.3
cdp203-100	11175.8	3284.6	14460.4	11469.7	17.8	11487.5	4154.1	20.6
rcdp101-100	4330.1	89.1	4419.2	4352.0	0.9	4352.9	162.7	1.5
rcdp102-100	3959.4	524.7	4484.1	3926.1	5.6	3931.7	227.7	12.3
rcdp103-100	4977.1	3963.9	8941.0	5340.5	250.3	5590.8	1000.0	37.5
rcdp201-100	5532.6	4304.8	9837.5	5836.7	254.1	6090.8	1382.2	38.1
rcdp202-100	5026.8	3920.5	8947.3	5584.1	258.9	5843.0	2150.3	34.7
rcdp203-100	4749.3	3617.8	8367.1	5152.2	217.3	5369.5	2949.7	35.8
rdp101-100	5294.4	952.2	6246.5	5374.6	33.9	5408.5	138.7	13.4
rdp102-100	4651.6	3027.0	7678.6	4945.9	72.1	5018.1	522.7	34.6
rdp103-100	4350.1	2618.9	6969.0	4534.7	107.0	4641.7	766.1	33.4
rdp201-100	4640.5	2342.5	6983.0	4933.5	129.8	5063.3	1536.6	27.5
rdp202-100	4289.5	2207.3	6496.7	4551.6	84.1	4635.6	1801.7	28.6
rdp203-100	4263.9	3433.2	7697.1	4573.5	243.0	4816.5	3224.3	37.4
Average	4138.7	1826.4	5965.1	4294.3	54.7	4348.9	694.3	26.6

Bibliography

[1] Ai, T. J. and Kachitvichyanukul, V. (2009). A particle swarm optimization for the vehicle routing problem with simultaneous pickup and delivery. *Computers & Operations Research*, 36(5):1693–1702.

[2] Angelelli, E. and Mansini, R. (2002). The vehicle routing problem with time windows and simultaneous pick-up and delivery. In *Quantitative approaches to distribution logistics and supply chain management*, pages 249–267. Springer.

[3] Archetti, C., Bianchessi, N., and Speranza, M. G. (2013). Optimal solutions for routing problems with profits. *Discrete Applied Mathematics*, 161(2):547–557.

[4] Azadian, F., Murat, A., and Chinnam, R. B. (2017). An unpaired pickup and delivery problem with time dependent assignment costs: Application in air cargo transportation. *European Journal of Operational Research*, 263(1):188 – 202.

[5] Balas, E. (1989). The prize-collecting traveling salesman problem. *Networks*, 19(6):621–636.

[6] Baldacci, R., Bartolini, E., and Mingozzi, A. (2011). An exact algorithm for the pickup and delivery problem with time windows. *Operations Research*, 59(2):414–426.

[7] Baldacci, R., Christofides, N., and Mingozzi, A. (2008). An exact algorithm for the vehicle routing problem based on the set partitioning formulation with additional cuts. *Mathematical Programming*, 115(2):351–385.

[8] Balseiro, S. R., Loiseau, I., and Ramont, J. (2011). An ant colony algorithm hybridized with insertion heuristics for the time dependent vehicle routing problem with time windows. *Computers and Operation Research*, 38:954–966.

[9] Barnhart, C., Johnson, E. L., Nemhauser, G. L., Savelsbergh, M. W. P., and Vance, P. H. (1998). Branch-and-price: Column generation for solving huge integer programs. *Operations Research*, 46(3):316–329.

[10] Barnhart, C. and Laporte, G. (2006). *Handbooks in operations research and management science: transportation*. Elsevier.

[11] Battarra, M., Cordeau, J. F., and Iori, M. (2014). Chapter 6: pickup-and-delivery problems for goods transportation. In *Vehicle Routing: Problems, Methods, and Applications, Second Edition*, pages 161–191. SIAM.

[12] Bianchessi, N. and Righini, G. (2007). Heuristic algorithms for the vehicle routing problem with simultaneous pick-up and delivery. *Computers & Operations Research*, 34(2):578–594.

[13] Bisiani., R. (1987). Beam search. In S.Shapiro, editor, *Encyclopedia of Artificial Intelligence*, pages 56–58. Wiley and Sons.

[14] Bol.com (2020). Delivery codes. `https://developers.bol.com/appendix-c-delivery-codes/`. Accessed: 2020-01-27.

[15] Boussier, S., Feillet, D., and Gendreau, M. (2007). An exact algorithm for team orienteering problems. *4 OR*, 5:211–230.

[16] Chang, M. S. (2005). A vehicle routing problem with time windows and stochastic demands. *Journal of the Chinese institute of engineers*, 28(5):783–794.

[17] Chen, J. F. and Wu, T. H. (2006). Vehicle routing problem with simultaneous deliveries and pickups. *Journal of the Operational Research Society*, 57(5):579–587.

[18] Cherkesly, M., Desaulnier, G., and Laporte, G. (2015). Branch-price-and-cut for the pickup and delivery problem with time windows and last-in-first-out loading. *Transportation science*, 49(4):752–66.

[19] Clement, J. (2020). Amazon: shipping costs 2012-2019. `https://www.statista.com/statistics/806498/amazon-shipping-costs/`. Accessed: 2020-06-08.

[20] Cordeau, J. F., Ghiani, G., and Guerriero, E. (2014). Analysis and branch-and-cut algorithm for the time-dependent travelling salesman problem. *Transportation Science*, 48(1):46–58.

[21] Cordeau, J. F. and Laporte, G. (2003). A tabu search heuristic for the static multi-vehicle dial-a-ride problem. *Transportation Research Part B: Methodological*, 37(6):579–594.

[22] Cordeau, J. F., Laporte, G., and Ropke, S. (2008). Recent models and algorithms for one-to-one pickup and delivery problems. In B.Golden, Raghavan, S., and Wasil, E., editors, *The Vehicle Routing Problem: Latest Advances and New Challenges*, pages 327–357, New York. Springer.

[23] Cortés-Murcia, D. L., Prodhon, C., and Afsar, H. M. (2019). The electric vehicle routing problem with time windows, partial recharges and satellite customers. *Transportation Research Part E: Logistics and Transportation Review*, 130:184–206.

[24] Crispim, J. and Brandão, J. (2005). Metaheuristics applied to mixed and simultaneous extensions of vehicle routing problems with backhauls. *Journal of the Operational Research Society*, 56(11):1296–1302.

[25] Dabia, S., Ropke, S., Van Woensel, T., and De Kok, T. (2013). Branch and price for the time-dependent vehicle routing problem with time windows. *Transportation science*, 47(3):380–396.

[26] Danzig, G. and Ramser, J. (1959). The truck dispatching problem. *Management Science*, 2:80–91.

[27] Dell' Amico, M., Maffioli, F., and Varbrand, P. (1995). On prize-collecting tours and the asymmetric traveling salesman problem. *International Transactions in Operational Research*, 2(3):297–308.

[28] Dell'Amico, M., Righini, G., and Salani, M. (2006). A branch-and-price approach to the vehicle routing problem with simultaneous distribution and collection. *Transportation science*, 40(2):235–247.

[29] Demir, E., Bektas, T., and Laporte, G. (2012). An adaptive large neighborhood search heuristic for the pollution-routing problem. *European Journal of Operational Research*, 223(2):346–59.

[30] Desaulniers, G., Desrosiers, J., and Solomon, M. M. (2006). *Column generation*, volume 5. Springer Science & Business Media.

[31] Dethloff, J. (2002). Relation between vehicle routing problems: an insertion heuristic for the vehicle routing problem with simultaneous delivery and pick-up applied to the vehicle routing problem with backhauls. *Journal of the Operational Research Society*, 53(1):115–118.

[32] Dimitrakos, T. D. and Kyriakidis, E. G. (2015). A single vehicle routing problem with pickups and deliveries, continuous random demands and predefined customer order. *European Journal of Operational Research*, 244(3):990–993.

[33] Donati, A. V., Montemanni, R., Casagrande, N., Rizzoli, A. E., and Gambardella, L. M. (2008). Time dependent vehicle routing problem with a muti ant colony system. *European Journal of Operational Research*, 185(3):1174–1191.

[34] Dumas, Y., Desrosiers, J., and Soumis, F. Y. (1991). The pickup and delivery problem with time windows. *European Journal of Operational Research*, 54(1):7–22.

[35] Dumitrescu, I., Ropke, S., and Cordeau, J. F. (2010). The traveling salesman problem with pickup and delivery: polyhedral results and branch-and-cut algorithm. *Math.Program.,Ser.A*, 121:269–305.

[36] Duquei, D., Lozano, L., and Medaglia, A. L. (2015). Solving the orienteering problem with time windows via the pulse framework. *Computers and Operation Research*, 54:168–176.

[37] eMarketer (2018). Worldwide retail and ecommerce sales: emarketer's updated forecast and new mcommerce estimates for 2016—2021. `https://www.emarketer.com/Report/Worldwide-Retail-Ecommerce-Sales-eMarketers-Updated-Forecast-New-Mcommerce-Estimates-20162021/2002182`. Accessed: 2020-10-12.

[38] Feillet, D., Dejax, P., and Gendreau, M. (2004). An exact algorithm for the elementary shortest path problem with resource constraints: Application to some vehicle routing problems. *Networks*, 44(3):216–229.

[39] Feillet, D., Dejax, P., and Gendreau, M. (2005). Traveling salesman problems with profits. *Transportation Science*, 39:188–205.

[40] Fomin, F. and Lingas, A. (2002). Approximation algorithms for time-dependent orienteering. *Information Processing Letters*, 83:57–62.

[41] Franceschetti, A., Honhon, D., Van Woensel, T., Bektaş, T., and Laporte, G. (2013). The time-dependent pollution routing problem. *Transportation Research Part B: Methodological*, 56:265–293.

[42] Fu, L. (2002). Scheduling dial-a-ride paratransit under time-varying, stochastic congestion. *Transportation Research Part B: Methodological*, 36(6):485–506.

[43] Gajpal, Y. and Abad, P. (2009). An ant colony system (ACS) for vehicle routing problem with simultaneous delivery and pickup. *Computers & Operations Research*, 36(12):3215–3223.

[44] Gansterer, M., Küçüktepe, M., and Hartl, R. F. (2017). The multi-vehicle profitable pickup and delivery problem. *OR Spectrum*, 39(1):303–319.

[45] Gavalas, D., Konstantopoulos, C., Mastakas, K., Pantziou, G., and Vathis, N. (2015). Heuristics for the time dependent team orienteering problem: Application to tourist route planning. *Computers & Operation Research*, 62:36–50.

[46] Gendreau, M., Ghiani, G., and Guerriero, E. (2015). Time-dependent routing problems: A review. *Computers and Operation Research*, 64:189–197.

[47] Gendreau, M., Laporte, G., and Séguin, R. (1996). Stochastic vehicle routing. *European Journal of Operational Research*, 88(1):3–12.

[48] Ghilas, V., Demir, E., and Van Woensel, T. (2016). An adaptive large neighborhood search heuristic for the pickup and delivery problem with time windows and scheduled lines. *Computers and Operations Research*, 72:12 – 30.

[49] Gromicho, J. A. S., van Hoorn, J. J., Kok, A. L., and Schutten, J. M. J. (2012). Restricted dynamic programming: A flexible framework for solving realistic vrps. *Computers and Operation Research*, 39(5):902–909.

[50] Gunawan, A., Lau, H. C., and Vansteenwegen, P. (2016). Orienteering problem: A survey of recent variants, solution approaches and applications. *European Journal of Operational Research*, 255(2):315–332.

[51] Haag, M. and Hu, W. (2019). 1.5 million packages a day: The internet brings chaos to N.Y. streets. `https://www.nytimes.com/2019/10/27/nyregion/nyc-amazon-delivery.html`. Accessed: 2020-06-14.

[52] Hashimoto, H., Yagiura, M., and Ibaraki, T. (2008). An iterated local search algorithm for the time-dependent vehicle routing problem with time windows. *Discrete Optimization*, 5(2):434–456.

[53] Henchiri, A., Bellalouna, M., and Khaznaji, W. (2014). A probabilistic traveling salesman problem: a survey. *FedCSIS Position Papers*, 3:55–60.

[54] Hickman, M. D. and Bernstein, D. H. (1997). Transit service and path choice models in stochastic and time-dependent networks. *Transportation Science*, 31(2):129–146.

[55] Ibaraki, T., Imahori, S., Nonobe, K., Sobue, K., Uno, T., and Yagiura, M. (2008). An iterated local search algorithm for the vehicle routing problem with convex time penalty functions. *Discrete Applied Mathematics*, 156(11):2050–2069.

[56] Ichoua, S., Gendreau, M., and Potvin, J. Y. (2003). Vehicle dispatching with time-dependent travel times. *European Journal of Operational Research*, 62:379–396.

[57] Jabali, O., Van Woensel, T., De Kok, A. G., Lecluyse, C., and Peremans, H. (2009). Time-dependent vehicle routing subject to time delay perturbations. *IIE Transactions*, 41(12):1049–1066.

[58] Jepsen, M., Petersen, B., Spoorendonk, S., and Pisinger, D. (2008). Subset-row inequalities applied to the vehicle routing problem with time windows. *Operation Research*, 56(2):497–511.

[59] Joo, S. J., Min, H., and Smith, C. (2017). Benchmarking freight rates and procuring cost-attractive transportation services. *The International Journal of Logistics Management*.

[60] Kleywegt, A. J., Shapiro, A., and Homem-de-Mello, T. (2002). The sample average approximation method for stochastic discrete optimization. *SIAM Journal on Optimization*, 12(2):479–502.

[61] Koç, Ç., Laporte, G., and Tükenmez, İ. (2020). A review on vehicle routing with simultaneous pickup and delivery. *Computers & Operations Research*, page 104987.

[62] Labadie, N., Mansini, R., Melechovskỳ, J., and Calvo, R. W. (2012). The team orienteering problem with time windows: An lp-based granular variable neighborhood search. *European Journal of Operational Research*, 220(1):15–27.

[63] Laporte, G. and Martello, S. (1990). The selective traveling salesman problem. *Discrete Applied Mathematics*, 26:193–207.

[64] Lecluyse, C., Van Woensel, T., and Peremans, H. (2009). Vehicle routing with stochastic time-dependent travel times. *4OR*, 7(4):363–377.

[65] Lei, H., Laporte, G., and Guo, B. (2011). The capacitated vehicle routing problem with stochastic demands and time windows. *Computers & Operations Research*, 38(12):1775–1783.

[66] Li, J. (2011). Model and algorithm for time-dependent team orienteering problem. in S.Lin and X.Huang(Eds). *Advanced research on computer education. Simulation and modeling, communications in computer and information science*, 175:1–7.

[67] Li, Y., Chen, H., and Prins, C. (2016). Adaptive large neighborhood search for the pickup and delivery problem with time windows, profits, and reserved requests. *European Journal of Operational Research*, 252(1):27–38.

[68] Liu, R., Xie, X., Augusto, V., and Rodriguez, C. (2013). Heuristic algorithms for a vehicle routing problem with simultaneous delivery and pickup and time windows in home health care. *European Journal of Operational Research*, 230(3):475–486.

[69] Lübbecke, M. E. (2010). Column generation. *Wiley encyclopedia of operations research and management science*.

[70] Mahmoudi, M. and Zhou, X. (2016). Finding optimal solutions for vehicle routing problem with pickup and delivery services with time windows: A dynamic programming approach based on state–space–time network representations. *Transportation Research Part B: Methodological*, 89:19 – 42.

[71] Malandraki, C. and Daskin, M. S. (1992). Time dependent vehicle routing problems: Formulations, properties, and heuristic algorithms. *Transportation Science*, 26(3):185–200.

[72] Malandraki, C. and Dial, R. B. (1996). A restricted dynamic programming heuristic algorithm for the time dependent traveling salesman problem. *European Journal of Operational Research*, 90(1):45–55.

[73] Min, H. (1989). The multiple vehicle routing problem with simultaneous delivery and pick-up points. *Transportation Research Part A: General*, 23(5):377–386.

[74] Mingyong, L. and Erbao, C. (2010). An improved differential evolution algorithm for vehicle routing problem with simultaneous pickups and deliveries and time windows. *Engineering Applications of Artificial Intelligence*, 23(2):188–195.

[75] Minis, I. and Tatarakis, A. (2011). Stochastic single vehicle routing problem with delivery and pick up and a predefined customer sequence. *European journal of operational research*, 213(1):37–51.

[76] Montané, F. A. T. and Galvao, R. D. (2006). A tabu search algorithm for the vehicle routing problem with simultaneous pick-up and delivery service. *Computers & Operations Research*, 33(3):595–619.

[77] Montero, A., Méndez-Díaz, I., and Miranda-Bront, J. J. (2017). An integer programming approach for the time-dependent traveling salesman problem with time windows. *Computers & Operations Research*, 88:280–289.

[78] Nahum, O. E. and Hadas, Y. (2009). Developing a model for the stochastic time-dependent vehicle routing problem. In *International conference on computers and industrial engineering*, CIE '09, pages 118–123, Troyes, France. IEEE.

[79] Nanry, W. P. and Barnes, J. W. (2000). Solving the pickup and delivery problem with time windows using reactive tabu search. *Transportation Research Part B: Methodological*, 34(2):107–121.

[80] Oyola, J., Arntzen, H., and Woodruff, D. L. (2016). The stochastic vehicle routing problem, a literature review, part i: models. *EURO Journal on Transportation and Logistics*, pages 1–29.

[81] Pandelis, D. G., Kyriakidis, E. G., and Dimitrakos, T. D. (2012). Single vehicle routing problems with a predefined customer sequence, compartmentalized load and stochastic demands. *European journal of operational research*, 217(2):324–332.

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

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

[84] Pecin, D., Pessoa, A., Poggi, M., and Uchoa, E. (2014). Improved branch-cut-and-price for capacitated vehicle routing. *Integer programming and combinatorial optimization*, Springer:393–403.

[85] Pickupp (2019). Delivering success: Style theory achieves a 98% customer satisfaction rate with hybrid delivery model. `https://sg.pickupp.io/delivering-success-high-customer-satisfaction-rate/`. Accessed: 2019-08-27.

[86] Pisinger, D. and Røpke, S. (2007). A general heuristic for vehicle routing problem. *Computers and Operation Research*, 34(8):2403–35.

[87] Reportlinker (2020). Covid-19 impact on logistics & supply chain industry market by industry verticals, mode of transport, region-global forecast to 2021. `https://www.reportlinker.com/p05892826/`. Accessed: 2020-06-28.

[88] Righini, G. and Salani, M. (2009). Decremental state space relaxation strategies and initialization heuristics for solving the orienteering problem with time windows with dynamic programming. *Computers and Operation Research*, 36(4):1191–203.

[89] Rodrigue, J. P., Comtois, C., and Slack, B. (2016). *The geography of transport systems*. Routledge.

[90] Røpke, S. (2012). Branching decisions in branch-and-cut-and-price algorithms for vehicle routing problems. *Presentation in Column Generation*.

[91] Røpke, S. and Cordeau, J. F. (2009). Branch and cut and price for the pickup and delivery problem with time windows. *Transportation Science*, 43(3):267–286.

[92] Røpke, S., Cordeau, J. F., and Laporte, G. (2009). Models and branch-and-cut algorithms for pickup and delivery problem with time windows. *Networks*, 49(4):258–272.

[93] Røpke, S. and Pisinger, D. (2006). An adaptive large neighborhood search heuristic for the pickup and delivery problem with time windows. *Transportation Science*, 40(4):455–72.

[94] Ruland, K. S. (1994). Polyhedral solution to the pickup and delivery problem. *Ph.D. Thesis*, pages Sever Institute, Washington University in St.Louis, MO.

[95] Ruland, K. S. and Rodin, E. Y. (1997). The pickup and delivery problem: Faces and branch-and-cut algorithm. *Computers & mathematics with applications*, 33(12):1–13.

[96] Savelsbergh, M. and Sol, M. (1995). The general pickup and delivery problem. *Transportation science*, 29(1):17–29.

[97] Schilde, M., Doerner, K. F., and Hartl, R. F. (2014a). Integrating stochastic time-dependent travel speed in solution methods for the dynamic dial-a-ride problem. *European journal of operational research*, 238(1):18–30.

[98] Schilde, M., F., D. K., and Hartl, R. F. (2014b). Integrating stochastic time-dependent travel time speed in solution methods for the dynamic dial-a-ride problem. *European Journal of Operation Reseach*, 238(1):18–30.

[99] Shaw, P. (1997). A new local search algorithm providing high quality solutions to vehicle routing problems. *APES Group, Dept of Computer Science, University of Strathclyde, Glasgow, Scotland, UK*.

[100] Shaw, P. (1998). Using constraint programming and local search methods to solve vehicle routing problems. In *International conference on principles and practice of constraint programming*, pages 417–431. Springer.

[101] Shi, Y., Boudouh, T., Grunder, O., and Wang, D. (2018). Modeling and solving simultaneous delivery and pick-up problem with stochastic travel and service times in home health care. *Expert Systems with Applications*, 102:218–233.

[102] Sigurd, M. and Pisinger, D. (2004). Scheduling transportation of live animals to avoid the spread of diseases. *Transportation Science*, 38:197–209.

[103] Sol, M. (1994). Column generation techniques for pickup and delivery problems. *PhD thesis, Technische Universiteit Eindhoven*.

[104] Solomon, M. (1987). Algorithms for the vehicle routing and scheduling problems with time window constraints. *Operations Research*, 35(2):254–265.

[105] Statistical Pocketbook (2019). *EU Transport in Figures, Mobility and Transport*. Publications Office of the European Union, Belgium.

[106] Subramanian, A., Drummond, L. M. D. A., Bentes, C., Ochi, L. S., and Farias, R. (2010). A parallel heuristic for the vehicle routing problem with simultaneous pickup and delivery. *Computers & Operations Research*, 37(11):1899–1911.

[107] Subramanian, A., Uchoa, E., Pessoa, A. A., and Ochi, L. S. (2011). Branch-and-cut with lazy separation for the vehicle routing problem with simultaneous pickup and delivery. *Operations Research Letters*, 39(5):338–341.

[108] Subramanian, A., Uchoa, E., Pessoa, A. A., and Ochi, L. S. (2013). Branch-cut-and-price for the vehicle routing problem with simultaneous pickup and delivery. *Optimization Letters*, 7(7):1569–1581.

[109] Sun, P., Veelenturf, L. P., Dabia, S., and Van Woensel, T. (2018a). The time-dependent capacitated profitable tour problem with time windows and precedence constraints. *European Journal of Operational Research*, 264(3):1058 – 1073.

[110] Sun, P., Veelenturf, L. P., Hewitt, M., and Van Woensel, T. (2018b). The time-dependent pickup and delivery problem with time windows. *Transportation Research Part B: Methodological*, 116:1 – 24.

[111] Taş, D., Dellaert, N., Van Woensel, T., and De Kok, T. (2014). The time-dependent vehicle routing problem with soft time windows and stochastic travel times. *Transportation Research Part C: Emerging Technologies*, 48:66–83.

[112] Turkeš, R., Sørensen, K., Hvattum, L. M., Barrena, E., Chentli, H., Coelho, L., Dayarian, I., Grimault, A., Gullhav, A., Iris, C., et al. (2019). Meta-analysis of metaheuristics: Quantifying the effect of adaptiveness in adaptive large neighborhood search. Technical report.

[113] UN.org (2018). 68% of the world population projected to live in urban areas by 2050, says UN. `https://www.un.org/development/desa/en/news/population/`

2018-revision-of-world-urbanization-prospects.html. Accessed: 2020-10-11.

[114] Van Woensel, T., Kerbache, L., Peremans, H., and Vandaele, N. (2008). Vehicle routing with dynamic travel times: a queueing approach. *European Journal of Operational Research*, 186(3):990–1007.

[115] Vansteenwegen, P. and Gunawan, A. (2019). *Orienteering problems: Models and algorithms for vehicle routing problems with profits*. Springer Nature.

[116] Vansteenwegen, P., Souffriau, W., and Van Oudheusden, D. (2011). The orienteering problem: a survey. *European Journal of Operation Reseach*, 209(1):1–10.

[117] Verbeeck, C., Aghezzaf, E. H., and Vansteenwegen, P. (2014). A fast solution method for the time-dependent orienteering problem with time windows. *European Journal of Operational Research*, 236:419–432.

[118] Verbeeck, C., Vansteenwegen, P., and Aghezzaf, E. H. (2016). Solving the stochastic time-dependent orienteering problem with time windows. *European Journal of Operational Research*, 255(3):699–718.

[119] Verhoeven, G. (2019). Dutch see traffic congestion increase with 17 %. https://newmobility.news/2019/12/30/dutch-traffic-congestion-on-the-rise/. Accessed: 2020-06-14.

[120] Vidal, T., Crainic, T. G., Gendreau, M., and Prins, C. (2015). Timing problems and algorithms: Time decisions for sequences of activities. *Networks*, 65(2):102–128.

[121] Visser, T. and Spliet, R. (2017). Efficient move evaluations for time-dependent vehicle routing problems. Technical Report EI2017-23, Erasmus School of Economics, Rotterdam, NL, Research Paper.

[122] Vu, D. M., Hewitt, M., Boland, N., and Savelsbergh, M. (2019). Dynamic discretization discovery for solving the time-dependent traveling salesman problem with time windows. *Transportation Science*.

[123] Wang, C., Mu, D., Zhao, F., and Sutherland, J. W. (2015). A parallel simulated annealing method for the vehicle routing problem with simultaneous pickup–delivery and time windows. *Computers & Industrial Engineering*, 83:111–122.

[124] Wang, H. F. and Chen, Y. Y. (2012). A genetic algorithm for the simultaneous delivery and pickup problems with time window. *Computers & Industrial Engineering*, 62(1):84–95.

[125] Wang, Z. (2018). Delivering meals for multiple suppliers: Exclusive or sharing logistics service. *Transportation Research Part E: Logistics and Transportation Review*, 118:496–512.

[126] Wollenberg, N., Borndörfer, R., and Gendreau, M. (2015). *Decision Making under Uncertainty in Routing Problems for Reverse Logistics*. PhD thesis, Universitätsbibliothek Duisburg-Essen.

[127] Xiang, Z., Chu, C., and Chen, H. (2008). The study of a dynamic dial-a-ride problem under time-dependent and stochastic environment. *European Journal of Operational Research*, 185(2):534–551.

[128] Xu, M. et al. (2017). Distribution logistics and logistics customer services of B2C e-tailing industry in the Chinese market.

[129] Yang, W. H., Mathur, K., and Ballou, R. H. (2000). Stochastic vehicle routing problem with restocking. *Transportation Science*, 34(1):99–112.

[130] Young, J. (2020). U.S. ecommerce sales grow 14.9% in 2019. `https://https://www.digitalcommerce360.com/article/us-ecommerce-sales/`. Accessed: 2020-10-08.

[131] Zhang, J., Lam, W. H., and Chen, B. Y. (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.

[132] Zhang, T., Chaovalitwongse, W. A., and Zhang, Y. (2012). Scatter search for the stochastic travel-time vehicle routing problem with simultaneous pick-ups and deliveries. *Computers & Operations Research*, 39(10):2277–2290.

[133] Zhu, L. and Sheu, J. B. (2018). Failure-specific cooperative recourse strategy for simultaneous pickup and delivery problem with stochastic demands. *European Journal of Operational Research*, 271(3):896–912.

Summary

In this thesis, we investigate short-haul freight transportation solutions for efficient and effective pickup and delivery services considering time-dependent travel times, from an operational and tactical planning perspective. The thesis is composed of four chapters.

In Chapter 2, we introduce the time-dependent capacitated profitable tour problem with time windows and precedence constraints and formally describe it as an arc-based mixed-integer program (MIP). Firstly, the proposed MIP is solved using an optimization software, considering small-size instances with up to 60 locations (30 pickup and delivery requests). To tackle larger instances, a tailored labeling algorithm is proposed. Several dominance criteria are also introduced to discard unpromising labels. Our computational results demonstrate that the algorithm can solve instances with up to 150 locations (75 pickup and delivery requests) to optimality. Additionally, we present a restricted dynamic programming heuristic to improve the computation time. This heuristic does not guarantee optimality but is able to find the optimal solution for 32 instances out of the 34 instances.

Chapter 3 studies a family of time-dependent pickup and delivery problems with time windows, extending the problem in Chapter 2 to the multiple vehicle cases. We aim to optimize the service of a transportation provider under two dimensions of operational flexibility. In the first, we consider problems wherein the transportation service provider can choose the transportation requests it serves in order to maximize profit. In the second, we consider problems wherein they can take advantage of periods of light traffic by dictating to drivers when their routes should begin. We also consider problems wherein these flexibilities are not present. We propose an exact solution approach for solving problems from this family that is based upon branch and price, wherein columns are generated via a tailored labeling algorithm. We augment the framework with adaptations of various speed-up techniques from the literature, including limited-memory subset-row cuts and route enumeration. With an extensive computational study, we assess the effectiveness of the proposed framework and the impact of the adapted techniques. We show that small to medium-size instances, with up to 45 freight requests (90 locations), can be optimally solved by the proposed exact algorithm.

In Chapter 4, we focus on the time-dependent profitable pickup and delivery problem with time windows. As one variant of problem studied in Chapter 3, it allows all routes start at

a flexible departure time and handles only the profitable requests. Moreover, we propose an adaptive large neighborhood search algorithm (ALNS). The general idea of ALNS is to iteratively improve a given solution by first partially deteriorating it and then repairing it. A destroy operator and an insertion operator are used, respectively. We use a total of ten removal and five insertion operators. Each operator is selected based on its past performance during the search process. Results of an extensive computational study show that the algorithm can quickly find high-quality solutions on instances with up to 75 transportation requests (150 locations). We also conduct a study of the impact on profits when explicitly recognizing traffic congestion during planning operations.

Chapter 5 investigates a more realistic representation of real-life applications, a time-dependent laundry routing problem with stochastic pickup demands (TDLRPSPD). As one of the most important possesses in laundry business, efficient laundry pickup and delivery will result in reduction in cost, time and improvement in service quality. It ultimately results in customers loyalty, economic efficiency and gain of a competitive advantage over competitor. However, the literature on this problem is quite limited. A two-stage stochastic programming with recourse is presented for this problem. We also propose a sample average approximation method together with an adaptive large neighborhood search algorithm to tackle this problem. Several experiments are conducted to show the effectiveness of our solution approach. Computational results on instances with up to 100 requests reveal that a cost savings of on average more than 50% can be achieved compared with a pure deterministic solution approach using expected pickup demands, if pickup demands uncertainty is taken into consideration during the planning process.

Conclusions are drawn in Chapter 6. We also discuss several possible directions for future research following the concepts presented in this thesis.

About the author

Peng Sun was born in Yiyang, a city in Hunan Province of China on January 31, 1988. He received his bachelor's degree in Automation from the College of Electrical and Information Engineering at Hunan University in 2009. He continued his studies at Huazhong University of Science and Technology and obtained his master's degree in System Engineering in 2012. After finishing his MSc, he worked at Huawei Device Co., as deliver engineering for 4 months. From September 2012, he started her PhD project in the Department of Industrial Engineering and Innovation Sciences at Eindhoven University of Technology. He focused on optimization and operation under the supervision of prof.dr. Tom van Woensel and dr. Luuk Veelenturf. He was also supervised by dr. Said Dabia between 2012 and 2013. During his PhD study, he visited Loyola University Chicago for 3 months to collaborate with dr. Mike Hewitt. From September 2017 to August 2018, he joined Kühne Logistics University and worked with dr. Asvin Goel as a post-doctoral researcher. In March 2019, Peng joined an online food ordering company, yd. yourdelivery GmbH (Just Eat Takeaway.com), in Berlin as a Business Analyst.