The pickup and delivery problem with time windows and scheduled lines : models and algorithms

The Pickup and Delivery Problem with Time Windows and Scheduled Lines: models and algorithms

Veaceslav Ghilas

Printed and cover design by Proefschriftmaken.nl || Uitgeverij BOXPress

This thesis is number D199 of the thesis series of Beta Research School for Operations Management and Logistics. Beta Research School is a joint effort of the School of Industrial Engineering and the Department of Mathematics and Computer Science at Eindhoven University of Technology, and Center for Production, Logistics and Operations Management at the University of Twente.

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

ISBN: 978–94–6295–528–8

This research has been funded by the Dutch Institute for Advanced Logistics (DINALOG), under the grant titled Cargo Hitching, #2011 4 086R.

The Pickup and Delivery Problem with Time Windows and Scheduled Lines: Models and algorithms

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 27 oktober 2016 om 16:00 uur

door

Veaceslav Ghilas

geboren te Straseni, Moldavië

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
copromotor(en):	dr. E. Demir
leden:	prof.dr. M. Savelsbergh (Georgia Institute of Technology)
	prof.dr. W.E.H. Dullaert (Vrije Universiteit Amstedam)
	prof.dr. A.G. de Kok
	prof.dr. J.–F. Cordeau (HEC, Montréal)

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

Acknowledgments

I would like to express my gratitude to all the people who contributed in one way or another to the completion of this thesis.

Firstly, I would like to thank my promotor Tom Van Woensel, who has guided me through four years of research. He has been a great support for me during the past years. His questions and advices greatly helped improve the quality of the work I have been doing i.e., research and teaching. Thank you for always supporting me with great ideas.

I was very lucky to have the chance to be supervised by Emrah Demir. Emrah is a great mentor and friend, from whom I have learned many things. Thank you for your time and patience spent on building this thesis piece by piece. Additionally, thank you for reading and giving good and timely feedback on the manuscripts.

During my PhD I had the great honour to work with Jean-François Cordeau, who supervised my work during my three-month stay at CIRRELT, Montreal, Canada. Thank you for your great guidance, ideas and for sharing your knowledge with me. Without you, Chapter 3 of this thesis would not have been as good as it is.

I kindly thank Wout Dullaert, Ton de Kok, and Martin Savelsbergh for being a part of my thesis committee. Thank you for your valuable comments, which have significantly improved the quality of the thesis.

I am grateful to all my current and former colleagues at the OPAC department. Especially, I would like to thank Peng Sun, with whom I shared my office during the past four years, for our interesting discussions. In addition, I would like to thank Fardin Dashty Saridarq for our chats about research and not only, during the stay in Montreal. I am also grateful to Taimaz Soltani, Baoxiang Li, Maryam Steadie Seifi, Anna Franceschetti, Yousef Ghiami, Engin Topan, Zumbul Atan, Luuk Veelenturf, Claudine Hulsman-Paul, Jose van Dijk-Kok, Jolanda Verkuijlen-Nelissen, Christel van Berlo-Verlijsdonk for all the discussions and help.

I would like to thank all my CIRRELT colleagues, which made my life in Montreal great. Thank you, Selene Silvestri, Aurelien Froger, Maria Isabel Restrepo Ruiz, Vinicius Morais, Julien Keutchayan, Xiaolu Liu, Delon Ge, Diego Pecin, Yunfei Wang, for all the good time spent together, enthusiasm, and for sharing your knowledge and experiences with me.

Special thanks go to my family for their support and love. Mom and dad, this thesis is our common achievement, due to your constant support and for cheering me up in difficult moments. I am also very grateful to my lovely sister, who always

fills me with positive energy. Thank you for being such a great sister. Finally, my outmost thanks go to my lovely wife Ecaterina, and our two daughters, Iana and Emma, for their support throughout everything. I can always rely on your encouragement and love!

Veaceslav Ghilas
Eindhoven, September 2016

To Iana and Emma

Contents

Chapter 1

Introduction

Transportation is the movement of people and goods from one place to another. It is essential for modern societies. Freight (goods) transportation is crucial for every modern supply chain, as raw materials and (semi-)finished products need to be moved between e.g., production facilities, or to final customers. Passenger transportation mainly involves travel for business/leisure purposes. It may be public, where operators provide scheduled services, or private. In terms of travel distance, both types may be short- and long-haul (Ghiani et al., 2004). Usually, long-haul transportation involves relatively long distances, e.g., between airports, countries, etc. On the other hand, short-haul implies services within a relatively small area, e.g., within a city, county, etc.

Nowadays the urban population considerably increases, and so does the demand for goods in urban areas. In addition, the need for passenger transportation also increases, thus leading to more private and public transport vehicles on the roads. Apart from provided services, both freight and passenger transportation in urban areas also result in increased traffic congestion, noise, air pollution (Demir et al., 2015).

On the other hand, the population in rural areas is declining, hence the use of public transportation significantly decreases (Santos et al., 2010). Operators can merely provide the services at socially acceptable levels in an economically viable way. Furthermore, goods delivery routes to individual consumers and retailers consist of fewer stops over longer distances (Harms et al., 2010). This leads to high delivery costs to meet rural demands.

In 2013, over 2,200 billion tonne-kilometres of freight (both short-haul and long-haul) were transported in the European Union, three quarters being transported

on the road segment (Eurostat, 2016). In addition, according to Coyle et al. (1996), transportation costs represent up to 10% of a product's selling price. Thus, transportation is a key logistics activity, whose efficiency plays a crucial role in total costs of supply chains. Furthermore, last-mile logistics, more specifically short-haul transportation, causes disproportionately high costs of up to 28% of the total transportation costs (Rodrigue et al., 2009). Thus, significant cost benefits might be achieved by increasing the efficiency of the last-mile segment.

To cope with this reality, innovative and effective 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).

This thesis aims to investigate integrated freight and public transportation solutions to perform pickup and delivery services in short-haul freight transportation environments, from an operational and tactical planning perspective. More specifically, we investigate an extension of the classical pickup and delivery problem, where requests may be transferred to available public transportation lines (also referred to as *scheduled lines* (SLs)). This transportation network concerns a small area, i.e., a city or a region (Ghiani et al., 2004). Given a fleet, each vehicle performs at most one tour per working day, transporting the freight requests from their origins to the corresponding destinations and public transportation can be used as a part of the freight's journey, as long as the customer requirements are satisfied. This specific transportation problem is titled the *Pickup and Delivery Problem with Time Windows and Scheduled Lines* (PDPTW-SL). Figure 1.1 presents an example of a considered network. This example includes three requests that need to be served, two SLs, and two depots with one vehicle each.

In this example, all requests are first picked up by one vehicle, and then transferred to SLs at T_2. Requests 1 and 2 are transported on the SL between T_2 and T_1. Request 3 is transported on SL that connects T_2 and T_3. Afterwards, another vehicle re-collects the requests from the corresponding transfer nodes, and delivers them to their final destinations.

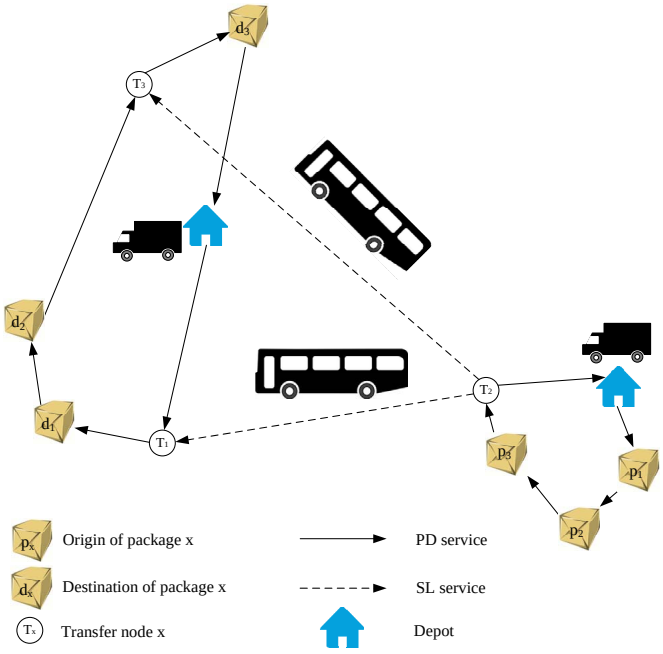

Figure 1.1 An example PDPTW-SL network

The rest of this chapter is organized as follows. Section 1.1 discusses the motivation of the research conducted 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 Motivation

Nowadays, municipalities become interested in developing instruments and policies to ensure efficient and effective mobility for passengers and freight. Since urban space is scarce, passenger and freight transportation flows overlap to a significant extent (Trentini and Malhene, 2010). Hence, the accessibility level decreases for both, passengers and freight, resulting in congestion and longer travel times. Furthermore, these lead to a number of negative side effects, such as noise, local air pollution, greenhouse gas (GHG) emissions (Demir et al., 2015). One way to tackle this trend is to adopt alternative ways to manage transportation networks to ensure smooth transportation flows.

According to the recommendations given by the European Commission in 2007 (Green Paper on Urban Mobility, 2007), local governmental institutions need to consider all urban logistics related to passenger and freight transportation as a single logistics system. However, in reality public transport is usually operated by the competent administrative body, while freight distribution is mainly performed by the private sector. In line with these recommendations, cities may invest resources in the followings: *(i)* improve the sharing of road space between public transportation flows and transport of goods (i.e., multi-use of the road lanes, night deliveries, shared bus and lorry lanes), *(ii)* increase the use of public transportation by passengers and goods (i.e., shared buses, shared subway, shared tramway, car sharing), *(iii)* introduce consolidation centres (i.e., shared delivery bays, goods lockers). A number of aforementioned shared solutions have already been tested in several cities, such as Barcelona, Genoa, Dresden, Amsterdam (Trentini and Malhene, 2010). This thesis is focused on points *(ii)*, and partly *(iii)*.

Rural areas may also benefit from integrated transport solutions. For example, as the population in rural areas is decreasing, so does the demand for public transport in those areas. Thus, it is increasingly difficult to keep rural public transportation viable at an acceptable cost. Integrating freight with public transport may lead to extra revenue to public transport operators, as freight will also be transported on e.g., buses.

Generally, a conventional focus on planning the freight transport activities is to reduce costs and, as a result increase the margins by considering e.g., fuel costs, driver wages (Greene and Wagener, 1997; Forkenbrock, 1999, 2001). However, growing concern about the environment, organizations start realizing the importance of the environmental and social impacts (e.g., air pollution, noise, and congestion) associated with transportation. According to Browne et al. (2012), freight transportation has a number of negative impacts on society and the most significant one is the total distance traveled by vehicles. This feature may be directly proportional to certain expenses, such as fuel costs and drivers' wages. Thus, reductions in total distance traveled may lead to decreases in operating costs and amounts of air pollution, noise and GHG emissions, as side effects.

Considering the aforementioned, we investigate in this thesis an integrated freight and public transportation network and analyze its cost benefits. The particular focus is on short-haul transportation, be it urban, or rural areas. Hence, we address the following research objective:

Research objective. Quantify the cost benefits of the integrated public and freight transportation from operational and tactical planning perspectives.

In order to achieve the aforementioned objective, a set of research questions are formulated and addressed in the corresponding chapters of the thesis. The

research questions are formulated as follows:

Research question 1. Which modeling framework can be used to formally define the investigated problem?

Research question 2. Which exact and heuristic algorithms are effective in solving the PDPTW-SL?

Research question 3. How can demand uncertainty be accounted for in the tactical planning process?

1.2 Conceptual model

Operational, and sometimes tactical, decisions in transportation mainly involve the Vehicle Routing Problem (VRP). It was first introduced by Danzig and Ramser (1959) and involves constructing the optimal delivery routes from a central depot to a set of geographically distributed customers. The pickup and delivery problem with time windows (PDPTW) is a generalization of the classical VRP. Generally, the PDPTW consists of designing a set of least cost vehicle routes, which start and end at a depot, in order to satisfy a set of *pickup and delivery requests*. The considered transportation network consists of depots, and request locations. Each request is characterized by an origin location, a destination location, desired time windows for both locations, and demand. Thus, vehicle routes need to satisfy the following constraints.

- Precedence relations: The pickup location of a request should be visited before its corresponding delivery location.
- The pickup and delivery nodes of each request need to be visited by the same vehicle.
- Each node is served within imposed time window.
- Vehicle capacity must not be violated at any time.

Due to time windows constraints, the problem of finding a feasible pickup and delivery plan is NP-hard as discussed in Savelsbergh and Sol (1995). 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 (2000); Røpke and Pisinger (2006)). 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 PDPTW (see e.g., Dumas et al. (1991); Røpke and Cordeau (2009)).

The PDPTW-SL is an extension of the PDPTW. More specifically, on top of the described transportation network, we consider a set of scheduled lines (SLs), which operate according to pre-determined routes and schedules (e.g., buses, trams, trains). No intermediate stops are considered, as loading/unloading operations may lead to delays, which negatively affect passenger service level. Each of the considered pickup and delivery request may be transferred to available public transport services, assuming the existence of a designated freight compartment with limited capacity. The schedule of each SL is also taken into account. Thus, the following additional constraints need to be satisfied:

- Unlike classical PDPTW, pickup and delivery nodes of the same request may be visited by different vehicles (i.e., relaxed precedence constraints). More specifically, a request can be picked up by a vehicle, and then transferred to public transportation at a transfer node. After being transported on the SLs, the request can be re-collected by another vehicle to be delivered to its destination node.
- The departure schedules of every public transportation service must be respected.
- The capacity of each SL must not be violated.

Note that the PDPTW-SL is related to VRP with synchronization constraints (Drexl, 2012), as time and load synchronization between vehicles and public transportation needs to be assured. This thesis provides a number of mathematical models and algorithms to tackle the considered problem on the operational and tactical decision levels.

1.3 Overview of the thesis

In Chapter 2, we introduce the PDPTW-SL and formally describe it as an arc-based mixed-integer program (MIP). The formulation is tightened using a number of valid inequalities. The proposed MIP is solved using an all-purpose solver (i.e., CPLEX 12.3), considering small-size instances. We show that instances with up to 11 requests could be optimally solved. Tackling larger instances using the proposed model remains challenging as the lower bounds obtained are weak. We also show that the proposed valid inequalities greatly improve the solution times. Finally, we quantify the cost benefits of the integrated transportation systems.

More specifically, cost savings of up to 20% can be achieved by using available SLs as a part of request journey, compared to the classical PDPTW, and conclude the potential benefits of such systems.

Chapter 3 provides a path-based MIP and optimally solves the model using a tailored Branch-and-Price (B&P) algorithm. Since it is prohibitively time-consuming to generate all possible routes (vehicle paths) and consequently solve the formulation, the proposed exact algorithm uses column generation to build promising routes. The column generator solves the *Elementary Shortest Path Problem with Resource and Precedence Constraints* (ESPPRPC), which is the natural pricing problem for the PDPTW-SL. *Precedence Constraints* imply that a request must be picked up from either its pickup, or a transfer node, and subsequently delivered to its delivery location, or a transfer node. A number of acceleration techniques are used to speed-up the proposed algorithm. To fully evaluate the effectiveness of the proposed B&P, we have used a library of PDPTW-SL instances. We have shown that small to medium-size instances, with up to 50 freight requests (100 locations), can be optimally solved by the proposed exact algorithm. Integrated system proved cost savings of up to 30% for some test instances, compared to classical transportation network.

In Chapter 4, we present an Adaptive Large Neighborhood Search (ALNS) heuristic to solve the PDPTW-SL. The general idea of ALNS is to iteratively improve a given solution by first partially deteriorating it (using a destroy operator) and then repairing it (using an insertion operator). In the case of the PDPTW-SL, a number of requests are removed from the vehicle routes, and then these are re-inserted back into the solution. We use a total of 10 removal and five insertion operators. The operators are selected based on their past performance during the search process. We propose a mechanism to efficiently verify time- and load-synchronization constraints. We evaluate the effectiveness of the proposed heuristic on a set of PDPTW-SL instances with up to 100 requests and up to three SLs. The conclusions found in Chapters 2 and 3 of the thesis remain valid for large instances as well.

Chapter 5 investigates a stochastic variant of the PDPTW-SL, titled the *Pickup and Delivery Problem with Time Windows, Scheduled Lines and Stochastic Demands* (PDPTW-SLSD). More specifically, PDPTW-SLSD concerns scheduling a set of vehicles on a tactical level to serve a set of requests (e.g., zip codes), whose expected demands are known in distribution when planning, but are only revealed with certainty upon the vehicles' arrival. Similarly to PDPTW-SL, a part of the transportation plan can be carried out on limited-capacity scheduled public transportation line services. The considered problem concerns tactical decisions, as the generated routes are supposed to be executed on a regular basis over a given horizon (e.g., every day) and the requests may represent a number of customers

from a small region (e.g., zip codes). We propose a scenario-based Sample Average Approximation (SAA) approach for the PDPTW-SLSD. An ALNS embedded into SAA method is used to generate good-quality solutions. Computational results on instances with up to 40 requests (i.e., 80 locations) reveal that the integrated transportation networks can lead to operational cost savings of up to 28% compared with classical pickup and delivery systems, if uncertainty is taken into consideration during the planning process.

To help clarity, Table 1.1 presents the overview of the thesis. Column "Methodology" indicates the types of algorithms used to solve the considered problem, "Maximum instance size" indicates the largest (in terms of the number of requests) instance solved, and finally "Stochastic aspect" indicates which feature of the problem is considered to be stochastic in the considered problem. Finally, column "Research questions" indicates which specific research question is answered in the corresponding chapter.

Table 1.1 An overview of the thesis

Chapter	Methodology	Maximum instance size	Stochastic aspect	Research questions		
				1	2	3
2	Solver	11	–	✓		
3	B&P	50	–	✓	✓	
4	ALNS	100	–		✓	
5	ALNS within SAA	40	customers/demands		✓	✓

The chapters of the thesis are based on the following papers.

Chapter 2: Ghilas, V, E Demir, T Van Woensel. The pickup and delivery problem with time windows and scheduled lines. INFOR: Information Systems and Operational Research 54(2), 147–167, 2016.

Chapter 3: Ghilas, V, J-F Cordeau, E Demir, T Van Woensel. Branch and price algorithm for the pickup and delivery problem with time windows and scheduled lines. Major revision in Transportation Science.

Chapter 4: Ghilas, V, E Demir, T Van Woensel. An adaptive large neighborhood search algorithm for the pickup and delivery problem with time windows and scheduled lines. Computers and Operations Research 72, 12–30, 2016.

Chapter 5: Ghilas, V, E Demir, T Van Woensel. A scenario-based planning for the pickup and delivery problem with time windows, scheduled lines and stochastic demands. Transportation Research Part B: Methodological 91, 34–51, 2016.

Chapter 2

The pickup and delivery problem with time windows and scheduled lines

2.1 Introduction

A successful integration of freight and public transportation creates a seamless movement for both people and goods (i.e., small parcels). The actual integration is already being observed in long-haul freight transportation (Levin et al., 2012). For example, Norwegian Hurtigruten carries freight and people at the same time (Hurtigruten, 2016). In short-haul transportation, however, people and freight rarely share transportation modes although they largely use the same infrastructure (Lindholm and Behrends, 2012). Knowing this reality, potential efficiency gains for an integrated approach can be further investigated. To the best of our knowledge, integrated transport solutions to short-haul transportation have not been sufficiently taken into consideration in the literature.

This chapter focuses on opportunities to make use of available public transportation, which operates according to predetermined routes and schedules, as a part of freight journey. We assume that each considered public transport vehicle has a finite carrying capacity for freight requests apart from available spaces destined for passengers. Therefore, transferring freight requests to available scheduled lines (SLs) could be beneficial for the whole transportation system. We consider two transport options: direct and indirect shipments. The first one

implies that origin and destination points of a request are visited by using only one pickup and delivery (PD) vehicle. On the other hand, the second type of shipment implies that a request is picked up by a PD vehicle and transferred to a transfer node, which is assumed to be located nearby. From there, the request continues its journey on SLs. Afterwards, the request is picked up again by another PD vehicle to be delivered to its destination point. In this chapter, we denote this specific transportation problem as the *Pickup and Delivery Problem with Time Windows and Scheduled Lines* (PDPTW-SL). A schematic overview of the example network is shown in Figure 2.1.

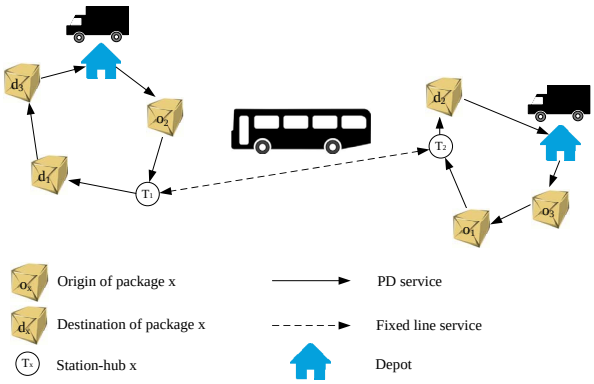

Figure 2.1 An illustration of the integrated transportation network

Figure 2.1 presents three requests that have their pickup and delivery locations close to two different transfer nodes. It would make sense to use a scheduled line that connects these two transfer nodes instead of using one PD vehicle as in the pickup and delivery problem (PDP). Hence, travel time savings of PD vehicles and operational cost reduction can be expected with the proposed integrated system. We note that in this thesis we focus on shared freight and public transportation application. It means that scheduled lines represent scheduled public transportation, e.g., buses, trains, etc. We will provide appropriate assumptions in Section 2.3. However we provide discussions regarding other applications of the PDPTW-SL in Chapter 6. In particular, integration of pickup and delivery operations with scheduled freight services.

It is noteworthy that small-sized freight (i.e., packages) is considered and the consolidation of the requests is done on both modes of transportation. In other words, a PD vehicle can perform several pickups before it delivers the requests to their delivery nodes, hence carrying larger quantities. Similarly, scheduled lines facilitate the consolidation in the start node, thus multiple freight requests can

be transported on the SL vehicles at a time (as long as the capacity constraint is not violated). Hence, freight consolidation leads to higher efficiency of the whole transportation system.

The contributions of this chapter are the following: *(i)* we introduce a mixed-integer program for the PDPTW-SL, *(ii)* we propose some effective valid inequalities for the given formulation and *(iii)* we analyze the benefits of using SLs in a pickup and delivery environment by comparing the solutions of the PDPTW with the solution of the PDPTW-SL. The remainder of the chapter is structured as follows. Section 2.2 provides a brief review of the related works. Section 2.3 introduces a mixed-integer programming formulation for the PDPTW-SL. Section 2.4 reports the results obtained from extensive computational experiments. Section 2.5 provides conclusions and future research directions.

2.2 Literature review

This section reviews the literature addressing the related problems of the PDPTW-SL. To the best of our knowledge, there is only limited literature on transfer options to scheduled line services. Thus, this thesis is one of the first attempts to investigate integrated short-haul freight transportation from an operational (tactical) planning perspective.

There are only a few practical initiatives to make use of the potential benefits of integrated transportation systems in urban environments. MULI was a demonstration project in the west of Germany between 1996 and 1999 (Trentini and Malhene, 2010). Special-design buses were used for both passengers and small-sized packages to reduce emissions. City Cargo Amsterdam was set up as a pilot experiment in 2007 (CargoTram, 2007). Two cargo trams were redesigned to transport packages in the city center of Amsterdam. In 2009, the project was abandoned due to lack of public funds.

In the context of PDP, we refer to Berbeglia et al. (2007) for a literature survey on solution methodologies. Cordeau and Laporte (2007) provide an overview of mathematical models and solution methodologies for single- as well as multi-vehicle dial-a-ride problem (DARP), which is an extension of the PDP. In a related study by Parragh (2010), the author investigates heterogeneity in vehicles and passengers along with waiting times of passengers. The author proposes a Branch & Cut (B&C) and a variable neighborhood search (VNS) algorithm to solve the DARP. Parragh et al. (2012) extend the heterogeneous DARP to consider driver-related constraints and propose a hybrid column generation and VNS heuristic algorithm.

One of the PDP extensions considering transfer opportunity between PD

vehicles is coined as PDP with transfers (PDP-T). Basically, a set of locations (i.e., transfer nodes), excluding pickup or delivery locations, can be used to perform request transfers from one vehicle to another. Other variants consider transfer oppotunity at every pickup and delivery node as well (Rais et al., 2013). In the PDP-T, a request can be picked up by one vehicle and delivered by another. A relevant study by Shang and Cuff (1996) is one of the earliest works on the PDP-T. The authors propose an insertion heuristic algorithm which leads to a cost reduction of up to 37% compared to the manual plans. In the work of Cortes et al. (2010), the authors propose an exact decomposition method, namely B&C, to solve the PDP-T. The proposed algorithm provides savings of up to 90% in terms of CPU time compared to traditional Branch & Bound (B&B) algorithm. In another work, Masson et al. (2012) introduce an adaptive large neighborhood search (ALNS) algorithm to solve the PDP-T. Their results indicate savings of up to 9% in terms of travel time, obtained by using intermediate transfers. In addition, Masson et al. (2014) propose an ALNS for the DARP with transfers (DARPT). The authors show that using transfers between vehicles may lead to significant savings in terms of operating costs (i.e., up to 8%). In addition, Petersen and Røpke (2011) investigate the PDP with cross-docking opportunities (PDPCD). The authors propose a large neighborhood search heuristic for the PDPCD. Their results suggest that improvements of up to 5% can be achieved due to cross-docking operations.

One of the first authors that conceptually introduced the idea of transporting freight using public transport was Nash (1982). Later, Liaw et al. (1996) investigate the opportunity to integrate scheduled line services with the DARP. The authors formulate the problem where transshipment to public scheduled transportation is allowed for passengers and wheel-chaired persons and propose two types of heuristics (i.e., online and offline). Their computational results show that improvements of up to 9% in terms of the number of serviced requests can be obtained using static planning and up to 7% with the help of online decision support. In another work, Aldaihani and Dessouky (2003) consider an integrated DARP with public transport and propose a two-stage heuristic algorithm (i.e., insertion and tabu search). Computational results indicate that solutions generated by the proposed algorithm lead to cost savings of 16.6%. In a related work of Häll et al. (2009), the authors introduce a mixed-integer programming (MIP) formulation to solve an integrated DARP. The authors consider transfers to fixed lines without modeling the schedule of the public transportation.

In our view, the PDP-T is a special case of the PDPTW-SL. In the latter, the requests are transferred to SL services at transfer nodes, by considering the timetable and capacity of these services. These transfer nodes are different physical locations. On the other hand, in the PDP-T the requests are transferred

between PD vehicles (without using an extra service) at specific locations (i.e., transfer nodes), without considering capacity or timing constraints. Thus, the PDP-T can be considered as a simplified PDPTW-SL, where transfer nodes of each scheduled line represent the same physical location, without timetabling and capacity limitations related to the transfer operations.

Trentini and Malhene (2010) study an overview of solutions for combining freight and passenger transportation that are used in practice. The authors divide transport solutions into three categories: shared road capacities (multi-use lanes, night deliveries, etc.), shared public transport services (buses, subway, etc.) and, finally, shared consolidation facilities (delivery bays, lockers, etc.). In a related study, Trentini et al. (2012) investigate a two-echelon vehicle routing problem (VRP) with transshipment in the context of public and freight integrated transportation system. The authors propose to use public transportation to ship the goods from a central distribution center to predefined stations (e.g., transshipment points). From there, a number of tricycles are used to deliver products to their final destinations. The authors propose ALNS along with a MIP formulation. Based on their results, the integrated solution proved to be more efficient in terms of pollution compared to traditional routing.

Table 2.1 provides an overview of the most relevant literature. We note that all papers in the table consider time-window aspect.

Table 2.1 An overview of the related literature

Research	Multi-depots	Hetero-geneous fleet	Transfers among PD vehicles	Transfers to fixed lines		Dyna-mics	Solution approach
				No sched.[1]	Sched.[2]		
Shang and Cuff (1996)	–	–	✓	–	–	–	Heuristic
Cortes et al. (2010)	✓	✓	✓	–	–	–	B&C
Masson et al. (2012)	–	–	✓	–	–	–	Metaheuristic
Masson et al. (2014)	✓	–	✓	–	–	–	Metaheuristic
Liaw et al. (1996)	–	–	–	–	✓	✓	Heuristic
Häll et al. (2009)	–	–	–	✓	–	–	B&B
Trentini et al. (2012)	–	–	–	–	✓	–	Metaheuristic
This chapter	✓	✓	–	–	✓	–	B&B

[1] Time schedules of the fixed lines are not considered; [2] Time schedules of the fixed lines are considered; "–" the aspect is not considered, "✓" the aspect is considered.

As seen in Table 2.1, the literature on scheduled line services is limited. Therefore, we aim at creating a building block for further research in this area, considering the extra complexity due to integrating SLs with current road transport flows.

2.3 Mathematical formulation for the PDPTW-SL

In this section, we introduce a mixed-integer programming formulation for the PDPTW-SL. All information (requests, demands, travel times, time windows) is assumed to be known in advance and an initial plan for the whole planning horizon (e.g., one day) is generated. More specifically, the plan is generated before operating day commences by considering only the requests known at the moment of planning (i.e., static planning). A solution to our model is a routing plan for the PD vehicles such that each request is served. Additionally, a time schedule for the PD vehicles and to-be-served requests is produced. The aim is to minimize the total operating costs. These include *(i)* operating (routing) cost of the PD vehicles, and *(ii)* the total transfer cost which is proportional to the number of shipped requests on the SLs.

2.3.1 Definitions and assumptions

We define the PDPTW-SL on a digraph $\mathcal{G} = (\mathcal{N}, \mathcal{A})$, where $\mathcal{N}$ represents the set of nodes and $\mathcal{A}$ represents the set of arcs. All other definitions and assumptions are described below.

Requests. Every request r involves a pickup node r and a delivery node $r + n$, with n being the total number of requests that need to be serviced. In addition, any request r has two required time windows: one for the pickup node ($[l_r, u_r]$), and one for the delivery node ($[l_{r+n}, u_{r+n}]$). The set of all requests is provided as $\mathcal{P}$. Note that we refer to a request by referring to its corresponding pickup node. Finally, demand d_r is known for each request.

PD Vehicles. The set $\mathcal{V}$ includes all available PD vehicles. Furthermore, Q_v corresponds to the carrying capacity of PD vehicle v. Index v in parameter Q_v indicates the fact that PD vehicles may be heterogeneous in terms of capacity, hence in terms of operating costs. The origin and destination of every PD vehicle is given as node o_v with its time window $[l_{o_v}, u_{o_v}]$.

Traveling and service times. Traveling times, along with service times are assumed to be static and deterministic. Constant Υ_{ij} represents the travel time from node i to node j. In addition, parameter s_i corresponds to the service time at node $i \in \mathcal{N}$.

Scheduled lines. Set $\mathcal{S}$ contains all physical transfer nodes (e.g., end-of-lines of the scheduled lines). Note that, for public transportation, intermediate stations may be disregarded, as otherwise package handling times might negatively affect passenger service level (e.g., delays). However, for other applications, e.g., container transportation where SLs represent container liners, considering

intermediate stops (e.g., port calls) can be considered by modeling each segment (between two consecutive stations) as a separate scheduled line. Set $\mathcal{E}$ represents all physical SLs. Each SL $(i,\ j)$ connects the end-stations (i.e., end-of-the-line stations) such that $i,\ j \in \mathcal{S}$. From the modeling perspective, note that arcs $(i,\ j)$ and $(j,\ i)$, $i,\ j \in \mathcal{S}$, represent two SLs. Each fixed line $(i,\ j)$ has a set of indices $\mathcal{K}^{ij}$ for the scheduled departures from node i, such that p_{ij}^{w} represents the departure time, $\forall\ (i,j) \in \mathcal{E},\ w \in \mathcal{K}^{ij}$. Note that each scheduled line may have different frequency than other lines, thus the size of $\mathcal{K}^{ij}$ may differ. Note that the considered scheduled lines may represent public transportation (e.g., buses, trains, trams, etc.), as well as freight scheduled transportation (e.g., liner shipping, rail, airlines, etc.).

Furthermore, it is assumed that SL vehicles are designed to carry a limited amount of requests, thus implying a finite carrying capacity $k_{ij},\ \forall\ (i,j) \in \mathcal{E}$. If considered in the shared public and freight transportation application, it is also assumed that freight carrying capacity is not influenced by the passenger demand. Similarly to air transportation, during low-demand season, airlines carry certain cargoes in the passenger compartment (i.e., dynamically modified compartment size), the considered scheduled lines may have a capacity dependent on the trip, e.g., k_{ij}^{w} which indicates the freight capacity during departure p_{ij}^{w}.

Each physical transfer node is assumed to be a consolidation location for the requests (e.g. DHL-Packstation (2016)). Namely, a secured area for temporary storage of the requests is available. The requests can be stored until their departure time (either by SLs or PD vehicle) at these locations. Furthermore, as multiple freight carriers may be using SL services, we assume that each carrier is assigned a part of the storage at the transfer nodes and SL vehicles capacity (e.g., contract-based agreement). Hence, the considered capacity is not affected by the demand of the other actors involved.

In addition, it is also assumed that cost η_{ij} per each unit of request shipped on the SL $(i,\ j)$ includes transportation, handling (transshipment) and storage costs. Handling of the requests (transfer to/from SLs) is assumed to be the responsibility of the SL service provider (e.g., a driver of the SL vehicle) and it might be the largest cost component of η_{ij}. Therefore, a natural way is to explicitly model the number of transfers. However, to trade-off complexity and practicality, we chose to consider η_{ij} as a cost for using SLs. Additionally, there might be a number of design issues related to integrated systems. First of all, choosing the right SL services (i.e., the number of SLs) and the right frequency. Furthermore, transfer node infrastructure, such as storage, temporary parking of the PD vehicles, transfer infrastructure (e.g, cranes), is another practical issue. Re-design of the SL (public transportation) vehicles might be an issue, as it might require investment. These decisions belong to strategic decision level.

For modeling purposes, each physical SL $(i, j) \in \mathcal{E}$ (e.g. arcs (1, 2) and (2, 1) in the example shown in Figure 2.2.a) gets n replications as in Häll et al. (2009). This is done to consider multiple visits and waiting times of the PD vehicles at the transfer nodes. Therefore, $\mathcal{F}$ contains all replicated SLs (e.g. arcs $(1a, 2a)$, $(1b, 2b)$, $(2a, 1a)$ and $(2b, 1b)$ in Figure 2.2.b).

We note that the replication process has some drawbacks. First of all, it increases the size of the considered network, thus increasing the number of arcs. In addition, it generates symmetry in the solution space, which consequently leads to degenerate solutions. To slightly reduce the complexity, each replication is assigned to one request, and only that specific request can be shipped on the assigned SL (refer to Figure 2.2.b). However, we modeled the transfer nodes differently (e.g., replicate the timing decision variables per number of visit at a transfer node), but it did not prove to be successful in considering waiting times at transfer nodes and different operations at transfer node at a time (e.g., re-collection and shipping a request). Otherwise, stricter assumptions needed to be considered regarding transfer nodes, e.g., only one visit per transfer node to be allowed.

As a consequence of the replication process, every physical transfer node in $\mathcal{S}$ (e.g., nodes 1 and 2 in Figure 2.2.a) gets n replications, thus obtaining the set of replicated transfer nodes as $\mathcal{T}$ (e.g., $\mathcal{T} \equiv \{1a, 1b, 2a, 2b\}$ in Figure 2.2.b). For modeling purposes, let $\psi^t, \forall\, t \in \mathcal{T}$ show the physical transfer node represented by t (e.g., $\psi^{1a} = \psi^{1b} = 1$). Consequently, $\mathcal{T}^t$ can be defined as $\{i \in \mathcal{T} | \psi^i = \psi^t, i \neq t\}$, $\forall\, t \in \mathcal{T}$ (e.g., in Figure 2.2.b, $\mathcal{T}^{2a} = \{2b\}$, as $2a$ and $2b$ represent physical transfer node 2). Note that $\mathcal{T}^t \subset \mathcal{T}, \forall\, t \in \mathcal{T}$.

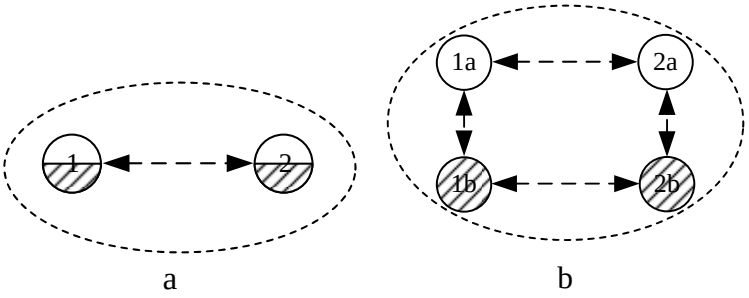

Figure 2.2 Physical and virtual scheduled lines

The proposed model can consider any complex SL network topology. Thus, we also assume that the requests can be transferred from one SL to another, by considering the scheduling and capacity constraints. Table 2.2 shows the parameters used in our model.

Table 2.2 Parameters

Notation	Definition
δ	Number of depots
Υ_{ij}	Traveling time from node i to node j
s_i	Service time at node i
k_{ij}	Package carrying capacity on the fixed line (i, j)
$[l_i, u_i]$	Time window at node i
p_{ij}^w	Departure time from node i, on the fixed line (i, j), indexed by w
τ	The amount of replicated transfer nodes (i.e. $\mid \mathcal{T} \mid$)
f_i^r	$\begin{cases} 1 & \text{if node } i \text{ is the origin node of request } r, \\ 0 & \text{if node } i \text{ is an intermediate node}, \forall i \in \mathcal{N}_2 \setminus \{r; r+n\} \text{(see Table 2.3)} \\ -1 & \text{if node } i \text{ is the destination of } r. \end{cases}$
ϕ_v	The routing cost of one time unit for vehicle v
η_{ij}	The cost of shipping one unit of parcel on the SL (i, j)
Q_v	Freight carrying capacity of vehicle v

The sets can be represented as in Table 2.3.

Table 2.3 Sets

Notation	Definition
$\mathcal{O}$	Set of depots, $\mathcal{O} \equiv [1, ..., \delta]$ (i.e. $o_v \in \mathcal{O}, \forall\ v \in \mathcal{V}$)
$\mathcal{P}$	Set of requests (pickup nodes), $\mathcal{P} \equiv [\delta + 1, \cdots, \delta + n]$
$\mathcal{D}$	Set of delivery nodes, $\mathcal{D} \equiv [\delta + n + 1, \cdots, \delta + 2n]$
$\mathcal{T}$	Set of replicated transfer nodes, $\mathcal{T} \equiv [\delta + 2n + 1, \cdots, \delta + 2n + \tau]$ (e.g., nodes 1a, 1b, 2a and 2b in Figure 2.2.b)
$\mathcal{T}^t$	Set of replicated transfer nodes that represent the same physical transfer node as t (e.g., in Figure 2.2.b, $\mathcal{T}^{2a} = \{2b\}$, $\mathcal{T}^{1a} = \{1b\}$, etc.)
$\mathcal{N}$	Set of nodes in $\mathcal{G}$; $\mathcal{P} \cup \mathcal{D} \cup \mathcal{O} \cup \mathcal{T} \equiv \mathcal{N}$
$\mathcal{N}_1$	$\mathcal{P} \cup \mathcal{D}$
$\mathcal{N}_2$	$\mathcal{P} \cup \mathcal{D} \cup \mathcal{T}$
$\mathcal{V}$	Set of PD vehicles
$\mathcal{E}$	Set of physical fixed lines which is defined as (i, j) with associated $\mathcal{K}^{ij}$ and k_{ij}
$\mathcal{K}^{ij}$	Set of indices for the departure times from the physical transfer node i on fixed line $(i, j) \in \mathcal{E}$
$\mathcal{F}$	Set of replicated fixed lines which is defined as (i, j) with associated $\mathcal{K}^{\psi^i \psi^j}$
$\mathcal{F}^r$	Set of replicated fixed lines associated with request r (e.g., in Figure 2.2 $\mathcal{F}^a = \{(1a, 2a), (2a, 1a)\}$)
$\mathcal{F}^t$	Set of replicated fixed lines connected to the replicated transfer node t (e.g., in Figure 2.2, $\mathcal{F}^{1a} = \{(1a, 2a), (2a, 1a)\}$)
$\mathcal{F}^{ij}$	Set of replicated fixed lines associated with a physical fixed line $(i, j) \in \mathcal{E}$ (e.g. in Figure 2.2, $\mathcal{F}^{1,2} = \{(1a, 2a), (1b, 2b)\}$, $\mathcal{F}^{2,1} = \{(2a, 1a), (2b, 1b)\}$)
$\mathcal{A}$	Set of arcs in $\mathcal{G}$ defined by $\mathcal{N} \times \mathcal{N}$, (note that $\mathcal{A} \setminus \mathcal{A}^1 \equiv \mathcal{F} \cup \{(i, j) \mid i \in \mathcal{O}, j \in \mathcal{N}_2\} \cup \{(i, j) \mid i \in \mathcal{N}_2, j \in \mathcal{O}\}$)
$\mathcal{A}^1$	$= \mathcal{N}_2 \times \mathcal{N}_2 \setminus \mathcal{F}$

Table 2.4 shows the decision variables used in the proposed formulation.

Table 2.4 Decision variables

Notation	Definition
x_{ij}^v	A binary variable with value 1 if arc (i, j) is traversed by PD vehicle v, 0 otherwise, $\forall\ (i, j) \in \mathcal{A}$, $v \in \mathcal{V}$
α_v	A continuous variable that indicates the time vehicle v returns to its depot, $\forall\ v \in \mathcal{V}$
β_i	A continuous variable that indicates the departure time of a PD vehicle from node i, $\forall\ i \in \mathcal{N}$
y_{ij}^r	A binary variable with value 1 if arc (i, j) is traversed by request r, 0 otherwise, $\forall\ i, j \in \mathcal{N}_2$, $r \in \mathcal{P}$
γ_i^r	A continuous variable that indicates the departure time of request r from node i, $\forall\ i \in \mathcal{N}_2$, $r \in \mathcal{P}$
q_{ij}^{rw}	A binary variable with value 1 if replicated SL (i, j) is used by request r that departs from i at time p_{ij}^w, 0 otherwise, $\forall\ r \in \mathcal{P}$, $(i, j) \in \mathcal{F}^r$, $w \in \mathcal{K}^{\psi^i \psi^j}$

2.3.2 An arc-based formulation

The PDPTW-SL is formalized as an arc-based MIP, as follows.

$$\min \quad \sum_{(i,j)\in\mathcal{A}} \sum_{v\in\mathcal{V}} \phi_v \Upsilon_{ij} x_{ij}^v + \sum_{r\in\mathcal{P}} \sum_{(i,j)\in\mathcal{F}^r} \sum_{w\in\mathcal{K}^{\psi^i\psi^j}} \eta_{ij} d_r q_{ij}^{rw} \tag{2.1}$$

Term (2.1) minimizes the total travel cost of the PD vehicles and the cost of using SLs for the transferred requests.

subject to

The routing and flow constraints

$$\sum_{i\in\mathcal{N}} \sum_{v\in\mathcal{V}} x_{ij}^v = 1 \qquad \forall j \in \mathcal{N}_1 \tag{2.2}$$

$$\sum_{i\in\mathcal{N}_2} x_{o_v,i}^v \leq 1 \qquad \forall v \in \mathcal{V} \tag{2.3}$$

$$\sum_{i\in\mathcal{N}} \sum_{v\in\mathcal{V}} x_{it}^v \leq 1 \qquad \forall t \in \mathcal{T} \tag{2.4}$$

$$\sum_{j\in\mathcal{N}} x_{ij}^v - \sum_{j\in\mathcal{N}} x_{ji}^v = 0 \qquad \forall i \in \mathcal{N}, v \in \mathcal{V} \tag{2.5}$$

$$\sum_{j\in\mathcal{N}_2} y_{ij}^r - \sum_{j\in\mathcal{N}_2} y_{ji}^r = f_i^r \qquad \forall r\in\mathcal{P}, i\in\mathcal{N}_2 \quad (2.6)$$

$$\sum_{i\in\mathcal{N}}\sum_{v\in\mathcal{V}} x_{it}^v \le \sum_{r\in\mathcal{P}}\sum_{(i,j)\in\mathcal{F}^t} y_{ij}^r \qquad \forall t\in\mathcal{T} \quad (2.7)$$

$$\sum_{r\in\mathcal{P}} d_r y_{ij}^r \le \sum_{v\in\mathcal{V}} Q_v x_{ij}^v \qquad \forall (i,j)\in\mathcal{A}^1 \quad (2.8)$$

In the first block of constraints, (2.2) assure that all pickup and delivery nodes are visited exactly once. Constraints (2.3) make sure that each considered PD vehicle is used at most once, and constraints (2.4) avoid visiting each replicated transfer node more than once. Flow balance for the PD vehicles is assured by constraints (2.5). Constraints (2.6) enforce flow balance for each request. Constraints (2.7) assure that a PD vehicle should pick up/drop off a request at a replicated transfer node related to a SL, if there is a flow on this specific SL. Constraints (2.8) force the capacity of each PD vehicle.

The scheduling constraints

$$y_{ij}^r = 1 \implies \gamma_j^r \ge \gamma_i^r + \Upsilon_{ij} + s_j \qquad \forall r\in\mathcal{P}, i,j\in\mathcal{N}_2 \quad (2.9)$$

$$\sum_{v\in\mathcal{V}} x_{ij}^v = 1 \implies \beta_j \ge \beta_i + \Upsilon_{ij} + s_j \qquad \forall i\in\mathcal{N}, j\in\mathcal{N}_2 \quad (2.10)$$

$$x_{i,o_v}^v = 1 \implies \alpha_v \ge \beta_i + \Upsilon_{i,o_v} + s_{o_v} \qquad \forall i\in\mathcal{N}_2, v\in\mathcal{V} \quad (2.11)$$

$$\beta_{r+n} \ge \beta_r + \Upsilon_{r,r+n} + s_{r+n} \qquad \forall r\in\mathcal{P} \quad (2.12)$$

$$l_i \le \beta_i - s_i \le u_i \qquad \forall i\in\mathcal{N}_1 \quad (2.13)$$

$$l_{g_v} \le \alpha_v \le u_{o_v} \qquad \forall v\in\mathcal{V} \quad (2.14)$$

$$\sum_{w\in\mathcal{K}^{\psi^i\psi^j}} q_{ij}^{rw} = y_{ij}^r \qquad \forall r\in\mathcal{P}, (i,j)\in\mathcal{F}^r \quad (2.15)$$

$$q_{ij}^{rw} = 1 \quad \textit{and} \quad y_{ij}^r = 1 \implies \gamma_i^r = p_{ij}^w \qquad \forall r\in\mathcal{P}, (i,j)\in\mathcal{F}^r, w\in\mathcal{K}^{\psi^i\psi^j} \quad (2.16)$$

$$\sum_{r\in\mathcal{P}}\sum_{(a,b)\in\mathcal{F}^{ij}} d_r q_{ab}^{rw} \le k_{ij} \qquad \forall (i,j)\in\mathcal{E}, w\in\mathcal{K}^{ij} \quad (2.17)$$

Timing for each request is considered in constraints (2.9). Similarly for PD vehicles, scheduling is updated in constraints (2.10) and (2.11). Constraints (2.12) assure the precedence constraints, and constraints (2.13)–(2.14) enforce the time windows. Constraints (2.15)–(2.16) assure the scheduling of the available SLs. Constraints (2.17) make sure that the package carrying capacity on the scheduled

line is not exceeded.

The synchronization constraints

$$\sum_{j \in \mathcal{N}_1} y_{ij}^r = 1 \implies \gamma_i^r = \beta_i \qquad \forall r \in \mathcal{P}, i \in \mathcal{T} \quad (2.18)$$

$$\sum_{j \in \mathcal{N}_2} y_{ij}^r = 1 \implies \gamma_i^r = \beta_i \qquad \forall r \in \mathcal{P}, i \in \mathcal{N}_1 \quad (2.19)$$

$$\sum_{i \in \mathcal{N}_2} y_{i,r+n}^r = 1 \implies \gamma_{r+n}^r = \beta_{r+n} \qquad \forall r \in \mathcal{P} \quad (2.20)$$

$$y_{tj}^r = 1 \implies \gamma_t^r = \beta_t \qquad \forall r \in \mathcal{P}, t \in \mathcal{T}, j \in \mathcal{T}^t \quad (2.21)$$

Time synchronization between requests and PD vehicles is considered in constraints (2.18)–(2.21). For clarity reasons, we present constraints (2.9)–(2.11), (2.16) and (2.18)–(2.21) as implications. However, well-known big-M technique can be used to linearize these constraints.

Decision variable domains

$$x_{ij}^v \in \{0,1\} \qquad \forall (i,j) \in \mathcal{A}, v \in \mathcal{V} \quad (2.22)$$

$$y_{ij}^r \in \{0,1\} \qquad \forall i,j \in \mathcal{N}_2, r \in \mathcal{P} \quad (2.23)$$

$$\alpha_v \in R^+ \qquad \forall v \in \mathcal{V} \quad (2.24)$$

$$\gamma_i^r \in R^+ \qquad \forall i \in \mathcal{N}_2, r \in \mathcal{P} \quad (2.25)$$

$$\beta_i \in R^+ \qquad \forall i \in \mathcal{N} \quad (2.26)$$

$$q_{ij}^{rw} \in \{0,1\} \qquad \forall r \in \mathcal{P}, (i,j) \in \mathcal{F}^r, w \in \mathcal{K}^{\psi^i \psi^j} \quad (2.27)$$

Since PDPTW-SL is an extension of the PDPTW, it is an NP-hard problem. We also note that an optimal solution for the PDPTW represents an upper bound for the PDPTW-SL because the opportunity of using SLs does not restrict the option of direct shipping. The complexity of the PDPTW-SL based on its input is as follows:

- $O(|\mathcal{N}|^2 \cdot |\mathcal{V}|)$, if $|\mathcal{V}| \geq n$,
- $O([|\mathcal{N} \setminus \mathcal{O}|]^2 \cdot n)$, if $|\mathcal{V}| < n$.

On the other hand, the complexity of the PDPTW based on its input is $O([|\mathcal{N} \setminus \mathcal{T}|]^2 \cdot |\mathcal{V}|)$. It is noted that $|\mathcal{T}| = |\mathcal{S}| \cdot n$ and in most of the cases $n > |\mathcal{V}|$. Hence, the complexity of the PDPTW-SL is greater then or equal to the complexity of the PDPTW (i.e., O(PDPTW-SL) $\geq$ O(PDPTW)).

Figure 2.3 presents an illustrative example of the synchronization constraints. A vehicle leaves its depot to pickup request r from transfer node T. This implies that request r will depart from T at the same time as the vehicle picks it up ($\beta_T = \gamma^r_T$). Afterwards, the vehicle delivers the request to its delivery node ($r + n$), where timing of vehicle and requests should also be synchronized.

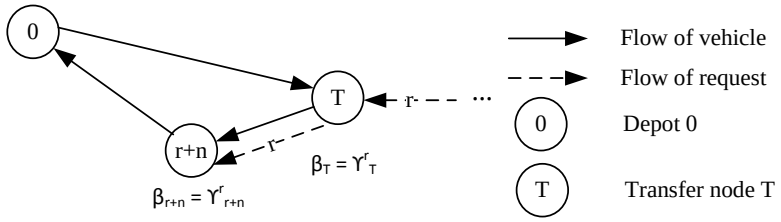

Figure 2.3 An example of synchronization constraints

Considering the explosion in complexity along with the increase in instance size, a certain preprocessing needs to be implemented in order to reduce the size of the network. The subsequent section will introduce several elimination rules.

Preprocessing

A preprocessing was implemented in order to reduce the number of decision variables. Hence, some of the infeasible arcs in graph $\mathcal{G}$ are removed.

- A vehicle cannot leave from and return to a depot other than its own.
- No vehicle can travel from the destination to the origin of the same request.
- No PD vehicle can travel between nodes i and j, if $(i, j) \in \mathcal{F}$.
- No request r can travel between origin and destination of a fixed line other than arc $(i, j) \in \mathcal{F}^r$.
- No request r can travel from the destination to the origin of the same request.
- No request is allowed to travel to/from any depot.
- No flow is allowed between nodes i and j, if $l_i + s_i + \Upsilon_{ij} > u_j$.
- No request can travel from its delivery node to any other node.
- No request can travel to its pickup node from any other node.

In addition, some elimination rules related to transfer nodes can be applied.

- An arc (t_1, t_2) is infeasible if $l_r + s_r + \Upsilon_{r,t_1} + s_{t_1} + \Upsilon_{t_1,t_2} + s_{t_2} + \Upsilon_{t_2,r+n} > u_{r+n}$, $\forall\, r \in \mathcal{P}$, $(t_1, t_2) \in \mathcal{F}^r$.

The number of departure times on the SLs for each request can be reduced by considering the time windows. For example, no request r can depart at a time earlier than $(l_r + s_r + \Upsilon_{r,t_1} + s_{t_1})$ on the SL (t_1, t_2), $\forall\, r \in \mathcal{P}$, $(t_1, t_2) \in \mathcal{F}^r$. Similarly, no request r can depart at a time later than $(u_{r+n} - s_{r+n} - \Upsilon_{t_1,r+n} - s_{t_1} - \Upsilon_{t_2,t_1} - s_{t_2})$ on a SL (t_2, t_1), $\forall\, r \in \mathcal{P}$, $(t_2, t_1) \in \mathcal{F}^r$.

Model tightening

As the proposed arc-based formulation grows very rapidly in size with the number of requests, number of SLs and departure times on each line, some additional constraints (valid inequalities) may be added to improve the formulation, hence obtain better lower bounds and smaller branch and bound trees. This section introduces three sets of cuts (i.e., flow tightening, vehicle service tightening, depot strengthening) that are valid for the PDPTW-SL.

We state Proposition 2.1 for *flow tightening* as follows.

Proposition 2.1 *The following inequality is valid for the PDPTW-SL:*

$$\sum_{v \in \mathcal{V}} x_{ij}^v \geq y_{ij}^r \qquad \forall (i,j) \in \mathcal{A}^1, r \in \mathcal{P} \tag{2.28}$$

These constraints can be explained as follows. For any request r and any arc (i,j), which can be traversed by a request, the value of the total vehicle flow (i.e., $\sum_{v \in \mathcal{V}} x_{ij}^v$) should be at least the value of the flow of request r (i.e., y_{ij}^r). In other words, these constraints maximize the values of vehicle flow variables, which serve request flows.

Proof: Let $(i,j) \in A^5$. In addition, let x_{ij}^v, $\forall\, v \in \mathcal{V}$, $(i,j) \in \mathcal{A}$ and y_{ij}^r, $\forall\, r \in \mathcal{P}$, $i, j \in \mathcal{N}_2$ be binary variables. We consider two cases: *(i)* PD vehicle does not traverse $(i,j) \in \mathcal{A}^1$ (i.e. $\sum_{v \in \mathcal{V}} x_{ij}^v = 0$) and *(ii)* PD vehicle travels on $(i,j) \in \mathcal{A}^1$ (i.e. $\sum_{v \in \mathcal{V}} x_{ij}^v = 1$).

In case *(i)*, due to capacity constraints (2.8), no request can travel on (i,j) (i.e. $y_{ij}^r = 0, \forall\, r \in \mathcal{P}$). In case *(ii)*, any request can travel on (i,j) as long as constraints (2.8) are satisfied. Therefore, $y_{ij}^r \in \{0, 1\}$, $\forall\, r \in \mathcal{P}$ ensure that $\sum_{v \in \mathcal{V}} x_{ij}^v \geq y_{ij}^r$, $\forall$

$r \in \mathcal{P}$. □

Example 2.1 Let there be a flow between i and j traversed by a vehicle v with its capacity equal to two units (i.e. $Q_v = 2$). In addition, two requests, r_1 and r_2, each having demand one (i.e., $d_{r_1} = d_{r_2} = 1$) travel on the same arc. Assume a linear programming (LP) relaxation solution with following values: $x_{ij}^v = 0.2$, $y_{ij}^{r_1} = 0.3$ and $y_{ij}^{r_2} = 0.1$. Hence, constraints (2.8) are satisfied ($0.3 + 0.1 \leq 2 \times 0.2$). By using constraints (2.28), x_{ij}^v is restricted to be at least 0.3 and $y_{ij}^{r_1} = 0.3$ and $y_{ij}^{r_2} = 0.1$, respectively. The resulting inequality is valid ($0.3 + 0.1 \leq 2 \times 0.3$).

A second proposition for *vehicle service tightening* is given in Proposition 2.2.

Proposition 2.2 *The following inequalities are valid for the PDPTW-SL:*

$$\sum_{i \in \mathcal{N}} x_{i,r}^v - \sum_{i \in \mathcal{N}} x_{r+n,i}^v \leq \sum_{(i,j) \in \mathcal{F}^r} y_{ij}^r \qquad \forall r \in \mathcal{P}, v \in \mathcal{V} \tag{2.29}$$

$$\sum_{i \in \mathcal{N}} x_{i,r}^v - \sum_{i \in \mathcal{N}} x_{r+n,i}^v \geq -\sum_{(i,j) \in \mathcal{F}^r} y_{ij}^r \qquad \forall r \in \mathcal{P}, v \in \mathcal{V} \tag{2.30}$$

The aim of these constraints is to improve the definition of precedence constraints, as in this formulation these are slightly relaxed (i.e. a request can be served by two PD vehicles). In the original formulation, precedence is assured by using timing variables (i.e., constraints (2.12)). Constraints (2.29) and (2.30) assure the use of SLs, if the pickup and delivery nodes of a request are visited by different vehicles.

PROOF: Let $r \in \mathcal{P}$ and $v \in \mathcal{V}$. In addition, let x_{ij}^v, $\forall\ v \in \mathcal{V}$, $(i,j) \in \mathcal{A}$ and y_{ij}^r, $\forall\ r \in \mathcal{P}$, $i,j \in \mathcal{N}_2$ be binary variables. We consider three cases: *(i)* r is served by PD vehicle v; *(ii)* r is served by two different PD vehicles, v and e.g. $v_1 \in \mathcal{V}$; *(iii)* r is served by other PD vehicle(s) v_1 and/or $v_2 \in \mathcal{V}$;

In case *(i)*, due to constraints (2.2), pickup ($\sum_{i \in \mathcal{N}} x_{i,r}^v = 1$) and delivery nodes ($\sum_{i \in \mathcal{N}} x_{r+n,i}^v = 1$) of r are visited by v, hence the left-hand side (LHS) of the constraints (2.29) and (2.30) take a value of zero. Furthermore, $\sum_{(i,j) \in \mathcal{F}^r} y_{ij}^r \geq 0$ ensures that even request r is served by v, it may still use a scheduled line as part of its journey.

In case *(ii)*, the pickup or the delivery node of r is visited by v. Hence, LHS of constraints (2.29) and (2.30) may take two values: –1 and 1. In order to satisfy constraints (2.7), $\sum_{(i,j) \in \mathcal{F}^r} y_{ij}^r > 0$, which implies that r must use a scheduled line as part of its journey.

In case *(iii)*, PD vehicle v does not visit the pickup, nor the delivery node of r, thus LHS gets value zero. Hence, $\sum_{(i,j)\in\mathcal{F}^r} y_{ij}^r \geq 0$ guarantees that r can use a scheduled line, as it may be served by one or two other PD vehicles. The result follows from these cases. □

Example 2.2 Let x be an optimal LP solution, such that constraints (2.2)–(2.21) are satisfied. In addition, in this solution vehicle v_1 visits pickup node r and

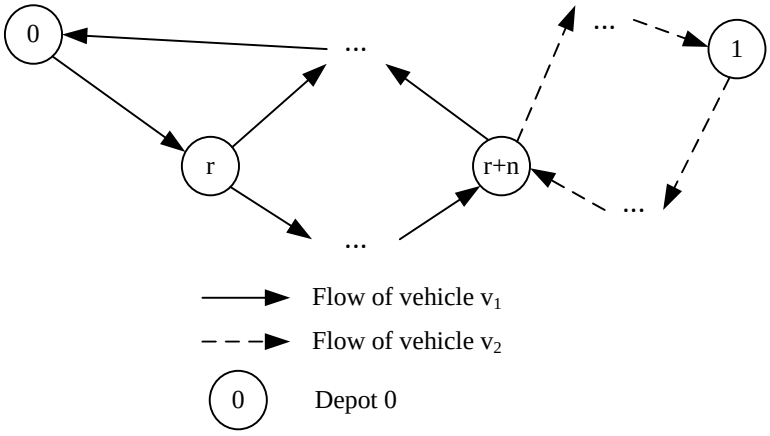

Figure 2.4 An example of violation for constraints (2.29) and (2.30)

delivery node $r+n$ with fractional flow values. Vehicle v_2 visits delivery node $r+n$ only, without visiting the corresponding pickup node. However, request r does not use the SLs (i.e., $\sum_{(i,j)\in\mathcal{F}^r} y_{ij}^r = 0$). The constraints above will avoid fractional routes similar to v_2 (i.e., violated precedence constraints), as the constraints are added for each request and vehicle.

A depot-related valid inequality is given in Proposition 2.3.

Proposition 2.3 *The following inequality is valid for the PDPTW-SL:*

$$M \sum_{j\in\mathcal{N}_2} x_{o_v,j}^v - \sum_{i\in\mathcal{N}_2} \sum_{j\in\mathcal{N}_2} x_{ij}^v \geq 0 \qquad \forall v \in \mathcal{V} \tag{2.31}$$

These constraints avoid vehicle sub-tours that do not leave/enter a depot.

PROOF: Let $v \in \mathcal{V}$ and x_{ij}^{v}, $\forall\, v \in \mathcal{V}$, $(i,j) \in \mathcal{A}$ be binary variable. Assume that each vehicle starts and ends its operating day at its depot (i.e., o_v). We know that any node $i \in \mathcal{N}_2$ cannot be visited before leaving the depot. Due to the timing constraints (2.10) and (2.11), this assumption holds and the departure of v from its depot should be smaller than the departure time from any other node. More specifically, if $\sum_{i\in\mathcal{N}_2}\sum_{j\in\mathcal{N}_2} x_{ij}^{v} > 0$, then $\sum_{j\in\mathcal{N}_2} x_{o_v,j}^{v}$ gets a value of 1 $\implies$ inequality (2.31) holds. □

Example 2.3 Assume a part of an LP solution with two PD vehicles, namely v_1, $v_2 \in \mathcal{V}$ and i, $j \in \mathcal{N}_2$ and $o_{v_1} = o_{v_2} = 0$ (see Figure 2.5). This LP solution satisfies constraints (2.2)–(2.21). Constraints (2.31) ensure that $x_{0i}^{v_1} > 0$ and/or $x_{0j}^{v_1} > 0$.

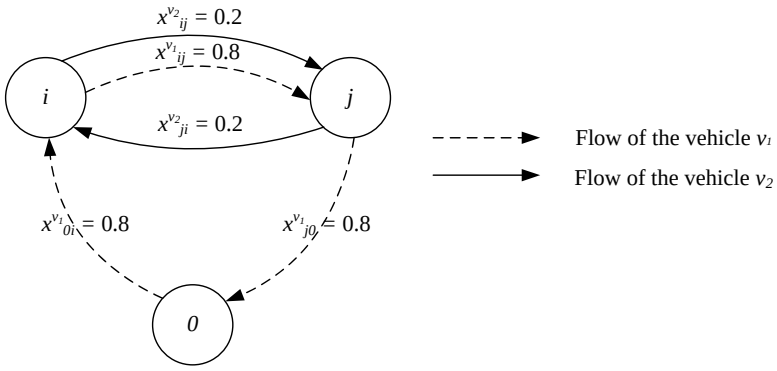

Figure 2.5 An example of violation for constraint (2.31)

Thus, the LP solution given in Figure 2.5 violates constraints (2.31). In addition, note that M can be substituted by $\mid \mathcal{N}_2 \mid$, which implies that a PD vehicle can visit each node in $\mathcal{N}_2$ at most once.

2.4 Computational results

This section presents the results obtained from solving the proposed arc-based formulation of the PDPTW-SL using CPLEX. The model is implemented in NetBeans IDE 7.1.1 using the corresponding CPLEX 12.1 libraries. Furthermore, all experiments are run on a server with four CPUs (2.4 GHz/6 cores) and 32 GB RAM.

2.4.1 Instance description

We have generated 3 sets of instances with up to three scheduled lines (triangular topology), namely R, C, and RC, which differ in the geographical distribution of the request nodes. More specifically, C consists of nodes positioned within at most 30 time units to one of the available three transfer nodes whereas RC includes nodes within 80 time units respectively, and R considers uniform-randomly distributed request nodes. The frequency of the SLs is one departure every 30 time units. Each instance contains up to 12 requests (i.e., 12 pickup and 12 delivery nodes) over 200×200 dimensional units on an Euclidean space. Instances are named in $G\_n\_sl\_v$ format, where G is the geographic characteristic of the customer nodes, n is the amount of freight requests that need to be serviced, sl is the amount of available SLs and v is the amount of available PD vehicles. In all cases, two depots with up to 3 heterogeneous PD vehicles each are considered.

We assume a planning horizon of 600 time units (i.e., 10 working hours). The widths of the time windows are generated randomly with an average of 40 time units. Service times are assumed to be up to three time units. The demand of each request is randomly chosen between one and three units. The carrying capacities of PD vehicles is randomly generated between 16 and 20 units. The SL capacity is assumed to be 15 demand units. The instance sets are available on `www.smartlogisticslab.nl`.

Tables 2.5 and 2.6 present further details regarding instances. In each table, the column *Instance* shows the identification of the instance. Table 2.5 provides the number of depots ($\mathcal{O}$), the number of PD vehicles ($\mathcal{V}$), the number of requests ($\mathcal{P}$), the number of fixed lines ($\mathcal{E}$) and their corresponding scheduled departure times. Table 2.6 provides the number of constraints, the number of variables in the resulting formulation, time to solve the LP relaxation and finally, objective function value of the LP relaxation. Corresponding values are computed with and without the proposed preprocessing.

We note that the driving cost per time unit of all PD vehicles is assumed to be 0.5 units. It seems reasonable considering operational costs such as fuel consumption, driver wage, insurance, and tax. The cost of each package request shipped on every scheduled line is set to 1 unit, which includes handling, storage and transportation costs.

As can be observed from Table 2.6, the results indicate that there are significant reductions in terms of the number of constraints and decision variables due to preprocessing. The average percentage of reductions for the number of constraints and variables is found to be 18.3% and 20.7%, respectively. Moreover, the improvement of the linear programming relaxation (LP) is on average 11.5%. In addition, the preprocessing decreases the number of constraints and variables for

Table 2.5 Input sets

Instance	$\lvert\mathcal{O}\rvert$	$\lvert\mathcal{V}\rvert$	$\lvert\mathcal{P}\rvert$	$\lvert\mathcal{E}\rvert$	$\lvert\mathcal{E}\rvert \cdot \lvert\mathcal{K}^{ij}\rvert$
C_6_1_4	2	4	6	2	36
C_7_1_4	2	4	7	2	36
C_8_1_4	2	4	8	2	36
C_9_1_4	2	4	9	2	36
C_10_1_4	2	4	10	2	36
C_11_1_4	2	4	11	2	36
C_12_1_4	2	4	12	2	36
RC_6_1_4	2	4	6	2	36
RC_7_1_4	2	4	7	2	36
RC_8_1_4	2	4	8	2	36
RC_9_1_4	2	4	9	2	36
RC_10_1_4	2	4	10	2	36
RC_11_1_4	2	4	11	2	36
RC_12_1_6	2	6	12	2	36
R_6_1_4	2	4	6	2	36
R_7_1_4	2	4	7	2	36
R_8_1_4	2	4	8	2	36
R_9_1_4	2	4	9	2	36
R_10_1_4	2	4	10	2	36
R_11_1_4	2	4	11	2	36
R_12_1_6	2	6	12	2	36

Table 2.6 The effect of preprocessing

Instance	No preprocessing					Preprocessing				
	PDPTW-SL			PDPTW		PDPTW-SL			PDPTW	
	Cons	Vars	LP	Cons	Vars	Cons	Vars	LP	Cons	Vars
C_6_1_4	5,208	5,055	188.72	910	382	4,363	4,206	241.18	892	286
C_7_1_4	7,714	7,406	205.48	1,199	508	6,412	6,052	224.29	1,105	376
C_8_1_4	11,069	12,448	220.06	1,528	646	9,244	10,199	240.52	1,460	466
C_9_1_4	15,123	16,629	232.57	1,897	788	12,727	13,671	254.59	1,803	552
C_10_1_4	20,059	21,589	237.87	2,306	934	17,008	17,822	262.26	2,156	634
C_11_1_4	26,008	27,485	247.40	2,755	1,120	21,590	22,104	267.91	2,671	752
C_12_1_4	33,063	34,321	266.29	3,244	1,302	27,702	27,834	291.58	3,175	854
RC_6_1_4	5,133	4,972	359.73	910	362	4,164	3,970	447.18	891	246
RC_7_1_4	7,619	7,304	358.80	1,199	488	6,169	5,770	374.64	1,092	328
RC_8_1_4	10,821	10,249	343.26	1,528	630	8,677	7,965	366.35	1,422	418
RC_9_1_4	14,753	13,790	355.60	1,897	772	12,129	10,991	388.38	1,752	504
RC_10_1_4	19,660	18,158	374.23	2,306	942	16,177	14,453	399.89	2,224	610
RC_11_1_4	25,429	23,209	409.71	2,755	1,112	21,122	18,636	438.84	2,641	708
RC_12_1_6	32,389	29,269	446.61	4,548	1,976	26,116	22,731	464.49	4,415	1,244
R_6_1_4	5,233	5,115	312.30	910	422	3,990	3,809	367.65	870	310
R_7_1_4	7,737	7,457	345.82	1,199	540	6,034	5,666	410.33	1,123	384
R_8_1_4	10,931	10,373	411.59	1,528	658	8,681	8,005	445.18	1,416	454
R_9_1_4	14,949	13,988	403.94	1,897	796	11,992	10,864	456.51	1,766	532
R_10_1_4	19,774	18,284	428.26	2,306	966	15,626	13,928	446.32	2,244	634
R_11_1_4	25,535	23,317	449.34	2,755	1,124	20,402	17,910	540.75	2,701	716
R_12_1_6	32,290	33,601	518.79	4,548	1,910	26,052	26,022	607.09	4,486	1,178
Average			338.87					377.90		

the PDPTW by 4.1% and 33.7%, respectively.

When comparing average numbers of constraints and variables of the PDPTW and PDPTW-SL, it can be observed that the latter model is more complex. On average, there are 85% more constraints and 95% more variables in PDPTW-SL than in the standard PDPTW. In addition, the solution space of the PDPTW-SL is larger than of the PDPTW because the optimal solution of the PDPTW represents an upper bound for the optimal solution of the PDPTW-SL (due to the

flexibility of using SLs).

2.4.2 CPLEX-standard versus CPLEX with user cuts

In Table 2.7, we indicate the lower bounds obtained at the root node (after applying preprocessing). Column *LP* shows the bounds obtained without applying any kind of cut generation. *CPX* shows the bounds obtained by activating automatic cut generation of CPLEX (without user-defined cuts). Finally, *Full* column shows the bounds obtained by applying both CPLEX and user-defined cuts. The last column indicates the improvement in the lower bound when using user cuts with CPLEX cuts at a time, compared to CPLEX cuts alone.

Table 2.7 Initial lower bounds

Instance	LP	CPX	Full	Improvement %
C_6_1_4	241.18	259.49	260.21	0.28
C_7_1_4	224.29	264.48	265.70	0.46
C_8_1_4	240.52	277.00	279.78	1.00
C_9_1_4	254.59	283.95	301.68	6.24
C_10_1_4	262.26	289.85	310.26	7.04
C_11_1_4	267.91	320.00	320.63	0.20
C_12_1_4	291.58	319.66	345.59	8.11
RC_6_1_4	447.18	487.68	487.76	0.02
RC_7_1_4	374.64	476.33	483.61	1.53
RC_8_1_4	366.35	475.43	481.60	1.30
RC_9_1_4	388.38	496.27	499.96	0.74
RC_10_1_4	399.89	489.07	503.43	2.94
RC_11_1_4	438.84	531.21	539.44	1.55
RC_12_1_6	464.49	564.20	571.24	1.25
R_6_1_4	367.65	412.01	416.16	1.01
R_7_1_4	410.33	426.34	439.02	2.97
R_8_1_4	445.18	481.26	504.35	4.80
R_9_1_4	456.51	492.14	514.09	4.46
R_10_1_4	446.32	497.07	515.62	3.73
R_11_1_4	540.75	602.04	622.92	3.47
R_12_1_6	607.09	671.09	681.27	1.52
	377.90	434.12	444.96	2.60

Tables 2.8 and 2.9 report the results obtained by using CPLEX with and without proposed cuts. In both cases, a time limit of 36,000 seconds (10 hours) was imposed to the solver. We indicate the objective function value, obtained GAP from the best bound, the best bound obtained, the CPU time (in seconds) and the number of explored nodes during the search process. Note that "–" means no integer solution was found within the imposed time limit.

Computational results obtained in this section show that instances with up to 11 requests can be solved to optimality in a reasonable time by using the proposed model with the valid inequalities introduced in Section 2.3.2. One can observe that taking advantage of the proposed cuts significantly improves the performance

Table 2.8 Results using CPLEX without user cuts

Instance	$\underline{z}$	$\overline{z}$	GAP %	Nodes	Time (s)
C_6_1_4	369.36	369.36	0.00	37,512	4,341
C_7_1_4	390.31	390.31	0.00	33,633	5,142
C_8_1_4	366.31	446.35	17.93	79,476	36,000
C_9_1_4	317.36	–	–	131,527	36,000
C_10_1_4	331.17	–	–	19,517	36,000
C_11_1_4	329.97	–	–	26,774	36,000
C_12_1_4	354.12	–	–	7,399	36,000
RC_6_1_4	572.76	572.76	0.00	12,708	404
RC_7_1_4	575.95	575.95	0.00	70,057	2,210
RC_8_1_4	585.32	585.32	0.00	310,184	13,392
RC_9_1_4	504.02	–	–	284,774	36,000
RC_10_1_4	512.08	–	–	104,181	36,000
RC_11_1_4	547.57	–	–	132,419	36,000
RC_12_1_6	578.09	–	–	107,558	36,000
R_6_1_4	416.16	416.16	0.00	981	1
R_7_1_4	473.06	473.06	0.00	295	1
R_8_1_4	558.17	558.17	0.00	980	35
R_9_1_4	632.42	632.42	0.00	14,941	2,116
R_10_1_4	636.05	636.05	0.00	200,928	16,729
R_11_1_4	661.39	748.29	11.61	16,9960	36,000
R_12_1_6	695.53	–	–	67,232	36,000

Table 2.9 Results using CPLEX with user cuts

Instance	$\underline{z}$	$\overline{z}$	GAP %	Nodes	Time (s)
C_6_1_4	369.36	369.36	0.00	20,790	3,334
C_7_1_4	390.31	390.30	0.00	20,789	3,762
C_8_1_4	384.93	446.35	13.76	47,787	36,000
C_9_1_4	327.43	493.66	33.67	24,290	36,000
C_10_1_4	335.89	–	–	9,081	36,000
C_11_1_4	347.68	–	–	4,278	36,000
C_12_1_4	366.81	–	–	2,100	36,000
RC_6_1_4	572.76	572.76	0.00	2,036	61
RC_7_1_4	575.95	575.95	0.00	2,208	161
RC_8_1_4	585.32	585.32	0.00	3,142	510
RC_9_1_4	593.70	593.70	0.00	70,292	32,204
RC_10_1_4	599.95	599.95	0.00	37,019	24,925
RC_11_1_4	624.48	624.48	0.00	12,879	18,128
RC_12_1_6	608.89	–	–	8,281	36,000
R_6_1_4	416.16	416.16	0.00	233	0
R_7_1_4	473.06	473.06	0.00	62	1
R_8_1_4	558.17	558.17	0.00	458	10
R_9_1_4	632.42	632.42	0.00	12,343	1,915
R_10_1_4	636.05	636.05	0.00	6,728	1,295
R_11_1_4	748.29	748.29	0.00	46,044	17,493
R_12_1_6	771.01	–	–	17,878	36,000

of CPLEX.

A total number of 10 instances were solved to optimality by both variants of the model within the time limit. On average, the CPU time was 4,437 seconds for CPLEX alone compared with 1,105 seconds for CPLEX with additional cuts.

The average number of explored nodes drops from 68,222 to 6,879. Proposed cuts helped in proving optimality within imposed time limit for the four instances that CPLEX without user-cuts failed to solve. For the data sets that were not solved to optimality by CPLEX alone, the user cuts helped obtaining tighter bounds or proving optimality.

2.4.3 Scenarios

In order to quantify the benefits of the proposed transportation system, we define and compare two scenarios. The first one considers proposed integrated transportation system (PDPTW-SL). The second scenario considers standard pickup and delivery system (PDPTW). CPLEX time limit is set to 24 hours and the best solution values are shown in Table 2.10.

We compare the scenarios by investigating: *(i)* total cost of the service provider and *(ii)* total driving time by PD vehicles. The values in bold indicate that CPLEX could not solve the instances to optimality within imposed time-limit.

Table 2.10 A comparison between PDPTW and PDPTW-SL

Instance	Total costs			Total driving time		
	PDPTW-SL	PDPTW	Savings %	PDPTW-SL	PDPTW	Savings %
C_6_1_4	369.36	461.70	20.0	728.7	923.40	21.1
C_7_1_4	390.31	464.49	16.0	768.59	928.98	17.3
C_8_1_4	446.35	483.50	7.7	882.67	967.00	8.7
C_9_1_4	**471.16**	498.19	–	**932.30**	996.37	–
C_10_1_4	**600.53**	515.25	–	**1,189.06**	1,030.49	–
C_11_1_4	–	524.69	–	–	1,049.38	–
C_12_1_4	–	598.55	–	–	1,197.10	–
RC_6_1_4	572.76	658.99	13.1	1,143.50	1,317.98	13.2
RC_7_1_4	575.95	662.18	13.0	1,149.88	1,324.36	13.2
RC_8_1_4	585.32	663.16	11.7	1,168.63	1,326.32	11.9
RC_9_1_4	593.70	703.30	15.6	1,181.38	1,407.00	16.0
RC_10_1_4	599.95	709.55	15.4	1,193.88	1,419.10	15.9
RC_11_1_4	624.48	742.52	15.9	1,242.94	1,485.04	16.3
RC_12_1_6	–	794.42	–	–	1,588.84	–
R_6_1_4	416.16	416.16	0.0	832.32	832.32	0.0
R_7_1_4	473.06	473.06	0.0	946.12	946.12	0.0
R_8_1_4	558.17	558.17	0.0	1,116.34	1,116.34	0.0
R_9_1_4	632.42	653.12	3.2	1,258.82	1,306.24	3.6
R_10_1_4	636.05	658.60	3.4	1,266.08	1,317.20	3.9
R_11_1_4	748.29	752.38	0.5	1,490.55	1,504.76	0.9
R_12_1_6	–	936.23	–	–	1,872.44	–

The PDPTW-SL proved, on average, 9% cost savings compared to the standard PDPTW. Note that for some R instances no savings were achieved as no packages were transferred to the scheduled lines. In this case, the optimal solution for the PDPTW is also found to be optimal for the PDPTW-SL. Overall, along with the increase in number of requests, the cost savings tend to remain significant (i.e., C, RC instances).

It is important to note that the effectiveness of the proposed system can be highly dependent on both, the spatial pattern of the requests and the configuration of scheduled lines. Although public transportation is integrated with vehicle

routing, in some cases no benefits can be achieved from an integrated system, as observed in some of R instances. Therefore, designing such a system involves strategic decisions related to the pattern of the scheduled lines (positioning of the transfer nodes relative to the demand cluster centres), the storage areas at the transfer nodes, and the re-design of the SL vehicles (e.g., extra freight capacity), that are not taken into consideration in this chapter (the focus is on operational costs of the system).

2.4.4 The effect of the number of SLs on the operating costs

In this section we evaluate the effect of the number of available SLs on the solution cost. A time limit of 24 hours has been imposed to CPLEX for the following experiments and the best solutions found are given in Table 2.11.

Table 2.11 The effect of the number of available SLs on the objective value function

Instance	One SL			Two SLs			Three SLs		
	$\overline{z}$	GAP %	$\underline{z}$	$\overline{z}$	GAP %	$\underline{z}$	$\overline{z}$	GAP %	$\underline{z}$
C_6	369.36	0.00	369.36	369.36	0.00	369.36	310.84	21.29	244.66
C_7	390.31	0.00	390.31	381.20	13.10	331.26	384.13	46.60	205.13
C_8	446.35	0.00	446.35	–	–	310.83	559.19	59.52	226.36
C_9	471.16	13.25	408.72	–	–	307.60	714.65	66.62	238.56
C_10	600.53	44.07	335.89	–	–	303.28	–	–	244.95
C_11	–	–	349.96	–	–	315.35	–	–	268.55
C_12	–	–	357.40	–	–	340.11	–	–	290.84
RC_6	572.76	0.00	572.76	572.76	0.00	572.76	520.10	0.00	520.10
RC_7	575.95	0.00	575.95	575.95	0.00	575.95	539.08	0.00	539.08
RC_8	585.32	0.00	585.32	585.32	0.00	585.32	548.45	16.89	455.82
RC_9	593.70	0.00	593.70	593.70	0.00	593.70	–	–	435.53
RC_10	599.95	0.00	599.95	–	–	512.73	–	–	442.22
RC_11	624.48	0.00	624.48	–	–	502.54	–	–	448.26
RC_12	–	–	608.89	–	–	504.78	–	–	475.81
R_6	416.16	0.00	416.16	416.16	0.00	416.16	416.16	0.00	416.16
R_7	473.06	0.00	473.06	473.06	0.00	473.06	470.15	0.00	470.14
R_8	558.17	0.00	558.17	535.27	0.00	535.27	490.77	0.00	490.77
R_9	632.42	0.00	632.42	579.38	0.00	579.38	647.27	20.49	514.64
R_10	636.05	0.00	636.05	584.89	0.00	584.89	584.89	4.42	559.06
R_11	748.29	0.00	748.29	687.36	4.10	659.36	–	–	571.73
R_12	–	–	795.18	–	–	683.56	–	–	627.47

The results indicate an intuitive insight: the more scheduled lines are considered, the more shipments on SLs are made. For the cases with optimal solutions, the average cost savings for the 2-SLs and 3-SLs are 2.4% and 6.2%, compared to the 1-SL case. Overall, the increase in savings can be up to 9% (i.e., RC_6_1_4) when comparing 1-SL case with 3-SLs case. We note that more available SLs lead to higher complexity. In other words, significantly fewer instances with 3 SLs could be solved, compared to e.g., 1- or 2-SLs cases.

2.4.5 The effect of the time windows on the operating costs

In this section we investigate the effect of the time-window tightness on the operational costs. Each pickup or delivery node is assigned a time window $[l_i^1$,

$u_i^1]$, such that $(u_i^1 - l_i^1) = (u_i - l_i)\rho$ and $|u_i^1 - u_i| = |l_i^1 - l_i|$, where $\rho \in \{0.5, 1.5, 2\}$. A time limit of 24 hours has been imposed to CPLEX for the following experiments. The results are shown in Tables 2.12 and 2.13.

Table 2.12 The effect of the time windows on the operating costs

Instance	$\rho = 0.5$				$\rho = 1$			
	$\overline{z}$	GAP %	$\underline{z}$	PDPTW	$\overline{z}$	GAP %	$\underline{z}$	PDPTW
C_6_1_4	458.05	0.00	458.05	470.56	369.36	0.00	369.36	461.70
C_7_1_4	463.72	0.00	463.72	478.65	390.30	0.00	390.30	464.49
C_8_1_4	480.01	0.00	480.01	541.38	446.35	0.00	446.35	483.50
C_9_1_4	509.34	0.00	509.34	556.16	471.16	13.25	408.72	498.19
C_10_1_4	–	–	413.86	667.58	600.53	44.07	335.89	515.25
C_11_1_4	753.73	45.03	414.32	670.30	–	–	349.96	524.69
C_12_1_4	–	–	403.86	737.40	–	–	357.40	598.55
RC_6_1_4	572.76	0.00	572.76	658.99	572.76	0.00	572.76	658.99
RC_7_1_4	575.95	0.00	575.95	662.18	575.95	0.00	575.95	662.18
RC_8_1_4	585.72	0.00	585.72	663.16	585.32	0.00	585.32	663.16
RC_9_1_4	594.09	0.00	594.09	724.88	593.70	0.00	593.70	703.30
RC_10_1_4	600.35	0.00	600.35	731.13	599.95	0.00	599.95	709.55
RC_11_1_4	624.88	0.00	624.88	773.34	624.48	0.00	624.48	742.52
RC_12_1_6	662.65	0.00	662.65	825.20	–	–	608.89	794.42
R_6_1_4	493.05	0.00	493.05	493.05	416.16	0.00	416.16	416.16
R_7_1_4	588.94	0.00	588.94	588.96	473.06	0.00	473.06	473.06
R_8_1_4	639.73	0.00	639.73	639.73	558.17	0.00	558.17	558.17
R_9_1_4	711.72	0.00	711.72	711.72	632.42	0.00	632.42	653.12
R_10_1_4	715.72	0.00	715.72	715.84	636.05	0.00	636.05	658.60
R_11_1_4	802.84	0.00	802.84	846.31	748.29	0.00	748.29	752.38
R_12_1_6	1,109.46	0.00	1,109.46	1,176.35	–	–	795.18	936.23

Table 2.13 The effect of the time windows on the operating costs

Instance	$\rho = 1.5$				$\rho = 2$			
	$\overline{z}$	GAP %	$\underline{z}$	PDPTW	$\overline{z}$	GAP %	$\underline{z}$	PDPTW
C_6_1_4	362.28	0.00	362.28	366.57	331.69	0.00	331.69	347.68
C_7_1_4	365.07	0.00	365.07	374.66	347.58	0.00	347.58	350.47
C_8_1_4	384.08	0.00	384.08	388.25	365.72	17.11	303.15	369.47
C_9_1_4	422.37	19.13	341.57	481.41	415.26	28.32	297.68	398.49
C_10_1_4	–	–	314.27	498.47	–	–	300.92	432.53
C_11_1_4	–	–	320.39	505.62	–	–	303.15	432.75
C_12_1_4	–	–	340.34	551.50	–	–	323.13	484.90
RC_6_1_4	572.76	0.00	572.76	658.99	547.75	0.00	547.75	630.45
RC_7_1_4	575.95	0.00	575.95	662.18	558.97	0.00	558.97	633.65
RC_8_1_4	583.60	0.00	583.60	663.15	568.35	0.00	568.35	634.62
RC_9_1_4	591.98	0.00	591.98	698.31	591.98	20.29	471.87	671.94
RC_10_1_4	714.18	27.47	518.03	704.57	–	–	458.38	679.95
RC_11_1_4	–	–	569.06	742.52	–	–	481.02	713.98
RC_12_1_6	–	–	571.59	794.42	–	–	511.02	769.79
R_6_1_4	416.16	0.00	416.16	416.16	397.16	0.00	397.16	397.16
R_7_1_4	468.02	0.00	468.02	468.02	411.77	0.00	411.77	411.77
R_8_1_4	552.51	0.00	552.51	558.17	535.32	0.00	535.32	558.17
R_9_1_4	571.74	0.00	571.74	571.74	571.74	13.99	491.76	571.74
R_10_1_4	575.85	0.00	575.85	575.85	575.85	7.93	530.20	575.85
R_11_1_4	711.68	7.67	657.11	711.68	–	–	597.38	707.53
R_12_1_6	–	–	684.08	837.14	–	–	619.05	837.14

According to the obtained results, we can conclude that the more relaxed time windows (TWs) are, the lower operating costs can be achieved. For instance, in the investigated cases, on average 3.7% (i.e., for 50% wider TWs) and 6.2% (i.e., 100% wider TWs) operating cost savings can be obtained if the TWs are wider compared to the original case (i.e., $\rho = 1$). The case with 50% tighter TWs led to an average of 7.7% deterioration in the objective function value.

However, by comparing the operating costs of the PDPTW-SL optimal solutions to the corresponding PDPTW solutions in all above cases (i.e., $\rho \in$

{0.5, 1.5, 2}), savings due to the use of available SLs still remain significant (i.e., approximately 7%).

Table 2.14 The effect of the time windows on the operating costs

Instance	PDPTW-SL	PDPTW
C_6_1_4	214.46	238.13
C_7_1_4	231.69	246.81
RC_6_1_4	399.16	445.84
RC_7_1_4	427.15	448.00
R_6_1_4	326.64	326.64
R_7_1_4	377.66	377.66
Average	329.53	357.49

In addition, all instances with six to seven requests and one SL could be optimally solved without considering the TWs of the requests. The obtained results (optimal solution costs) are shown in Table 2.14. On average, 5.1% savings in terms of operating costs can be achieved by using available SL compared to the PDP environment, with a maximum of 10% (i.e., RC_6_1_4). Therefore, based on the obtained results, we can conclude that TWs play an important role in the potential savings obtained by the PDPTW-SL solutions compared to the classical PDPTW. However, no relationship could be observed between the TWs wideness and operating cost savings when comparing the PDPTW-SL and the PDPTW. Note that disregarding time windows for the PDPTW-SL can lead to an average drop of 29% in operating costs, compared to the original case (i.e., $\rho = 1$).

2.4.6 The effect of the SLs frequency on the operating costs

In this section we investigate the effect of the departure frequency of the available SLs on the operating costs. We consider three cases, namely the original frequency (once every 30 time units), a departure every 7 time units and, finally, every 180 time units. The obtained results are shown in Table 2.15. Note that numbers in bold indicate suboptimal values and "–" is used to show that no integer solution was found within a time limit of 24 hours.

According to the obtained results, we can conclude another intuitive insight: less frequent SL services lead to higher operating costs, with the PDPTW optimal solution cost as an upper bound.

Table 2.15 The effect of the SLs frequency on the operating costs

Instance	7 time units	30 time units	180 time units	PDPTW
C_6_1_4	369.36	369.36	441.42	461.70
C_7_1_4	390.31	390.31	452.82	464.49
C_8_1_4	**441.66**	446.35	483.50	483.50
C_9_1_4	**504.87**	**471.15**	498.19	498.19
C_10_1_4	**579.81**	**600.53**	510.01	515.25
C_11_1_4	–	–	**524.69**	524.69
C_12_1_4	–	–	–	598.55
RC_6_1_4	572.76	572.76	658.99	658.99
RC_7_1_4	575.95	575.95	662.18	662.18
RC_8_1_4	585.32	585.32	663.16	663.16
RC_9_1_4	593.70	593.70	703.30	703.30
RC_10_1_4	599.95	599.95	709.55	709.55
RC_11_1_4	624.48	624.48	742.52	742.52
RC_12_1_6	–	–	794.42	794.42
R_6_1_4	416.16	416.16	416.16	416.16
R_7_1_4	473.06	473.06	473.06	473.06
R_8_1_4	558.17	558.17	558.17	558.17
R_9_1_4	632.42	632.42	632.42	653.12
R_10_1_4	636.05	636.05	636.05	658.60
R_11_1_4	748.29	748.29	748.29	752.38
R_12_1_6	–	–	913.68	936.23
Average	555.42	555.42	607.00	612.67

2.5 Conclusions and future research directions

This chapter presented an extension to the PDPTW in the context of package transportation with the opportunity to use public scheduled transportation. A mixed-integer formulation for the PDPTW-SL was proposed. However, due to high computational complexity, the PDPTW-SL can be optimally solved only for small instances. By using the proposed arc-based model, two scenarios were analyzed in terms of operating costs.

The proposed transportation system leads to significant operating cost reductions compared to the standard PDPTW environment. Overall, this chapter has demonstrated that the operational costs can be reduced by up to 20% by making use of public transportation for carrying small packages.

The proposed model is yet essential for further research. In terms of solution methodologies, a potential research direction could be developing a specialized B&C algorithm for the current arc-based formulation by using the cuts existing in the literature along with the ones proposed in this chapter. In addition, meta-heuristic algorithms that generate good-quality solutions for reasonable-sized instances could be investigated. Moreover, as the considered transportation environment is subject to changes (e.g. traffic jams, new requests, etc.), algorithms for solving the dynamic version of the proposed problem could be studied as well. The model can be extended in terms of realistic aspects, such as time-dependent travel times, uncertainty in the input data and driver-related constraints.

Chapter 3

Branch-and-price for the pickup and delivery problem with time windows and scheduled lines

3.1 Introduction

In this chapter we present an exact solution approach to optimally solve small- to medium-size instances of the PDPTW-SL, which was formally introduced in Chapter 2. Unlike the arc-based formulation presented earlier, we re-formulate the PDPTW-SL as a path-based mixed-integer program (MIP). Since it is prohibitively time-consuming to generate all possible routes and consequently solve the formulation, we propose a branch-and-price (B&P) algorithm, which uses column generation method to construct promising routes. The column generator solves the *Elementary Shortest Path Problem with Resource and Precedence Constraints* (ESPPRPC), which is the natural pricing problem for the PDPTW-SL, in order to generate promising routes.

The scientific contributions of this chapter are threefold: *(i)* we re-formulate the PDPTW-SL as a path-based MIP, *(ii)* we develop a B&P algorithm to solve small to medium-size PDPTW-SL instances, and *(iii)* as a by-product, we show how the algorithm can be applied to the PDPTW-T.

The remainder of the chapter is structured as follows. Section 3.2 provides a brief review of related work. Section 3.3 introduces the MIP formulation for the PDPTW-SL, which is followed by the proposed solution methodology in Section 3.4. Section 3.5 reports the results obtained from extensive computational experiments. Conclusions are given in Section 3.6.

3.2 Literature review

In this section, we present recent studies on column generation algorithms applied to PDPs.

3.2.1 Column generation algorithms

For a detailed overview of column generation algorithms, the interested reader is referred to Lübbecke and Desrosiers (2005). The first branch-and-price algorithm for the PDPTW was proposed by Dumas et al. (1991). In their formulation, a set-partitioning master problem considers columns which represent feasible routes. The resulting pricing problem is a shortest path problem with resource and precedence constraints. Later, Savelsbergh and Sol (1995) proposed an improved B&P algorithm by using heuristics in the pricing problem, by reducing the master problem size (e.g., removing unused columns), and finally by applying more effective branching rules. Xu et al. (2003) and Sigurd et al. (2004) applied column generation to address variants of the PDPTW arising in long-haul transportation and in the transportation of live animals, respectively. Mues and Pickl (2005) proposed a column generation method for the PDP-T, but the implementation of this idea was not provided and no results were reported. In another study, Røpke and Cordeau (2009) proposed a branch-and-cut-and-price algorithm to solve the PDPTW and successfully tackled large instances with up to 500 requests. The authors investigated two pricing problems, namely the elementary shortest path problem with resource and precedence constraints (ESPPRPC) and the non-elementary variant of the problem (SPPRPC), respectively. They concluded that solving the SPPRPC as pricing problem for the PDPTW does not yield significant benefits. This is mainly due to the fact that SPPRPC is strictly NP-hard. In order to speed up the algorithm used to solve the elementary shortest path problem with resource constraints (ESPPRC), Righini and Salani (2006, 2008) proposed various techniques, including bi-directional search and decremental state space relaxation methods. Later, Baldacci et al. (2011a) proposed another exact approach to solve the PDPTW. The authors describe a bounding procedure by using heuristics combined with a cut-and-column generation procedure. The generated routes

have reduced cost between the upper bound and the lower bound achieved. The resulting problem is solved either by a solver, either using a branch-and-cut-and-price, depending on the problem size.

Even though there were several attempts to develop exact solution algorithms for the PDPTW with transfers, solving medium to large-size instances remains challenging. To the best of our knowledge, this chapter is the first attempt to apply column generation to the PDPTW-SL and PDPTW-T.

3.3 Description of the PDPTW-SL

In this section, we present the set partitioning formulation of the PDPTW-SL, along with the definitions used.

3.3.1 Definitions and assumptions

Let $\mathcal{P}$ and $\mathcal{D}$ denote the sets of pickup and delivery nodes, respectively. We consider a set of n requests, where each request is associated with a pickup node $r \in \mathcal{P}$ and a delivery node $r + n \in \mathcal{D}$. We refer to a specific request r by its pickup node, e.g., $r \in \mathcal{P}$. Each request r is associated with two desired time windows: one for the origin ($[l_r, u_r]$), and one for the destination ($[l_{r+n}, u_{r+n}]$). In addition, each request r has a demand d_r. Let $\mathcal{V}$ denote the set of vehicles. Each vehicle $v \in \mathcal{V}$ has a carrying capacity Q_v and an assigned depot $o_v \in \mathcal{O}$ with a time window ($[l_{o_v}, u_{o_v}]$).

The set of all physical transfer nodes is given by $\mathcal{S}$ while the set $\mathcal{E}$ contains all physical scheduled lines. Line $(i, j) \in \mathcal{E}$ is represented by a directed arc between the start and the end of the line and it has a set $\mathcal{K}^{ij}$ of indices w for the departure times p_{ij}^w from terminal i. We note that each SL may have a different frequency, thus the size of the sets $\mathcal{K}^{ij}$ may differ. Furthermore, it is assumed that SL vehicles are designed to carry a limited number of requests, thus implying a carrying capacity Q_{ij} for every $(i, j) \in \mathcal{E}$.

In order to model the waiting times of the PD vehicles and to allow multiple visits at transfer nodes, each physical SL $(i, j) \in \mathcal{E}$ is replicated n times as in Häll et al. (2009). The set of all replicated SLs is denoted by $\mathcal{F}$. In addition, in order to reduce the number of decision variables, each request r is assigned one replication of each SL $(i, j) \in \mathcal{E}$. The set of replicated SLs associated with request r is given by $\mathcal{F}^r$. Note that the replication process implies that each physical transfer node in $\mathcal{S}$ is replicated n times. The set of all replicated transfer nodes is denoted by $\mathcal{T}$.

To simplify the modeling, we also introduce the following additional notation. Let ψ^t, $\forall t \in \mathcal{T}$, be the physical transfer node represented by t (i.e., $\psi^t \in \mathcal{S}$). Set $\mathcal{T}^r$ represents the replicated transfer nodes associated with request r. In addition, $\mathcal{E}^t$ gives the set of physical scheduled lines that start at transfer node t and $\mathcal{E}_t$ represents the set of physical scheduled lines that end at transfer node t.

To formulate the PDPTW-SL as a set-partitioning problem, we let Ω denote the set of all feasible routes (paths) satisfying precedence, elementarity, time windows and PD-vehicle capacity constraints. Moreover, we let Ω^v be the set of feasible paths that can be executed by PD vehicle $v \in \mathcal{V}$. The routing cost of each path $p \in \Omega$ is given by c_p. In addition, constant z_p^i takes value 1 if node $i \in \mathcal{P} \cup \mathcal{D}$ is visited in path p, 0 otherwise. Constant a_p^{ij} takes value 1 if path p contains arc (i, j), 0 otherwise. Finally, m_p^{it} is a parameter taking value 1 if request node $i \in \mathcal{P} \cup \mathcal{D}$ and replicated transfer node $t \in \mathcal{T}$ are visited in path p by respecting precedence constraints, 0 otherwise.

The decision variables used to represent the routing and scheduling of the PD vehicles, along with the timing of the requests, are given as follows. Binary variable x_p takes value 1 if path $p \in \Omega$ is in the solution, 0 otherwise. If request r departs on the SL (i, j) at scheduled departure time p_{ij}^w, binary variable q_{ij}^{rw} takes value 1, otherwise 0. Continuous variables β_i, γ_i and α_v indicate the departure time of a vehicle from node i, the departure time of request r from transfer node i, and the time at which vehicle v returns to its depot, respectively.

Finally, the PDPTW-SL can be defined on a digraph $\mathcal{G} = (\mathcal{N}, \mathcal{A})$, where $\mathcal{N}$ is the set of nodes (i.e, $\mathcal{N} \equiv \mathcal{O} \cup \mathcal{P} \cup \mathcal{D} \cup \mathcal{T}$) and $\mathcal{A} \equiv \mathcal{A}_1 \cup \mathcal{A}_2 \cup \mathcal{A}_3 \cup \mathcal{A}_4$ is the set of arcs defined as follows:

$$\mathcal{A}_1 = ((\mathcal{P} \cup \mathcal{D})) \times (\mathcal{P} \cup \mathcal{D})) \backslash \{(r + n, r) \colon r \in \mathcal{P}\}$$
$$\mathcal{A}_2 = \{(i, j) \colon i \in \mathcal{O}, j \in \mathcal{P}\} \cup \{(i, j) \colon i \in \mathcal{D}, j \in \mathcal{O}\} \cup (\mathcal{O} \times \mathcal{T})$$
$$\mathcal{A}_3 = ((\mathcal{P} \cup \mathcal{D})) \times \mathcal{T}) \backslash (\{(j, r) \colon r \in \mathcal{P}, j \in \mathcal{T}^r\} \cup \{(r + n, j) \colon r \in \mathcal{P}, j \in \mathcal{T}^r\})$$
$$\mathcal{A}_4 = \{(i, j) \colon i, j \in \mathcal{T}, (\psi^i, \psi^j) \notin \mathcal{E}\}.$$

This definition of the set of arcs considers all possible connections except those that would be infeasible (e.g., direct paths from a depot to a delivery node). Figure 3.1 presents an example with one depot, two requests and one SL along with all corresponding arcs. Sets $\mathcal{A}_1$ and $\mathcal{A}_2$ are given in Figure 3.1.a whereas sets $\mathcal{A}_3$ and $\mathcal{A}_4$ are given in Figure 3.1.b. Finally, each arc $(i, j) \in \mathcal{A}$ has a deterministic travel time Υ_{ij} and service time at node $i \in \mathcal{N}$ is given by s_i. In addition, η_{ij} represents the cost per demand unit shipped on the SL (i, j).

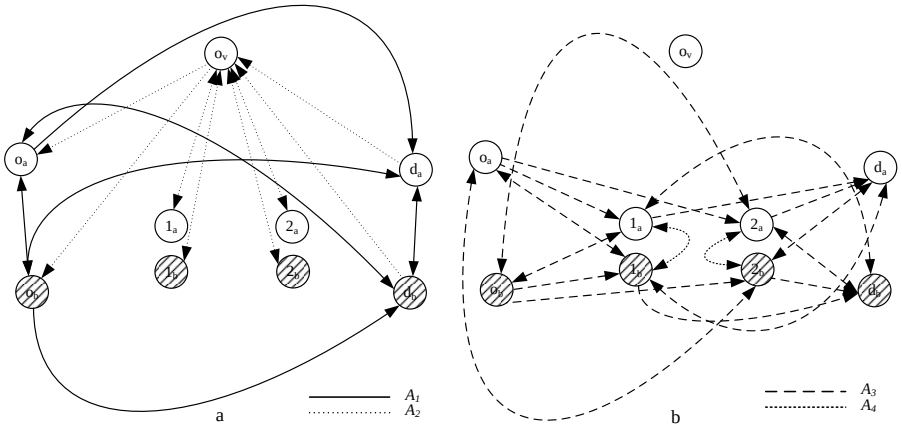

Figure 3.1 An example network with two requests and one SL

3.3.2 A set-partitioning formulation

The PDPTW-SL can be formulated as the following set-partitioning problem:

$$\text{Minimize} \quad \sum_{p\in\Omega} c_p x_p + \sum_{r\in\mathcal{P}} \sum_{(i,j)\in\mathcal{E}} \sum_{w\in\mathcal{K}^{ij}} \eta_{ij} d_r q_{ij}^{rw} \tag{3.1}$$

subject to

$$\sum_{p\in\Omega^v} x_p \leq 1 \qquad \forall v \in \mathcal{V} \tag{3.2}$$

$$\sum_{p\in\Omega} z_p^i x_p = 1 \qquad \forall i \in \mathcal{P}\cup\mathcal{D} \tag{3.3}$$

$$\sum_{p\in\Omega} m_p^{rt} x_p \leq \sum_{p\in\Omega} \sum_{t_1\in\mathcal{T}^r\setminus\{t\}} m_p^{r+n,t_1} x_p \qquad \forall r \in \mathcal{P}, t \in \mathcal{T}^r \tag{3.4}$$

$$\sum_{p\in\Omega} m_p^{rt} x_p + \sum_{(i,j)\in\mathcal{E}_{\psi^t}} \sum_{w\in\mathcal{K}^{ij}} q_{ij}^{rw} = \sum_{p\in\Omega} m_p^{r+n,t} x_p + \sum_{(i,j)\in\mathcal{E}^{\psi^t}} \sum_{w\in\mathcal{K}^{ij}} q_{ij}^{rw} \qquad \forall r \in \mathcal{P}, t \in \mathcal{T}^r \tag{3.5}$$

$$\sum_{r\in\mathcal{P}} d_r q_{ij}^{rw} \leq Q_{ij} \qquad \forall (i,j)\in\mathcal{E}, w\in\mathcal{K}^{ij} \quad (3.6)$$

$$\sum_{w\in\mathcal{K}^{\psi^i,\psi^j}} q_{\psi^i,\psi^j}^{rw} = 1 \implies \gamma_j^r \geq \gamma_i^r + \Upsilon_{ij} + s_j \qquad \forall r\in\mathcal{P}, (i,j)\in\mathcal{F}^r \quad (3.7)$$

$$\sum_{p\in\Omega} a_p^{ij} x_p = 1 \implies \beta_j \geq \beta_i + \Upsilon_{ij} + s_j \qquad \forall i\in\mathcal{N}, j\in\mathcal{P}\cup\mathcal{D}\cup\mathcal{T} \quad (3.8)$$

$$\sum_{p\in\Omega^v} a_p^{i,o_v} x_p = 1 \implies \alpha_v \geq \beta_i + \Upsilon_{i,o_v} + s_{o_v} \qquad \forall i\in\mathcal{P}\cup\mathcal{D}\cup\mathcal{T}, v\in\mathcal{V} \quad (3.9)$$

$$\beta_{r+n} \geq \beta_r + \Upsilon_{r,r+n} + s_{r+n} \qquad \forall r\in\mathcal{P} \quad (3.10)$$

$$l_i \leq \beta_i - s_i \leq u_i \qquad \forall i\in\mathcal{P}\cup\mathcal{D} \quad (3.11)$$

$$l_{o_v} \leq \alpha_v \leq u_{o_v} \qquad \forall v\in\mathcal{V} \quad (3.12)$$

$$q_{\psi^i,\psi^j}^{rw} = 1 \implies \gamma_i^r = p_{\psi^i,\psi^j}^w \qquad \forall r\in\mathcal{P}, (i,j)\in\mathcal{F}^r, w\in\mathcal{K}^{\psi^i,\psi^j} \quad (3.13)$$

$$\sum_{(i,j)\in\mathcal{E}_{\psi^t}} \sum_{w\in\mathcal{K}^{\psi^i,\psi^j}} q_{ij}^{rw} = 1 \implies \gamma_t^r = \beta_t \qquad \forall r\in\mathcal{P}, t\in\mathcal{T}^r \quad (3.14)$$

$$\beta_t \leq \gamma_t^r \qquad \forall r\in\mathcal{P}, t\in\mathcal{T}^r \quad (3.15)$$

$$x_p \in \{0,1\} \qquad \forall p\in\Omega \quad (3.16)$$

$$q_{ij}^{rw} \in \{0,1\} \qquad \forall r\in\mathcal{P}, (i,j)\in\mathcal{E}, w\in\mathcal{K}^{ij} \quad (3.17)$$

$$\beta_i \geq 0 \qquad \forall i\in\mathcal{N} \quad (3.18)$$

$$\gamma_i^r \geq 0 \qquad \forall r\in\mathcal{P}, i\in\mathcal{T} \quad (3.19)$$

$$\alpha_v \geq 0 \qquad \forall v\in\mathcal{V}. \quad (3.20)$$

The objective (3.1) is the sum of two cost functions. The first function is related to routing and the second one considers SL-related costs. Constraints (3.2) assure that each vehicle is used at most once. Constraints (3.3) ensure that every request node (pickup or delivery) is visited exactly once. Constraints (3.4) enforce the visit of transfer nodes if the pickup and delivery nodes of a request are visited in different paths. Constraints (3.5) are the flow balance constraints for each transfer node. The capacity of the SLs is considered in constraints (3.6). Constraints (3.7) set timing variables related to SLs. The scheduling of the PD vehicles is considered in constraints (3.8) and (3.9). Constraints (3.10) assure that the pickup node is visited earlier than the associated delivery node. Constraints (3.11) and (3.12) force time windows to be respected. The timetabling of the available SLs is considered in constraints (3.13). Constraints (3.14) assure that once a transfer node is a destination transfer node, the request departs at the same time as a PD vehicle. Constraints (3.15) enforce the departure time of a request from a transfer node to be at least the departure time of a PD vehicle from that node. Note that constraints (3.7)–(3.9) and (3.13)–(3.14) are written as implications and

standard linearization techniques can be used to express them as one or two linear inequalities.

3.4 Branch-and-price algorithm

In this section, we introduce the B&P algorithm used to solve the PDPTW-SL. We first describe the column generation method, which includes the pricing problem definition, the labeling algorithm used to solve it and some acceleration techniques. Then, we present the branching strategy to explore the enumeration tree, and we discuss potential improvements to the algorithm.

3.4.1 Column generation

Column generation is used to solve the linear programming (LP) relaxation in each node of the branch-and-bound tree. A lower bound (LB) on the optimal value of the PDPTW-SL can be obtained by solving the LP relaxation of (3.1)–(3.20), which is obtained by replacing the integrality constraints (3.16) and (3.17) with the following:

$$0 \leq x_p \leq 1 \qquad \forall p \in \Omega \tag{3.21}$$

$$0 \leq q_{ij}^{rw} \leq 1 \qquad \forall r \in \mathcal{P}, (i,j) \in \mathcal{E}, w \in \mathcal{K}^{ij}. \tag{3.22}$$

Since the size of Ω is usually very large, it is almost impossible to solve or even to explicitly represent model (3.1)–(3.20). Instead, a restricted master problem (RMP) is obtained by considering a subset of paths $\overline{\Omega} \subseteq \Omega$ in each iteration. Because $\overline{\Omega}$ might not yield a feasible solution, artificial variables with a large cost are added to the RMP to ensure that primal and dual solutions can be obtained. Once the RMP is solved, its dual solution is used to define the objective function of the pricing subproblems. The subproblem for each PD vehicle aims to generate negative reduced cost variables (columns) x_p with respect to this dual solution. If such variables are found, they are added to the RMP, which is then solved again to start a new iteration. Otherwise, when no subproblem can find any such column, the column generation process stops and the obtained solution to the current RMP is optimal for the master problem.

3.4.2 Pricing problem formulation

The pricing problem of the PDPTW-SL is a variant of the *Elementary Shortest Path Problem with Resource and Precedence Constraints* (ESPPRPC). The main

difference in the ESPPRPC for the PDPTW-SL and for the classical PDPTW is that in the former problem, each request can be served in two ways: direct (i.e., by one vehicle from its origin to its destination) or indirect (i.e., using a scheduled line as part of the journey, thus the vehicle needs to visit a transfer node). In particular, in each of the generated paths, the requests are served as follows: *(i)* classical way – first visit its pickup node $r \in \mathcal{P}$, and then deliver it to its delivery node $r + n \in \mathcal{D}$. In case *(ii)*, first visit its pickup node $r \in \mathcal{P}$, and consequently deliver it to a replicated transfer node e.g., $t_1 \in \mathcal{T}^r$. At this point, the labeling algorithm does not assure that this request is re-collected by another route. The synchronization takes place in the master problem. Finally, *(iii)* a request r is picked up from a transfer node e.g., $t_2 \in \mathcal{T}^r$, and subsequently is delivered to its delivery node $r+n \in \mathcal{D}$. Again, the pricing problem does not assure the synchornization between the routes, but only the precedence, time windows and capacity constraints. Since each physical transfer node is replicated for each request, it is possible that all requests can be transferred at/picked up from a transfer point by visiting the corresponding replicated transfer nodes sequentially. Figure 3.2 illustrates the possible delivery options in the pricing problem, where the flow from t_1 to t_2 is assured in the master problem only.

Note that the triangle inequality in terms of reduced cost does not hold for the pricing problem, meaning that $\overline{c}_{r,r+n}$ may be larger than $\overline{c}_{r,t_1} + \overline{c}_{t_2,r+n} + \eta_{t_1,t_2}$, where $\overline{c}_{ij}$ is the reduced cost of arc (i, j), $\forall r \in \mathcal{P}$, $t_1, t_2 \in \mathcal{T}^r$.

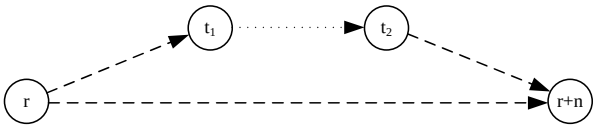

Figure 3.2 An illustration of possible shipment options

Since the vehicles may be heterogeneous and multiple depots are available, the ESPPRPC can be solved for each PD vehicle v. We note that the set of nodes considered in the pricing problem is $\mathcal{N} \equiv \mathcal{P} \cup \mathcal{D} \cup \mathcal{T} \cup \{o_v\} \cup \{o_v^{'}\}$, where $o_v^{'}$ is a replicated depot of vehicle v. The aim of the ESPPRPC is to generate an elementary shortest path from node o_v to $o_v^{'}$ by considering the available resources, such as time windows and capacity.

Let x_{ij} be a binary variable which takes value 1 if arc (i, j) is in the path, and 0 otherwise. In addition, let q_i be a continuous decision variable, which indicates the total load of vehicle v after visiting node $i \in \mathcal{N}$. The pricing problem can now be formulated for each PD vehicle $v \in \mathcal{V}$ as the following MIP:

$$\text{Minimize} \quad \sum_{(i,j)\in\mathcal{A}} \overline{c}_{ij} x_{ij} \tag{3.23}$$

subject to

$$\sum_{i\in\mathcal{N}} x_{o_v,i} = \sum_{i\in\mathcal{N}} x_{i,o'_v} = 1 \tag{3.24}$$

$$\sum_{j\in\mathcal{N}} x_{ij} = \sum_{j\in\mathcal{N}} x_{ji} \qquad \forall i \in \mathcal{P}\cup\mathcal{D}\cup\mathcal{T} \tag{3.25}$$

$$\sum_{i\in\mathcal{N}} x_{ir} \le \sum_{i\in\mathcal{N}} \sum_{t\in\mathcal{T}^r} x_{it} + \sum_{i\in\mathcal{N}} x_{i,r+n} \qquad \forall r \in \mathcal{P} \tag{3.26}$$

$$\sum_{i\in\mathcal{N}} x_{i,r+n} \le \sum_{i\in\mathcal{N}} \sum_{t\in\mathcal{T}^r} x_{it} + \sum_{i\in\mathcal{N}} x_{ir} \qquad \forall r \in \mathcal{P} \tag{3.27}$$

$$x_{ij} = 1 \implies \beta_j \ge \beta_i + \Upsilon_{ij} + s_j \qquad \forall i,j \in \mathcal{N} \tag{3.28}$$

$$\sum_{i\in\mathcal{N}\setminus\mathcal{T}^r} x_{ir} = \sum_{i\in\mathcal{N}} x_{it} = 1 \implies \beta_t \ge \beta_r + \Upsilon_{rt} + s_t \qquad \forall r \in \mathcal{P}, t \in \mathcal{T}^r \tag{3.29}$$

$$\sum_{i\in\mathcal{N}\setminus\{r\}} x_{i,r+n} = \sum_{i\in\mathcal{N}\setminus\{r\}} x_{it} = 1 \implies$$

$$\beta_{r+n} \ge \beta_t + \Upsilon_{t,r+n} + s_{r+n} \qquad \forall r \in \mathcal{P}, t \in \mathcal{T}^r \tag{3.30}$$

$$\beta_{r+n} \ge \beta_r + \Upsilon_{r,r+n} + s_{r+n} \qquad \forall r \in \mathcal{P} \tag{3.31}$$

$$l_i \le \beta_i - s_i \le u_i \qquad \forall i \in \mathcal{N} \tag{3.32}$$

$$\sum_{i\in\mathcal{N}\setminus\mathcal{T}^r} x_{ir} = x_{jt} = 1 \implies q_t^v = q_j^v - d_r \qquad \forall r \in \mathcal{P}, t \in \mathcal{T}^r, j \in \mathcal{N} \tag{3.33}$$

$$\sum_{i\in\mathcal{N}\setminus\{r\}} x_{i,r+n} = x_{jt} = 1 \implies q_t^v = q_j^v + d_r \qquad \forall r \in \mathcal{P}, t \in \mathcal{T}^r, j \in \mathcal{N} \tag{3.34}$$

$$x_{ir} = 1 \implies q_r^v = q_i^v + d_r \qquad \forall r \in \mathcal{P}, i \in \mathcal{N} \tag{3.35}$$

$$x_{i,r+n} = 1 \implies q_{r+n}^v = q_i^v - d_r \qquad \forall r \in \mathcal{P}, i \in \mathcal{N} \tag{3.36}$$

$$q_i^v \le Q_v \qquad \forall i \in \mathcal{P}\cup\mathcal{D}\cup\mathcal{T} \tag{3.37}$$

$$x_{ij} \in \{0,1\} \qquad \forall i,j \in \mathcal{N} \tag{3.38}$$

$$\beta_i \ge 0 \qquad \forall i \in \mathcal{N} \tag{3.39}$$

$$q_i \ge 0 \qquad \forall i \in \mathcal{N}. \tag{3.40}$$

Let A_v, B_i, C_t^r, D_t^r, E_{ij} and F_i^v be the dual variables associated with constraints (3.2)–(3.5), (3.8) and (3.9), respectively, represented in a linear form (using big-M technique). Moreover, let r^j be the request related to node $j \in \mathcal{P}\cup\mathcal{D}\cup\mathcal{T}$, and M_{ij}, M_i be large coefficients. The reduced cost of a path p can be calculated as follows:

$$\bar{c}_p^v = c_p - \sum_{i \in N} \sum_{j \in \mathcal{P} \cup \mathcal{D}} a_p^{ij}(B_j + M_{ij}E_{ij})$$
$$- \sum_{i \in N} \sum_{j \in \mathcal{T}} a_p^{ij} \left(M_{ij}E_{ij} + m_p^{r^j,j} C_j^{r_j} - \sum_{k \in \mathcal{T}^{r^j}: j \neq k} m_p^{r^j+n,k} C_k^{r^j} + m_p^{r^j,j} D_j^{r^j} - m_p^{r^j+n,j} D_j^{r^j} \right)$$
$$- \sum_{i \in \mathcal{P} \cup \mathcal{D} \cup \mathcal{T}} a_p^{i,o_v}(M_i F_i^v) - A_v, \quad \forall v \in \mathcal{V}, p \in \Omega^v. \tag{3.41}$$

The objective function (3.23) minimizes the reduced cost of a path. Constraints (3.24) assure that there is exactly one outgoing arc from o_v and one incoming arc to $o_v^{'}$. The flow balance for each node is given in constraints (3.25). Precedence constraints are imposed by (3.26) and (3.27). These constraints make sure that each request can be picked up either *(i)* from its pickup node, and consequently delivered to its delivery node or to one of the related transfer nodes, or *(ii)* from one of the related transfer nodes and transferred to its delivery node. Time windows are enforced in constraints (3.28)–(3.32). Capacity constraints are imposed by (3.33)–(3.37). To help clarity, some constraints are written as implications and can be linearized by using big-M techniques. In addition, expressions (3.23) and (3.41) can be made equivalent by adding additional decision variables in model (3.23)–(3.40) and linearizing the constraints written as implications.

Resource constrained shortest path problems used in column generation approaches are usually solved using dynamic programming methods called label-setting (or labeling) algorithms. In the following sections we explain how the ESPPRPC can be solved by using the proposed labeling algorithm.

3.4.3 Labeling algorithm

The objective of the ESPPRPC is to find a cheapest path from a source node s to a sink node t in a graph $\mathcal{G} = (\mathcal{N}, \mathcal{A})$. Every arc in this graph is associated with a cost and the cost of a path is the sum of the costs of all arcs traversed by the path. The path must satisfy the available resources, such as cost, capacity and time, used between the source node and sink node. An overview of resource constrained shortest path problems and of appropriate solution methodologies was given by Irnich and Desaulniers (2005).

Labeling algorithms build partial paths in graph $\mathcal{G}$. Each such path starts at s and ends at any node $i \in \mathcal{N}$. Existing partial paths are extended along the arcs leaving the end node of the partial path. In general, labeling algorithms generate all possible feasible paths in the graph. To speed up the labeling algorithm, *dominated* paths can be removed during the process. A path p dominates another path p' if both end at the same node $i \in \mathcal{N}$, resource consumption (including the

cost) in p is smaller than or equal to what it is in p', and all feasible extensions of p' by a partial path to the sink node are also feasible for p. Hence, *sufficient* conditions can be used to determine dominance criteria. The conditions used in this chapter are discussed in the corresponding sections below. Partial paths are represented by *labels* that contain information about resource consumption at the end node of the partial path and dominance is verified in terms of the labels.

In this section, we introduce a new labeling algorithm for the ESPPRPC that takes advantage of restricted dominance conditions, since the triangle inequality does not hold for the cost. We assume that the source and sink nodes are o_v and $o_v^{'}$, $\forall v \in \mathcal{V}$, respectively. To speed up the labeling algorithm, a bi-directional search is performed where labels are extended both forward from o_v to its successors and backward from $o_v^{'}$ to its predecessors. Both forward and backward labels are not allowed to cross a certain threshold (e.g., the middle of the planning horizon). At the end, forward and backward labels are merged to construct entire feasible paths. It has been observed that a bi-directional search may in practice lead to substantially shorter running times for the related ESPPRC as discussed by Righini and Salani (2006, 2008) and by Dabia et al. (2015).

The forward labeling algorithm. In the forward labeling algorithm, labels are extended from the source (i.e., o_v) to its successor nodes. For each forward label (L_f), we store the following data:

η	last node of the partial path;
t	departure time from the last node;
q	total load after visiting the last node;
c	accumulated cost;
$\mathcal{O}$	set of requests that have been started, but not completed (i.e., the request has been picked up either from the pickup node or from a transfer node, but not delivered to its delivery or transfer node);
$\mathcal{C}$	set of completed requests (i.e., $\mathcal{C} \cap \mathcal{O} \equiv \emptyset$);
$\mathcal{O}^T$	set of started requests from a transfer node, such that $\mathcal{O}^T \subseteq \mathcal{O}$.

We also store the reference of the parent label in each label. The resources are t, q, c, $\mathcal{O}$, and $\mathcal{C}$. Note that the resources are initialized as follows: $t = 0$, $q = 0$, $c = 0$, and finally, $\mathcal{O} \equiv \mathcal{O}^T \equiv \mathcal{C} \equiv \emptyset$. The notation t_{L_f} is used to refer to the departure time in forward label L_f and similar notation is used for the rest of the resources (e.g., η_{L_f}, q_{L_f}, etc.).

When extending a forward label L_f along an arc (η_{L_f}, j), the extension is feasible if:

$$t_{L_f} + t_{\eta_{L_f},j} \leq u_j, \tag{3.42}$$

$$q_{L_f} + d_j \leq Q_v, \tag{3.43}$$

where $t_{\eta_{L_f},j}$ is the traveling time between nodes η_{L_f} and j, and u_j is the upper bound of node j's time window. Moreover, d_j is the demand of node j, and Q_v is the capacity of PD vehicle v.

The capacity constraints are satisfied by expression (3.43). We note that time window constraints are satisfied in the master problem due to synchronization constraints between multiple routes. In the subproblem, time windows are used to eliminate infeasible label extensions and to reduce the solution space. Moreover, L_f and j must satisfy one of the following five conditions, which ensure that the extension is feasible with the sets $\mathcal{O}$, $\mathcal{O}^T$ and $\mathcal{C}$:

$$j \in \mathcal{P} \wedge j \notin \mathcal{O}_{L_f} \wedge j \notin \mathcal{C}_{L_f} \tag{3.44}$$

$$j \in \mathcal{D} \wedge j - n \in \mathcal{O}_{L_f} \tag{3.45}$$

$$j \in \mathcal{T} \wedge r^j \notin \mathcal{O}_{L_f} \wedge r^j \notin \mathcal{C}_{L_f} \tag{3.46}$$

$$j \in \mathcal{T} \wedge r^j \in \mathcal{O}_{L_f} \setminus \mathcal{O}^T_{L_f} \wedge r^j \notin \mathcal{C}_{L_f} \tag{3.47}$$

$$j = o_v^{'} \wedge \mathcal{O}_{L_f} \in \emptyset. \tag{3.48}$$

Expression (3.44) states that visiting a pickup node $j \in \mathcal{P}$ is feasible if request j is not started (has not been picked up earlier in the partial path from either a transfer node $t \in \mathcal{T}^j$, either the pickup node $j \in \mathcal{P}$), and if this request is not completed (has not been served) in this partial path. Expression (3.45) implies that extension to a delivery node j is feasible if the corresponding request is started (has been picked up, either from its origin node $j-n \in \mathcal{P}$, either from a replicated transfer node $t \in \mathcal{T}^{j-n}$). In addition, expression (3.46) implies that an extension to a replicated transfer node t is feasible if the request related to t is not started (has not been picked up in the partial path) and the request is not completed. In this case, the request related to the transfer node t will be picked up from t. Furthermore, expression (3.47) states that an extension to a transfer node t is feasible if the request related to t is started from its pickup node, and not from a transfer node, and if it is not completed. In this case, the request will be transferred to t. Finally, expression (3.48) assures that no partial path ends, before all started requests are served.

When label η_{L_f} and node j satisfy the conditions (3.42) and (3.43), along with one of the following conditions, the corresponding updates on sets $\mathcal{O}$, $\mathcal{O}^T$ and $\mathcal{C}$ are performed as follows:

- expression (3.44): $\mathcal{O}_{L_f} \leftarrow \mathcal{O}_{L_f} \cup \{j\}$.
- expression (3.45): $\mathcal{C}_{L_f} \leftarrow \mathcal{C}_{L_f} \cup \{j-n\}$, $\mathcal{O}_{L_f} \leftarrow \mathcal{O}_{L_f} \setminus \{j-n\}$, and $\mathcal{O}^T_{L_f} \leftarrow \mathcal{O}^T_{L_f} \setminus \{j-n\}$.
- expression (3.46): $\mathcal{O}_{L_f} \leftarrow \mathcal{O}_{L_f} \cup \{r^j\}$ and $\mathcal{O}^T_{L_f} \leftarrow \mathcal{O}^T_{L_f} \cup \{r^j\}$.

- expression (3.47): $\mathcal{C}_{L_f} \leftarrow \mathcal{C}_{L_f} \cup \{r^j\}$ and $\mathcal{O}_{L_f} \leftarrow \mathcal{O}_{L_f} \setminus \{r^j\}$.
- expression (3.48).

The following dominance criteria are valid for the PDPTW-SL.

Proposition 3.1 (Dominance 1) *A label L_f^2 is dominated by label L_f^1 if:*

1. $\eta_{L_f^1} = \eta_{L_f^2}$,
2. $t_{L_f^1} \leq t_{L_f^2}$,
3. $c_{L_f^1} \leq c_{L_f^2}$,
4. $\mathcal{O}_{L_f^1} \equiv \mathcal{O}_{L_f^2}$,
5. $\mathcal{C}_{L_f^1} \subseteq \mathcal{C}_{L_f^2}$.

The aforementioned conditions make sure that any feasible extension of L_f^2 is also feasible for L_f^1 and resource consumption in any extension of L_f^2 is larger than or equal to the corresponding consumption in L_f^1. The dominance criteria proposed here are inspired from those of Dumas et al. (1991) and Røpke and Cordeau (2009). We note that the triangle inequality in terms of reduced costs does not hold for the pricing problem of the PDPTW-SL, since there is the possibility of serving requests in two ways. In other words, a transfer node can either be an origin node for a request r (i.e., $r \in \mathcal{O}_{L_f}$ and $\{r\} \notin \mathcal{C}_{L_f}$) or can be a destination node for r (i.e., $\{r\} \in \mathcal{C}_{L_f}$). Therefore, the dominance criteria (1)–(5) are restrictive.

PROOF: Let L_f^1 and L_f^2 be two labels that satisfy the five conditions in Proposition 3.1. In addition, let $R_{L_f^1}$ and $R_{L_f^2}$ be the sets of all feasible extensions of labels L_f^1 and L_f^2, respectively. We show that *(i)* any feasible extension L of L_f^2 is also a feasible extension of L_f^1 (i.e., $R_{L_f^2} \subseteq R_{L_f^1}$) and *(ii)* for any feasible extension L of L_f^2 we have that $c_{L_f^1 \oplus L} \leq c_{L_f^2 \oplus L}$.

(i) Because of an elementarity assumption and conditions (2), (4) and (5) of the dominance test, the relation between the sets of all feasible extensions of L_f^1 and L_f^2 implies that $R_{L_f^2} \subseteq R_{L_f^1}$.

(ii) Because of condition (3) of the dominance test, the best extension of L_f^1 will never be worse than the best extension of L_f^2. Hence, L_f^1 dominates label L_f^2. Finally, we note that it is not necessary to consider the load q in the dominance test because of condition (4). □

The backward labeling algorithm. In the backward labeling algorithm, labels are extended from the sink (i.e., $o_v^{'}$) to its predecessors. It is quite similar to the

forward labeling algorithm. To a backward label L_b, we associate the following data:

η	last node of the path;
t	departure time from the node;
q	total load after visiting the last node;
c	accumulated cost;
$\mathcal{O}$	set of requests that have been finished, but not started (i.e., the request has been delivered either to the delivery node or to a transfer node, but not picked up from its pickup or transfer node);
$\mathcal{C}$	set of completed requests;
$\mathcal{O}^T$	set of finished requests at a transfer node, such that $\mathcal{O}^T \subseteq \mathcal{O}$.

Similarly to forward labeling, the resources are t, q, c, $\mathcal{O}$, $\mathcal{O}^T$, and $\mathcal{C}$. These are initialized as follows: $t = u_{o_v}$, $q = Q_v$, $c = 0$, and $\mathcal{O} \equiv \mathcal{O}^T \equiv \mathcal{C} \equiv \emptyset$.

When extending a backward label L_b along an arc $(j,\ \eta_{L_b})$, the extension is feasible only if:

$$t_{L_b} - s_{\eta_{L_b}} - t_{j,\eta_{L_b}} \geq l_j + s_j \tag{3.49}$$

$$q_{L_b} + d_j \geq 0. \tag{3.50}$$

In this case, we ensure the capacity constraints. Similarly to forward labeling, time windows are used to eliminate infeasible extensions. Moreover, L_b and j must satisfy one of the following five conditions, which ensure that the extension is compatible with the sets $\mathcal{O}$ and $\mathcal{C}$:

$$j \in \mathcal{D} \wedge j - n \notin \mathcal{O}_{L_b} \wedge j - n \notin \mathcal{C}_{L_b} \tag{3.51}$$

$$j \in \mathcal{P} \wedge j \in \mathcal{O}_{L_b} \tag{3.52}$$

$$j \in \mathcal{T} \wedge r^j \notin \mathcal{O}_{L_b} \wedge r^j \notin \mathcal{C}_{L_b} \tag{3.53}$$

$$j \in \mathcal{T} \wedge r^j \in \mathcal{O}_{L_b} \setminus \mathcal{O}^T_{L_b} \wedge r^j \notin \mathcal{C}_{L_b} \tag{3.54}$$

$$j = o_v \wedge \mathcal{O}_{L_b} \in \emptyset. \tag{3.55}$$

Condition (3.51) states that visiting a delivery node $j \in \mathcal{D}$ is feasible if the corresponding request $j - n$ has not been delivered at the delivery node, nor at a transfer node earlier in the partial path, and this specific request is not completed. Expression (3.52) implies that extending the label to a pickup node $j \in \mathcal{P}$ is feasible if j has already been delivered to its delivery node, or a transfer node in the partial path. Condition (3.53) states that visiting a transfer node $j \in \mathcal{T}$ is feasible if the corresponding request $r^j \in \mathcal{P}$ has not been delivered to its delivery node, nor a transfer node in the partial path, and if this request has not been completed. Expression (3.54) states that extending to a transfer node $j \in \mathcal{T}$ is feasible if the corresponding request $r^j \in \mathcal{P}$ has been delivered to its delivery node, and not to

a transfer node, and if this request has not been completed in the partial path. Finally, Condition (3.55) implies that a partial path cannot be ended at the origin depot (i.e., o_v), if at least one request is delivered, but not picked up (from the origin, or from a transfer node) in the partial path.

When label η_{L_b} and node j satisfy the conditions (3.49) and (3.50), along with one of the following conditions, the corresponding updates on sets $\mathcal{O}$, $\mathcal{O}^T$ and $\mathcal{C}$ are performed as follows:

- expression (3.51): $\mathcal{O}_{L_b} \leftarrow \mathcal{O}_{L_b} \cup \{j - n\}$.
- expression (3.52): $\mathcal{C}_{L_b} \leftarrow \mathcal{C}_{L_b} \cup \{j\}$, $\mathcal{O}_{L_b} \leftarrow \mathcal{O}_{L_b} \setminus \{j\}$, and $\mathcal{O}^T_{L_b} \leftarrow \mathcal{O}^T_{L_b} \setminus \{j\}$.
- expression (3.53): $\mathcal{O}_{L_b} \leftarrow \mathcal{O}_{L_b} \cup \{r^j\}$ and $\mathcal{O}^T_{L_b} \leftarrow \mathcal{O}^T_{L_b} \cup \{r^j\}$.
- expression (3.54): $\mathcal{C}_{L_b} \leftarrow \mathcal{C}_{L_b} \cup \{r^j\}$ and $\mathcal{O}_{L_b} \leftarrow \mathcal{O}_{L_b} \setminus \{r^j\}$.
- expression (3.55).

The dominance condition used in the backward label setting algorithm is the following: a label L^1_b dominates a label L^2_b if conditions (3.1) and (3.1)–(3.1) are satisfied, along with the following condition:

$$t_{L^1_b} \geq t_{L^2_b}. \tag{3.56}$$

The proof is similar to that given for the forward dominance test.

Merging forward and backward labels. When all forward and backward labels are generated, they are merged to construct feasible paths with negative reduced cost. A forward label L_f and a backward label L_b can be merged if the following conditions are satisfied:

$$\mathcal{O}_{L_f} \equiv \mathcal{O}_{L_b} \tag{3.57}$$

$$\mathcal{O}^T_{L_f} \cap \mathcal{O}^T_{L_b} \equiv \emptyset \tag{3.58}$$

$$\mathcal{C}_{L_f} \cap \mathcal{C}_{L_b} \equiv \emptyset \tag{3.59}$$

$$t_{L_f} + t_{\eta_{L_f},\eta_{L_b}} + s_{\eta_{L_b}} \leq t_{L_b}. \tag{3.60}$$

Condition (3.57) assures that the started requests (but not completed) in the forward label are the same as the finished requests (but not started) in the backward label. Expression (3.58) makes sure that a request cannot be picked up at a transfer node and delivered to another transfer node within the same route. Condition (3.59) makes sure that no request is served twice (once in each of the forward and backward labels) in the considered route. Finally, Condition (3.60) assures that merging the two labels leads to feasible scheduling.

Since we use the dominance test, not all feasible paths may be generated. However, a path that minimizes the reduced cost will necessarily be created. Labels are discarded only if another label exists that can be extended in the same ways as the discarded label. Hence, the extension of the dominating label will always lead to a path with lower or equal reduced cost compared to the same extension of the discarded label.

3.4.4 Acceleration techniques

To speed up the column generation, in addition to the bi-directional search in the labeling algorithm, we apply the following ideas:

Pricing problem heuristics. The performance of the B&P algorithm can be accelerated by using heuristics to solve the pricing problem. Heuristics search for easy-to-find negative-reduced cost paths and add them to the master problem. When heuristics fail to find any such path, the exact algorithm is used. Ideally, for every node in the branching tree, the exact algorithm should be called only once to prove optimality, i.e., that no more paths with negative reduced cost exist. In our branch-and-price algorithm, we employ two heuristic algorithms.

First, *H1* is a truncated labeling algorithm as in Dabia et al. (2015), in which only a limited number of labels (with the best cost) for each node are stored and considered for possible extension. Here, we store only 50 labels at each node.

Second, *H2* uses the proposed labeling algorithm with stronger dominance criteria. More specifically, for both the forward and backward labeling algorithms, we use the same dominance criteria as before except condition $\mathcal{C}_{L_f^1} \subseteq \mathcal{C}_{L_f^2}$.

Pre-processing and label elimination. We provide some important problem-specific pre-processing techniques that improve the performance of the algorithm. These are valid under the assumption that triangle inequality holds in terms of travel time, namely, $\Upsilon_{r,r+n} \leq \Upsilon_{r,k} + \Upsilon_{k,r+n}$.

Let $r \in \mathcal{P}$ and $t \in \mathcal{T}^r$ be a destination transfer node of r, then E_t^d is the earliest arrival time at t on a scheduled line as a destination transfer node. It is computed as follows:

$$E_t^d = \text{argmin}_{k \in \mathcal{O}, (i,j) \in \mathcal{F}^t} \{\lceil (\text{argmax}\{\Upsilon_{k,r} + s_r; l_r + s_r\} + \Upsilon_{r,i} + s_i) \rceil + \Upsilon_{i,j} + s_j\}, \tag{3.61}$$

where $\mathcal{F}^t$ is the set of replicated scheduled lines that have the destination node t.

In addition, let L_t^o be the latest feasible departure time of request r on a scheduled line from transfer node t, $\forall r \in \mathcal{P}$, $t \in \mathcal{T}^r$. It is computed as follows:

$$L_t^o = \text{argmin}_{(i,j) \in \mathcal{F}_t} \{\lfloor (u_{r+n} + s_{r+n} - \Upsilon_{j,r+n} - s_j - \Upsilon_{i,j}) \rfloor\}, \tag{3.62}$$

where $\mathcal{F}_t$ is the replicated scheduled line set that contains SLs which have the origin in node t.

Note that E_t^d and L_t^o are both used in the labeling algorithm to restrict the solution space. In other words, if e.g., arrival time at a *destination* transfer node is smaller than E_t^d, then the departure time is set to E_t^d. Similarly, if the arrival time at an *origin* transfer node is larger than L_t^o, than this extension is considered as infeasible.

Moreover, for each request r and each replicated transfer node $t \in \mathcal{T}^r$, the time window of node t is updated as follows:

$$l_t = \text{argmin}_{k \in \mathcal{O}}\{(\text{argmax}\{\Upsilon_{k,r} + s_r; l_r + s_r\} + \Upsilon_{r,t})\}, \tag{3.63}$$

$$u_t = \underset{k \in \mathcal{O}}{\text{argmax}}\{(\text{argmin}\{u_k - \Upsilon_{r+n,k} - s_{r+n}; u_{r+n} + s_{r+n}\} - \Upsilon_{t,r+n} - s_t)\}. \tag{3.64}$$

We note that if $L_t^o < l_t$ and $E_t^d > u_t$ or $u_t < l_t$, there is no feasible solution by visiting node t. Therefore, any label extension to node t can be eliminated. In addition, any arc related to t is also eliminated from the graph.

Similarly to Dumas et al. (1991), we investigate all requests $r \in \mathcal{O}_{L_f}$. If there is at least one request r that cannot be delivered from η_{L_f} to its delivery node $r+n$ by satisfying its time window, then the forward label is eliminated. Similarly, if there is at least one request $r \in \mathcal{O}_{L_b}$ that cannot be picked up before η_{L_b} within its time window, then the backward label is removed.

In addition, for each replicated transfer node, label extension is reduced by breaking the symmetry for replicated transfer nodes. For example, consider a label that represents a replicated transfer node t. Let the other replicated transfer node t_1 represent the same physical transfer node as t (i.e., $\psi^t = \psi^{t_1}$) and let t_1 have the same functionality as t (origin or destination transfer node). Then, the extension from t to t_1 is valid only if $t_1 > t$. This is done to avoid symmetric sequences of nodes visited within the paths (e.g., [..., t, t_1, ...] and [..., t_1, t, ...], $\forall t, t_1 \in \mathcal{T}$, such that $\psi^t = \psi^{t_1}$ and t, t_1 have the same functionality, i.e., origin or destination transfer nodes).

Infeasible arcs can be eliminated from the graph by considering time windows. A simple analysis leads to the following observations:

- arcs $(o_v, i+n)$, (i, o_v) and $(i+n, i)$ are infeasible for $i \in \mathcal{P}$, $v \in \mathcal{V}$;
- arc (i, j) with $i, j \in \mathcal{P} \cup \mathcal{D} \cup \mathcal{T}$ is infeasible if $l_i + s_i + \Upsilon_{ij} > u_j$.

More relaxed versions of the elimination rules proposed by Dumas et al. (1991) for PDPTW are also valid for the PDPTW-SL as follows:

- arcs (i, j) and $(j, i + n)$ are infeasible if $i \in \mathcal{P}$, $j \in \mathcal{P} \cup \mathcal{D} \cup \mathcal{T} \setminus \{i + n\}$ and path $\{i, j, i + n\}$ is infeasible.

3.4.5 Branching

The branch and bound tree is explored using the best bound strategy. The algorithm first branches on arcs (i, j). It imposes two branches: $\sum_{p\in\Omega} a_p^{ij} x_p \geq 1$ and $\sum_{p\in\Omega} a_p^{ij} x_p \leq 0$. Strong branching is used, i.e., the impact of branching on several candidates is investigated every time a branching decision has to be made. For each candidate, we estimate the lower bound in the two child nodes by solving the associated LP relaxation using a quick pricing heuristic, namely *H1*. The branch that maximizes the lower bound in the weakest of the two child nodes is chosen. We consider 10 branching candidates at every node. When all arcs have integer values, it can happen that some x variables still have fractional values, as the same route can be traversed by multiple vehicles (e.g., path p can be assigned to vehicles v_1 and v_2, respectively, both with value 0.5). In this case, the algorithm looks for an arc (i, j) traversed by a vehicle v that has a fractional value and imposes two branches: $\sum_{p\in\Omega^v} a_p^{ij} x_p \geq 1$ and $\sum_{p\in\Omega^v} a_p^{ij} x_p \leq 0$. If x variables have integer values, then the MIP formulation (3.1)–(3.20) is solved for the considered routes (x which are integer) only. Let $\bar{\Omega}_x$ be the restricted set of routes, which may not necessarily be feasible in terms of time windows (by considering time synchronization between routes) and SL-capacity constraints. If the considered MIP model finds a feasible solution, then the solution is accepted. Otherwise, the following valid inequality is added to the formulation to disregard this solution:

$$\sum_{p\in\bar{\Omega}_x} x_p \leq |\bar{\Omega}_x| - 1. \tag{3.65}$$

The inequality is similar to a Benders combinatorial cut, where a solution that is not feasible is discarded from the search tree. Thus, at most $|\bar{\Omega}_x| - 1$ of the considered paths can be a part of the solution. In the path-based formulations, adding such inequalities to the model will make the considered paths regenerate in the following column generation iteration. Therefore, expression (3.65) should be reformulated in terms of arcs, consequently pricing out the considered arcs within the paths in the subproblem. Hence, the expression (3.65) can be reformulated to

$$\sum_{p\in\bar{\Omega}_x} \nu_p x_p \leq \sum_{p\in\bar{\Omega}_x} \nu_p - \mu, \tag{3.66}$$

where ν_p represents the number of arcs in path p and μ is the number of arcs within the path $p \in \bar{\Omega}_x$ that visits the fewest nodes.

Proposition 3.2 *Inequality (3.66) is equivalent to (3.65) in terms of integer feasible solutions.*

PROOF: Let x be an infeasible routing solution in terms of time windows or SL capacity. These routes are infeasible due to the inter-route synchronization constraints, time or load. In addition, let $\bar{\Omega}_x$ be the set of the infeasible combination of the considered routes (paths), where each PD vehicle is used at most once. Furthermore, let ν_p be the number of arcs within path $p \in \bar{\Omega}_x$ and $\mu = \text{argmin}_{p \in \bar{\Omega}_x}\{\nu_p\}$. Expression (3.65) assures that at least one path $p \in \bar{\Omega}_x$ will take value zero. This means that at least μ arcs from the considered paths $p \in \bar{\Omega}_x$ will take value zero, since the path that visits the fewest nodes involves μ arcs. Thus, (3.66) is equivalent to (3.65) in terms of integer feasible solutions. □

The dual information of inequalities (3.66) is considered in the pricing problem for each considered arc in $\bar{\Omega}_x$. Generally, this procedure is performed to avoid branching on q variables, which is time-consuming and leads to a slow convergence of the algorithm.

We have tested different branching schemes, as follows. *(i)* branch on q binary variables (SL scheduling), then *(ii)* branch on arcs and on arcs per vehicle. In addition, we have tested slightly similar approach: *(i)* branch on $\sum_{(i,j)\in\mathcal{E}} \sum_{w\in\mathcal{K}^{ij}} q_{ij}^{rw}$ for request r with the most fractional value of this expression. The considered request is either forced to use SLs, or is forbidden. When all values of $\sum_{(i,j)\in\mathcal{E}} \sum_{w\in\mathcal{K}^{ij}} q_{ij}^{rw}$ are 1 or 0, then we branched on arcs, and finally, use a MIP to determine the feasibility of the routes (when all have integer values). These branching schemes provided worse results than the one described above, as the lower bound converged slower (usually $\sum_{(i,j)\in\mathcal{E}} \sum_{w\in\mathcal{K}^{ij}} q_{ij}^{rw}$ has either value 1 or 0 in a linear relaxation solution, without intermediate values).

3.4.6 Potential improvements

We note that two stabilization techniques have been tested in our implementation. These include the smoothing method proposed by Pessoa et al. (2013) and the box method introduced by Marsten et al. (1975). The use of these techniques did not provide significant benefits to the overall performance of the algorithm, therefore no stabilization is used in the computational experiments presented in the following section.

Recently, Desaulniers et al. (2008) and Baldacci et al. (2011b) suggested to relax the elementarity conditions of the pricing problem to speed up the labeling algorithm for some type of VRPs. The elementarity constraints are then imposed in the master problem. Allowing cycles in the pricing problem may lead to an easier problem to solve, for which pseudo-polynomial algorithms are known (see for example Desrochers et al. (1992)). We have implemented and tested the *ng*-path relaxation of Baldacci et al. (2011b) but it did not show promising results. According to Røpke and Pisinger (2007), the shortest path problem with pairing and precedence constraints is still a strictly NP-hard problem.

We tested the effect of a warm start on the performance of the B&P, given an initial solution (i.e., constructed by using a greedy heuristic). Basically, two schemes were tried: *(i)* starting just with the routes of the initial solution (and the corresponding upper bound) and *(ii)* setting an initial column pool with multiple feasible routes (e.g., 1,000 routes). The first scheme proved insignificant effect on the solution time, as a solution (several routes) does not provide significant dual information in the first iterations of the column generation. In addition, the given upper bound from the initial solution did not reduce the number of explored B&B nodes, as we apply best first strategy. The second scheme showed a positive effect on computational time of the B&P, as the initial column pool provides dual information that reduces the number of iterations of the column generation in each B&B node. The overall CPU time slightly decreased. The proposed B&P may benefit from a good heuristic, which would provide a good initial pool of vehicle routes. In the computational experiments, initial column pool (in the root node) was empty.

3.5 Computational experiments

We have implemented the branch-and-price algorithm in Java. Since the subproblems defined on each PD vehicle are independent from each other, they are solved in parallel using four threads. All experiments were conducted on a 2.6 GHz Intel Core i5 processor with 4 GB of RAM. The LP relaxations of the restricted master problems were solved with CPLEX 12.3. We note that a limit of three hours has been imposed on the solution time. For the experiments we used generated datasets, which can be found on `www.smartlogisticslab.nl`. Some of the considered datasets have been used in Chapter 2 and additional ones (e.g. with 15 and 40 requests) were generated in the same way as in the second chapter of this thesis.

The values of the cost parameters used in this section equal to the ones used in Chapter 2 (i.e., driving cost of the PD vehicles is 0.5€ per time unit and the

cost of SL service per demand unit is equal to 1€).

3.5.1 Bi-directional vs. mono-directional labeling algorithm

In this section, we study the benefits of using a bi-directional search over the mono-directional search. The detailed results can be found in Table 3.4 of the Appendix. The bi-directional labeling algorithm performs better than the mono-directional version: the lower bounds obtained within the imposed time limit by the bi-directional labeling algorithm are better over all instances. Unlike the mono-directional algorithm, the bi-directional search found lower bounds for all instances within time limit. This is mainly due to the fact that the number of labels that need to be processed in the bi-directional labeling algorithm is considerably smaller compared with the mono-directional version. Lastly, bi-directional search proved to be twice faster than mono-directional search, based on instances that were optimally solved by both algorithms. Overall, the best implementation leads to strong lower bounds in the root node, on average being 1.29% away from the optimal solution.

3.5.2 The impact of acceleration techniques

In this section, we investigate the effect of the preprocessing, label elimination techniques and the proposed valid inequality on the overall performance of the algorithm. More specifically, we define three algorithmic setups, as follows: *S1* – without preprocessing and label elimination, but with the cut proposed in Section 3.4.4, *S2* – with the preprocessing and label elimination, but without the valid inequality, and finally *All* – with all acceleration techniques described in this chapter. Table 3.1 presents the obtained results within a three-hour time limit. Columns "$\underline{z}$" and "$\overline{z}$" indicate the best lower and upper bounds found within a three-hour time limit. "Nodes" indicate the number of branch-and-bound nodes explored during the search and, finally, "Time" indicates the computational times in seconds.

The preprocessing and label elimination significantly improve the overall performance of the algorithm. The former tightens the formulation (i.e., better root lower bounds by on average 9.56%), and thus leads to fewer nodes to be processed in the search tree. In addition, it reduces the solution space of the problem, hence decreases the computational effort needed. The latter (i.e., label elimination) has a positive impact on the CPU time to process each B&B node. As can be seen in Table 3.1, *S1* could solve only three out of 18 instances, compared to *S2* and *All* setups, with 13 and 14, respectively. Finally, the cut proposed in Section 3.4.4 helped close the gap for one instance (i.e., C7_3_4). In addition, the

Table 3.1 The impact of acceleration techniques.

Instance	S1				S2				All			
	$\underline{z}$	$\overline{z}$	Nodes	Time (s)	$\underline{z}$	$\overline{z}$	Nodes	Time (s)	$\underline{z}$	$\overline{z}$	Nodes	Time (s)
C6_3_4	282.51	–	3,111	10,800	310.85	310.85	7,807	5,183	310.85	310.85	1,741	2,632
C7_3_4	263.92	–	1,817	10,800	313.37	–	7,577	10,800	313.41	313.41	2,301	5,795
C8_3_4	279.70	–	763	10,800	336.33	–	5,559	10,800	337.12	–	1,821	10,800
C9_3_4	300.02	–	17	10,800	355.56	–	2,057	10,800	356.36	–	1,793	10,800
C10_3_4	–	–	–	10,800	364.82	–	1,317	10,800	364.82	–	1,317	10,800
C11_3_4	–	–	–	10,800	380.08	–	947	10,800	380.08	–	947	10,800
RC6_3_4	520.11	520.11	1	2	520.11	520.11	1	1	520.11	520.11	1	1
RC7_3_4	482.55	–	2,813	10,800	539.08	539.08	3	1	539.08	539.08	3	1
RC8_3_4	482.94	–	1,443	10,800	548.45	548.45	3	1	548.45	548.45	3	1
RC9_3_4	488.64	–	259	10,800	552.71	552.71	3	1	552.71	552.71	3	1
RC10_3_4	558.96	558.96	3	1,181	558.96	558.96	3	1	558.96	558.96	3	1
RC11_3_4	585.72	–	1	10,800	585.72	585.72	7	11	585.72	585.72	7	11
R6_3_4	416.16	416.16	125	125	416.16	416.16	1	0	416.16	416.16	1	0
R7_3_4	452.74	–	2,164	10,800	470.15	470.15	21	12	470.15	470.15	17	8
R8_3_4	465.92	–	623	10,800	490.77	490.77	3	1	490.77	490.77	3	1
R9_3_4	510.59	–	67	10,800	579.38	579.38	7	5	579.38	579.38	7	5
R10_3_4	511.93	–	11	10,800	584.89	584.89	7	4	584.89	584.89	7	4
R11_3_4	–	–	–	10,800	687.36	687.36	3	2	687.36	687.36	3	2

cut decreased the number of explored nodes in the B&B tree and improved the best lower bound found within time limit.

3.5.3 PDPTW-SL vs. PDPTW

This section discusses the computational results obtained when solving the PDPTW-SL (with one SL) and PDPTW instances using the proposed algorithm. A similar comparison was made in Chapter 2 but the results were optimal only for small instances (up to 11 requests). In this section, we confirm that the previously found conclusions still hold for the optimal solution of larger instances. The detailed results are shown in Tables 3.5 and 3.6 of the Appendix.

According to the results, PDPTW-SL solutions proved to be cheaper compared to the PDPTW solutions. However, the cost savings are dependent on the relative positioning of the request nodes and SLs, time windows and available capacities. More specifically, C instances provided the most savings, and R instances the least. Overall, the results support the findings given in Chapter 2: the use of SLs benefits the pickup and delivery network. In order to maximize the benefits of such systems, strategic decisions, such as configuration of the SLs, storage areas at transfer nodes and re-design of the SL vehicles, must be properly considered.

Regarding the ride times of the requests, no specific relationship has been found between the use of SLs and the standard pickup and delivery operations. More specifically, in some instances, the average ride time in the optimal PDPTW-SL solution is higher than in the optimal PDPTW solution. It may be explained by the fact that using SLs, considering corresponding timetables, may lead to extra waiting and distance to be traveled for the requests that are transferred. In other

cases, the average ride time in the optimal PDPTW-SL solution is lower than in the corresponding PDPTW solution. It makes sense, since the vehicle routes without SLs are longer, hence longer distances are traveled to perform the services. Overall instances, the average request ride time is 223.2 and 221.1 time units, respectively, for the PDPTW-SL and PDPTW solutions.

3.5.4 PDPTW-SL with multiple scheduled lines

This section illustrates the effect of having multiple available SLs on the operational costs of the system. Table 3.7 of the Appendix indicates the results obtained when solving the instances with one, two (i.e., two connected lines) and three (i.e., triangular network) available SLs within a three-hour time limit. The routing costs are assumed to be homogeneous. As expected, the complexity drastically increases with the number of available SLs as the increase in graph size makes the pricing problem more difficult to solve. As a side effect, more SLs lead to significantly slower convergence during the search. The results support the observations made in Chapter 2 for similar settings: more SLs lead to more cost savings compared to PDPTW solutions (up to 32% for C6_3_4).

3.5.5 Comparison to the arc-based MIP

In this section, we present a computational comparison of the proposed (path-based) MIP formulation using the B&P algorithm to the arc-based MIP presented in Chapter 2 solved using CPLEX. Note that the arc-based MIP is solved with the valid inequalities proposed along with the preprocessing introduced in Section 3.4.4 of this chapter. Table 3.2 presents the results obtained. The columns are self-explanatory.

The results indicate that the MIP re-formulation in terms of vehicle routes provides stronger lower bounds. On the other hand, the arc-based MIP proved relatively weak lower bounds, i.e., on average 7.58% from the optimal solution. The proposed B&P algorithm solved 34 out of 37 instances, and CPLEX (using arc-based formulation) could solve only 22 of them. In addition, B&P solved significantly faster the instances that were solved by CPLEX as well.

3.5.6 PDPTW with transfers

As a by-product, we can simplify the proposed set partitioning master problem and solve the PDPTW-T. In this context, transfers among PD vehicles can be made only at pre-defined locations. For the formal description of the problem the

Table 3.2 Comparison between models.

Instance	B&P			arc-based MIP		
	$\underline{z}$	$\overline{z}$	Time (s)	$\underline{z}$	$\overline{z}$	Time (s)
C6_1_4	369.36	369.36	0	369.36	369.36	68
C7_1_4	390.31	390.31	0	390.31	390.31	160
C8_1_4	446.35	446.35	4	446.35	446.35	365
C9_1_4	471.16	471.16	140	471.16	471.16	6,351
C10_1_4	510.01	510.01	67	501.25	536.50	10,800
C11_1_4	522.85	522.85	9	453.44	528.08	10,800
C12_1_4	541.60	541.60	4	470.04	–	10,800
C15_1_6	649.93	649.93	13	546.29	–	10,800
C20_1_8	751.23	751.23	83	544.95	–	10,800
C25_1_8	879.94	879.94	2,114	–	–	10,800
C40_1_8	1,199.31	–	10,800	–	–	10,800
RC6_1_4	572.76	572.76	0	572.76	572.76	3
RC7_1_4	575.95	575.95	0	575.95	575.95	3
RC8_1_4	585.32	585.32	0	585.32	585.32	5
RC9_1_4	593.70	593.70	0	593.70	593.70	5
RC10_1_4	599.95	599.95	0	599.95	599.95	8
RC11_1_4	624.48	624.48	1	624.48	624.48	8
RC12_1_6	662.03	662.03	1	662.03	662.03	11
RC15_1_6	1,068.16	1,068.16	1	1,068.16	1,068.16	65
RC20_1_8	1,200.36	1,200.36	6	1,062.37	–	10,800
RC25_1_8	1,530.43	1,530.43	28	1,085.66	–	10,800
RC40_1_12	1,969.45	1,969.45	738	–	–	10,800
RC50_1_14	2,200.88	2,200.88	1,235	–	–	10,800
RC60_1_16	2,636.18	–	10,800	–	–	10,800
R6_1_4	416.16	416.16	0	416.16	416.16	4
R7_1_4	473.06	473.06	1	473.06	473.06	4
R8_1_4	558.17	558.17	0	558.17	558.17	4
R9_1_4	632.42	632.42	2	632.42	632.42	39
R10_1_4	636.05	636.05	1	636.05	636.05	24
R11_1_4	748.29	748.29	1	748.29	748.29	39
R12_1_6	913.68	913.68	2	913.68	913.68	182
R15_1_6	1,083.50	1,083.50	15	1,083.50	1,083.50	317
R20_1_8	1,542.93	1,542.93	12	1,280.74	–	10,800
R25_1_8	1,636.12	1,636.12	50	1,326.77	–	10,800
R40_1_12	1,969.77	1,969.77	895	–	–	10,800
R50_1_14	2,595.14	2,595.14	580	–	–	10,800
R60_1_16	2,858.69	–	10,800	–	–	10,800
Average			177			4,673

reader is referred to Cortes et al. (2010). From the modeling perspective, each transfer node can be considered as a SL with zero traveling time (i.e., the same physical location) and with no imposed schedule and capacity constraints. In other words, PDPTW-T is a special case of PDPTW-SL, where multiple individual SLs of line network topology can exist. However, certain service times at these nodes exist (i.e., loading/unloading operations).

Unfortunately, the instances previously used in the literature are not available. In addition, the authors of the related articles used random elements in their instance generators and thus, we could not re-create the same instances in order to compare the performances of the algorithms. Therefore, we have generated a new set of instances with up to 50 requests, two depots and two transfer nodes (available at `www.smartlogisticslab.nl`). The request nodes were randomly generated on a 200×200 Euclidean space. Transfer nodes were positioned in the center of the considered area. Time windows, service times and PD-vehicle capacities were randomly generated. Instances follow a naming convention of *pdptwt_n_m_v*, where n is the number of requests, m is the number of transfer nodes, and v is the number of vehicles.

In order to remove the SL scheduling part from the model, variables q_{ij}^{rw} are simplified to q_{ij}^{r}, binary variables taking value 1 if request r is transferred at node i, and 0 otherwise $\forall r \in \mathcal{P}$, $(i, j) \in \mathcal{E}$. The objective function of the PDPTW-T represents only the first term of expression (3.1), i.e., minimize routing costs. The constraints of the restricted master problem remain the same as in formulation (3.2)–(3.20), without constraints (3.6) and (3.13). The results are reported in Table 3.3.

Table 3.3 Comparison between PDPTW-T and PDPTW.

	PDPTW-T						PDPTW			
Instance	$\underline{z}$	$\overline{z}$	# of SLs	# of vehicles	Time (s)	Savings (%)	$\underline{z}$	$\overline{z}$	# of vehicles	Time (s)
pdptwt_20_2_8	1,256.04	1,256.04	10	7	291	2.84	1,292.80	1,292.80	7	3
pdptwt_20_2_10	1,326.29	1,326.29	7	7	25	11.77	1,503.16	1,503.16	8	5
pdptwt_20_2_10.1	1,477.37	1,477.37	1	9	94	2.27	1,511.71	1,511.71	9	3
pdptwt_25_2_10	1,629.68	1,629.68	4	9	3,021	7.59	1,763.54	1,763.54	9	4
pdptwt_25_2_8	1,463.86	1,463.86	3	8	841	6.05	1,558.08	1,558.08	8	48
pdptwt_25_2_10.1	1,409.90	1,409.90	14	8	7,501	5.31	1,489.00	1,489.00	8	34
pdptwt_30_1_12	1,829.70	1,829.70	1	9	255	2.63	1,879.07	1,879.07	9	96
pdptwt_30_2_10	1,831.37	1,831.37	8	9	674	4.84	1,924.51	1,924.51	9	9
pdptwt_30_1_12.1	1,838.15	1,838.15	10	10	72	4.06	1,915.98	1,915.98	10	6
pdptwt_35_1_10	1,838.17	1,838.17	7	9	9,405	5.57	1,946.68	1,946.68	10	9
pdptwt_35_1_12	2,084.32	2,084.32	6	10	146	3.44	2,158.53	2,158.53	11	10
pdptwt_35_1_12.1	1,986.74	1,986.74	3	9	3,360	2.21	2,031.65	2,031.65	10	52
pdptwt_40_1_12	2,120.48	2,120.48	12	11	1,631	6.30	2,263.00	2,263.00	11	11
pdptwt_40_1_12.1	2,146.15	2,146.15	4	11	231	5.54	2,271.95	2,271.95	12	7
pdptwt_40_1_12.2	2,135.27	–	–	–	10,800	–	2,195.56	2,195.56	11	9
pdptwt_50_1_14	2,445.66	–	–	–	10,800	–	2,501.05	2,501.05	12	33
pdptwt_50_1_16	2,439.23	–	–	–	10,800	–	2,469.07	2,469.07	12	149
pdptwt_50_1_14.1	2,559.78	–	–	–	10,800	–	2,590.23	2,590.23	13	205
Average	1,731.30	1,731.30	6	9	1,968	5	1,822.12	1,822.12	9	21

The results indicate that PDPTW-T instances with up to 40 requests can be solved to optimality. Similarly to the PDPTW-SL, transfer opportunities lead to cost savings, compared to the PDPTW solutions. More specifically, average savings of 5% can be achieved with one or two available transfer nodes. In addition, a slight decrease in the number of vehicles used is observed.

3.6 Conclusions

We have proposed a B&P algorithm to solve the PDPTW-SL. To fully evaluate the effectiveness of the algorithm, we have used a library of PDPTW-SL instances. We have shown that small to medium-size instances, with up to 50 freight requests, can be optimally solved by the proposed algorithm. Moreover, average savings of 10% (with a maximum of 32%) can be achieved by using available SLs to perform the deliveries, compared to the corresponding PDPTW solutions. Additionally, a special case of the problem, the PDPTW-T, is solved using the proposed algorithm. Instances with up to 40 requests provided average operational savings of 5% compared to the PDPTW solutions.

3.A Comparison between bi-directional versus mono-directional labeling algorithms

Table 3.4 reports detailed results regarding the performance of the bi-directional and the mono-directional labeling algorithms.

Table 3.4 Comparison between bi-directional and mono-directional algorithms.

	Bi-directional				Mono-directional		
Instance	Root	$\underline{z}$	$\overline{z}$	Time (s)	$\underline{z}$	$\overline{z}$	Time (s)
C6_1_4	369.36	369.36	369.36	0	369.36	369.36	1
C7_1_4	390.31	390.31	390.31	0	390.31	390.31	1
C8_1_4	416.93	446.35	446.35	4	446.35	446.35	6
C9_1_4	441.20	471.16	471.16	117	471.16	471.16	491
C10_1_4	471.40	510.01	510.01	67	510.01	510.01	85
C11_1_4	498.14	522.85	522.85	9	522.85	522.85	46
C12_1_4	541.60	541.60	541.60	4	541.60	541.60	7
C15_1_6	649.93	649.93	649.93	13	649.93	649.93	37
C20_1_8	745.38	751.23	751.23	83	751.23	751.23	98
C25_1_8	874.93	879.94	879.94	2,114	879.94	879.94	5,093
C40_1_8	1,199.31	1,199.31	–	10,800	–	–	10,800
RC6_1_4	572.76	572.76	572.76	0	572.76	572.76	0
RC7_1_4	575.95	575.95	575.95	0	575.95	575.95	0
RC8_1_4	585.32	585.32	585.32	0	585.32	585.32	0
RC9_1_4	593.70	593.70	593.70	0	593.70	593.70	1
RC10_1_4	599.95	599.95	599.95	0	599.95	599.95	1
RC11_1_4	624.48	624.48	624.48	1	624.48	624.48	1
RC12_1_6	662.03	662.03	662.03	1	662.03	662.03	3
RC15_1_6	1,068.16	1,068.16	1,068.16	1	1,068.16	1,068.16	0
RC20_1_8	1,200.36	1,200.36	1,200.36	6	1,200.36	1,200.36	8
RC25_1_8	1,528.02	1,530.43	1,530.43	28	1,530.43	1,530.43	34
RC40_1_12	1,955.79	1,969.45	1,969.45	738	1,969.45	1,969.45	2,827
RC50_1_14	2,199.79	2,200.88	2,200.88	1,235	2,200.88	2,200.88	2,169
RC60_1_16	2,611.51	2,636.18	–	10,800	2,611.51	–	10,800
R6_1_4	416.16	416.16	416.16	0	416.16	416.16	0
R7_1_4	458.28	473.06	473.06	1	473.06	473.06	1
R8_1_4	547.45	558.17	558.17	0	558.17	558.17	1
R9_1_4	607.48	632.42	632.42	2	632.42	632.42	4
R10_1_4	619.83	636.05	636.05	1	636.05	636.05	3
R11_1_4	746.03	748.29	748.29	1	748.29	748.29	2
R12_1_6	897.60	913.68	913.68	2	913.68	913.68	6
R15_1_6	1,058.97	1,083.50	1,083.50	15	1,083.50	1,083.50	7
R20_1_8	1,533.21	1,542.93	1,542.93	12	1,542.93	1,542.93	7
R25_1_8	1,613.81	1,636.12	1,636.12	50	1,636.12	1,636.12	64
R40_1_12	1,948.56	1,969.77	1,969.77	895	1,969.77	1,969.77	948
R50_1_14	2,592.15	2,595.14	2,595.14	580	2,595.14	2,595.14	797
R60_1_16	2,854.99	2,864,75	–	10,800	2,858.69	–	10,800
Average				176			375

3.B Comparison between PDPTW-SL and PDPTW solutions

In Table 3.5, under columns "$\underline{z}$" and "$\overline{z}$" we indicate the best lower and upper bounds found within the time limit. The column titled "# of SLs" shows the number of demand units shipped on scheduled lines. In addition, "# of vehicles" indicates the number of vehicles used in the solutions. Finally, "Time" and "Savings" show the time needed to obtain a solution and cost savings from using SLs as part of freight journey, respectively.

Table 3.5 Comparison between PDPTW-SL and PDPTW with homogeneous routing costs.

	PDPTW-SL						PDPTW			
Instance	$\underline{z}$	$\overline{z}$	# of SLs	# of vehicles	Time (s)	Savings (%)	$\underline{z}$	$\overline{z}$	# of vehicles	Time (s)
C6_1_4	369.36	369.36	5	3	0	20.00	461.70	461.70	3	1
C7_1_4	390.31	390.31	6	3	0	15.97	464.49	464.49	3	1
C8_1_4	446.35	446.35	5	4	4	7.68	483.50	483.50	3	0
C9_1_4	471.16	471.16	7	4	117	5.43	498.19	498.19	3	1
C10_1_4	510.01	510.01	4	4	67	1.02	515.25	515.25	3	1
C11_1_4	522.85	522.85	4	4	9	0.35	524.69	524.69	3	1
C12_1_4	541.60	541.60	5	4	4	9.51	598.55	598.55	3	5
C15_1_6	649.93	649.93	8	4	13	10.25	724.12	724.12	4	1
C20_1_8	751.23	751.23	2	5	83	5.45	794.53	794.53	5	2
C25_1_8	879.94	879.94	11	5	2,114	18.27	1,076.71	1,076.71	6	32
C40_1_8	1,199.31	–	–	–	10,800	–	1,428.62	1,428.62	8	2,203
RC6_1_4	572.76	572.76	1	3	0	13.09	658.99	658.99	4	0
RC7_1_4	575.95	575.95	1	3	0	13.02	662.18	662.18	4	0
RC8_1_4	585.32	585.32	1	3	0	11.74	663.16	663.16	4	0
RC9_1_4	593.70	593.70	3	3	0	15.58	703.30	703.30	4	1
RC10_1_4	599.95	599.95	3	3	0	15.45	709.55	709.55	4	0
RC11_1_4	624.48	624.48	3	3	1	15.90	742.52	742.52	4	1
RC12_1_6	662.03	662.03	3	4	1	16.66	794.42	794.42	5	1
RC15_1_6	1,068.16	1,068.16	0	6	1	0.00	1,068.16	1,068.16	6	1
RC20_1_8	1,200.36	1,200.36	4	7	6	16.48	1,437.13	1,437.13	7	4
RC25_1_8	1,530.43	1,530.43	6	8	28	2.65	1,572.09	1,572.09	7	12
RC40_1_12	1,969.45	1,969.45	2	10	738	3.16	2,033.66	2,033.66	10	87
RC50_1_14	2,200.88	2,200.88	5	10	1,235	3.08	2,270.83	2,270.83	10	55
RC60_1_16	2,636.18	–	–	–	10,800	–	2,682.66	2,682.66	12	1,534
R6_1_4	416.16	416.16	0	3	0	0.00	416.16	416.16	3	0
R7_1_4	473.06	473.06	0	3	1	0.00	473.06	473.06	3	0
R8_1_4	558.17	558.17	0	3	0	0.00	558.17	558.17	3	0
R9_1_4	632.42	632.42	3	4	2	3.17	653.12	653.12	3	1
R10_1_4	636.05	636.05	3	4	1	3.42	658.60	658.60	4	1
R11_1_4	748.29	748.29	3	4	1	0.54	752.38	752.38	4	1
R12_1_6	913.68	913.68	3	5	2	2.41	936.23	936.23	5	3
R15_1_6	1,083.50	1,083.50	0	6	15	0.00	1,083.50	1,083.50	6	4
R20_1_8	1,542.93	1,542.93	0	8	12	0.00	1,542.93	1,542.93	8	1
R25_1_8	1,636.12	1,636.12	3	8	50	4.73	1,717.39	1,717.39	8	39
R40_1_12	1,969.77	1,969.77	0	9	895	0.00	1,969.77	1,969.77	9	220
R50_1_14	2,595.14	2,595.14	2	12	580	0.98	2,620.89	2,620.89	13	388
R60_1_16	2,858.69	–	–	–	10,800	–	2,959.71	2,959.71	14	1,084
Average		909	3	5	176	7		966	5	25

Table 3.6 illustrates the results obtained after solving the proposed instances with heterogeneous routing costs. The minimum-capacity vehicle is assumed to cost 0.5 per operating time unit. Larger vehicles are assigned a cost that linearly increases with the carrying capacity. The columns are self-explanatory, similar to the previously presented table.

Table 3.6 Comparison between PDPTW-SL and PDPTW with heterogeneous routing costs.

	PDPTW-SL						PDPTW			
Instance	$\underline{z}$	$\overline{z}$	# of SLs	# of vehicles	Time (s)	Savings (%)	$\underline{z}$	$\overline{z}$	# of vehicles	Time (s)
C6_1_4	385.49	385.49	5	3	0	21.07	488.38	488.38	3	0
C7_1_4	410.76	410.76	6	3	1	16.82	493.81	493.81	3	1
C8_1_4	468.30	468.30	5	4	4	8.75	513.19	513.19	3	0
C9_1_4	495.64	495.64	7	4	32	6.25	528.70	528.70	3	0
C10_1_4	535.01	535.01	4	4	39	2.14	546.71	546.71	3	1
C11_1_4	548.55	548.55	4	4	47	1.46	556.67	556.67	3	0
C12_1_4	568.30	568.30	5	4	4	9.46	627.67	627.67	3	1
C15_1_6	980.46	980.46	8	5	108	18.81	1,207.67	1,207.67	4	9
C20_1_8	890.02	890.02	6	4	132	15.49	1,053.12	1,053.12	5	134
C25_1_8	1,330.48	1,330.48	13	6	7,395	20.64	1,676.53	1,676.53	6	67
C40_1_8	1,441.80	–	–	–	10,800	–	1,803.10	1,803.10	8	9,054
RC6_1_4	593.52	593.52	1	3	0	13.66	687.39	687.39	4	0
RC7_1_4	596.89	596.89	1	3	0	13.59	690.76	690.76	4	0
RC8_1_4	606.78	606.78	1	3	0	12.28	691.73	691.73	4	0
RC9_1_4	615.50	615.50	3	3	1	15.90	731.87	731.87	4	0
RC10_1_4	621.75	621.75	3	3	1	15.81	738.47	738.47	4	0
RC11_1_4	646.28	646.28	3	3	1	16.42	773.28	773.28	4	0
RC12_1_6	693.75	693.75	3	4	1	15.72	823.19	823.19	5	0
RC15_1_6	1,586.84	1,586.84	0	6	1	1.44	1,610.10	1,610.10	6	1
RC20_1_8	1,639.68	1,639.68	4	7	6	18.89	2,021.58	2,021.58	7	6
RC25_1_8	2,403.87	2,403.87	6	8	61	4.16	2,508.24	2,508.24	7	14
RC40_1_12	3,145.84	3,145.84	1	10	8,964	3.19	3,249.36	3,249.36	10	150
RC50_1_14	3,442.28	3,442.28	5	10	8,172	6.03	3,663.11	3,663.11	10	85
RC60_1_16	4,024.96	–	–	–	10,800	–	4,077.42	4,077.42	13	2,451
R6_1_4	416.16	416.16	0	3	0	0.00	416.16	416.16	3	0
R7_1_4	473.06	473.06	0	3	1	0.00	473.06	473.06	3	1
R8_1_4	558.17	558.17	0	3	1	0.00	558.17	558.17	3	0
R9_1_4	632.42	632.42	3	4	6	3.17	653.12	653.12	3	1
R10_1_4	636.05	636.05	3	4	2	3.42	658.60	658.60	4	1
R11_1_4	748.29	748.29	3	4	3	0.54	752.38	752.38	4	0
R12_1_6	946.53	946.53	3	5	5	2.46	970.42	970.42	5	3
R15_1_6	1,377.04	1,377.04	0	6	21	0.00	1,377.04	1,377.04	6	4
R20_1_8	1,839.34	1,839.34	1	8	152	0.05	1,840.22	1,840.22	7	1
R25_1_8	2,011.86	2,011.86	1	8	52	5.80	2,135.65	2,135.65	8	52
R40_1_12	2,518.05	2,518.05	0	9	1,162	1.80	2,564.27	2,564.27	9	471
R50_1_14	3,182.19	3,182.19	2	12	3,478	1.40	3,227.26	3,227.26	13	877
R60_1_16	3,479.13	–	–	–	10,800	–	3,639.99	3,639.99	16	2,402
Average		1,134	3	5	878	9		1,221	5	55

3.C The effect of multiple SLs

Table 3.7 indicates the results obtained when solving the instances with up to three SLs. The average is computed over all instances that we optimally solved in all SL configurations. As expected, multiple SLs lead to more savings compared to corresponding PDPTW solutions. In particular, 2-SLs and 3-SLs cases led to cost savings of 4.58%, and 8.39% respectively. In is important to note that the cost savings are mainly obtained from using available capacities, thus minimizing the total traveling time of the PD vehicles. In addition, the average number of vehicles used did not significantly change compared to the classical system.

Table 3.7 The effect of multiple available SLs.

Instance	One SL (sl=1)				Two SLs (sl=2)				Three SLs (sl=3)			
	$\bar{z}$	# of vehicles	# of SLs	Time (s)	$\bar{z}$	# of vehicles	# of SLs	Time (s)	$\bar{z}$	# of vehicles	# of SLs	Time (s)
C6_sl_4	369.36	3	5	0	369.36	3	5	2	310.85	3	8	2,657
C7_sl_4	390.31	3	6	0	381.21	3	6	349	313.41	3	9	5,795
C8_sl_4	446.35	4	5	4	415.84	3	5	1,507	–	–	–	10,800
C9_sl_4	471.16	4	7	117	–	–	–	10,800	–	–	–	10,800
RC6_sl_4	572.76	3	1	0	572.76	3	1	0	520.11	4	4	1
RC7_sl_4	575.95	3	1	0	575.95	3	1	0	539.08	4	4	1
RC8_sl_4	585.32	3	1	0	585.32	3	1	1	548.45	4	4	1
RC9_sl_4	593.70	3	3	0	593.70	3	3	1	552.71	4	6	1
RC10_sl_4	599.95	3	3	0	599.95	3	3	2	558.96	4	6	1
RC11_sl_4	624.48	3	3	1	624.48	3	3	2	585.72	4	7	11
RC12_sl_6	662.03	4	3	1	662.03	4	3	5	619.59	5	7	11
RC15_sl_6	1,068.16	6	0	1	1,059.38	6	1	2	1,059.38	6	1	2
RC20_sl_8	1,200.36	7	4	6	1,127.02	8	8	4,677	1,127.02	8	8	1,439
RC25_sl_8	1,530.43	8	6	28	1,388.18	7	8	35	1,371.47	7	14	947
RC40_sl_12	1,969.45	10	2	738	–	–	–	10,800	–	–	–	10,800
R6_sl_4	416.16	3	0	0	416.16	3	0	0	416.16	3	0	0
R7_sl_4	473.06	3	0	1	473.06	3	0	1	470.15	3	1	8
R8_sl_4	558.17	3	0	0	535.27	4	1	14	490.77	3	2	1
R9_sl_4	632.42	4	3	2	579.38	4	3	2	579.38	4	3	5
R10_sl_4	636.05	4	3	1	584.89	4	3	2	584.89	4	3	4
R11_sl_4	748.29	4	3	1	687.36	4	3	1	687.36	4	3	2
R12_sl_6	913.68	5	3	2	854.80	7	5	4,253	822.64	5	5	133
R15_sl_6	1,083.50	6	0	15	1,023.99	6	3	27	921.56	6	6	1,072
R20_sl_8	1,542.93	8	0	12	1,425.42	8	3	310	1,378.70	7	8	10,283
R25_sl_8	1,636.12	8	3	50	1,495.46	8	9	23	1,493.46	8	7	60
R40_sl_12	1,969.77	9	0	895	1,922.00	9	4	1,291	–	–	–	10,800
Average	792	4	2	6	755	5	3	441	725	5	5	1,183

Chapter 4

An adaptive large neighborhood search heuristic for the pickup and delivery problem with time windows and scheduled lines

4.1 Introduction

The aim of this chapter is to present a metaheuristic algorithm to solve the PDPTW-SL, which was formally presented in Chapters 2 and 3. In particular, we apply adaptive large neighborhood search (ALNS). The ALNS algorithm is based on the "destroy and re-create" principle. More specifically, given an initial solution, for a number of iterations, the algorithm partially destroys the solution, and then repairs it using several operators, until a certain stopping criterion is satisfied. The ALNS is a state-of-the-art heuristic, which proved good results for many combinatorial optimization problems, in particular vehicle routing problems (Røpke and Pisinger, 2006; Demir et al., 2012; Masson et al., 2012).

The contributions of the chapter are as follows: *(i)* to adapt the classical ALNS heuristic algorithm to solve PDPTW-SL, and *(ii)* to efficiently handle synchronization constraints within the proposed ALNS framework. The rest of

this chapter is structured as follows. Section 4.2 describes the proposed ALNS heuristic algorithm for the PDPTW-SL. The experimental results are given in Section 4.3. Finally, conclusions are provided in Section 4.4.

4.2 An ALNS algorithm for the PDPTW-SL

The proposed metaheuristic (ALNS) is an extension of the Large Neighborhood Search (LNS) heuristic, which is first introduced by Shaw (1998). The ALNS metaheuristic was first used to solve several vehicle routing problems by Røpke and Pisinger (2006); Pisinger and Røpke (2007). Unlike LNS, where one large neighborhood is usually used, the ALNS applies several neighborhood operators. The neighborhood of a given set of feasible routes is investigated by removing some requests and reinserting them back to the solution. The operators that performed well in previous iterations, are given higher chances to being chosen than the ones that proved poor performance. Thus, each operator is assigned a *probability* of being selected. The newly-generated solution is accepted if the criteria defined by the master search framework is satisfied.

4.2.1 Initialization stage

A greedy insertion heuristic is used to obtain a feasible initial solution to the PDPTW-SL. All requests are initially stored in a list $\mathcal{L}$ and inserted one by one in a random order in their best available positions (for more detail on greedy insertion (GI)–see Section 4.2.4). The feasibility with regard to the capacity of PD and SL vehicles and time windows is always maintained.

4.2.2 Probability adjustment procedure

A roulette-wheel mechanism is used to control the selection of the neighborhood operators. At the beginning, the probability of removal and insertion operators are uniformly assigned. As in the implementation of Røpke and Pisinger (2006), every N_w iterations, each operator is given a score π_i. The *probability* is updated as $P_d^{t+1} = P_d^t\,(1 - r_p) + r_p\,\pi_i/\omega_i$, where r_p is the adjustment parameter, π_i is the accumulated score of operator i and ω_i shows the amount of times the operator was used. The score of the operators is increased by σ_1 if a new best solution is obtained. The score is increased by σ_2 if the newly-generated solution outperforms the current solution. Finally, if a deteriorated solution is accepted, the score is increased by σ_3.

4.2.3 Removal stage

Ten removal operators are used in our ALNS heuristic algorithm. All operators are adapted from or inspired by Shaw (1998); Røpke and Pisinger (2006) and Demir et al. (2012). The removal stage mainly consists of removing ϕ requests from the current solution and adding them to so-called *removal list* $\mathcal{L}$ as shown in Algorithm 1. The algorithm starts with a given feasible routing solution X and returns a solution where some requests are disregarded. A chosen operator is used to remove a set of requests (i.e., pickup, delivery and transfer nodes if used) from the solution. The parameter ϕ defines the number of requests to be removed. Removed pickup and delivery nodes are then inserted into list $\mathcal{L}$.

Algorithm 1: The overall structure of the removal operators

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

1 *Initialize removal list ($\mathcal{L} \leftarrow \emptyset$)*
2 **for** *ϕ iterations* **do**
3 *Apply removal operator to remove a request r (includes two nodes; pickup and delivery)*
4 $\mathcal{L} \leftarrow \mathcal{L} \cup r$

The removal operators used in our implementation are introduced below.

- **Random Removal (RR)**: This operator randomly removes ϕ requests from the solution, and runs for $\phi \in [\underline{\phi}, \overline{\phi}]$ iterations, where $\underline{\phi}$ and $\overline{\phi}$ are the lower and the upper limits on the number of requests to be removed and ϕ is a random integer number within the specified range.
- **Route Removal (ROR)**: This operator removes a complete route from a given solution. The route that is to be removed is randomly selected. The ROR operator then iteratively selects a node j from the selected route until it is emptied. The corresponding node of j, i.e., its pickup or delivery node, is removed irrespective of which route it is positioned in.
- **Late-Arrival Removal (LAR)**: For every request r, this operator calculates the difference between the service start time and l_r, l_{r+n} respectively, and then removes the request with the largest total difference (i.e., $r^* = argmax_{r \in \mathcal{P}}\{|[\beta_r - s_r - l_r] + [\beta_{r+n} - s_{r+n} - l_{r+n}]|\}$, where β_x is the departure time from node x). Similarly to RR, this operator runs for $\phi \in [\underline{\phi}, \overline{\phi}]$ iterations.
- **Worst-Distance Removal (WDR)**: This operator repeatedly removes costly requests. The cost is considered to be the total distance from the

pickup and delivery nodes of a request to the prior and succeeding nodes within given routes, i.e., it removes node $r^* = argmax_{r\in\mathcal{P}}\{|\Upsilon_{i-1,r}+\Upsilon_{r,i+1}+\Upsilon_{j-1,r+n}+\Upsilon_{r+n,j+1}|\}$, where $i-1$ and $j-1$, are predecessors and $i+1$ and $j+1$ successors of the pickup and delivery nodes, respectively.

- **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., r_1 or r_1+n) and another to r_2 (i.e., r_2 or r_2+n) are in the same vehicle route, and 1 otherwise. This operator picks the request $r^* = argmin_{r_2\in\mathcal{P}}\{\Pi_1[\Upsilon_{r_1,r_2}+\Upsilon_{r_1+n,r_2+n}]+\Pi_2[|\beta_{r_1}-\beta_{r_2}|+|\beta_{r_1+n}-\beta_{r_2+n}|]+\Pi_3 l_{r_1,r_2}+\Pi_4|d_{r_1}-d_{r_2}|\}$, where Π_1–Π_4 are normalized weights. The operator is applied $\phi \in [\underline{\phi},\overline{\phi}]$ times by selecting a request, which is not yet added in $\mathcal{L}$ and which is the most similar to the last added request.

- **Proximity-Based Removal (PR)**: This operator removes a set of requests that are similar in terms of distance. In other words, it is a special case of SR operator with $\Pi_1 = 1$, and $\Pi_2 = \Pi_3 = \Pi_4 = 0$.

- **Demand-Based Removal (DR)**: This operator is a special case of SR with $\Pi_1 = \Pi_2 = \Pi_3 = 0$, and $\Pi_4 = 1$.

- **Time-Based Removal (TR)**: This operator is a special case of SR with $\Pi_1 = \Pi_3 = \Pi_4 = 0$, and $\Pi_2 = 1$.

- **Historical Knowledge Removal (HR)**: This operator stores the position cost of every request r. The cost is assumed to be the total distance from the origin and destination nodes of request r to the prior and succeeding nodes within the routes, and determined as $c_r = \Upsilon_{i-1,r} + \Upsilon_{r,i+1} + \Upsilon_{j-1,r+n} + \Upsilon_{r+n,j+1}$ at every iteration of the algorithm. Note that $i-1$ and $j-1$ are the preceding nodes of the origin and, respectively, destination nodes of r and $i+1$ and $j+1$ are their successors in the corresponding PD-vehicle routes. The best position cost c_r^* of request r is set to be the minimum of all values found so far. The HR operator then selects the request r^* with the largest cost difference from its best position cost, i.e., $r^* = argmax_{r\in\mathcal{P}}\{c_r - c_r^*\}$. Request r^* is added to $\mathcal{L}$. This operator iterates $\phi \in [\underline{\phi},\overline{\phi}]$.

- **Worst Removal (WR)**: This operator removes $\phi \in [\underline{\phi},\overline{\phi}]$ requests with the highest cost. The cost in this case is computed as follows: given a solution, the cost of a request r is the difference in the objective function between the current solution (with r) and the same solution without serving r. Note that the difference between WR and WDR is that WR uses the total cost (including transfer cost) of the requests as objective, whereas WDR focuses only on the distance. In addition, WR also takes into account the length

of the newly introduced arcs $(i-1, i+1)$ and $(j-1, j+1)$ when removing nodes i and j from a route.

4.2.4 Insertion stage

We have used 5 insertion operators in the proposed ALNS heuristic algorithm. With the help of insertion operators, the removed requests in $\mathcal{L}$ can be inserted into the existing routes, if possible. The general schematic overview of an insertion procedure is shown in Algorithm 2. It is noteworthy that in Line 5 of the insertion algorithm, we do not consider every possible insertion. In order to avoid redundant computations in *repairTransfers*$(X,\ r,\ T^r)$ and *feasibleSchedule*(X)) due to time windows, a preliminary time window check is applied as described in Braekers et al. (2014). Due to the fact that PD-vehicle capacity constraints can be fully verified only after applying *repairTransfers*(X, r, T^r) procedure, capacity violations for the routes with no transferable requests can be easily detected. Note that *feasibleSchedule(X)* and *feasibleCapacity(X, c(X), η_{ij})* (see Line 11 in Algorithm 2) methods are described in Section 4.2.5.

Algorithm 2: The generic structure of an *insertion* i^* *(X^*_{new}, $\mathcal{L}$, $X_{current}$)* procedure

input : A partially destroyed current solution X^*_{new}, a list of removed requests $\mathcal{L}$, and current solution $X_{current}$
output: A feasible solution obtained after the insertion procedure

```
1  for (each request r in L) do
2      X_new ← NULL
3      c(X*_new) ← +∞
4      for (each route Q and route M) do
5          for (each position q within a route Q and each position m within a route M) do
6              Insert r and r + n in positions q and m, respectively, in solution X*_new
7              if (Q ≠ M) then
8                  (X_t, t*_r, t*_{r+n}) ← repairTransfers(X*_new, r, T^r)
9              else
10                 X_t ← X*_new
11             if (feasibleCapacity(X_t, c(X_t), η_ij) and feasibleSchedule(X_t)) then
12                 if (acceptance criteria of i* is satisfied) then
13                     X_new ← X_t
14     if (X_new ≠ NULL) then
15         X*_new ← X_new
16     else
17         return X_current
18 return X*_new
```

Note that the insertion operators of the PDPTW-SL are more complicated than the ones for the classical PDPTW. In particular, the pickup and delivery nodes of a request can be inserted in two different routes. It means that such requests must use SLs, and therefore corresponding transfer nodes need to be inserted in the considered routes. For example, pickup node r of a request is inserted in route A, and delivery node $r+n$ in route B, respectively. In order to make this solution feasible (in terms of precedence constraints), a transfer node needs to be inserted in each of the routes: A and B, respectively. In particular, one transfer node is inserted in route A, sequenced after pickup node r (i.e., first the request is picked up from r, and then transferred). Another transfer node is inserted in route B, sequenced before delivery node $r+n$ (i.e., first the request is recollected from a transfer node, and then delivered to its final destination). For modeling convenience, each original transfer node is replicated n (number of requests) times, hence each request gets assigned a list of replicated transfer nodes (see Chapter 2 for more details). Therefore, *repairTransfers(X, r, T^r)* procedure is run to insert corresponding replicated transfer nodes related to the considered request in a greedy fashion. The overview of the repair mechanism is shown in Algorithm 3. Obviously, tailored classical insertion operators involve extra computational complexity, as the solution neighborhoods in the PDPTW-SL are larger.

Algorithm 3: The generic structure of the *repairTransfers(X, r, T^r)* procedure

input : A partial solution X, a request r and T^r replicated transfer nodes related to r
output: A feasible solution X_t and a origin and a destination transfer nodes (t^*_r, t^*_{r+n}) of r in the current solution X_t

1 $Q \leftarrow$ *the route related to* r
2 $M \leftarrow$ *the route related to* $r+n$
3 $t^*_r \leftarrow$ *transfer node* $\in T^r$ *with the cheapest insertion in route* Q*, sequenced after* r
4 $T^r = T^r \setminus t^*_r$
5 $t^*_{r+n} \leftarrow$ *transfer node* $\in T^r$ *with the cheapest insertion in route* M*, sequenced earlier than* $r+n$
6 $X_t \leftarrow$ *updated* X *given minimal cost insertions of* t^*_r *and* t^*_{r+n}
7 *return* (X_t, t^*_r, t^*_{r+n})

A transferable request r is assigned two related replicated transfer nodes: t^*_r (i.e., origin transfer node), and t^*_{r+n} (i.e., destination transfer node).

We now describe the insertion operators. We note that the requests are randomly chosen from the removal list $\mathcal{L}$.

- **Greedy Insertion (GI)**: This operator iteratively inserts a request (both pickup and delivery nodes) in the best possible position of the routes. $\Delta^r_{I_i J_j}$ is the objective function value when the pickup node of r is inserted in route

I (position i) and its corresponding delivery node is inserted in route J (position j). Thus, ${\Delta^r_{I_i J_j}}^* = argmin\{\Delta^r_{I_i J_j}\}$.

- **Second-best Insertion (SI)**: This new operator chooses the second best insertion for randomly selected unassigned request. The main idea of using this operator is to diversify the search.

- **Greedy Insertion and Noise function (GIN)**: This operator is similar to the GI and involves certain flexibility in selecting the best insertion of a request. This flexibility is obtained by changing the cost for request r: $Updated\,Cost = Actual\,Cost + \bar{d}\,\mu\,\epsilon$, where $\bar{d}$ is the largest distance between nodes, μ is a noise parameter used for diversification and is set equal to 0.1, and ϵ is a random number between $[-1, 1]$. *Updated Cost* is computed for each request in $\mathcal{L}$.

- **Second-best Insertion and Noise function (SIN)**: This operator is similar to the SI and involves the same noise function that is used in GIN operator.

- **Best out of λ Feasible Insertions (λFI)**: This new operator is similar to GI but chooses the best insertion out of the first λ feasible insertions. The parameter λ is a randomly generated integer number between 1 and ψ. For each unassigned request we generate a set of possible insertions associated with corresponding distance-based costs (disregarding distance to the corresponding transfer nodes), while considering precedence constraints. These positions are not necessary feasible in terms of time windows. Additionally, in order to filter out some more of the infeasible insertions, the time windows of the subsequent nodes visited within the considered routes are verified. The generated set is sorted in increasing order based on insertion costs. Hence the operator investigates the cheapest insertions overall routes first.

Moreover, five additional variants of these operators are also used, where the requests are sorted in $\mathcal{L}$ in terms of their time flexibility (i.e., the least flexible first). The flexibility of request r can be computed as $|u_{r+n} - l_r|$.

4.2.5 Feasibility of the routes and schedules

Compared to the classical PDPTW, scheduling (time windows) constraints bring extra complexity in the PDPTW-SL since the requests may be picked up by a PD vehicle and delivered by another one. This implies that PD vehicles and SLs must be synchronized. This problem can be handled as shown in Algorithm 4.

Algorithm 4: The generic structure of the *feasibleSchedule(X)* procedure

input : A solution X
output: Boolean value: $TRUE$ if feasible, $FALSE$ if infeasible

```
1  list ← ∅
2  β_{t_r^*} ← 0, β_{t_{r+n}^*} ← 0, ∀ r ∈ P
3  for (each route I in X) do
4      for (each node i in I) do
5          β_i ← max{β_{p_i} + ϒ_{p_i,i} + s_i; l_i + s_i}
6          if (β_i > u_i + s_i) then
7              return FALSE
8          if (i = t*_{r+n}, ∀r ∈ P) then
9              list ← list ∪ i
10             i ← end route
11 while (list ≠ ∅) do
12     for (t*_{r+n} in list) do
13         if (β_{t_r^*} > 0) then
14             path_r ← generatePath(t*_r, t*_{r+n})
15             β_{t*_{r+n}} ← determineTime(β_{t*_r}, path_r)
16             for (each successor node i of t*_{r+n}) do
17                 β_i ← max{β_{p_i} + ϒ_{p_i,i} + s_i; l_i + s_i}
18                 if (β_i > u_i + s_i) then
19                     return FALSE
20                 if (i = t*_{r1+n}, ∀r1 ∈ P) then
21                     list ← list ∪ i
22                     i ← end route
23             list ← list \ t*_{r+n}
24     if (β_{t_r^*} = 0, ∀ β_{t*_{r+n}} ∈ list) then
25         return FALSE
26 return TRUE
```

All β values for the transfer nodes are reset to 0 in Line 2. The first block of the algorithm (Lines 3–10) sets the time for all nodes at the earliest possible value (depending on the start of the time window and the departure time from the preceding node p_i). The algorithm switches to the next route whenever a destination transfer node is reached. This node is added to a *list* that contains all destination transfer nodes which do not have an updated synchronized time. The second part of the algorithm (Lines 11–25) runs until the *list* of destination transfer nodes is emptied or a cycle (see Figure 4.2) is found.

- For each $t^*_{r+n} \in list$ with the corresponding origin transfer node (i.e., t^*_r) that has already received a timing, generate a shortest path (Dijkstra (1959))

from t_r^* to t_{r+n}^* (Line 14). Given the shortest path and the departure time from t_r^*, the algorithm computes the time at t_{r+n}^* by considering earliest scheduled departure time later than $\beta_{t_r^*}$ (i.e., $\lceil \beta_{t_r^*} \rceil$) for every intermediate line used (Line 15). Considered t_{r+n}^* with a synchronized timing is removed from the *list*. Finally, all succeeding nodes of t_{r+n}^* get assigned a value until a destination transfer node is reached and added to the *list*.

- Whenever all $\beta_{t_r^*}$ is zero $\forall\ t_{r+n}^* \in list$, the algorithm finds a cycle (see Figure 4.2) and stops. Hence, no feasible schedule is possible for a given solution X.

For a better understanding, we refer to the example shown in Figure 4.1. Three requests, namely a, b, and c, are to be delivered to destination nodes $a+n$, $b+n$ and $c+n$, respectively. All the requests are transferable, thus each request is shipped on a scheduled line. All these transfer nodes are replications of the original transfer nodes and each replication is assigned to only one request. In the present example, T_a represents the origin transfer node of a, and T_{a+n} is its destination transfer node. For the sake of simplicity, each arc has one time unit and each node does not require any service time. The numbers on top of the nodes indicate the departure time from that specific node. In this example, three PD vehicles are needed for the transportation of these three requests.

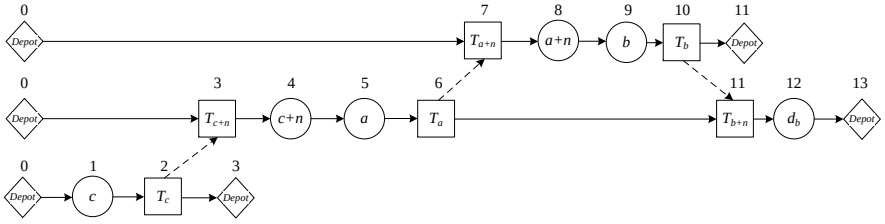

Figure 4.1 An interdependency relation of the routes

By referring to Figure 4.1, the feasibility of the schedule is checked as follows: the process starts in the first route. The algorithm reaches the transfer node T_{a+n} that is a destination transfer node of request a. The algorithm moves to the second route and reaches T_{c+n}, which is a destination transfer node of request c. Finally, it moves to the last route and updates the timing for the whole route as no destination transfer node is encountered. The *list* contains two destination transfer nodes, i.e., T_{a+n} and T_{c+n}. A shortest path is generated from T_c to T_{c+n}, and the departure time $\beta_{T_{c+n}}$ is updated (i.e., 3). Afterwards, the subsequent nodes of T_{c+n} for a given synchronized time $\beta_{T_{c+n}}$ get assigned time values until T_{b+n} is reached. Similarly, the timing of T_{a+n} is updated and followed by T_{b+n}.

Note that a cycle implies that the precedence constraints are violated for at least two transferable requests, since such requests need to be picked up and dropped off twice (see Figure 4.2). In addition, cycles may be composed of multiple routes and requests.

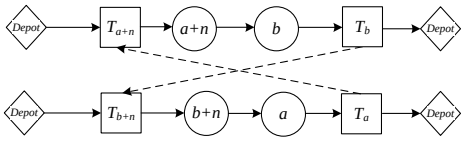

Figure 4.2 An example of a two-request cycle

The procedure *determineTime*($\beta_{t_r^*}$, $path_r$) (see Algorithm 4, Line 15) computes departure time from t^*_{r+n} for a given departure time from t^*_r. Note that unlike the classical case, in the PDPTW-SL the whole solution needs to be verified. The main reason for that is the fact that a change in a route, may significantly affect the scheduling in other routes as well, as there may be transferable requests. Therefore, the synchronization constraints require extra computational effort to investigate a solution.

The capacity constraints of both PD and SL vehicles are verified in Algorithm 5. First, the algorithm checks SL capacity by first generating scheduled line paths along with every scheduled departure time from the considered transfer nodes (Line 3). The capacity variables are updated for each SL and each possible scheduled departure. The cost of the solution is updated in Line 6. Whenever the capacity is violated (Line 7), the algorithm stops and returns *FALSE* value. The PD-vehicle capacity constraints can be checked in a standard way with some preprocessing (Line 9). In particular, the destination transfer node of a transferable request r (i.e., t^*_{r+n}) becomes an origin node on a different route, hence it will have a positive demand (i.e., d_r) whereas the origin transfer node of r (i.e., t^*_r) becomes a destination node and it is assigned a negative demand (i.e., $-d_r$).

4.2.6 Master search framework

In the proposed ALNS, we used simulated annealing (SA). The overall pseudo-code of the proposed algorithm is provided in Algorithm 6.

The notation X_{best} shows the best solution obtained, $X_{current}$ presents the current solution, and X_{new} shows a newly-generated solution that can be accepted or rejected at each iteration. Each solution X is associated with a cost $c(X)$. Any solution X_{new} can have three outcomes: *(i)* if $c(X_{new}) < c(X_{current})$, then it

Algorithm 5: The generic structure of *feasibleCapacity*$(X,\ \mathrm{c}(X),\ \eta_{ij})$ procedure

input : A solution X, routing cost of X, and cost of using SL η_{ij} per unit shipped on (i,j)
output: A boolean value θ

Initialize m^w_{ij} array for the capacity used on the scheduled line (i,j) at time p^w_{ij} and $\overline{\mathcal{P}}^t$ $\leftarrow$ set of transferable requests in solution X
for *(r in $\overline{\mathcal{P}}^t$)* **do**
    set $(\overline{\mathcal{K}}^r, \gamma)$ $\leftarrow$ scheduled lines (i,j) used by r and corresponding scheduled departure times γ^r_{ij}
    for *($(i,j) \in \overline{\mathcal{K}}^r$)* **do**
        set $m^w_{ij} \leftarrow m^w_{ij} + d_r$
        set $c(X) \leftarrow c(X) + d_r\ \eta_{ij}$
        if *($m^w_{ij} > k_{ij}$)* **then**
            return FALSE
if *($PDvehicleCapacityViolated(X)$)* **then**
    return FALSE
return TRUE

is accepted, *(ii)* if $c(X_{new}) > c(X_{current})$ and $e^{-(c(X_{new})-c(X_{current}))/T}$ is greater than a random number within [0,1], with T–*temperature*, then X_{new} is accepted as well, and finally, *(iii)* it is rejected. The initial temperature is selected as T. The temperature T diminishes at each iteration by a rate κ, where $0 < \kappa < 1$. The algorithm runs for a given number of iterations (i.e., 10,000 iterations).

4.3 Computational experiments

To analyze the performance of the proposed ALNS, we run a series of experimental tests and present the obtained results in this section.

4.3.1 Data generation

Three sets of instances, namely R, C, and RC, with three centrally-positioned scheduled lines in a triangular topology and a frequency of one departure of every 30 time units are considered. Each instance contains up to 100 requests (i.e., 100 pickup and 100 delivery nodes) over 200×200 time units on an Euclidean space. Some of the considered instances (with 6–12, 15, 20, 25, 40, 50, and 60 requests) were used in Chapters 2 and 3. Instances with 75 and 100 requests are generated in the same way as described in previous chapters. In particular, instances follow

Algorithm 6: The pseudo-code of the ALNS with SA

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

Generate an initial solution by using the Greedy insertion algorithm
Initialize probability P_d^t for each destroy operator $d \in D$ and probability P_i^t for each insertion operator $i \in I$
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 P_d^t
  Let X_{new}^* be the solution generated after using operator d^* to $X_{current}$
  Choose an insertion operator $i^* \in I$ with probability P_i^t
  Let X_{new} be the newly-generated solution after applying operator i^* to X_{new}^*
  if $c(X_{new}) < c(X_{current})$ **then**
    $X_{current} \leftarrow X_{new}$
    if $c(X_{current}) < c(X_{best})$ **then**
      $X_{best} \leftarrow X_{current}$
  else
    Let $\nu \leftarrow e^{-(c(X_{new})-c(X_{current}))/T}$
    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*

a naming convention of $G\_n\_sl\_v$, where G is the geographic distribution of the customers, n is the number of requests that needs to be served, sl is the number of SLs, and v is the number of available PD vehicles. Instances are classified with regard to the location of the requests. For example, C involves clustered nodes around transfer nodes, R considers uniform-randomly distributed request nodes, and finally RC involve randomly clustered nodes. More specifically, C and RC have the nodes positioned within at most 30, respectively 80 time units to one of its three available transfer nodes. In all cases, two depots with 30 heterogeneous PD vehicles each are considered.

The planning horizon is set to 600 time units (e.g., 10 working hours if 1 time unit = 1 minute). The widths of the time windows are randomly generated between 26 and 91 time units. Service times are considered to be up to three time units. Each demand is assigned between one and three units. The capacity of PD vehicle is generated between six and 20 units. The carrying capacity on the considered SLs is assumed to be 15 demand units. The datasets (i.e., R, C, and RC) can be found on the web page `www.smartlogisticslab.nl`.

4.3.2 Parameter tuning

The proposed algorithm is implemented in NetBeans IDE 7.1.1 using Java. All experiments were conducted on an Intel Core i5 with 2.6 GHz speed and 4 GB RAM. A more detailed tuning was done on the five most sensitive parameters as shown in Table 4.1. These were identified during some preliminary tests. Three instances of small sizes (i.e., 10, 15 and 25 requests) were solved 10 times each for the given combination of parameters. The values for the rest of the parameters were inspired from Røpke and Pisinger (2006); Demir et al. (2012), as these proved good performance on related routing problems.

Table 4.1 Results of parameter tuning

Destroy rate %	σ_1	σ_2	σ_3	Average $\overline{z}$ 10K iterations	Average $\overline{z}$ 15K iterations
15	1	0	1	1,087.82	1,087.82
15	1	0	5	1,096.85	1,096.84
15	1	3	1	1,113.41	1,113.41
15	1	3	5	1,089.38	1,089.38
15	5	0	1	1,103.03	1,099.89
15	5	0	5	1,093.48	1,093.48
15	5	3	1	1,084.83	1,081.69
15	5	3	5	1,098.52	1,098.52
25	1	0	1	1,072.20	1,072.20
25	1	0	5	1,072.20	1,072.20
25	1	3	1	1,076.08	1,076.08
25	**1**	**3**	**5**	**1,072.19**	**1,072.19**
25	5	0	1	1,073.16	1,073.16
25	5	0	5	1,072.20	1,072.20
25	5	3	1	1,072.20	1,072.20
25	5	3	5	1,078.07	1,078.07
35	1	0	1	1,078.10	1,078.10
35	1	0	5	1,081.02	1,078.04
35	1	3	1	1,077.61	1,077.61
35	1	3	5	1,077.89	1,077.89
35	5	0	1	1,093.95	1,093.95
35	5	0	5	1,079.30	1,079.30
35	5	3	1	1,077.99	1,077.99
35	5	3	5	1,080.99	1,080.99
Average				1,083.44	1,083.05

In total, the ALNS algorithm involves 16 parameters which are listed in Table 4.2.

Similarly to previous chapters, a driving cost of the PD vehicles is assumed to be 0.5€ per minute. It seems reasonable considering all operational costs of a medium-size van, such as fuel consumption, driver wage, insurance, and tax. The cost of each demand unit of package request shipped on a SL is set to 1€, which includes handling, storage and transportation costs. The parameters used in the ALNS algorithm are grouped into two categories below.

Table 4.2 Used parameters in the computational experiments

Group	Parameter	Value
I	The total number of iterations (N_i)	10,000
	The number of route-wheel iterations (N_w)	200
	The roulette-wheel constant (r_p)	0.1
	New best solution (σ_1)	1
	Improving solution (σ_2)	3
	Deteriorating solution (σ_3)	5
II	Initial temperature (T)	200
	Cooling degree (κ)	0.9995
	Lower limit of removable requests ($\underline{\phi}$)	2.5% of $\lvert\mathcal{P}\rvert$
	Upper limit of removable requests ($\overline{\phi}$)	25% of $\lvert\mathcal{P}\rvert$
	First Shaw parameter (Π_1)	0.5
	Second Shaw parameter (Π_2)	0.2
	Third Shaw parameter (Π_3)	0.1
	Fourth Shaw parameter (Π_4)	0.2
	Noise parameter (μ)	0.1
	Number of feasible insertions (ψ)	30

- Group I contains the parameters related to the selection procedure of the operators. The values of σ_1, σ_2 and σ_3 do not satisfy $\sigma_1 \geq \sigma_2 \geq \sigma_3$. In our implementation and similar to Demir et al. (2012, 2013), we have chosen (based on parameter tuning) a parameter setting which rewards more acceptance of a worse solutions than discovering of better ones. It turns out to be an effective way for diversifying the search.
- Group II parameters are used to fine-tune the SA master search and the operators. We use 10,000 iterations (and not 15,000) for the master search, as the average performance difference is insignificant, compared to extra time needed to run 5,000 iterations more (see Table 4.1).

Table 4.3 and Table 4.4 indicate the number of times each operator was used within the ALNS. In addition, these tables show in the parenthesis the total time spent to run each operator. We note that the results are obtained using only one instance of each different size in terms of the number of requests.

The results shown in Table 4.3 show that operators RR, SR, PR and HR are used almost equally often. In many cases, the most widely used operators are LAR, TR and WR.

Table 4.3 Number of iterations and the computational times needed by each removal operator

Instance	RR	ROR	LAR	WDR	SR
R_25_1_8	986(0.0)	800(0.0)	**1,358(0.1)**	766(0.0)	979(0.0)
R_50_1_14	946(0.0)	947(0.0)	900(0.0)	951(0.0)	**1,051(0.0)**
R_75_1_20	995(0.0)	893(0.0)	**1,104(0.0)**	943(0.1)	958(0.1)
R_100_1_24	1,024(0.0)	640(0.0)	**1,323(0.1)**	**1,294(0.1)**	907(0.1)
	PR	DR	TR	HR	WR
R_25_1_8	938(0.0)	900(0.0)	**1,121(0.0)**	904(0.0)	**1,248(0.0)**
R_50_1_14	963(0.0)	826(0.0)	**1,241(0.0)**	946(0.0)	**1,229(0.0)**
R_75_1_20	933(0.1)	769(0.0)	**1,196(0.1)**	936(0.1)	**1,273(0.0)**
R_100_1_24	926(0.1)	743(0.1)	**1,205(0.1)**	1,096(0.1)	842(0.1)

Table 4.4 Number of iterations and the computational times needed by each insertion operator

Instance	GI	SI	GIN	SIN	λFI
R_25_1_8	860(0.2)	855(0.2)	537(0.1)	510(0.2)	451(0.0)
R_50_1_14	655(0.7)	631(0.5)	312(0.1)	414(0.4)	453(0.1)
R_75_1_20	636(1.4)	532(0.9)	404(0.6)	263(0.5)	286(0.0)
R_100_1_24	531(4.2)	485(2.1)	212(1.3)	252(1.1)	273(0.2)
	oGI	oSI	oGIN	oSIN	oλFI
R_25_1_8	**1,843(0.9)**	**1,945(1.2)**	1,024(0.6)	**1,045(0.6)**	930(0.3)
R_50_1_14	**2,359(11.2)**	**2,332(10.8)**	**971(4.5)**	941(4.5)	932(1.1)
R_75_1_20	**2,697(44.9)**	**2,643(42.5)**	**1,040(13.5)**	832(13.0)	667(1.5)
R_100_1_24	**3,315(285.0)**	**3,095(263.8)**	**875(56.7)**	510(42.5)	452(4.7)

As seen in Table 4.4, most used insertion operators are the ones that have ordered $\mathcal{L}$. Since least flexible requests are inserted first, these operators have higher chances to insert all requests within the existing routes. More specifically, oGI, oSI and their variants with noise functions are widely selected. It is noted that randomly-ordered insertion operators do not perform well, as many times the algorithm cannot insert all unassigned requests back to the solution due to their order of insertion.

Tables 4.5–4.6 indicate the number of times an operator has found the best and a better solution compared to the current one, respectively. It is noted that the number in parenthesis indicates the number of times a current solution is improved, but not to become a best known.

Table 4.5 Number of global best solutions found and number of improving solutions achieved by the removal operators

Instance	RR	ROR	LAR	WDR	SR	PR	DR	TR	HR	WR
R_25_1_8	1(61)	0(47)	1(29)	0(4)	0(77)	0(43)	**2(48)**	**4(97)**	1(60)	**2(152)**
R_50_1_14	**24(179)**	2(111)	**6(205)**	3(68)	4(188)	**5(137)**	1(72)	5(257)	1(56)	1(154)
R_75_1_20	**14(195)**	6(94)	3(208)	1(154)	**7(235)**	6(178)	4(42)	6(290)	**9(189)**	4(168)
R_100_1_24	**17(226)**	**9(73)**	**9(292)**	4(107)	7(228)	7(196)	2(32)	4(306)	8(163)	7(256)

The results indicate that all removal operators, to some extent, contribute to

Table 4.6 Number of global best solutions found and number of improving solutions achieved by the insertion operators

Instance	GI	SI	GIN	SIN	λFI	oGI	oSI	oGIN	oSIN	oλFI
R_25_1_8	1(7)	1(7)	0(3)	0(2)	0(0)	**3(253)**	**4(210)**	**2(73)**	0(63)	0(0)
R_50_1_14	**18(36)**	0(10)	0(3)	0(1)	0(0)	**19(755)**	**15(575)**	0(25)	0(22)	0(0)
R_75_1_20	**8(21)**	0(5)	0(0)	0(2)	0(0)	**28(1069)**	**23(628)**	1(15)	0(13)	0(0)
R_100_1_24	**12(27)**	0(6)	0(1)	0(0)	0(0)	**32(1211)**	**30(630)**	0(2)	0(2)	0(0)

achieving improved solutions. On the other hand, some insertion operators (i.e., λFI and oλFI) do not improve a current solution at any time. However, as it will be shown below, these operators are needed to diversify the search and achieve better overall performance of the algorithm. Furthermore, as expected, greedy operators (i.e., best and second-best) help obtaining improved solutions.

In order to identify the usefulness of the new insertion operators proposed in this article, we tested four configurations of the ALNS. These are shown in Table 4.7, along with the average objective function values over ten runs of the algorithm. We used the same instances as in Table 4.1.

Table 4.7 A tuning of insertion operators

Configuration	Average $\overline{z}$
Without SI and SIN	949.61
Without λFI and oλFI	952.15
Without SI, SIN, λFI and oλFI	956.13
With all operators	**948.25**

According to the obtained results, using SI, SIN and λFI operators leads to the best performance of the algorithm. These operators are mainly destined to diversify the search. As seen in Table 4.7, it seems imperative to use such operators along with the unconventional scoring setting. More specifically, as can be seen in Figure 4.3, λFI operators are mainly used in the initial iterations of the algorithm. Thus, it can be concluded that diversification in initial phase of the run is crucial to obtain good quality solutions.

4.3.3 Results of the proposed algorithm on the PDPTWs

In this section, we provide in Tables 4.8–4.10 computational results on the PDPTW benchmark instances (i.e., Li and Lim (2003)), which come in three sets: R, C and RC classified with respect to the positioning of the customers (i.e., random, clustered and randomly-clustered). The reason for choosing these instances is that Røpke and Cordeau (2009) and Baldacci et al. (2011a) provided their results by using the minimization of operating costs as our algorithm does. Tables 4.8–4.10 compare published results to the ones obtained by our ALNS heuristic algorithm.

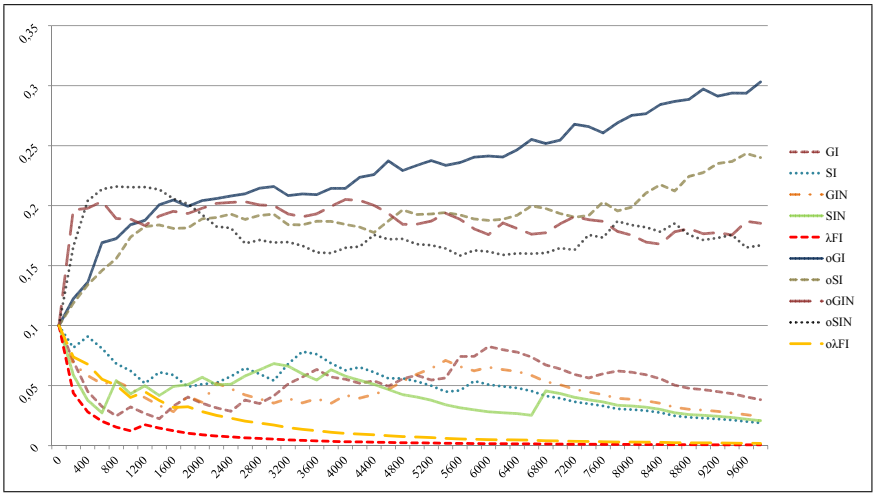

Figure 4.3 Insertion operator probabilities over the run of the algorithm

The values of the best known (or even optimal) solutions obtained from (Røpke and Cordeau, 2009; Baldacci et al., 2011a; Røpke and Pisinger, 2006) are given in under column "Best known value". Note that the values in bold emphasize that the proposed algorithm found the best known solution. The symbol "*" indicates that the values are not necessary optimal and are obtained by Røpke and Pisinger (2006). Moreover, we note that all figures presented in these tables use a single decimal digit. For the ALNS algorithm, we then present the best solution value obtained in column "ALNS best found". In addition, we indicate "ALNS average value" after 10 runs with the corresponding average GAP (%) (i.e., let $v(ALNS)$ be the value of the solution found by the proposed ALNS, then, the GAP (%) $= 100\,(v(ALNS) - v(\text{Best}))\ / v(ALNS)$, where v(Best) is the value of the best solution known). Also, we indicate the worst solution found after 10 runs of the algorithm in "ALNS worst found". Finally, we show the average CPU time required to solve the instances using the algorithm.

Table 4.8 Results of the proposed algorithm on benchmark PDPTW-C instances

Li and Lim (2003) instances	# of requests	Best known $\overline{z}$	Time (s)	ALNS best $\overline{z}$	ALNS average $\overline{z}$	ALNS worst $\overline{z}$	GAP %	Time (s)
LC1_2_1	106	2,704.6	3	**2,704.6**	**2,704.6**	**2,704.6**	0.00	47
LC1_2_2	105	2,764.6	22	**2,764.6**	2,764.8	2,766.9	0.00	51
LC1_2_3	103	2,772.2	115	**2,772.2**	2,772.3	2,774.3	0.00	121
LC1_2_4	105	2, 661.4*	209	**2,661.4**	2,663.4	2,664.6	0.07	123
LC1_2_5	107	2,702.0	5	**2,702.0**	**2,702.0**	2,702.0	0.00	40
LC1_2_6	107	2,701.0	7	**2,701.0**	**2,701.0**	2,701.0	0.00	42
LC1_2_7	107	2,701.0	8	**2,701.0**	**2,701.0**	2,701.0	0.00	38
LC1_2_8	105	2,689.8	16	**2,689.8**	**2,689.8**	2,689.8	0.00	43
LC1_2_9	105	2,724.2	55	**2,724.2**	2,738.1	2,750.0	0.51	63
LC1_2_10	104	2,741.6	137	2,743.9	2,763.8	2,803.3	0.80	67
Average			58				0.14	66

Table 4.9 Results of the ALNS heuristic on benchmark PDPTW-RC instances

Li and Lim (2003) instances	# of requests	Best known $\overline{z}$	Time (s)	ALNS best $\overline{z}$	ALNS average $\overline{z}$	ALNS worst $\overline{z}$	GAP %	Time (s)
LRC1_2_1	106	3,606.1	3	**3,606.1**	3,607.9	3,612.0	0.05	45
LRC1_2_2	103	3,292.4	322	**3,292.4**	3,293.9	3,299.3	0.05	67
LRC1_2_3	105	3, 079.5*	183	3,108.5	3,121.2	3,136.2	1.34	107
LRC1_2_4	106	2, 525.8*	284	2,552.1	2,583.2	2,605.1	2.22	190
LRC1_2_5	107	3,715.8	42	3,715.8	3,738.0	3,771.9	0.59	55
LRC1_2_6	105	3,360.9	7	3,382.4	3,401.0	3,411.2	1.18	39
LRC1_2_7	106	3,317.7	408	3,344.3	3,377.1	3,424.9	1.76	52
LRC1_2_8	104	3,086.5	1,563	3,129.5	3,143.7	3,199.0	1.82	63
LRC1_2_9	104	3,053.8	1,757	3,073.8	3,099.4	3,139.7	1.47	54
LRC1_2_10	105	2, 837.5*	156	2,857.2	2,871.7	2,881.5	0.51	51
Average			473				1.10	72

Table 4.10 Results of the ALNS heuristic on benchmark PDPTW-R instances

Li and Lim (2003) instances	# of requests	Best known $\overline{z}$	Time (s)	ALNS best $\overline{z}$	ALNS average $\overline{z}$	ALNS worst $\overline{z}$	GAP %	Time (s)
LR1_2_1	105	4,819.1	2	**4,819.1**	**4,819.1**	4,819.1	0.00	42
LR1_2_2	105	4,093.1	21	**4,093.1**	4,101.4	4,129.0	0.20	96
LR1_2_3	104	3,486.8	3,691	**3,486.8**	3,494.2	3,505.5	0.21	162
LR1_2_4	105	2, 830.7*	228	2,839.1	2,858.1	2,875.9	0.95	201
LR1_2_5	106	4,221.6	3	**4,221.6**	4,239.2	2,263.4	0.42	42
LR1_2_6	107	3,763.0	181	**3,763.0**	3,769.9	3,829.6	0.18	70
LR1_2_7	103	3, 112.9*	173	**3,112.9**	3,115.7	3,119.1	0.09	107
LR1_2_8	103	2, 645.4*	226	2,652.4	2,664.7	2,726.3	0.72	159
LR1_2_9	105	3,953.5	15	**3,953.5**	3,961.0	3,976.1	0.19	52
LR1_2_10	104	3,386.3	1,377	3,390.4	3,411.2	3,445.1	0.73	62
Average			592				0.37	99

As shown in Table 4.8–4.10, the ALNS heuristic proved good performance on the PDPTW instances. For 18 out of 30 instances, our heuristic algorithm obtained the best known solutions in at least one out of the 10 runs. For the remaining 12 instances, the GAPs are found to be not greater than 2.79%. The average CPU time needed to solve the instances is found to be 78 seconds.

4.3.4 Results of the generated instances

In this section, we provide the results on the four generated sets of PDPTW-SL instances. For the first group (with up to 60 requests and one SL), each instance was solved 10 times with the proposed heuristic and once with the B&P algorithm proposed in Chapter 3 with an imposed time-limit of three hours. The following

three groups were solved 10 times using the proposed ALNS in the context of PDPTW-SL and PDPTW. The detailed results of these experiments are presented in Tables 4.11–4.14.

Table 4.11 Computational results for the instances with up to 60 requests

Instance	B&P			ALNS		
	$\underline{z}$	$\overline{z}$	Time (s)	$\overline{z}$	Time (s)	GAP %
C6_2_4	369.36	369.36	0	369.36	1	0.00
C7_2_4	390.31	390.31	0	390.31	1	0.00
C8_2_4	446.35	446.35	4	446.35	2	0.00
C9_2_4	471.16	471.16	140	472.68	2	0.32
C10_2_4	510.01	510.01	67	510.01	2	0.00
C11_2_4	522.85	522.85	9	522.84	2	0.00
C12_2_4	541.60	541.60	4	541.60	2	0.00
C15_2_6	649.93	649.93	13	659.30	3	1.42
C20_2_8	751.23	751.23	83	751.23	4	0.00
C25_2_8	879.94	879.94	2,114	879.94	8	0.00
C40_2_8	1,199.31	–	10,800	1,226.77	8	–
RC6_2_4	572.76	572.76	0	572.75	1	0.00
RC7_2_4	575.95	575.95	0	575.95	1	0.00
RC8_2_4	585.32	585.32	0	585.32	1	0.00
RC9_2_4	593.70	593.70	0	593.69	2	0.00
RC10_2_4	599.95	599.95	0	599.94	2	0.00
RC11_2_4	624.48	624.48	1	624.47	2	0.00
RC12_2_6	662.03	662.03	1	662.03	2	0.00
RC15_2_6	1,068.16	1,068.16	1	1,068.16	1	0.00
RC20_2_8	1,200.36	1,200.36	6	1,203.98	2	0.30
RC25_2_8	1,530.43	1,530.43	28	1,530.43	2	0.00
RC40_2_12	1,969.45	1,969.45	738	1,988.36	6	0.95
RC50_2_14	2,200.88	2,200.88	1,235	2,214.09	10	0.60
RC60_2_16	2,636.18	–	10,800	2,646.34	13	–
R6_2_4	416.16	416.16	0	416.16	1	0.00
R7_2_4	473.06	473.06	1	473.05	1	0.00
R8_2_4	558.17	558.17	0	558.17	1	0.00
R9_2_4	632.42	632.42	2	632.41	1	0.00
R10_2_4	636.05	636.05	1	636.05	2	0.00
R11_2_4	748.29	748.29	1	748.29	2	0.00
R12_2_6	913.68	913.68	2	934.73	2	2.30
R15_2_6	1,083.50	1,083.50	15	1,083.50	1	0.00
R20_2_8	1,542.93	1,542.93	12	1,542.93	1	0.00
R25_2_8	1,636.12	1,636.12	50	1,638.42	1	0.14
R40_2_12	1,969.77	1,969.77	895	1,969.77	5	0.00
R50_2_14	2,595.14	2,595.14	580	2,595.14	7	0.00
R60_2_16	2,858.69	–	10,800	2,890.34	12	–
Average			172		2	0.17

Table 4.11 indicates that in most of the cases, the ALNS algorithm generated optimal solutions, but in a significantly shorter CPU times. For some instances, sub-optimal solutions were obtained, the optimality GAP not exceeding 2.3%. For the instances solved to optimality, the average CPU time required by B&P is approximately 172 seconds whereas ALNS needed approximately 2 seconds.

Tables 4.12–4.14 provide the results obtained for larger instances. The column

Instance indicates the instance identification. The *Best found* $\overline{z}$ column indicates the best objective value found after 10 runs of the algorithm. In addition, columns *Average* $\overline{z}$ and *Average GAP* show the average objective values over 10 runs and respectively the average relative GAP from the best solution found (i.e., GAP = $100(\mathrm{c}(ALNS_{average}) - \mathrm{c}(ALNS_{best}))/\mathrm{c}(ALNS_{average})$). The *Cost savings* column indicates the cost savings of the best PDPTW-SL solution compared to the best corresponding PDPTW solution. The *Best driving time* column shows the total driving time of the best solution found and the *Driving time savings* indicate the savings with regard to the total driving time. *Time* indicates the average computational time for solving the instances. *#Vehs* and *#SLs* provide the number of PD vehicles used and the number of shipments (demand units) on the available SLs in the best solution found.

Table 4.12 Results of *C* instances

Instance	Best found $\overline{z}$	Average $\overline{z}$	Average GAP %	Cost savings %	Best driving time	Driving time savings %	Time (s)	#Vehs	#SLs
C25_0_8	1,076.70	1,083.40	0.62		2,153.40		1	6	
C25_1_8	879.94	882.63	0.30	18.27	1,737.88	19.30	7	5	11
C25_2_8	878.86	881.26	0.27	18.37	1,723.72	19.95	8	7	17
C25_3_8	753.03	754.33	0.17	30.06	1,468.06	31.83	14	5	19
C_50_0_12	1,529.37	1,541.56	0.79		3,058.73		4	8	
C_50_1_12	1,415.60	1,425.67	0.71	7.44	2,793.19	8.68	124	10	14
C_50_2_12	1,342.23	1,356.01	1.02	12.24	2,634.47	13.87	121	10	50
C_50_3_12	1,214.16	1,229.30	1.23	20.61	2,354.31	23.03	144	10	74
C_75_0_16	2,040.54	2,069.10	1.38		4,081.09		12	10	
C_75_1_16	1,807.33	1,829.14	1.19	11.43	3,558.66	12.80	411	12	56
C_75_2_16	1,686.85	1,702.30	0.91	17.33	3,283.70	19.54	528	12	90
C_75_3_16	1,621.18	1,641.60	1.24	20.55	3,112.36	23.74	650	13	130
C_100_0_20	2,349.46	2,378.79	1.23		4,698.91		34	12	
C_100_1_20	2,112.73	2,126.42	0.64	10.08	4,167.47	11.31	2378	13	58
C_100_2_20	2,040.36	2,069.71	1.42	13.16	3,964.72	15.62	2224	14	116
C_100_3_20	1,915.40	1,939.87	1.26	18.47	3,662.80	22.05	2357	14	168

Table 4.13 Results of *RC* instances

Instance	Best found $\overline{z}$	Average $\overline{z}$	Average GAP %	Cost savings %	Best driving time	Driving time savings %	Time (s)	#Vehs	#SLs
RC_25_0_8	1,572.08	1,572.08	0.00		3,144.16		1	7	
RC_25_1_8	1,530.43	1,530.43	0.00	2.65	3,048.86	3.03	3	8	6
RC_25_2_8	1,413.06	1,414.44	0.10	10.12	2,808.12	10.69	4	7	9
RC_25_3_8	1,377.54	1,382.58	0.36	12.37	2,727.08	13.27	6	7	14
RC50_0_14	2,270.83	2,270.83	0.00		4,541.66		3	10	
RC50_1_14	2,213.06	2,213.54	0.02	2.54	4,416.12	2.76	11	10	5
RC50_2_14	2,213.06	2,213.85	0.04	2.54	4,416.12	2.76	12	10	5
RC50_3_14	2,211.89	2,214.13	0.10	2.60	4,409.78	2.90	15	10	7
RC_75_0_16	2,863.05	2,868.34	0.18		5,726.11		9	12	
RC_75_1_16	2,814.39	2,825.63	0.40	1.70	5,616.78	1.91	143	12	12
RC_75_2_16	2,814.39	2,836.39	0.78	1.70	5,616.78	1.91	155	12	12
RC_75_3_16	2,814.39	2,839.05	0.87	1.70	5,616.78	1.91	181	12	12
RC_100_0_18	3,114.35	3,119.10	0.15		6,228.70		25	12	
RC_100_1_18	3,088.07	3,093.66	0.18	0.84	6,164.13	1.04	794	12	12
RC_100_2_18	3,088.07	3,100.22	0.39	0.84	6,164.13	1.04	677	12	12
RC_100_3_18	3,088.07	3,134.30	1.48	0.84	6,164.13	1.04	985	12	12

The proposed algorithm is relatively fast. For example, instances of up to 100 requests are solved in less than 40 minutes. As it can be noticed from the Tables 4.12–4.14, the ALNS algorithm for the PDPTW-SL is substantially

Table 4.14 Results of R instances

Instance	Best found $\overline{z}$	Average $\overline{z}$	Average GAP %	Cost savings %	Best driving time	Driving time savings %	Time (s)	#Vehs	#SLs
R_25_0_8	1,717.39	1,724.71	0.42		3,449.42		1	8	
R_25_1_8	1,636.12	1,637.72	0.10	4.73	3,266.24	5.31	2	8	3
R_25_2_8	1,597.75	1,601.97	0.26	6.97	3,185.50	7.65	2	8	5
R_25_3_8	1,540.96	1,545.94	0.32	10.27	3,069.92	11.00	3	8	6
R_50_0_14	2,620.88	2,649.70	1.09		5,241.76		2	13	
R_50_1_14	2,595.14	2,604.2	0.35	0.98	5,186.28	1.06	8	12	2
R_50_2_14	2,412.89	2,420.67	0.32	7.94	4,803.78	8.36	20	12	11
R_50_3_14	2,392.81	2,403.52	0.45	8.70	4,759.62	9.20	21	12	13
R_75_0_20	3,337.85	3,337.85	0.00		6,675.70		7	14	
R_75_1_20	3,337.85	3,355.41	0.52	0.00	6,675.70	0.00	69	14	0
R_75_2_20	3,321.14	3,361.77	1.21	0.50	6,630.28	0.68	98	15	12
R_75_3_20	3,321.14	3,349.07	0.83	0.50	6,630.28	0.68	110	15	12
R_100_0_24	3,643.07	3,646.89	0.10		7,286.14		26	15	
R_100_1_24	3,643.07	3,665.31	0.61	0.00	7,286.14	0.00	538	15	0
R_100_2_24	3,628.87	3,646.65	0.49	0.39	7,245.74	0.55	690	16	12
R_100_3_24	3,628.87	3,637.37	0.23	0.39	7,245.74	0.55	725	16	12

slower than the same algorithm in case of the PDPTW. The main reason is the extra complexity that is induced by having the flexibility of using available scheduled lines, thus making multiple PD-vehicle routes depend on each other. In particular synchronization constraints (i.e., time windows and capacity) require extra computation time (e.g., 20–30% of the CPU time). Due to fact that changes in scheduling in one route may affect other routes as well, it might not be enough to investigate just the route(s) in which a request is inserted, as other requests may have to be transferred in the considered routes.

Overall, the results indicate the potential operating costs due to available scheduled lines. In particular, the savings range from 0 to 30% with regard to operating costs and from 0 to 31% in terms of driving time. Note that in this study the amount of CO_2e emissions is directly proportional to total driving time as we disregard the extra emissions produced by the SLs due to extra carried weight (i.e., packages).

Most of the savings can be achieved by shipping requests on the available SLs. However, the number of PD vehicles used slightly increases in PDPTW-SL compared to the solutions for PDPTW, especially in C instances. It is explained by the fact that the number of vehicles used depends on the time windows, capacities, and demands. Moreover, we note that savings decrease along with the increase in the number of requests for R and RC instances. This can be explained by the increasing density of the requests over the considered area (200×200 time units). Hence, driving time from one demand node to another becomes shorter. For C instances the savings remain significant for larger instances as well. Hence, we can conclude that the more demand points are clustered around transfer nodes, the better performance of the system is. An obvious point that is supported by the results is that the more SLs are available, more savings can be achieved compared to the classical PDPTW.

4.3.5 The effect of heterogeneous routing costs on the algorithm performance

In this section, we show the effectiveness of the proposed ALNS on instances with heterogeneous vehicle-routing costs. The vehicle routing costs are based on the vehicle capacity. The minimum-capacity vehicle is assumed to cost 0.5€ per operating time unit. Larger vehicles are assigned a cost that increases linearly along with the carrying capacity. Each instance is run ten times and the results are given in Table 4.15. The columns are self-explanatory, similar to previously presented tables.

Table 4.15 An analysis of heterogeneous vehicle routing costs

Instance	Best $\overline{z}$	Average $\overline{z}$	Average GAP %	Cost savings %	Time (s)	#Vehs	#SLs
25_C_0_8	1,676.53	1,693.02	0.97		2	6	
25_C_1_8	1,333.99	1,354.51	1.51	20.43	6	6	13
25_RC_0_8	2,513.39	2,540.65	1.07		1	7	
25_RC_1_8	2,403.87	2,421.96	0.75	4.36	4	8	6
25_R_0_8	2,145.41	2,171.02	1.18		2	8	
25_R_1_8	2,019.78	2,035.34	0.76	5.86	2	8	1
50_C_0_12	1,623.09	1,635.45	0.76		3	8	
50_C_1_12	1,459.64	1,489.81	2.03	10.07	37	9	17
50_RC_0_14	3,751.17	3,784.39	0.88		5	10	
50_RC_1_14	3,570.35	3,611.17	1.13	4.82	12	10	5
50_R_0_14	3,271.88	3,281.55	0.29		4	12	
50_R_1_14	3,231.22	3,258.57	0.84	1.24	7	13	6
75_C_0_16	2,278.57	2,309.17	1.33		8	11	
75_C_1_16	1,955.53	1,972.39	0.85	14.18	698	11	25
75_RC_0_16	3,164.14	3,183.45	0.61		7	12	
75_RC_1_16	3,144.66	3,155.30	0.34	0.62	122	12	6
75_R_0_20	3,583.39	3,603.99	0.57		5	14	
75_R_1_20	3,572.31	3,586.76	0.40	0.31	138	15	2
100_C_0_20	3,985.45	4,017.42	0.80		22	12	
100_C_1_20	3,557.39	3,576.59	0.54	10.74	2178	13	36
100_RC_0_18	3,420.74	3,461.23	1.17		9	13	
100_RC_1_18	3,344.28	3,362.95	0.56	2.24	644	12	6
100_R_0_24	3,758.27	3,792.44	0.90		13	15	
100_R_1_24	3,758.27	3,825.33	1.75	0.00	668	15	0

According to the obtained results, the proposed ALNS seems to perform stable when considering heterogeneous costs, leading to solutions with an average GAP of 0.91% compared to the best known solutions. In addition, the results indicate that objective function values tend to increase due to larger routing costs (heterogeneous costs) for the considered vehicles, compared to homogeneous case (i.e., 0.5 units for all vehicles). In this context, making use of available SLs leads to average cost savings of 6.24% compared to the corresponding best-known PDPTW solutions.

4.3.6 An application study

In this section we investigate the performance of the PDPTW-SL environment on a realistic scheduled lined system. In particular, we solve one instance of 100 randomly generated requests on a 60×60 time-units area, three depots and 20 PD vehicles. The scheduled lines graph is shown in Figure 4.4 and it is inspired from the current metro system in Amsterdam (see Figure 4.4). The distances are considered Euclidean and time windows are randomly generated.

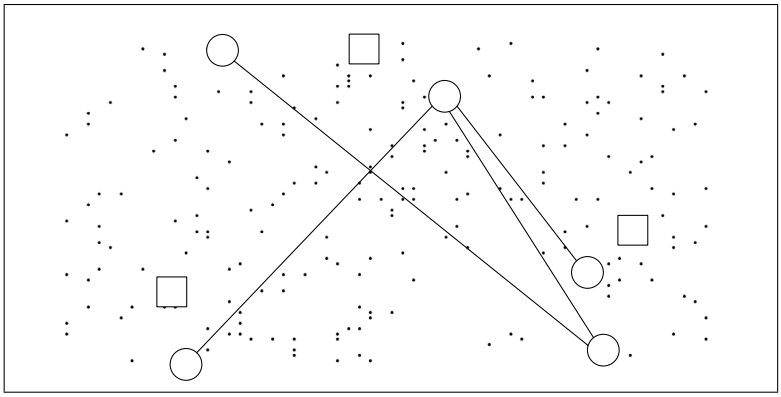

Figure 4.4 An illustrative representation of Amsterdam's metro system

Table 4.16 Results on a realistic real-life scheduled line system

Instance	PDPTW-SL					PDPTW			
	Driving time	#SLs	$\overline{z}$	Time (s)	#Vehs	Driving time	$\overline{z}$	Time (s)	#Vehs
Amsterdam 100 requests	1,767.22	36	919.61	1,258	10	1,945.69	972.85	34	7

The results shown in Table 4.16 are obtained after five runs of the algorithm and indicate the best solutions found for both PDPTW-SL and PDPTW. In particular, the PDPTW-SL integrated transportation system led to 5% savings in terms of operating costs and 9% in shorter total driving time. Even though PDPTW-SL system can lead to operating costs and CO_2e emissions benefits, the number of vehicles used is increased as compared to PDPTW.

4.4 Conclusions

We proposed a metaheuristic algorithm to solve the PDPTW-SL. To investigate the performance of the algorithm, we have used several sets of PDPTW-SL

instances. Compared to the existing solutions on a set of PDPTW instances, the proposed algorithm performed well in terms of both, solution quality (with a maximum GAP of 2.3%) and CPU time (78 seconds on average). Furthermore, we have also shown that small PDPTW-SL instances can be solved optimally by the proposed formulation. The solutions obtained from solving larger instances, up to 100 requests, were compared to their corresponding PDPTW solutions and it is concluded that the flexibility of using scheduled line services leads to significant cost savings and fewer CO_2e emissions. However, note that the performance of the PDPTW-SL system may be highly dependent on the relative positioning of the scheduled lines to the request nodes. In addition, investment costs needed for implementing such system may affect its outcome.

The reliability of such a system may decrease due to extra causes of delays and cargo damages (i.e., transfers to/from SLs, delays in SL schedules). Therefore, shippers may not be willing to use PDPTW-SL system. In order to tackle such issues, more advanced planning tools are needed, which consider stochastic aspects of the problem. The current research state of the considered problem is yet young and further industry collaborations are to be done in order to learn more about practical issues that may or may not prevent the implementation and the execution of such a system.

Overall the numerical experiments indicate that the proposed algorithm leads to high-quality routing solutions for relatively large instances in a reasonable amount of time. Regarding extensions of the investigated problem, additional aspects may be considered such as driver-related constraints, stochastic aspects (demands, travel times) or passenger requests, and the other related constraints (e.g., maximum ride-time). In addition, since the results show that the number of PD vehicles used may increase when SLs are introduced, it would be interesting to introduce fixed costs for the PD vehicles as well to make more clear the trade-off between using fewer PD vehicles or using SLs with more PD vehicles.

Chapter 5

A scenario-based approach for the pickup and delivery problem with time windows, scheduled lines and stochastic demands

5.1 Introduction

This chapter investigates the *Pickup and Delivery Problem with Time Windows, Scheduled Lines and Stochastic Demands* (PDPTW-SLSD), in which a set of geographically-spread freight requests need to be transported to their destinations using a fleet of heterogeneous pickup and delivery (PD) vehicles (i.e., a priori optimization). Moreover, capacitated SL services can be used as a part of requests' journey without affecting passenger service level. This characteristic of the PDPTW-SLSD implies that a request can be served in two ways, direct delivery or transferred through SLs. Furthermore, the exact quantities demanded by each customer are only learned upon vehicles' arrival at the corresponding pickup locations. Depending on the demand realizations, there are two possible violation outcomes: *(i)* PD vehicle may arrive at a pickup location without enough carrying capacity, and *(ii)* SL service capacity might not be sufficient for the realized demand. In such cases, called *route failures*, corrective (or recourse)

actions need to be applied in order to recover feasibility. These actions obviously lead to extra costs which can be charged by LSPs or freight carriers. Note that this problem considers routing on a tactical level, meaning that generated routes will be executed on a regular basis for a planning horizon (e.g., one month). In addition, pickup and delivery nodes may represent e.g., zip codes, as gathering historical data for each customer may be inacurate due to low individual demands.

In order to consider demand uncertainty, a *Sample Average Approximation* (SAA) method along with an adaptive large neighborhood search (ALNS) algorithm is proposed. The SAA is a scenario-based framework to deal with stochastic optimization problems (Kleywegt et al., 2001). The basic steps of the SAA are as follows: *(i)* solving the SAA problem for a given restricted set of scenarios (i.e., a subset of a larger set of scenarios), *(ii)* evaluating the found solution on a larger set of scenarios and approximating the expected value function by the sample average function, and *(iii)* iterating (*step (i)–(ii)*) until the defined stopping criterion is reached. We note that exact as well as heuristic algorithms (i.e., ALNS) can be used to solve the SAA problem in *step (i)*. We chose SAA combined the ALNS for two main reasons. First, the complexity of the problem, namely the deterministic PDPTW-SL is difficult enough to be able to optimally solve medium-size instances in short computational time. Therefore, the SAA problem for the PDPTW-SLSD is solved using an ALNS algorithm to keep it tractable. Finally, implementation simplicity of the SAA is another reason.

The scientific contribution of this study is three-fold: (*i*) we propose a solution approach for the integrated short-haul transportation problem in a stochastic environment, (*ii*) we quantify the benefits of the integrated system in stochastic setting, and (*iii*) we present the effect of considering uncertainty in the planning process. The remainder of this chapter is organized as follows. Section 5.2 provides a brief literature review on related topics. The problem description and the mathematical formulation are discussed in Section 5.3. The solution approach is given in Section 5.4. Section 5.5 presents the results of extensive computational experiments. Conclusions are stated in Section 5.6.

5.2 Literature review

In this section, we briefly present recent works on stochastic PDPs.

5.2.1 Stochastic PDPs

Over the years, there has been an increasing interest for the VRP with stochastic demands (VRPSD) (see, e.g., Laporte et al., 2002; Verweij et al., 2003; Secomandi and Margot, 2009). The most advanced solution approaches can be categorized into two groups according to the use of stochastic techniques. The most commonly used approach is *stochastic programming with recourse*, which is a well-known framework for modeling optimization problems that involve uncertainty. Since some data is unknown at the moment of planning, one decision is made and expected (or recourse) costs of the consequences of the plan are minimized. The second most-used approach and the one related to this chapter is the *multi-scenario approach* (or Monte Carlo simulation). This method is a common way to estimate expected costs when a closed-form expression is not available. Therefore, an approximation is made by evaluating a solution on a set of generated scenarios. In order to implement a multi-scenario stochastic optimization approach, metaheuristic algorithms can be efficiently engaged with sampling approaches. The interested readers are referred to Bianchi et al. (2009) and Gutjahr (2011) for metaheuristic algorithms proposed in the domain of stochastic combinatorial optimization.

To the best of our knowledge, limited literature exists on PD-related problems (e.g., one-to-one PDP, DARP) with stochastic demands. One of the first works was studied by Powell et al. (1988), who considered the dynamic PDP with stochastic demands in the context of long haul freight transportation. The computational results indicated that substantial profits can be achieved by considering uncertainty in the planning process while increasing the service level compared with the previous deterministic planning approach. Other examples are the single vehicle PDP (or 1-TSP) with stochastic demands by Louveaux and Gonzalez (2009), the DARP with stochastic customers' delays by Heilporn et al. (2011), and the stochastic DARP with expected return transports by Schilde et al. (2011).

5.3 Problem description and mathematical formulation

The PDPTW-SLSD is an extension of the classical PDPTW (see, e.g., Dumas et al., 1991; Nanry and Barnes, 2000; Lu and Dessouky, 2006). The PDPTW-SLSD consists of routing and scheduling a set of vehicles to transport a set of geographically-spread requests from corresponding origins to their destinations. In addition, a set of limited-capacity scheduled line vehicles operates according

to pre-defined routes and schedules. Each request may use any of the available SL services as a part of its journey. In other words, a request may be collected by one PD vehicle, transferred to a scheduled line and afterwards delivered to its final destination after being re-collected by another PD vehicle. Furthermore, each request has to be served within its corresponding time windows.

It is assumed that demand of each request is unknown by the time of planning. The exact quantities demanded by each request are only learned upon the vehicle's arrival at the corresponding pickup locations. However, the demand of each request is assumed to follow a known probability distribution.

Due to aforementioned source of uncertainty, capacity constraints might be violated at any time for a given *a priori* route. More specifically, two types of capacity violations might occur. In case *(i)*, at an origin or a transfer node, there might be an insufficient capacity on a PD vehicle to perform the pickup operation. In this case, it is assumed that the to-be-picked-up request will be served by an outsourced service at a certain cost, which is dependent on the direct distance from the point of failure to its destination. In case *(ii)*, there might be an insufficient capacity on a SL service. In this case, the excessive demand can be transported by using the subsequent service of the SL. We also assume that demand cannot be split in this recourse action. If time windows are violated or the capacity of the subsequent SL service is exceeded, the service is outsourced similarly to case *(i)*. Below we introduce the notations and assumptions used throughout the chapter.

- *Request:* Every request is associated with an origin node $r \in \mathcal{P}$ and a destination node $r + n \in \mathcal{D}$, where $n = |\mathcal{P}|$ (the number of requests to be served). Each request has one time window for the pickup node ($[l_r, u_r]$), and one time window for the delivery node ($[l_{r+n}, u_{r+n}]$). Furthermore, each freight request has demand quantity d_r that is defined with a probability distribution function. We assume that each demand unit corresponds to a package of a standardized small size (disregarding the nature of its content, e.g., dry or frozen, and its weight). In addition, we refer to a request by its corresponding pickup node.
- *Vehicle:* The set of PD vehicles is denoted by $\mathcal{V}$. In addition, each vehicle $v \in \mathcal{V}$ is associated with a capacity Q_v (i.e., maximum number of packages), the depot $o_v \in \mathcal{O}$, where $\mathcal{O}$ is the set of depots, and its corresponding time window $[l_{o_v}, u_{o_v}]$. Finally, each vehicle is assigned a routing cost per one time unit θ_v.
- *Travel and service time:* Travel and service times are known beforehand and remain unchanged during the planning horizon. Each arc (i, j) is associated with travel time Υ_{ij}. The service time at node i is represented by s_i.
- *Scheduled line:* The set of all physical transfer nodes is given as $\mathcal{S}$. The set of all physical scheduled lines is denoted by $\mathcal{E}$, which is defined by a

directed arc between the start and the end of the line (i, j), where $i, j \in \mathcal{S}$. For example, between two transfer nodes i and j, two scheduled lines with opposite directions are considered as (i, j) and (j, i). $\mathcal{K}^{ij}$ represents a set of indices for the departure times from origin node i of SL (i, j), such that the departure time is denoted by p^w_{ij} (e.g., $p^0_{T_1,T_2} = 30$ units). Furthermore, the capacity of the SLs is denoted by Q_{ij}, $\forall\ (i, j) \in \mathcal{E}$. Note that scheduled lines are considered to be heterogeneous in terms of frequency and capacity. Shipping one unit of package on SL (i, j) costs η_{ij}.

Additional practical assumptions related to SLs are given as follows.

- Every SL vehicle is assumed to have a separate compartment for carrying packages, thus the passenger capacity is not influenced.
- A storage space (DHL-Packstation, 2016) for to-be-shipped packages is always available at transfer nodes. We also assume that transfers are only allowed only at the end-of-line stations in order to avoid long delays (i.e., loading/unloading) at the intermediate stops. Furthermore, we assume that multiple PD vehicles can be serviced at a time at every transfer node.
- In case of multiple freight carriers using SLs, it is assumed that each carrier has a limited storage space at a transfer node and on SLs (e.g., contract-based agreement).
- Cost η_{ij} includes transportation, handling (transshipment) and storage costs.
- Since the focus of the present problem is on routing decisions, investment costs, such as the re-design of the SL vehicles and the storage capacities at the transfer nodes are disregarded.

For modeling purposes, every physical SL is assigned n copies as in Häll et al. (2009). Figure 5.1.a illustrates an example with one physical SL (i.e., arcs (1, 2) and (2, 1)) and two requests. Each replication is assigned to one request, and only that specific request can travel on the assigned SL (see Figure 5.1.b). This is done to reduce the number of decision variables in the formulation. Therefore, a set of all replicated scheduled lines is denoted by $\mathcal{F}$ (e.g., $(1a, 2a)$, $(1b, 2b)$, $(2a, 1a)$ and $(2b, 1b)$ in Figure 5.1.b). The set of replicated SLs associated with request r is given as $\mathcal{F}^r$ (e.g., in Figure 5.1, $\mathcal{F}^a = \{(1a, 2a), (2a, 1a)\}$). In addition, the set of replicated SLs related to the replicated transfer node t is given as $\mathcal{F}^t$ (e.g., $\mathcal{F}^{1a} = \{(1a, 2a), (2a, 1a)\}$ in Figure 5.1). Finally, $\mathcal{F}^{ij}$ includes all replicated SLs associated with a physical SL $(i, j) \in \mathcal{E}$ (e.g., $\mathcal{F}^{1,2} = \{(1a, 2a), (1b, 2b)\}$, $\mathcal{F}^{2,1} = \{(2a, 1a), (2b, 1b)\}$).

Furthermore, each transfer node in $\mathcal{S}$ (e.g., nodes 1 and 2 in Figure 5.1.a) is copied n times. Hence, $\mathcal{T}$ represents the set of all replicated transfer nodes (e.g.,

$\mathcal{T} \equiv \{1a, 1b, 2a, 2b\}$ in Figure 5.1.b). For the sake of modeling efficiency, we use ψ^t, $\forall\ t \in \mathcal{T}$ as the physical transfer node represented by the replicated transfer node t (e.g., $\psi^{1a} = \psi^{1b} = 1$). Therefore, set $\mathcal{T}^t$ is $\{i \in \mathcal{T} |\ \psi^i = \psi^t\ and\ i \neq t\}$, $\forall\ t \in \mathcal{T}$ (e.g., $\mathcal{T}^{2a} = \{2b\}$ in Figure 5.1.b). Finally, let f_i^r be a constant taking value 1 if node i is the pickup node of request r, value –1 if it is the delivery node of r, or value 0 if it is none of them (pickup nor delivery node of r).

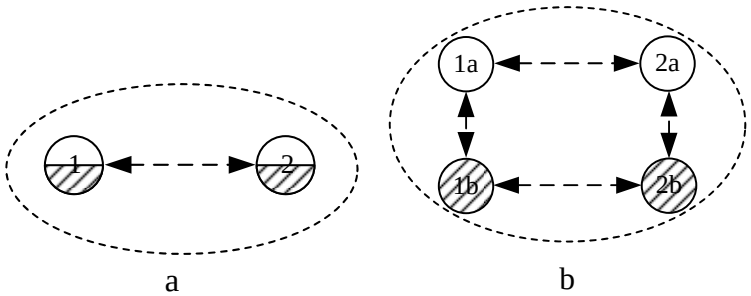

Figure 5.1 Physical and virtual scheduled lines

The proposed formulation is not limited to use of only one scheduled line. Any complex network topology (i.e., square, triangle, star, etc.) might be considered, and the requests are allowed to be transferred from one SL to another.

Finally, the PDPTW-SLSD can be defined on a digraph $\mathcal{G} = (\mathcal{N}, \mathcal{A})$, where $\mathcal{N}$ is the set of nodes (i.e, $\mathcal{O} \cup \mathcal{P} \cup \mathcal{D} \cup \mathcal{T}$) and the set of arcs $\mathcal{A} \equiv \mathcal{A}_1 \cup \mathcal{A}_2 \cup \mathcal{A}_3 \cup \mathcal{A}_4$ defined as follows:

$$\mathcal{A}_1 = ((\mathcal{P} \cup \mathcal{D})) \times (\mathcal{P} \cup \mathcal{D}))\backslash\{(r+n,\ r):\ r \in \mathcal{P}\}$$
$$\mathcal{A}_2 = \{(i,\ j):\ i \in \mathcal{O},\ j \in \mathcal{P}\} \cup \{(i,\ j):\ i \in \mathcal{D},\ j \in \mathcal{O}\} \cup (\mathcal{O} \times \mathcal{T})$$
$$\mathcal{A}_3 = ((\mathcal{P} \cup \mathcal{D})) \times \mathcal{T})\backslash(\{(j,\ r):\ r \in \mathcal{P},\ j \in \mathcal{T}^r\} \cup \{(r+n,\ j):\ r \in \mathcal{P},\ j \in \mathcal{T}^r\})$$
$$\mathcal{A}_4 = \{(i,\ j):\ i,\ j \in \mathcal{T},\ (\psi^i,\ \psi^j) \notin \mathcal{E}\}.$$

These sets of arcs consider all feasible connections in terms of precedence constraints. Figure 5.2 presents a small example of a network with one depot, two requests and one SL service along with all possible arcs. Sets $\mathcal{A}_1$ and $\mathcal{A}_2$ are shown in Figure 5.2.a and sets $\mathcal{A}_3$ and $\mathcal{A}_4$ are presented in Figure 5.2.b.

The binary variable x_{ij}^v equals to 1 if arc $(i,\ j)$ is used by PD vehicle v, 0 otherwise, $\forall\ (i,\ j) \in \mathcal{A}$, $v \in \mathcal{V}$. Variable y_{ij}^r takes value 1 if arc $(i,\ j)$ is used by request r, 0 otherwise, $\forall\ i,\ j \in \mathcal{P} \cup \mathcal{D} \cup \mathcal{T}$, $r \in \mathcal{P}$. In addition, q_{ij}^{rw} takes value 1 if replicated SL $(i,\ j)$ is used by request r that departs from i at time p_{ij}^w, 0 otherwise, $\forall\ r \in \mathcal{P}$, $(i,\ j) \in \mathcal{F}^r$, $w \in \mathcal{K}^{\psi^i \psi^j}$. Timing decisions are assured by: α_v

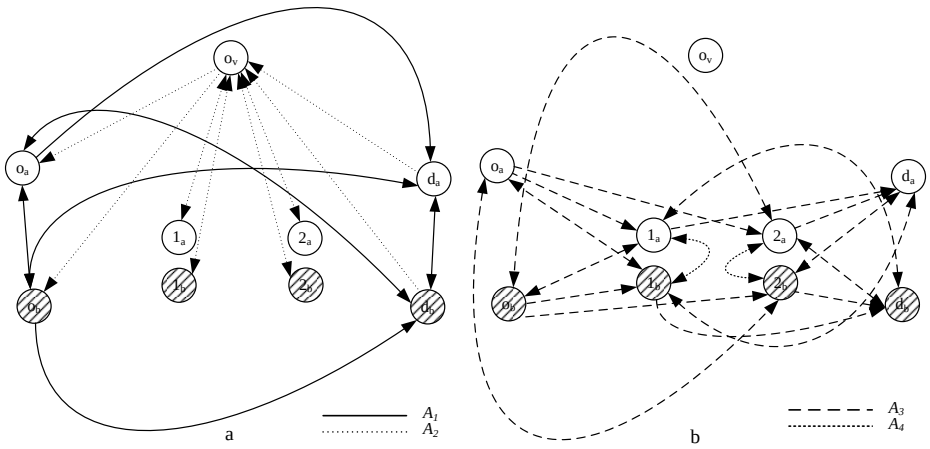

Figure 5.2 An example network with two requests and one SL service

that indicates the time at which vehicle v arrived back to the depot, $\forall\ v \in \mathcal{V}$, by β_i that shows the time of departure of a PD vehicle from node i, $\forall\ i \in \mathcal{N}$, and finally, γ_i^r that indicates the time at which request r departed from node i, $\forall\ i \in \mathcal{P} \cup \mathcal{D} \cup \mathcal{T}$, $r \in \mathcal{P}$.

The two-stage stochastic PDPTW-SLSD formulation is shown in the following two subsections.

5.3.1 The first-stage model

$$Minimize \quad \sum\nolimits_{(i,j)\in\mathcal{A}} \sum\nolimits_{v\in\mathcal{V}} \theta_v \Upsilon_{ij} x_{ij}^v \tag{5.1}$$

$$+ E[Q(\boldsymbol{x}, \xi, \eta)] \tag{5.2}$$

The objective function minimizes the total operating costs incurred by PD vehicles (5.1) and the *recourse* costs (5.2). In the recourse function, $\boldsymbol{x}$ is the given routing vector, ξ is the set of scenarios, and η is the cost vector for using SLs per unit shipped.

subject to

Routing and flow constraints

$$\sum_{i\in\mathcal{N}}\sum_{v\in\mathcal{V}} x_{ij}^{v} = 1 \qquad \forall j \in \mathcal{P}\cup\mathcal{D} \quad (5.3)$$

$$\sum_{i\in\mathcal{P}\cup\mathcal{D}\cup\mathcal{T}} x_{o_v,i}^{v} \leq 1 \qquad \forall v \in \mathcal{V} \quad (5.4)$$

$$\sum_{i\in\mathcal{N}}\sum_{v\in\mathcal{V}} x_{it}^{v} \leq 1 \qquad \forall t \in \mathcal{T} \quad (5.5)$$

$$\sum_{j\in\mathcal{N}} x_{ij}^{v} - \sum_{j\in\mathcal{N}} x_{ji}^{v} = 0 \qquad \forall i \in \mathcal{N}, v \in \mathcal{V} \quad (5.6)$$

$$\sum_{j\in\mathcal{P}\cup\mathcal{D}\cup\mathcal{T}} y_{ij}^{r} - \sum_{j\in\mathcal{P}\cup\mathcal{D}\cup\mathcal{T}} y_{ji}^{r} = f_i^r \qquad \forall r \in \mathcal{P}, i \in \mathcal{P}\cup\mathcal{D}\cup\mathcal{T} \quad (5.7)$$

$$\sum_{i\in\mathcal{N}}\sum_{v\in\mathcal{V}} x_{it}^{v} \leq \sum_{r\in\mathcal{P}}\sum_{(i,j)\in\mathcal{F}^t} y_{ij}^{r} \qquad \forall t \in \mathcal{T} \quad (5.8)$$

Constraints (5.3) state that all pickup and delivery nodes are visited exactly once. Constraints (5.4) ensure that each vehicle leaves its depot at most once and constraints (5.5) state that each replicated transfer node is visited at most once. Flow conservations for PD vehicles and requests are considered in constraints (5.6) and (5.7), respectively. Constraints (5.8) ensure that if a request uses a scheduled line, a PD vehicle should pick up or drop off the request at the corresponding transfer node.

Scheduling constraints

$$y_{ij}^{r} = 1 \implies \gamma_j^r \geq \gamma_i^r + \Upsilon_{ij} + s_j \qquad \forall r \in \mathcal{P}, i, j \in \mathcal{P}\cup\mathcal{D}\cup\mathcal{T} \quad (5.9)$$

$$\sum_{v\in\mathcal{V}} x_{ij}^{v} = 1 \implies \beta_j \geq \beta_i + \Upsilon_{ij} + s_j \qquad \forall i \in \mathcal{N}, j \in \mathcal{P}\cup\mathcal{D}\cup\mathcal{T} \quad (5.10)$$

$$x_{i,o_v}^{v} = 1 \implies \alpha_v \geq \beta_i + \Upsilon_{i,o_v} + s_{o_v} \qquad \forall i \in \mathcal{P}\cup\mathcal{D}\cup\mathcal{T}, v \in \mathcal{V} \quad (5.11)$$

$$\beta_{r+n} \geq \beta_r + \Upsilon_{r,r+n} + s_{r+n} \qquad \forall r \in \mathcal{P} \quad (5.12)$$

$$l_i \leq \beta_i - s_i \leq u_i \qquad \forall i \in \mathcal{P}\cup\mathcal{D} \quad (5.13)$$

$$l_{g_v} \leq \alpha_v \leq u_{o_v} \qquad \forall v \in \mathcal{V} \quad (5.14)$$

$$\sum_{w\in\mathcal{K}^{\psi^i\psi^j}} q_{ij}^{rw} = y_{ij}^{r} \qquad \forall r \in \mathcal{P}, (i,j) \in \mathcal{F}^r \quad (5.15)$$

$$q_{ij}^{rw} = 1 \quad and \quad y_{ij}^{r} = 1 \implies \gamma_i^r = p_{ij}^{w} \qquad \forall r \in \mathcal{P}, (i,j) \in \mathcal{F}^r, w \in \mathcal{K}^{\psi^i\psi^j} \quad (5.16)$$

Timing variables of each request are considered in constraints (5.9). Similarly, schedules for PD vehicles are considered in constraints (5.10) and (5.11). Constraints (5.12) define the precedence relations for each request. Constraints (5.13) and (5.14) enforce time windows restrictions. Constraints (5.15)–(5.16) state that if a request uses a scheduled line, SL vehicle must only depart at its scheduled

departure time.

Synchronization constraints

$$\sum\nolimits_{j\in\mathcal{P}\cup\mathcal{D}} y_{ij}^r = 1 \implies \gamma_i^r = \beta_i \qquad \forall r \in \mathcal{P}, i \in \mathcal{T} \quad (5.17)$$

$$\sum\nolimits_{j\in\mathcal{P}\cup\mathcal{D}\cup\mathcal{T}} y_{ij}^r = 1 \implies \gamma_i^r = \beta_i \qquad \forall r \in \mathcal{P}, i \in \mathcal{P}\cup\mathcal{D} \quad (5.18)$$

$$\sum\nolimits_{i\in\mathcal{P}\cup\mathcal{D}\cup\mathcal{T}} y_{i,r+n}^r = 1 \implies \gamma_{r+n}^r = \beta_{r+n} \qquad \forall r \in \mathcal{P} \quad (5.19)$$

$$y_{tj}^r = 1 \implies \gamma_t^r = \beta_t \qquad \forall r \in \mathcal{P}, t \in \mathcal{T}, j \in \mathcal{T}^t \quad (5.20)$$

The set of constraints (5.17)–(5.20) ensure the synchronization between requests' and vehicles' schedules.

Decision variable domains

$$x_{ij}^v \in \{0,1\} \qquad \forall (i,j) \in \mathcal{A}, v \in \mathcal{V} \quad (5.21)$$

$$y_{ij}^r \in \{0,1\} \qquad \forall i,j \in \mathcal{P}\cup\mathcal{D}\cup\mathcal{T}, r \in \mathcal{P} \quad (5.22)$$

$$\alpha_v \in R^+ \qquad \forall v \in \mathcal{V} \quad (5.23)$$

$$\gamma_i^r \in R^+ \qquad \forall i \in \mathcal{P}\cup\mathcal{D}\cup\mathcal{T}, r \in \mathcal{P} \quad (5.24)$$

$$\beta_i \in R^+ \qquad \forall i \in \mathcal{N} \quad (5.25)$$

$$q_{ij}^{rw} \in \{0,1\} \qquad \forall r \in \mathcal{P}, (i,j) \in \mathcal{F}^r, w \in \mathcal{K}^{\psi^i\psi^j}. \quad (5.26)$$

We note that constraints (5.9)–(5.11), (5.16) and (5.17)–(5.20) are formulated as implications. Standard linearization techniques can be easily used to express them using one or two linear inequalities, as follows:

$$\gamma_j^r \geq \gamma_i^r + \Upsilon_{ij} + s_j - M_{ij}\left(1 - y_{ij}^r\right) \qquad \forall r \in \mathcal{P}, i,j \in \mathcal{P}\cup\mathcal{D}\cup\mathcal{T} \quad (5.27)$$

$$\beta_j \geq \beta_i + \Upsilon_{ij} + s_j - M_{ij}\left(1 - \sum\nolimits_{v\in\mathcal{V}} x_{ij}^v\right) \qquad \forall i \in \mathcal{N}, j \in \mathcal{P}\cup\mathcal{D}\cup\mathcal{T} \quad (5.28)$$

$$\alpha_v \geq \beta_i + \Upsilon_{i,o_v} + s_{o_v} - M_{i,o_v}\left(1 - x_{i,o_v}^v\right) \qquad \forall i \in \mathcal{P}\cup\mathcal{D}\cup\mathcal{T}, v \in \mathcal{V} \quad (5.29)$$

$$\gamma_i^r \leq p_{ij}^w + M_i\left(2 - q_{ij}^{rw} - y_{ij}^r\right) \qquad \forall r \in \mathcal{P}, (i,j) \in \mathcal{F}^r, w \in \mathcal{K}^{ij} \quad (5.30)$$

$$\gamma_i^r \geq p_{ij}^w - M_i\left(2 - q_{ij}^{rw} - y_{ij}^r\right) \qquad \forall r \in \mathcal{P}, (i,j) \in \mathcal{F}^r, w \in \mathcal{K}^{ij} \quad (5.31)$$

$$\gamma_i^r \leq \beta_i + M_i\left(1 - \sum\nolimits_{j\in\mathcal{P}\cup\mathcal{D}} y_{ij}^r\right) \qquad \forall r \in \mathcal{P}, i \in \mathcal{T} \quad (5.32)$$

$$\gamma_i^r \geq \beta_i - M_i\left(1 - \sum\nolimits_{j\in\mathcal{P}\cup\mathcal{D}} y_{ij}^r\right) \qquad \forall r \in \mathcal{P}, i \in \mathcal{T}. \quad (5.33)$$

Constraints (5.18)–(5.20) can be linearized in the same way as constraints (5.17) (i.e., (5.32) and (5.33)).

5.3.2 The second-stage decisions

Given an a priori routing solution vector $\boldsymbol{x}$, the expected costs $\mathrm{E}[\mathrm{Q}(\boldsymbol{x}, \xi, \eta)]$ can be computed in two steps. In Step 1, capacity violations of the PD vehicles are evaluated. As a consequence, some requests (including transferable ones) may not be served by the available PD vehicles due to given demand realizations and capacity restrictions. Hence, extra costs are required for outsourcing. Further in Step 2, for remaining set of requests that are supposed to be transferred, SLs' capacity violations are verified. Each step is explained in detail below.

Step 1: For PD vehicles, capacity violations might happen either *(i)* at the pickup node of a request r (e.g., p_1 in Figure 5.3), or *(ii)* at the transfer node of a request t^d, which is a destination transfer node (e.g., T_1^d in Figure 5.3). In either way, it is assumed that an outsourced vehicle (e.g., taxi) could deliver the request to its destination from the point of *failure* at a cost of $P_1 \Upsilon_{l,r+n}$, $\forall\ l \in \{p_1, T_1^d\}$. Moreover, the destination node of r does not need to be visited and the shortened route leads to reduced operational costs.

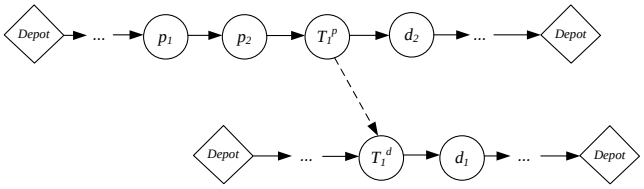

Figure 5.3 An illustrative routing example of the PDPTW-SLSD

Example. We now consider four possible situations: *(i)* let the PD vehicle's capacity be violated at node p_2 (see Figure 5.3). In this case, node d_2 will not be visited as the demand from the origin p_2 cannot be picked up by a PD vehicle. Hence, the transport service has to be outsourced. *(ii)* It is assumed that the vehicle's capacity is violated at node p_1. Hence, no related nodes to request 1 are visited: T_1^p, T_1^d, and d_1. *(iii)* Let the capacity be violated at node T_1^d. In this case, node d_1 cannot be visited. *(iv)* Let the demand of request 1 be zero. Similarly to aforementioned cases, T_1^p, T_1^d, d_1 cannot be visited. It is noted that, in cases *(i)–(iii)* the considered routes are shortened while incurring additional outsourcing costs, whereas case *(iv)* only leads to reduction in operational costs, due to lack of demand, thus no other related nodes need to be visited.

Step 2: As a result of the previous step, a restricted set of transferable requests $\overline{\mathcal{P}}^t$ is obtained. The capacity of SL service is only violated when the total demand of a set of transferable requests exceeds the capacity at a given scheduled departure time, i.e. $\sum_{r \in \overline{\mathcal{P}}^t} \sum_{(a,b) \in \mathcal{F}^{ij}} d_r q_{ab}^{rw} > k_{ij}\ \forall\ (i,\ j) \in \mathcal{E},\ w \in \mathcal{K}^{ij}$. In such case, the

excessive demand can be stored at the origin of the SL until the next scheduled departure. If waiting leads to violation of the time window constraints, or if the SL capacity on the next trip is exceeded, then the service is outsourced as in *Step 1* (from the origin of the SL to the demand's destination). The recourse cost of a given routing solution can be calculated as shown in Algorithm 7.

Algorithm 7: The generic structure of **Recourse**($\boldsymbol{x}$, ξ, η) procedure

input : A routing solution $\boldsymbol{x}$, a set of scenarios ξ
output: The average *recourse* cost $E[Q(\boldsymbol{x}, \xi, \eta)]$

1 Initialize $E[Q(\boldsymbol{x}, \xi, \eta)] \leftarrow 0$
2 Let c($\boldsymbol{x}$) be the routing costs of solution $\boldsymbol{x}$
3 **for** *scenario s in* ξ **do**
4 $\quad \mathbf{X} \leftarrow \boldsymbol{x}$
5 $\quad cost \leftarrow 0$
6 $\quad \overline{\mathcal{P}}^t \leftarrow$ set of transferable requests in $\boldsymbol{x}$
7 $\quad$ **for** *each node i in* $\boldsymbol{X}$ **do**
8 $\quad\quad r \leftarrow$ getRelatedRequest(i)
9 $\quad\quad$ **if** $i \in \mathcal{P}$ *and capacity violated* **then**
10 $\quad\quad\quad \mathbf{X} \leftarrow \mathbf{X} \setminus i + n$
11 $\quad\quad\quad \mathbf{X} \leftarrow \mathbf{X} \setminus (T_i^o \cup T_i^d)$
12 $\quad\quad\quad \overline{\mathcal{P}}^t \leftarrow \overline{\mathcal{P}}^t \setminus i$
13 $\quad\quad\quad cost \leftarrow cost + P_1 \Upsilon_{i,i+n}$
14 $\quad\quad$ **else if** *i is a destination transfer node and related request r is picked up and capacity violated* **then**
15 $\quad\quad\quad \mathbf{X} \leftarrow \mathbf{X} \setminus r + n$
16 $\quad\quad\quad cost \leftarrow cost + P_1 \Upsilon_{i,r+n}$
17 $\quad\quad$ **else if** $i \in \mathcal{P}$ *and* $d_i = 0$ **then**
18 $\quad\quad\quad \mathbf{X} \leftarrow \mathbf{X} \setminus i + n$
19 $\quad\quad\quad \mathbf{X} \leftarrow \mathbf{X} \setminus (T_i^o \cup T_i^d)$
20 $\quad\quad\quad \overline{\mathcal{P}}^t \leftarrow \overline{\mathcal{P}}^t \setminus i$
21 $\quad cost \leftarrow cost -$ c($\boldsymbol{x}$) + c($\mathbf{X}$)
22 $\quad cost \leftarrow cost + getSLCost(\overline{\mathcal{P}}^t, \eta)$
23 $\quad$ set $E[Q(\boldsymbol{x}, \xi, \eta)] \leftarrow E[Q(\boldsymbol{x}, \xi, \eta)] + \frac{cost}{|\xi|}$
24 return $E[Q(\boldsymbol{x}, \xi, \eta)]$

The *recourse* cost variable is initialized in Line 1. For each considered scenario (Line 3), a set of routes is also stored (Line 4) and a set of transferable requests from the given routing solution is selected (Line 6). Three cases of capacity violation of a PD vehicle may exist as explained earlier. The penalty cost for not serving a request is updated in Lines 13 and 16. Once the routes are shortened, routing cost savings are updated in Line 21, where c($\mathbf{X}$) is the routing cost of the shortened set of routes.

The extra costs incurred by exceeding SLs capacities is updated in $getSLCost(\mathcal{P}^t)$

(Line 22). More specifically, for each SL $(i,\ j) \in \mathcal{E},\ w \in \mathcal{K}^{ij}$ with violated SL capacity, the requests with the latest end time window of the corresponding delivery nodes (i.e., excess demand) are re-scheduled for the next trip on the same SL service. If time windows or SL capacity constraints are violated, the service is outsourced at an extra cost dependent on the distance to destination of the exceeded demand. As a result, the vehicle routes are shortened, i.e., the end-of-the-line station and the corresponding delivery node are not visited. Finally, line 23 updates the sample average approximation function and line 24 returns the expected *recourse* costs for a given routing solution $\boldsymbol{x}$ and a set of scenarios ξ.

5.3.3 An illustrative example

We now provide an example to show the necessity of using recourse actions during the transport planning. Assume an instance with two vehicles, six requests and two transfer nodes (i.e., T_1 and T_2). Nodes p_1,.., p_6 represent pickup locations and nodes d_1,.., d_6 represent the corresponding delivery locations. For simplicity reasons, assume the traveling time on each arc (including SLs) to be one time unit. Two different solutions with the same routing costs are shown in Figure 5.4. These solutions differ in terms of recourse costs. The numbers on top (or bottom) of each node, which are shown in italic, represent the departure times from the corresponding nodes and the numbers on top of dotted arrows indicate the request flows on the SLs.

We note that throughout the chapter we refer to *stochastic* solutions as the ones that are generated using the proposed methodology in this chapter, whose objective function is to minimize the routing and the recourse expected costs. On the other hand, *deterministic* solutions are generated using the methodology proposed in Chapter 4 (using average demands), which disregards the cost of recourse actions, thus minimizing routing and SL costs only.

In both cases, five requests, i.e., 1, 3, 4, 5 and 6, are being transported through SLs. However, the expected costs of these solutions differ. More specifically, the solution shown in Figure 5.4.a has a lower recourse cost as *Vehicle 1* first picks up requests 1, 3, and 4 and ships them on the SL service (i.e., T_1–T_2, see dashed arrows in Figure 5.4.a). Then, *Vehicle 1* picks up request 6 (which was picked up by *Vehicle 2* and shipped from T_2) from T_1, and finally requests 5 and 2 from their pickup locations. Request 5 is shipped on the SL from T_1. In this case, the total expected cost of this solution is found to be 385.40 units for a given set of scenarios.

On the other hand, in the deterministic solution (Figure 5.4.b), *Vehicle 1* first picks up requests 1, 3, 4 from their pickup locations and request 6 from station

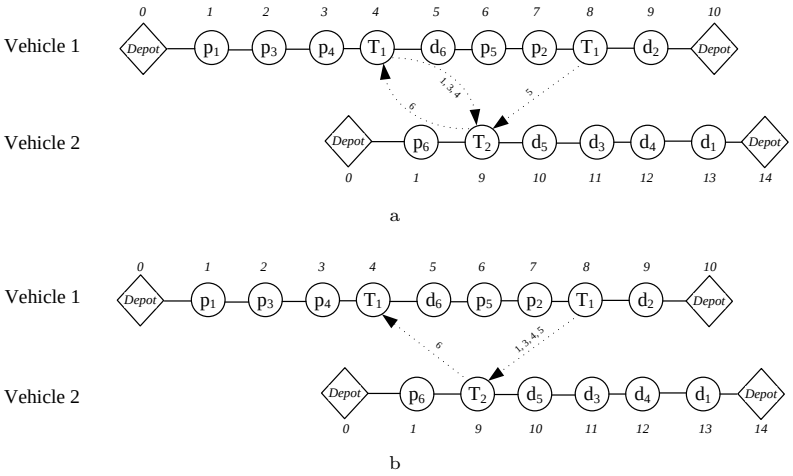

Figure 5.4 Two illustrative solutions with the same routing cost

T_1. Then the vehicle delivers request 6 to its destination node 13 and picks up requests 5 and 2 from their pickup locations to transfer them on the SL by visiting the transfer node T_1. This solution has a total expected cost of 462.48 units and the difference from the stochastic solution is triggered by the lack of available capacity at the pickup location of request 5. Therefore, one can observe that in addition to the routing decisions, a sequence of actions (i.e., departure times on the SLs) is also important when considering the expected cost of a transport plan.

5.4 The ALNS-based solution methodology for the PDPTW-SLSD

In this section, we provide a solution methodology for the PDPTW-SLSD. As aforementioned, our approach consists of a scenario-based SAA framework combined with an ALNS algorithm.

5.4.1 The sample average approximation method

The SAA is an iterative sampling approach proposed by Kleywegt et al. (2001) to solve stochastic discrete optimization problems, where the number of possible scenarios is very large. Given a large scenario set $\Omega^{'}$, which is intractable, the

SAA consists of solving a set of smaller and tractable problems (referred to as SAA problems) with samples of size $|\Omega| \ll |\Omega^{'}|$. In other words, instead of solving one difficult problem with huge number of scenarios ($\Omega^{'}$), the algorithm iteratively solves a series of problems with fewer scenarios (Ω). For the PDPTW-SLSD, a sample contains a number of scenarios, which represent demand vectors for the considered requests, given probability distributions. The SAA procedure used to calculate the approximate optimality gap (or SAA gap) for the PDPTW-SLSD is as follows:

1. The algorithm takes the following input: *(i)* a sample size $|\Omega|$ to generate the SAA solution (by solving the SAA problem), *(ii)* a larger set of scenarios $\Omega^{'}$ to approximate the value of the optimal solution of the SAA problem, and *(iii)* the number of iterations M.

2. For $m = 1 \rightarrow M$, do the following:

   2.1 Generate a sample set Ω, which includes $|\Omega|$ demand vectors (scenarios) for the requests. Solve the SAA problem by considering scenarios $\in \Omega$ to obtain the objective value v_{Ω}^{m} and the solution x^{m}, using the ALNS algorithm that was described in Chapter 4, tailored to consider demand uncertainty.

   2.2 Using Algorithm 7, determine the upper bound on the solution x^{m} for the sample $\Omega^{'}$, which is denoted by $v_{\Omega'}(x^{m})$. In other words, x^{m} is evaluated on $\Omega^{'}$ to obtain an estimation of the upper bound. Let $\hat{x}$ be the solution that has the best value of the upper bound $v_{\Omega'}(\hat{x})$ found after m iterations. The variance of this estimated upper bound can be computed as follows:

$$\sigma^{2}_{\Omega'}(\hat{x}) = \frac{1}{|\Omega'|(|\Omega'-1|)} \sum_{i \in \Omega'} \left(v_i(\hat{x}) - v_{\Omega'}(\hat{x})\right)^2, \tag{5.34}$$

   where $v_i(\hat{x})$ is the objective function value of the solution $\hat{x}$ for scenario i.

   2.3 Calculate the average of the objective function values obtained in previous iterations, denoted by $\hat{v}_{\Omega}^{m}$, and their corresponding variance as shown below.

$$\hat{v}_{\Omega}^{m} = \frac{1}{m} \sum_{i=1}^{m} v_{\Omega}^{i}, \tag{5.35}$$

$$\sigma^{2}_{\hat{v}_{\Omega}^{m}} = \frac{1}{m(m-1)} \sum_{i=1}^{m} \left(v_{\Omega}^{i} - \hat{v}_{\Omega}^{m}\right)^2. \tag{5.36}$$

   The value $\hat{v}_{\Omega}^{m}$ is a statistical lower bound for the optimal solution value of the sample $\Omega^{'}$ as given in Kleywegt et al. (2001).

2.4 Calculate the SAA gap $\epsilon^{SAA}(\Omega, \Omega^{'})$ and the variance of the gap as follows:

$$\epsilon^{SAA}(\Omega, \Omega^{'}) = v_{\Omega^{'}}(\hat{x}) - \hat{v}^{m}_{\Omega} \tag{5.37}$$

$$\sigma^{2}_{\epsilon^{SAA}(\Omega,\Omega^{'})} = \sigma^{2}_{\Omega^{'}}(\hat{x}) + \sigma^{2}_{\hat{v}^{m}_{\Omega}}. \tag{5.38}$$

The best value of $\epsilon^{SAA}(\Omega, \Omega^{'})$ and its corresponding $\sigma^{2}_{\epsilon^{SAA}(\Omega,\Omega^{'})}$ are stored during the process.

3. Return the best solution $\hat{x}$.

We tailored the ALNS described in Chapter 4 to consider demand uncertainty. In particular, most of the effort in tailoring the algorithm was evaluating the objective function. Apart from routing costs, the objective function contains also recourse costs, which are quantified using Algorithm 7. Therefore, the insertion operators presented in Chapter 4 were naturally adapted using Algorithm 7 to compute the total expected cost at every iteration. In addition, several removal operators were tailored to consider demand uncertainty, such as Shaw Removal (SR), Demand-Based Removal (DR), and Worst Expected Removal (WER). The first two operators (SR, DR), unlike the deterministic version, consider mean demand along with variance of the demand for each request. WER removes a number of requests with the highest total expected costs. The total expected cost is determined as follows: for a given routing solution, the cost of request r is the difference in the objective function values (i.e., routing and recourse costs) between two solutions, namely one where request r is served and the one were request r is not served (removed).

5.5 Computational experiments

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.

5.5.1 Data and experimental setting

Three sets of instances R, C, and RC were adapted from previous chapters to consider stochastic demands. Each instance contains up to three SLs in a triangular topology positioned in the central part of the 200×200 Euclidean space. The SLs have a frequency of one departure every 30 time units. Each instance contains up to 40 requests (i.e., up to 40 pick-up and 40 delivery nodes). Instances are named as $G\_n\_sl\_v$, where G is the geographic distribution of the customers,

n is the number of to-be-served requests, sl is the number of available SLs, and v is the number of available PD vehicles. By geographic distribution G we mean the following: C – clustered request nodes around transfer nodes, R – uniform-randomly distributed request nodes, and finally RC – randomly clustered nodes. More specifically, C has nodes positioned within at most 30 time units to one of the three available transfer nodes whereas RC has nodes within 80 time units respectively. In all cases, two depots with up to six PD vehicles each are considered.

The planning horizon is set to 600 time units. The time windows are randomly generated with an average width of 40 units. Service times are set to at most three time units. The request demands are assumed to follow a discrete triangular distribution for a given minimum value a, mean b and maximum value c. The capacity of each PD vehicle is generated between six and nine units. The carrying capacity on the considered SLs is assumed to be five demand units. The instances can be found on the web page of SmartLogisticsLab (2016).

The parameters used in the computational experiments are shown in Table 5.1. Heuristic-specific parameters have been assigned the same values as in Chapter 4, which were used to solve the deterministic PDPTW-SL. The settings used generally worked well on the test problems with uncertain demands. We assume a driving time unit cost for PD vehicles to be 0.5€. It seems quite reasonable when considering the driver's wage, fuel consumption, insurance, taxes, etc. Furthermore, the cost of using SL service is set to 1€ and it includes transportation, handling and storage costs. The recourse cost for outsourcing the service, which is considered per driving time unit, is assumed to be 3€.

Table 5.1 Parameters used in the methodology

Notation	Definition	Value	Notation	Definition	Value
θ_v	Cost of vehicle v	0.5	$\|\Omega'\|$	The size of the large set of scenarios	10,000
η_{ij}	Cost of SL (i, j)	1	$\|\Omega\|$	The size of the small set of scenarios	60
P_1	Cost for outsourcing	3	M	Number of SAA iterations	10
k_{ij}	Capacity of SL (i, j)	5	σ_1	Score for new best solution	1
	Number of ALNS interations	10,000	σ_2	Score for improving the current solution	3
λ	Number of feasible insertions (λFI)	30	σ_3	Score for accepting a worse solution	5

Kleywegt et al. (2001) 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. (2016) 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.

5.5.2 The performance of the operators

To show the number of times each removal and insertion operator was called within the ALNS, we give relevant information in Tables 5.2 and 5.3. These tables indicate the usage frequency as a percentage of the total number of iterations, and the total time spent (shown in parenthesis) of each operator, respectively. We note that the results are obtained using only three instances of different sizes.

Table 5.2 Number of iterations as a percentage of the total number iterations and the CPU times required by the removal operators

Instance	RR	ROR	LAR	WDR	SR
C10_1_4	7.89 (0.00)	**10.52 (0.00)**	**21.05 (0.01)**	5.26 (0.00)	5.26 (0.00)
RC15_1_6	3.44 (0.00)	6.89 (0.00)	1.72 (0.00)	**22.41 (0.01)**	8.62 (0.01)
R20_1_8	1.72 (0.00)	3.12 (0.00)	10.93 (0.01)	**12.49 (0.01)**	1.56 (0.00)
	PR	DR	TR	HR	WER
C10_1_4	5.26 (0.00)	7.89 (0.00)	5.26 (0.00)	**10.52 (0.00)**	**21.05 (0.01)**
RC15_1_6	6.89 (0.00)	5.17 (0.00)	8.62 (0.01)	**24.16 (0.01)**	**12.08 (0.01)**
R20_1_8	7.81 (0.01)	**12.49 (0.01)**	4.68 (0.00)	**14.06 (0.01)**	**31.14 (0.02)**

Table 5.2 shows that WDR, HR and WER are the most frequently used operators. The first operator is used to consider traveling distance aspect of the problem whereas the HR and WER operators consider the objective function value (including uncertainty aspect) of the PDPTW-SLSD.

Table 5.3 Number of iterations as a percentage of the total number iterations and the CPU times required by the insertion operators

Instance	GI	SI	GIN	SIN	λFI
C10_1_4	5.89 (0.89)	7.62 (1.15)	8.22 (1.24)	3.64 (0.92)	3.64 (0.55)
RC15_1_6	2.86 (0.36)	6.13 (0.78)	6.26 (0.80)	1.64 (0.58)	1.85 (0.23)
R20_1_8	7.89 (2.83)	7.29 (2.51)	9.38 (2.99)	1.34 (0.51)	3.03 (0.97)
	oGI	oSI	oGIN	oSIN	oλFI
C10_1_4	**22.07 (3.33)**	**17.25 (2.30)**	**17.11 (3.68)**	3.66 (0.90)	10.90 (1.94)
RC15_1_6	**26.29 (3.37)**	**22.24 (2.85)**	**19.04 (2.54)**	1.18 (0.57)	12.51 (1.86)
R20_1_8	**23.26 (7.43)**	**24.06 (7.68)**	**16.94 (5.41)**	1.27 (0.59)	5.54 (2.09)

Table 5.3 shows that the most frequently used insertion operators are oGI, oSI and oGIN. Ordered insertion operators tend to perform better than randomly-ordered ones. Furthermore, insertion operators with noise functions are used less than the operators without noise function.

5.5.3 Comparison between the PDPTW-SD and the PDPTW-SLSD

In this section, we identify the effects of using SLs on the total operational costs (including expected recourse costs). Table 5.4 shows the obtained results

for four different scenarios: *(i)* the classical PDPTW with stochastic demands, where no transfers are allowed, *(ii) – (iv)* the PDPTW-SLSD with one, two and three available SLs. The results indicate the average value over the 10 runs of the algorithm. Column *E(x)* indicates the total expected cost, *#Vehs* shows the number of vehicles used, and finally *#SLs* indicates the number of demand units (i.e., mean value of the given probability distribution of each request) that used SL services. Finally, the average CPU time needed to solve the PDPTW-SLSD is shown in the last column.

Table 5.4 Comparison between the PDPTW-SD and the PDPTW-SLSD with different number of SLs

Instance	sl=0		sl=1			sl=2			sl=3			CPU
	E(x)	#Vehs	E(x)	#Vehs	#SLs	E(x)	#Vehs	#SLs	E(x)	#Vehs	#SLs	(s)
C10_sl_4	561.69	3	541.68	4	4	527.73	4	8	435.95	4	12	216
C15_sl_6	721.31	4	660.46	4	9	615.50	4	17	576.25	6	21	786
C20_sl_8	890.36	6	805.54	6	6	805.35	6	6	741.84	6	10	1,387
C25_sl_8	1,192.01	6	991.44	8	13	986.36	7	13	857.38	7	22	2,105
C40_sl_8	1,609.99	8	1,516.60	8	8	1,515.41	8	8	1,349.39	8	18	3,351
RC10_sl_4	703.70	4	592.09	3	3	592.93	3	3	570.80	4	5	219
RC15_sl_6	1,149.93	6	1,149.45	6	0	1,141.57	6	1	1,116.85	6	1	368
RC20_sl_8	1,455.86	7	1,200.12	7	6	1,198.49	6	10	1,189.30	7	10	921
RC25_sl_8	1,665.57	8	1,660.77	8	6	1,513.65	8	13	1,469.06	8	14	1,446
RC40_sl_12	2,328.38	10	2,293.31	11	1	2,291.58	12	7	2,182.29	12	18	3,866
R10_sl_4	675.14	4	675.43	4	0	675.59	4	0	675.01	4	0	243
R15_sl_6	1,121.84	6	1,121.31	6	0	1,103.99	6	1	1,104.96	6	1	423
R20_sl_8	1,576.70	8	1,576.25	8	0	1,518.15	8	2	1,500.84	8	3	875
R25_sl_8	1,762.23	8	1,698.82	8	4	1,669.69	8	8	1,668.95	8	6	1,289
R40_sl_12	2,214.91	10	2,195.71	11	3	2,151.73	10	4	2,115.67	11	7	3,213
Average	1,308.64	6.5	1,245.27	6.8	4.2	1,220.52	6.7	6.7	1,170.30	7.0	9.9	1,381

According to the obtained results, the flexibility of using SLs as a part of transport plan leads to reasonable amount of operational cost savings. More specifically, on average, the total expected costs can be decreased by 8%. In some cases (e.g., C25_3_8) using SLs may even lead to 28% cost savings. However, in some cases (e.g., R10_1_4), the integrated networks do not lead to any benefits as SLs are not used. There may be several reasons for not using the scheduled lines. For example: time windows of the requests may not have sufficient slack time for using SLs, or the pickup and delivery nodes are close to the same transfer node. Figure 5.5 illustrates the average cost savings by considering all three scenarios (i.e., 1-, 2-, and 3-SLs) on three instance sets (C, RC, and R).

As can be observed from Table 5.4 and Figure 5.5, if more request nodes are clustered around the transfer nodes, larger operational cost savings can be achieved. More specifically, integrated networks yield larger benefits in C and RC instances (i.e., up to 28% and 20%, respectively), compared to R instances (i.e., up to 5%). In addition, the more SLs are available, the larger integration between the transportation networks is, hence larger cost savings can be achieved. In other words, all instance sets with one available SL lead to more costly solutions, compared to the 2-SLs and 3-SLs cases. It can be observed in Table 5.4 that the more SLs are available, the more freight units are served using public

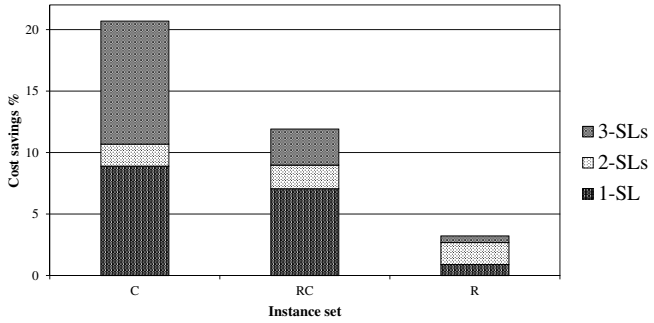

Figure 5.5 Average savings by using different number of SLs

transportation services, thus decreasing the expected costs.

5.5.4 SAA and ALNS solutions

This section presents the effects recourse actions (costs) have on the routing decisions during the planning process. We evaluate the best solutions found by the ALNS proposed in Chapter 4 (which disregards recourse costs and considers mean demands only), and by the proposed SAA considering demand uncertainty (i.e. recourse costs). Table 5.5 indicates the obtained results for the instances with one SL and homogeneous routing costs. Column $\overline{z}$ presents the operating costs of the best solution found by ALNS. Column $E(x)$ presents the total expected cost of the given solution (including recourse costs). In particular, $E(x)$ under ALNS presents the expected total operating cost of the best deterministic solutions found (solved by ALNS considering mean demands). $E(x)$ under SAA indicates the total expected cost of the solutions generated by SAA (considering recourse actions). Other columns are self-explanatory.

The solutions obtained using SAA proved three trends, compared to the solutions generated by ALNS: they contain slightly longer routes (i.e., on average 4% longer total driving times), the average number of vehicles used is slightly increased and, finally, the average number of requests being shipped using SLs is slightly decreased. In all cases, at least one of these trends is respected (see e.g. C20_1_8, where more vehicles are used and more requests are shipped on SLs). In terms of request ride time (i.e., time difference between arrival at the delivery node and its corresponding pickup node), the SAA solutions tend to have 0.12% longer total ride time overall requests, compared to ALNS solutions. This is explained by longer vehicle routes. Generally, certain buffers in terms of PD vehicle capacity usage are generated, thus slightly more PD vehicles are used in SAA solutions.

Table 5.5 Comparison between ALNS and SAA solutions

Instance	ALNS					SAA			
	$\bar{z}$	#Vehs	#SLs	Driving time	E(x)	E(x)	#Vehs	#SLs	Driving time
C10_1_4	510.01	4	4	1,012.03	541.68	541.68	4	4	1,012.03
C15_1_6	660.30	4	9	1,302.59	660.46	660.46	4	9	1,302.59
C20_1_8	751.23	5	2	1,498.46	884.91	805.54	6	6	1,581.97
C25_1_8	892.03	6	13	1,758.06	1,079.16	991.44	8	13	1,971.78
C40_1_8	1,255.99	8	8	2,487.98	1,516.60	1,516.60	8	8	2,487.98
RC10_1_4	599.95	3	3	1,193.90	616.45	592.09	3	3	1,225.46
RC15_1_6	1,068.16	6	0	2,136.33	1,149.45	1,149.45	6	0	2,136.33
RC20_1_8	1,218.91	7	6	2,425.82	1,269.75	1,200.12	7	6	2,430.63
RC25_1_8	1,530.43	8	6	3,048.87	1,845.94	1,660.77	8	6	3,152.49
RC40_1_12	2,015.16	10	1	4,028.31	3,032.51	2,293.31	11	1	4,331.47
R10_1_4	640.05	4	7	1,266.11	699.93	675.43	4	0	1,362.84
R15_1_6	1,083.50	6	0	2,167.00	1,121.31	1,121.31	6	0	2,167.00
R20_1_8	1,546.44	8	0	3,092.88	1,780.70	1,576.25	8	0	3,211.40
R25_1_8	1,641.42	8	4	3,274.83	1,776.01	1,698.82	8	4	3,450.06
R40_1_12	1,997.31	9	3	3,988.62	2,406.04	2,195.71	11	3	4,330.35
Average	1,160.73	6.4	4.4	2,312.12	1,358.73	1,245.27	6.8	4.2	2,410.29

Overall, on average 8.3% savings in terms of expected operating costs can be achieved, if uncertainty is taken into considerations during the planning.

To get a clearer view on the effects the uncertainty has, we present Figures 5.6 and 5.7. The figures illustrate two solutions obtained by solving $RC10\_3\_4$ instance: one using ALNS (with mean demands only), and another using SAA (considering demand uncertainty). In each graph, x-axis represents the sequence of vehicle visits. Note that p_x and d_x represent the pickup, and delivery nodes of request x, respectively. The spaces between the consecutive visits approximates the driving time between the two nodes. On the other hand, y-axis represents the capacity utilization of the vehicles. The capacity (Q) of each vehicle is represented by the horizontal dotted line. The upper (dashed) line in the graph represents the worst case scenario, considering largest demand realization for each request. In addition, the lower line illustrates the capacity utilization considering mean demands.

The SAA solution outperforms the one generated by the ALNS (i.e., considering mean demands) in terms of total expected costs, i.e., 570.78 and 617.01, respectively. However, the total traveling time increases from 1,105.91 to 1,130.01 (i.e., 2%) in the stochastic solution. However, the recourse costs of both solutions are different. In particular, the SAA solution leads to significant reductions in terms of recourse actions needed, namely from 64.05 to 5.78. In addition, in the SAA solution, fewer requests (i.e., five mean demand units) are shipped via available SLs, compared to the ALNS solution (i.e., six mean demand units). It implies that certain capacity slack is generated on the SLs, to avoid outsourcing in case of SL capacity violations.

Specifically, we note that ALNS solution is worse due to vehicle route number one (see Figure 5.6, upper graph). Namely, when visiting pickup node p_9, and considering maximum demand realizations, the vehicle's capacity would be

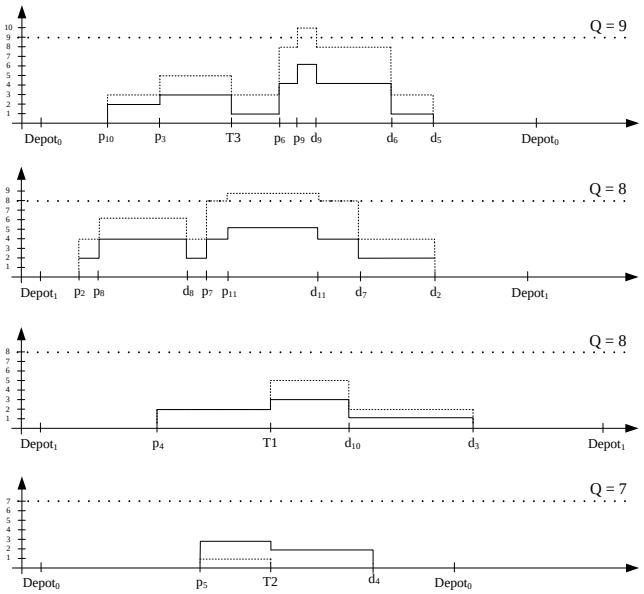

Figure 5.6 The solution obtained by ALNS, considering mean demands

exceeded by one unit. Hence, we could conclude that ALNS solution would experience capacity violations in more scenarios than the SAA solution.

Generally, the capacity buffers are generated by: *(i)* performing fewer consecutive pickups by the vehicles, thus leaving some slack capacity aboard, *(ii)* using more vehicles to execute the deliveries, and *(iii)* shipping fewer requests via available SLs.

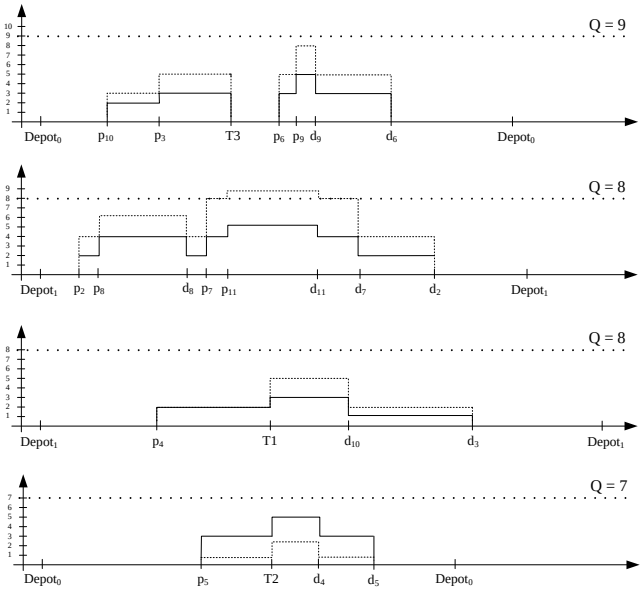

Figure 5.7 The solution obtained by SAA, considering demand uncertainty

5.5.5 The relationship between SL capacity and total expected costs

In this section, we investigate the effect of SLs capacity on the total expected costs. We test three scenarios with different SLs capacity: *five*, 10, 15 demand units. The average results out of ten runs of the algorithm are shown in Table 5.6.

Table 5.6 The effect of SLs capacity on the total expected operational costs

	$k_{ij} = 5$			$k_{ij} = 10$			$k_{ij} = 15$		
Instance	E(x)	#Vehs	#SLs	E(x)	#Vehs	#SLs	E(x)	#Vehs	#SLs
C10_1_4	541.68	4	4	530.73	4	5	530.49	4	5
C15_1_6	660.46	4	9	659.24	4	9	658.33	4	9
C20_1_8	805.54	6	6	805.79	6	6	804.77	6	6
C25_1_8	991.44	8	13	986.15	7	13	978.58	7	13
C40_1_8	1,516.60	8	8	1,513.27	8	8	1,513.42	8	8
RC10_1_4	592.09	3	3	591.54	3	3	591.36	3	3
RC15_1_6	1,149.45	6	0	1,149.64	6	0	1,149.92	6	0
RC20_1_8	1,200.12	7	6	1,199.56	7	6	1,199.41	7	6
RC25_1_8	1,660.77	8	6	1,654.11	8	6	1,653.81	8	6
RC40_1_12	2,293.31	11	1	2,292.11	10	3	2,290.96	10	3
R10_1_4	675.43	4	0	648.68	4	7	648.76	4	7
R15_1_6	1,121.31	6	0	1,120.37	6	0	1,120.25	6	0
R20_1_8	1,576.25	8	0	1,574.09	8	0	1,574.37	8	0
R25_1_8	1,698.82	8	4	1,698.37	8	4	1,698.19	8	4
R40_1_12	2,195.71	11	3	2,184.64	11	5	2,184.02	11	5
Average	1,245.27	6.8	4.2	1,240.55	6.7	5.0	1,239.78	6.7	5.0

On average, one can observe a trend, which indicates that larger SL service

capacity lead to fewer violations, hence fewer recourse actions needed to tackle SL service capacity violations. With an extra capacity, the PDPTW-SLSD solutions become cheaper (up to 3.94%) and slightly more freight units are shipped via SLs.

It is important to note that, in some cases (e.g., RC15_1_6), statistically insignificant difference in terms of total expected cost can be observed. This is explained by the fact that each solution is evaluated on different sets of independently generated scenarios for a given demand distribution.

5.5.6 The effect of demand variance on the solution costs

In this section, we investigate the impact of demand variance on the expected solution costs. We test four different cases; by decreasing the difference between the mean and maximum value of the triangular distribution by half (i.e., changing the maximum value), by increasing it by 1.5, and finally by 2. The results are shown in Table 5.7 and indicate the average results over ten runs of the algorithm.

Table 5.7 The effect of demand variance on the total expected costs

Instance	0.5		1		1.5		2	
	ALNS	SAA	ALNS	SAA	ALNS	SAA	ALNS	SAA
C10_1_4	506.00	492.71	541.68	541.68	631.04	592.21	828.89	751.92
C15_1_6	652.41	644.82	660.46	660.46	688.73	681.86	859.89	787.69
C20_1_8	726.57	730.95	884.91	805.54	1,183.85	1,014.77	1,644.75	1,238.84
C25_1_8	872.04	866.00	1,079.16	991.44	1,690.58	1,138.41	2,052.67	1,500.30
C40_1_8	1,267.50	1,232.17	1,516.60	1,516.60	2,043.25	1,849.88	3,243.78	2,553.26
RC10_1_4	622.60	591.28	616.45	592.09	643.36	626.18	838.85	749.46
RC15_1_6	1,056.45	1,056.45	1,149.64	1,149.45	1,398.13	1,323.03	1,867.67	1,682.03
RC20_1_8	1,270.33	1,199.94	1,269.75	1,200.12	1,555.76	1,420.24	2,113.75	1,829.17
RC25_1_8	1,562.91	1,519.00	1,845.94	1,660.77	2,196.52	1,888.83	2,683.43	2,317.58
RC40_1_12	2,217.55	2,003.29	3,032.51	2,293.31	3,917.60	2,825.61	4,808.62	3,557.54
R10_1_4	643.73	635.77	699.93	675.43	796.74	763.19	1,008.92	930.10
R15_1_6	1,191.31	1,042.84	1,121.31	1,121.31	1,573.25	1,339.52	1,789.42	1,619.84
R20_1_8	1,565.92	1,495.40	1,780.70	1,576.25	2,160.16	1,760.31	2,405.61	2,127.80
R25_1_8	1,655.81	1,655.59	1,776.01	1,698.82	1,992.74	1,770.75	2,412.15	2,141.38
R40_1_12	2,013.87	1,989.26	2,406.04	2,195.71	3,094.95	2,511.15	4,085.73	3,187.78
Average	1,188.33	1,143.70	1,358.73	1,245.27	1,704.44	1,433.73	2,176.28	1,798.31

As expected, deterministic solutions tend to perform poorer with the increase in demand variance. Furthermore, such trend is also observed for stochastic solutions, but to a smaller extent. The results seem to be intuitive since high variance might lead to more capacity violations. Along with the increase in demand variance, the solutions generated by SAA tend to contain more capacity usage buffers, as fewer consecutive pickups are performed and fewer requests are shipped using SLs.

5.5.7 The SLs frequency analysis

In this section, we quantify the effect of the SLs frequency on the total expected costs of the PDPTW-SLSD. We test three situations, namely one departure every

ten time units, the original case (i.e., every 30 time units) and, finally, one departure every 180 time units. The average over 10 runs of the algorithm is shown for each instance in Tables 5.8 and 5.9.

Table 5.8 The effect of the SLs frequency on the total expected costs

	7 units			30 units		
Instance	E(x)	#Vehs	#SLs	E(x)	#Vehs	#SLs
C10_1_4	511.96	4	8	541.68	4	4
C15_1_6	658.11	4	9	660.46	4	9
C20_1_8	805.53	6	6	805.54	6	6
C25_1_8	985.90	7	13	991.44	8	13
C40_1_8	1,515.88	8	8	1,516.60	8	8
RC10_1_4	592.74	3	3	592.09	3	3
RC15_1_6	1,149.08	6	0	1,149.45	6	0
RC20_1_8	1,200.65	7	6	1,200.12	7	6
RC25_1_8	1,652.56	8	6	1,660.77	8	6
RC40_1_12	2,288.50	11	1	2,293.31	11	1
R10_1_4	675.32	4	0	675.43	4	0
R15_1_6	1,118.64	6	0	1,121.31	6	0
R20_1_8	1,571.57	8	1	1,576.25	8	0
R25_1_8	1,698.63	8	4	1,698.82	8	4
R40_1_12	2,195.36	11	5	2,195.71	11	3
Average	1,241.36	6.7	4.7	1,245.27	6.8	4.2

Table 5.9 The effect of the SLs frequency on the total expected costs

	180 units			no SLs	
Instance	E(x)	#Vehs	#SLs	E(x)	#Vehs
C10_1_4	561.22	3	0	561.69	3
C15_1_6	660.11	4	9	721.31	4
C20_1_8	875.08	7	2	890.36	6
C25_1_8	1,106.74	7	5	1,192.01	6
C40_1_8	1,537.02	8	3	1,609.99	8
RC10_1_4	703.95	4	0	703.70	4
RC15_1_6	1,149.37	6	0	1,149.93	6
RC20_1_8	1,455.33	7	0	1,455.86	7
RC25_1_8	1,665.10	8	0	1,665.57	8
RC40_1_12	2,327.60	10	0	2,328.38	10
R10_1_4	675.27	4	0	675.14	4
R15_1_6	1,121.27	6	0	1,121.84	6
R20_1_8	1,575.95	8	0	1,576.70	8
R25_1_8	1,698.74	8	4	1,762.23	8
R40_1_12	2,198.81	10	3	2,214.91	10
Average	1,287.44	6.7	1.7	1,308.64	6.5

According to the obtained results, lower operational costs can be achieved with higher frequency of the SLs. The cost savings increase due to the fact that more requests are shipped via SLs. On the other hand, higher frequency implies more SLs capacity, hence higher integration between the two transportation networks that leads to fewer SLs capacity violations.

5.5.8 An analysis on heterogeneous PD vehicle routing costs

In this section, we investigate the benefits of the integrated networks considering heterogeneous routing costs of the PD vehicles and one SL. More specifically, the minimum-capacity vehicle is assumed to cost 0.5 per operating time unit. Larger vehicles are assigned a cost that linearly increases with the carrying capacity. The average results over the 10 runs of the algorithm are presented in Table 5.10. The columns are self-explanatory.

Table 5.10 Heterogeneous PD vehicle routing costs

Instance	PDPTW-SLSD			PDPTW-SD		
	E(x)	#Vehs	#SLs	E(x)	#Vehs	Savings %
C10_1_4	665.13	4	0	665.13	4	0.00
C15_1_6	752.86	5	9	906.66	4	16.96
C20_1_8	1,006.83	6	6	1,099.14	6	8.40
C25_1_8	1,260.59	7	13	1,548.27	6	18.58
C40_1_8	1,938.28	8	10	2,061.83	8	5.99
RC10_1_4	649.04	3	3	786.06	4	17.43
RC15_1_6	1,500.01	6	0	1,502.01	6	0.13
RC20_1_8	1,632.94	7	6	1,872.39	7	12.79
RC25_1_8	2,216.67	8	0	2,241.86	8	1.12
RC40_1_12	3,002.31	10	1	3,034.01	11	1.04
R10_1_4	682.06	4	0	682.12	4	0.01
R15_1_6	1,248.42	6	0	1,248.78	6	0.03
R20_1_8	2,012.47	8	1	2,022.69	8	0.51
R25_1_8	2,277.16	8	4	2,356.54	8	3.37
R40_1_12	2,509.08	10	3	2,559.60	10	1.97
Average	1,556.92	6.7	3.7	1,639.14	6.7	5.89

The results indicate that operating cost savings of up to 18% can be achieved compared to the corresponding PDPTW-SD solutions. On average, using the same amount of vehicles as in classical pickup and delivery system, and the available SLs may lead to average cost savings of 5.9%.

5.5.9 Case study of Amsterdam metro system

In this section, we investigate the performance of the PDPTW-SLSD on a realistic scheduled-line network. More specifically, we replicate the Amsterdam metro system as a case study. In addition, one instance with 30 requests and two depots with three vehicles each is considered. We evaluate the systems' performance on different cost (i.e., routing and SL) settings. Note that outsourcing cost P_1 is set to $6 \times \theta$, where θ is considered to be the same for all PD vehicles. Table 5.11 indicates the obtained results.

Figure 5.8 illustrates the relationship between operational cost savings and the ratio $\theta \backslash \eta$. As can be seen, the larger routing cost is, the more savings can

Table 5.11 Sensitivity analysis on the ratio of costs

$\theta\backslash\eta$	1			10			30			PDPTW-SD	
	E(x)	#Vehs	#SLs	E(x)	#Vehs	#SLs	E(x)	#Vehs	#SLs	E(x)	#Vehs
0.5	303.35	5	13	316.29	5	2	323.12	4	0	323.10	4
1	584.82	6	23	651.24	4	4	653.51	4	0	653.66	4
5	3,168.91	4	11	3,173.09	5	7	3,261.78	5	2	3,714.39	5

be achieved from the integrated transportation networks (see 5\1, 5\10, etc. in Figure 5.8).

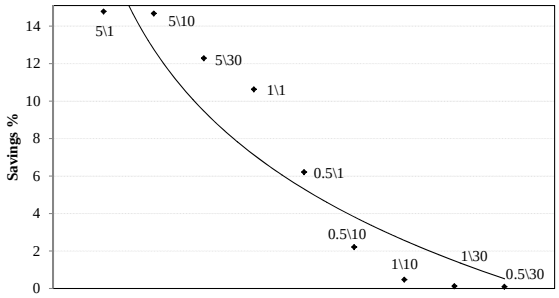

Figure 5.8 Relationship between the operating cost savings and $\theta\backslash\eta$

For a given SL cost η, the larger θ is, the more requests will be shipped via available SLs to perform the deliveries, and thus more cost savings can be achieved. On the other hand, smaller θ leads to insignificant incentives to make use of SLs, unless their cost is very small. Otherwise, the cost of using SLs will trade-off the routing cost savings, thus reducing the benefits of integrating system to none.

5.6 Conclusions

We proposed a scenario-based solution methodology to solve the considered short-haul transportation problem, where demand uncertainty is also taken into consideration. ALNS heuristic algorithm is used within the SAA framework to solve the PDPTW-SLSD. The computational results on instances with up to 40 requests (i.e., 80 locations) indicate significant reduction in expected operational costs, compared to the corresponding PDPTW stochastic routing solutions (i.e., up to 28%). In addition, using more SLs leads to shorter vehicle routes and fewer capacity violations. As a consequence of using extra capacity, lower operational costs can be achieved. Intuitively, larger SL capacity or higher SL frequency benefits the transportation system since fewer recourse actions may need to be taken. Furthermore, the results indicate that the relationship between routing

and SL costs drives the cost savings.

Investigating exact algorithms for the PDPTW-SLSD may be an interesting area for further research. An efficient mathematical formulation for the recourse problem is needed to apply exact algorithms based on stochastic programming techniques. The effects of uncertainty in travel times on the operational aspects may also be interesting to investigate. Last, but no least, focusing on environmental aspects, i.e., minimizing the emissions instead of operating costs, may be an interesting research area.

Chapter 6

Conclusions

Transportation is vital to modern societies. However, there are many undesirable side effects associated with it. These include increasing amount of emissions, traffic congestion, noise levels and amount of fatalities due to traffic accidents. As discussed by Browne et al. (2012), the total amount of kilometers driven impacts the most of the aforementioned side effects.

Generally, urban areas are densely covered by public transportation networks e.g., bus, tram and metro lines. Thus, freight and public transportation trips overlap to a significant extent (Trentini and Malhene, 2010), causing increased traffic congestion and noise level. On the other hand, public transport coverage is considerably smaller in rural areas, due to population scarceness and hence low demands. As a consequence, public transport operators hardly manage to provide high service level at socially acceptable prices. In addition, freight carriers perform fewer stops over longer distances, leading to high operating costs per demand served. To cope with this reality, this thesis focuses on cost-efficient integrated freight and public transportation solutions, which can be applied to urban, as well as rural areas.

More specifically, we investigate the effects of the integrated pickup and delivery operations with public transportation, from a routing and scheduling perspective. In other words, we quantify the cost benefits of having the flexibility of using available public transportation lines for carrying freight on an operational planning level. Real aspects such as schedules and capacities of the public transport vehicles are accounted for. The considered problem in this thesis is an extension of the classical pickup and delivery problem, coined as the PDPTW-SL.

Research question 1. Which modeling framework can be used to formally

define the investigated problem?

In Chapters 2 we show how the PDPTW-SL can be formalized as an arc-based MIP. The model is general, as it considers heterogeneous (in terms of capacity and routing cost) vehicles, heterogeneous (in terms of capacity) public transportation vehicles and lines (in terms of frequency). The proposed formulation is tightened using several families of valid inequalities, which helped improve solution times using an all-purpose solver (i.e., CPLEX). We perform extensive computational experiments on small-size (up to 12 requests) instances to assess the potential of such integrated transportation systems. We found that the benefits increase along with the number of available public transport lines, and their frequencies. In some cases, significant reductions of up to 20% in operating costs can be achieved by not increasing the number of vehicles used to perform the service. In the following chapters, we investigate solution methodologies to effectively solve the PDPTW-SL.

In addition, Chapter 3 provides a different MIP formulation for the PDPTW-SL. In particular, the problem is defined in terms of vehicle routes. Since it is prohibitively time-consuming to generate all possible routes, and consequently solve the formulation, we propose a tailored B&P algorithm. Each branch-and-bound node is solved using column generation. The columns are constructed by solving a variant of the elementary shortest path problem with resource and precedence constraints, using a labeling algorithm. The overall performance of the B&P is enhanced by applying several techniques, such as pricing problem heuristic, label elimination, pre-processing and valid inequalities. We note that the PDPTW-SL has the PDPTW with transfers (PDPTW-T) as a special case. To the best of our knowledge, the proposed B&P is the first approach based on column generation that can be applied to the PDPTW-T.

Research question 2. Which exact and heuristic algorithms are effective in solving the PDPTW-SL?

In Chapters 2 and 3, different models are presented and two exact algorithms (i.e., all-purpose solver and B&P) are proposed to solve the problem. As expected, arc-based formulation (Chapter 2) performs significantly worse than the model given in Chapter 3. The B&P algorithm is able to solve PDPTW-SL instances with up to 50 requests and PDPTW-T instances with up to 40 requests, whereas the arc-based formulation could determine optimal solutions for instances with up to 11 requests only.

In Chapter 4, we develop an ALNS heuristic to find relatively good-quality solutions for large-size instances. Time- and load-synchronization constraints are efficiently considered within the proposed framework. We make use of several destroy and insertion operators, which are adapted or inspired from the related

literature. Instances with up to 100 requests and up to three public transport lines are solved. According to the obtained results, the quality of the solutions tends to be relatively good (i.e., up to 2.3% GAP).

Research question 3. How can demand uncertainty be accounted for in the tactical planning process?

Finally, in Chapter 5, we investigate a variant of the PDPTW-SL, where the request demands are uncertain at the moment of planning, given corresponding distributions. The demands are revealed upon vehicle's arrival at the pickup locations of the requests. The problem considers tactical decision level, which means that generated routes will be executed on a regular basis over a planning horizon. The customer nodes might represent zip codes (a group of customers in an neighborhood), as approximating individual demand might be innacurate, due to e.g. low individual demand. We develop a scenario-based solution methodology. In particular, we extend the ALNS proposed in Chapter 4 of this thesis and embed it into a sample average approximation framework. The ALNS operators are updated to account for stochastic elements of the problem. We solve instances with up to 40 requests and up to three public transport lines. The operating cost benefits of the integrated transportation system remain significant if the uncertainty is taken into account during the planning process.

Research objective. Quantify the cost benefits of the integrated public and freight transportation from operational and tactical planning perspectives.

This thesis quantifies the operational cost benefits of the integrated freight and public transportation networks (or scheduled lines). As shown throughout the thesis, logistics service providers may experience operating cost savings of up to 32%, compared to the traditional operation. Note that these savings are achieved by shorter vehicle routes, as scheduled services are used as part of the freight's journey. Additional benefits, such as the decrease in the amount of air pollution, noise, congestion, may also result, as these are directly related to total driven kilometers of the vehicles. Furthermore, public transport operators may generate additional revenue, as freight is also carried next to passengers. Thus, servicing rural areas may become slightly less costly, as the cooperation between LSPs and public transport operators may lead to benefits for both parties.

As mentioned earlier, considered problem may be encountered in long-haul transportation as well, where scheduled services may be freight scheduled transportation. For example, requests may represent containers and scheduled services may be container liners (i.e., ships), which nowadays operate according to fixed routes and schedules. In this case, certain assumptions used in this thesis for urban transportation may be relaxed. In particular, the infrastructure needed to perform the transfers at the transfer nodes exist (i.e., gantry cranes, storage

yards). In addition, the ports are secured by high fences to ensure security. Furthermore, transfer operations will become more costly and they might be needed to be re-modeled, as these costs are dependent on the number of moves (i.e., loading/unloading) as well, and not only on the number of SLs used.

Alternatively, scheduled services may represent airline flights between certain airports, transporting either passengers or freight, or both. In this case, loading/unloading infrastructure for transferring freight exists in the airports and aforementioned discussions are valid for this application as well. Regarding passenger transportation, transfer might involve only costs for using SLs, and not the transfer operations, as passengers can transfer by themselves.

6.1 Challenges

Due to enormous complexity of real-life systems and limited computing/ methodological resources available nowadays to tackle real-life problems, we made certain simplifying assumptions that relatively well represent reality. However, some challenges still exist due to potential legal practicalities. For example, we assume the existence of relevant infrastructure for performing the transfer operations to/from SLs. In some countries (e.g., France) loading/unloading of freight cannot be performed at the same locations (i.e., stations) where passengers get on public transport vehicles. To address such a limitation, public transport lines may need to be modified, to stop at a transfer location, which is nearby, before traveling to the end-of-the line to pick up the passengers.

In addition, we also assume that the responsibility of transfer operations belong to public transport drivers. However, in many countries, drivers' rights are protected by labor unions and it may be difficult to assign them extra responsibilities, unless high extra costs are paid, which obviously may affect the cost-efficiency of the proposed transportation system.

Furthermore, security issues are crucial nowadays. Therefore an integrated transportation system investigated in this thesis may require additional measures, e.g., scanners, to tackle this issue, especially considering applications in urban logistics.

Generally, the planning problems related to integrated transport systems become significantly more complex. In particular, more resources (i.e., vehicles and public transport lines) need to be accounted for during the planning process. Uncertainty related to e.g., travel times and capacity, leads to additional complexity to the planning problem, as shown in Chapter 5. Scheduled services, such as public transportation involve extra uncertainty in travel times between

start- and end-of-lines, due to traffic conditions. Therefore, routing plans need to be robust in terms of timing, in order to be able to meet customers' service time windows.

6.2 Further research directions

The research presented in this thesis can be extended in multiple directions. In this thesis, we focused on operational decision support tools for an integrated freight and public transportation system. Many unexpected events (e.g., changes in travel times) may cause difficulties in executing an initial plan. Thus, considering dynamic aspects of the problem and development of appropriate algorithms (heuristic or exact) may be an interesting direction for further research. In addition, such dynamic algorithms may be also used as recourse actions within methodologies focused on stochastic variants of the problem.

Strategic decisions related to operation of such systems may be imperative to study. In particular, decisions related to which and how many public transportation lines should be used to carry freight apart from the passengers. Investment costs associated with setting up the transfer nodes, storage capacities, re-design of the public transport vehicles, should be accounted for in more detail.

From a methodological point of view, the exact algorithm presented in Chapter 3 can be improved by adding appropriate valid inequalities to tighten the lower bound. In addition, since 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.

As mentioned earlier, in urban areas travel times are dynamic and stochastic. Thus, scheduled services might not be as accurate as assumed. This makes synchronization between PD and SL vehicles more complicated, thus leading to higher chances of time-window violations. Therefore, appropriate methodology (e.g. SAA) may be needed to tackle this uncertainty. In particular, the recourse actions described in Chapter 5 may be extended to consider penalty costs for violating the time windows. Such an approach would complicate the problem even more, thus leading to higher computational effort needed.

Bibliography

Y Adulyasak, J-F Cordeau, and R Jans. Benders decomposition for production routing under demand uncertainty. *Operations Research*, 63(4):851–867, 2015.

M Aldaihani and M Dessouky. Hybrid scheduling methods for paratransit operations. *Computers & Industrial Engineering*, 45(1):75–96, 2003.

R Baldacci, E Bartolini, and A Mingozzi. An exact algorithm for the pickup and delivery problem with time windows. *Operations Research*, 59(2):414–426, March 2011a. ISSN 0030-364X.

R Baldacci, A Mingozzi, and R Roberti. New route relaxation and pricing strategies for the vehicle routing problem. *Operations Research*, 59(5):1269–1283, 2011b.

G Berbeglia, J-F Cordeau, I Gribkovskaia, and G Laporte. Static pickup and delivery problems: a classification scheme and survey. *TOP: An Official Journal of the Spanish Society of Statistics and Operations Research*, 15(1):1–31, 7 2007.

L Bianchi, M Dorigo, L M Gambardella, and W Gutjahr. A survey on metaheuristics for stochastic combinatorial optimization. *Natural Computing*, 8 (2):239–287, 2009.

K Braekers, A Caris, and G Janssens. Exact and meta-heuristic approach for a general heterogeneous dial-a-ride problem with multiple depots. *Transportation Research Part B*, 67:166–186, 2014.

M Browne, J Allen, T Nemoto, D Patier, and J Visser. Reduing social and environmental impacts of urban freight transport: A review of some major cities. *Procedia - Social and Behavioral Sciences*, 39:19–33, 2012.

CargoTram. Official webpage (accessed on 11.01.2016), 2007. URL `http://www.eltis.org/discover/case-studies/delivering-goods-cargo-tram-amsterdam-netherlands`.

J-F Cordeau and G Laporte. The dial-a-ride problem: models and algorithms. *Annals OR*, 153(1):29–46, 2007.

E-C Cortes, M Matamala, and C Contardo. The pickup and delivery problem with transfers: Formulation and a branch-and-cut solution method. *European Journal of Operational Research*, 200(3):711–724, 2010.

J Coyle, E Bardi, and C Langley. The management of business logistics, 1996.

S Dabia, E Demir, and T Van Woensel. An exact approach for a variant of the pollution-routing problem. *Accepted in Transportation Science*, page Forthcoming, 2015.

G Danzig and J Ramser. The truck dispatching problem. *Management Science*, 2:80–91, 1959.

E Demir, T Bektaş, and G Laporte. An adaptive large neighborhood search heuristic for the pollution-routing problem. *European Journal of Operational Research*, 223(2):346–359, 2012.

E Demir, T Bektaş, and G Laporte. The bi-objective Pollution-Routing Problem. *European Journal of Operational Research*, 232(3):464–478, 2013.

E Demir, Y Huang, S Scholts, and T Van Woensel. A selected review on the negative externalities of the freight transportation: Modeling and pricing. *Transportation Research Part E: Logistics and Transportation Review*, 77:95–114, 2015.

G Desaulniers, F Lessard, and A Hadjar. Tabu search, partial elementarity, and generalized k-path inequalities for the vehicle routing problem with time windows. *Transportation Science*, 42(3):387–404, 2008.

M Desrochers, J Desrosiers, and M Solomon. A new optimization algorithm for the vehicle routing problem with time windows. *Operations Research*, 40(2): 342–354, 1992.

DHL-Packstation. DHL official webpage (accessed on February 21, 2016), 2016. Available at: `www.dhl.de/en/paket/pakete-empfangen/packstation.html`.

E-W Dijkstra. A note on two problems in connexion with graphs. *Numerische Mathematik*, 1:269 – 271, 1959.

M Drexl. Synchronization in vehicle routing - A survey of vrps with multiple synchronization constraints. *Transportation Science*, 46(3):297–316, 2012.

Y Dumas, J Desrosiers, and F Soumis. The pickup and delivery problem with time windows. *European Journal of Operational Research*, 54:7–22, 1991.

Eurostat. Eurostat official webpage (accessed on February 25, 2016), 2016. Available at: `www.ec.europa.eu/eurostat/statistics-explained/index.php/Freight_transport_statistics`.

D-J Forkenbrock. External costs of intercity truck freight transportation. *Transportation Research Part A: Policy and Practice*, 33:505–526, 1999.

D-J Forkenbrock. Comparison of external costs of rail and truck freight transportation. *Transportation Research Part A: Policy and Practice*, 35:321–337, 2001.

G Ghiani, G Laporte, and R Musmanno, editors. *Introduction to Logistics Systems Planning and Control*. Wiley & Sons Ltd., 2004.

Green Paper on Urban Mobility. Green paper on urban mobility, 2007. Available at: `www.ec.europa.eu/transport/urban/urban_mobility/green_paper/doc/2007_09_25_gp_urban_mobility_memo_en.pdf`.

D-L Greene and M Wagener. Sustainable transport. *Journal of Transport Geography*, 5:177–190, 1997.

W Gutjahr. Recent trends in metaheuristics for stochastic combinatorial optimization. *Central European Journal of Computer Science*, 1:58–66, 2011.

CH Häll, H Andersson, JT Lundgren, and P Varbrand. The integrated dial-a-ride problem. *Public Transportation*, 1:39–54, 2009.

L Harms, M Jose O Kalter, and P Jorritsma. Krimp en mobiliteit (Shrinkage and mobility). *Ministerie van Verkeer en Waterstaat*, 2010.

G Heilporn, J-F Cordeau, and G Laporte. An integer L-shaped algorithm for the dial-a-ride problem with stochastic customer delays. *Discrete Applied Mathematics*, 159(9):883–895, 2011.

Hurtigruten. Hurtigruten official webpage (accessed on February 11, 2016), 2016. Available at: `www.hurtigruten-web.com/index_en.html`.

S Irnich and G Desaulniers. Shortest path problems with resource constraints. In G Desaulniers, J Desrosiers, and M M. Solomon, editors, *Column Generation*, pages 33–65. Springer US, 2005.

A Kleywegt, A Shapiro, and T Homem-De-Mello. The sample average approximation method for stochastic discrete optimization. *SIAM Journal of Optimization*, 2:898–913, 2001.

G Laporte, F Louveaux, and L van Hamme. An integer L-shaped algorithm for the capacitated vehicle routing problem with stochastic demands. *Operations Research*, 50(3):415–423, 2002.

Y Levin, M Nediak, and H Topaloglu. Cargo capacity management with allotments and spot market demand. *Operations Research*, 60(2):351–365, 2012.

B Li, D Krushinsky, T Van Woensel, and H Reijers. The share-a-ride problem with stochastic travel times and stochastic delivery locations. *Transportation Research Part C: Emerging Technologies*, 67:95 – 108, 2016.

H Li and A Lim. A metaheuristic for the pickup and delivery problem with time windows. *International Journal on Artificial Intelligence Tools*, 12(2):173–186, 2003.

C-F Liaw, C White, and J Bander. A decision support system for the bimodal dial-a-ride problem. *IEEE Transactions on Systems, Man, and Cybernetics, Part A*, 26(5):552–565, 1996.

M Lindholm and S Behrends. Challenges in urban freight transport planning – a review in the Baltic Sea Region. *Journal of Transport Geography*, 22:129–136, 2012.

F Louveaux and J J S Gonzalez. On the one-commodity pickup-and-delivery traveling salesman problem with stochastic demands. *Mathematical Programming*, 119(1):169–194, 2009.

Q Lu and M Dessouky. A new insertion-based construction heuristic for solving the pickup and delivery problem with time windows. *European Journal of Operational Research*, 175(2):672–687, 2006.

ME Lübbecke and J Desrosiers. Selected topics in column generation. *Operations Research*, 53(6):1007–1023, 2005.

R Marsten, W Hogan, and J Blankenship. The boxstep method for large-scale optimization. *Operations Research*, 23(3):389–405, 1975.

R Masson, F Lehuede, and O Peton. An adaptive large neighborhood search for the pickup and delivery problem with transfers. *Transportation Science*, 47:1 – 12, 2012.

R Masson, F Lehuede, and O Peton. The dial-a-ride problem with transfers. *Computers & Operations Research*, 41:12 – 23, 2014.

C Mues and S Pickl. Transshipment and time windows in vehicle routing. In *8th International Symposium Parallel Architectures, Algorithms and Networks, Piscataway, NJ. Proceedings*, pages 113–119, 2005.

W Nanry and J W Barnes. Solving the pickup and delivery problem with time windows using reactive tabu search. *Transportation Research Part B: Methodological*, 34(2):107–121, 2000.

C A Nash, editor. *Economics of public transport*. Longman, London, UK, 1982.

S Parragh, J-F Cordeau, K Doerner, and R Hartl. Models and algorithms for the heterogeneous dial-a-ride problem with driver-related constraints. *OR Spectrum*, 34:593 – 633, 2012.

S N Parragh. Introducing heterogeneous users and vehicles into models and algorithms for the dial-a-ride problem. *Transportation Research Part C: Emerging Technologies*, pages 912–930, 2010.

A Alves Pessoa, R Sadykov, E Uchoa, and F Vanderbeck. In-out separation and column generation stabilization by dual price smoothing. In *Experimental Algorithms, 12th International Symposium, SEA 2013, Rome, Italy, June 5-7, 2013. Proceedings*, volume 7933 of *Lecture Notes in Computer Science*, pages 354–365. Springer, 2013.

H L Petersen and S Røpke. The pickup and delivery problem with cross-docking opportunity. In *Computational Logistics*, volume 6971 of *Lecture Notes in Computer Science*, pages 101–113. Springer Berlin Heidelberg, 2011.

D Pisinger and S Røpke. A general heuristic for vehicle routing problems. *Computers & Operations Research*, 34(8):2403–2435, 2007.

W B Powell, Y Sheffi, KS Nickerson, K Butterbaugh, and S Atherton. Maximizing profits for north american van lines' truckload division: A new fremwork for pricing and operation. *Interfaces*, 1(18):21–41, 1988.

A Rais, F Alvelos, and M S Carvalho. New mixed integer-programming model for the pickup-and-delivery problem with transshipment. *European Journal of Operational Research*, 235:530–539, 2013.

G Righini and M Salani. Symmetry helps: bounded bi-directional dynamic programming for the elementary shortest path problem with resource constraints. *Discrete Optimization*, 3(3):255–273, 2006.

G Righini and M Salani. New dynamic programming algorithms for the resource constrained elementary shortest path problem. *Networks*, 51(3):155–170, 2008.

J–P Rodrigue, C Comtois, and B Slack. *The Geography of Transport Systems*. Taylor & Francis, 2009. ISBN 9780203884157.

S Røpke and J-F Cordeau. Branch and cut and price for the pickup and delivery problem with time windows. *Transportation Science*, 43(3):267–286, 2009.

S Røpke and D Pisinger. An adaptive large neighborhood search heuristic for the pickup and delivery problem with time windows. *Transportation Science*, 40(4): 455–472, 2006.

S Røpke and D Pisinger. Complexity of shortest path problems arising in column generation for vehicle routing problems with pairing and precedence constraints. Technical report, DTU: Technical University of Denmark, 2007.

G Santos, H Behrends, and A Teytelboym. Part II: Policy instruments for sustainable road transport. *Research in Transportation Economics*, 28:46–91, 2010.

M Savelsbergh and M Sol. The general pickup and delivery problem. *Transportation Science*, 29:17–29, 1995.

M Schilde, K F Doerner, and R F Hartl. Metaheuristics for the dynamic stochastic dial-a-ride problem with expected return transports. *Computers & Operations Research*, 38(12):1719–1730, 2011.

N Secomandi and F Margot. Reoptimization approaches for the vehicle-routing problem with stochastic demands. *Operations Research*, 57(1):214–230, 2009.

J Shang and C Cuff. Multicriteria pickup and delivery problem with transfer opportunity. *Computers & Industrial Engineering*, 30(4):631 – 645, 1996.

P Shaw. Using constraint programming and local search methods to solve vehicle routing problems. In *Proceedings of the 4th International Conference on Principles and Practice of Constraint Programming*, pages 417–431. Springer, New York, 1998.

M Sigurd, D Pisinger, and M Sig. Scheduling transportation of live animals to avoid the spread of diseases. *Transportation Science*, 38:197–209, 2004.

SmartLogisticsLab. Official webpage (accessed on February 21, 2016), 2016. Available at: `www.smartlogisticslab.nl`.

A Trentini and N Malhene. Toward a shared urban transport system ensuring passengers & goods cohabitation. *Trimestrale del Laboratorio Territorio Mobilita e Ambiente - TeMALab*, 4:37–44, 2010.

A Trentini, R Masson, F Lehuede, N Malhene, O Peton, and H Tlahig. A shared "passenger & goods" city logistics system. *4th International Conference on Information Systems, Logistics and Supply Chain, Quebec, Canada*, 2012.

B Verweij, S Ahmed, A Kleywegt, G Nemhauser, and A Shapiro. The sample average approximation method applied to stochastic routing problems: a computational study. *Computational Optimization and Applications*, 24:289–333, 2003.

H Xu, Z-L Chen, S Rajagopal, and S Arunapuram. Solving a practical pickup and delivery problem. *Transportation Science*, 37(3):347–364, 2003.

Summary

The Pickup and Delivery Problem with Time Windows and Scheduled Lines: models and algorithms

A successful integration of freight and public transportation creates a seamless movement for both people and goods. This integration achieves socially desirable economically viable transport options in urban areas as it reduces congestion and air pollution. This paper focuses on opportunities to make use of available public transportation services, which operate according to predetermined routes and schedules, as a part of freight journey in the pickup and delivery transport systems. With the possibility of using scheduled line services (SLs), there can be two delivery options for serving the transport requests. The first one implies that the origin and destination points of a request are visited by using only one pickup and delivery (PD) vehicle. The second type of delivery implies that a request is picked up by a PD vehicle and transported to a transfer node. From there, the request continues its journey on SLs. Afterwards, the request is picked up again by another PD vehicle to be delivered to its destination point. In this thesis, we denote this specific transportation problem as the *Pickup and Delivery Problem with Time Windows and Scheduled Lines* (PDPTW-SL).

In Chapter 2 we formalize the problem as an arc-based mixed-integer program (MIP). In this context, all data (i.e., requests, demands, travel times, service times, etc.) are assumed to be known in advance. A state-of-the-art solver is used to solve instances with up to 11 requests to optimality. Based on the obtained results, we conclude significant operating-cost savings of up to 20% due to the use of available SLs.

In Chapter 3 we propose an advanced exact algorithm to optimally solve larger instances. The problem is first re-formulated as a path-based MIP. Since it is prohibitively time-consuming to generate all possible paths (i.e., vehicle routes) and consequently solve the formulation, the proposed exact algorithm uses column

generation to build promising routes in a branch-and-price (B&P) framework. The column generator solves the *Elementary Shortest Path Problem with Resource and Precedence Constraints*, which is the natural pricing problem for the PDPTW-SL. We show the effectiveness of the proposed algorithm by optimally solving medium-size instances of up to 50 requests. In addition, the obtained results indicate that operating cost savings remain significant for medium-size instances.

Chapter 4 aims to propose an advanced metaheuristics for the solution of the PDPTW-SL. We developed an adaptive large neighborhood search (ALNS) heuristic. The ALNS is based on the idea of iteratively destroying an existing solution, and then re-constructing it using certain operators. The proposed heuristics provide relatively good-quality solutions within short computational time for instances with up to 100 requests. Finally, we conclude that operating-cost savings of up to 30% can be achieved for large instances as well.

Chapter 5 presents an extended version of the problem at hand, which considers request demands to be uncertain. The demands are assumed to be given as a probability distribution. We extend the ALNS proposed in previous chapter, and embed it into a sample average approximation (SAA) framework to generate good-quality solutions for the PDPTW-SL with stochastic demands. We conclude that making use of available SLs makes sense, as long as the demand uncertainty is considered in the planning process.

About the author

Veaceslav Ghilas was born on October 18, 1986 in Straseni, Moldova. He received his B.Sc. (2009) and M.Sc. (2011) in Maritime Navigation and Transport from Maritime University of Constanta, Romania. In addition, he received a second M.Sc. (2012) in Transportation and Logistics from Technical University of Denmark. His master's thesis on container stowage problem was supervised by Stefan Røpke.

In 2012, he started as PhD student at Eindhoven University of Technology under the supervision of Tom Van Woensel and Emrah Demir. During his PhD programme, Veaceslav spent three months as a visiting researcher at CIRRELT (Interuniversity Research Center on Enterprise Network, Logistics and Transportation), Canada, and collaborated with Jean-François Cordeau (HEC Montréal, Canada). On October 27, 2016 Veaceslav will defend his PhD thesis at Eindhoven University of Technology.