KU LEUVEN

ARENBERG DOCTORAL SCHOOL
Faculty of Engineering Technology

Vehicle Routing Problems in Logistic Companies

Vinícius Gandra Martins Santos

Supervisors:
Prof. dr. ir. G. Vanden Berghe
Prof. dr. H. Çalik
Prof. dr. M. A. M. Carvalho
Prof. dr. T. A. M. Toffolo

Dissertation presented in partial fulfillment of the requirements for the degree of Doctor of Engineering Technology (PhD)

May 2021

Vehicle Routing Problems in Logistic Companies

Vinícius GANDRA MARTINS SANTOS

Examination committee:
Prof. dr. ir. The Chairman, chair
Prof. dr. ir. G. Vanden Berghe, supervisor
Prof. dr. H. Çalik, supervisor
Prof. dr. M. A. M. Carvalho, supervisor
Prof. dr. T. A. M. Toffolo, supervisor
Prof. dr. ir. The One
Prof. dr. ir. The Other
Prof. dr. External Jurymember
(Far Away)

Dissertation presented in partial fulfillment of the requirements for the degree of Doctor of Engineering Technology (PhD)

May 2021

Abstract

The vehicle routing problem (VRP) was first introduced nearly 60 years ago and delineates a class of problems which remains an active research area of operations research. In simple terms, the VRP aims to minimize the cost of delivery routes from a central depot to a set of geographically scattered customers using a fleet of homogeneous vehicles subject to capacity constraints. The VRP is an $\mathcal{NP}$-hard problem and thus poses a challenge to both exact and heuristic approaches. Despite its complexity, the VRP in its original form is unable to capture the level of detail which characterizes real-world applications where various operational rules are encountered. In order to bridge the gap between academic theory and practical applications the original VRP is often expanded with real-world constraints, leading to a range of computationally challenging VRP variants.

The research presented in this thesis is motivated by distribution challenges arising in logistic companies which have not yet been completely captured in existing problems. This thesis introduces VRP generalizations which combine multiple constraints occurring at different decision levels to tackle these real-world problems. These levels of decisions include, besides the routing, the location decisions concerning where to transfer goods and loading decisions which aim to produce feasible two-dimensional load plans. Such problem generalizations require more decisions to be taken into account, significantly expanding the search space and posing a great challenge for companies which currently rely on off-the-shelf solvers or human operators to produce solutions. To obtain high-quality solutions in short computational runtimes, heuristics are developed which comprise of tailored methods capable of efficiently handling problem-specific constraints and which deploy state-of-the-art algorithms to solve the VRP component of the problems. This thesis also investigates the impact of the newly introduced constraints on solution quality. Computational experiments are performed to evaluate the quality of the proposed methods and solutions obtained using instances derived from real-world data which have been made publicly available to stimulate further research.

List of Abbreviations

2E-LRP Two-echelon location-routing problem. vi, ix, 3, 42–45, 47, 50, 52, 62, 67–71, 74

2E-LRP2L Two-echelon location-routing problem with two-dimensional loading constraints. 3, 4, 43, 45–50, 58–62, 67, 69, 71, 73, 74, 77

2E-LRPSD Two-echelon location-routing problem with a single platform. 44, 45

2E-VRP Two-echelon vehicle routing problem. 2

2L-CVRP Vehicle routing problem with two-dimensional loading constraint. 45, 46

MDVRP Multi-depot vehicle routing problem. 1

OVRP Open vehicle routing problem. 2

PDP Vehicle routing problem with pickup and delivery. 2, 8

PDPT Pickup and delivery problem with transshipment. 8, 9

PDPTW Pickup and delivery problem with time windows. 5

SDVRP Split delivery vehicle routing problem. 2

TTRP Truck-and-trailer routing problem. 10

VRP Vehicle routing problem. 1–3, 5, 10, 73–75

VRPTW Vehicle routing problem with time windows. 2

VSBR Vessel Swap-Body Routing Problem. 3, 5–12, 20–22, 25, 31, 37, 39, 73, 74

Contents

Abstract i

List of Abbreviations iii

List of Symbols v

Contents v

List of Figures vii

List of Tables ix

1 Introduction 1

2 Vessel swap-body routing problem 5
2.1 Introduction 5
2.2 Related work 8
2.2.1 Pickup and delivery problems with transshipment . . . 8
2.2.2 Vehicle routing problems with trailers and transshipment 10
2.3 Notation and problem formulation 11
2.3.1 A modified network construction 12
2.3.2 Decision variables 14
2.3.3 A mixed integer programming formulation for the VSBR 14
2.4 A heuristic for the VSBR 20
2.4.1 Acceptance criterion 21
2.4.2 Initial solution 22
2.4.3 Best insertion method for requests 22
2.4.4 Transfer phase 23
2.4.5 Routing phase 30
2.5 Computational experiments 30
2.5.1 Instance sets 31

2.5.2 Detailed results 33
2.6 Conclusions 39

3 Two-echelon location routing problem with loading constraints 41
3.1 Introduction 41
3.2 Related work 43
3.2.1 Selected studies concerning the 2E-LRP 44
3.2.2 Selected studies concerning the 2L-CVRP 45
3.3 Problem description 46
3.4 Methodology 50
3.4.1 Initial solution 51
3.4.2 Location phase 53
3.4.3 Lower bound 54
3.4.4 Routing phase 56
3.4.5 Load plan method 58
3.4.6 Loading strategies 58
3.5 Computational study 60
3.5.1 New instance set 61
3.5.2 Detailed results 62
3.5.3 Result discussion 65
3.6 The 2E-LRP special case 67
3.6.1 Benchmark instances 68
3.6.2 ILS-LR for the 2E-LRP 68
3.6.3 Parameters and setup 69
3.6.4 Comparison with the state-of-the-art 2E-LRP methods . 69
3.7 Conclusions 70

4 Conclusion 73
4.1 Future research for this thesis 75

A 77
A.1 2E-LRP formulation 77
A.2 2E-LRP2L solution cost breakdown 79
A.3 CMLRP formulation 79

Bibliography 83

List of Figures

1.1 VRP variants present in the 2E-LRP2L. 4

2.1 Example of a vessel towing three bodies. 6
2.2 VSBR solution with body transfers. 7
2.3 A modified network representation for the model 13
2.4 Body transfer insertion. 26
2.5 Solution values for the I_300 instances. 37
2.6 Relationship between travel cost and percentage of unserved requests. 38

3.1 Two-echelon routes with load plan. 42
3.2 Loading constraints and violations. 48
3.3 Cost for the combinations of two routes and two load plans. . . 49
3.4 Second-echelon routes with different satellite configurations. . . 55
3.5 Second echelon minimum routing cost construction. 56
3.6 An infeasible load plan for items whose total area is less than that of the vehicle. 59
3.7 Different loading strategies for the first echelon. 63
3.8 Different loading strategies for the second echelon. 64
3.9 Different loading strategies for the second echelon. Traveling time and penalty analysis. 66
3.10 Best solution comparison and run time of different loading strategies. 67

A.1 The average cost breakdown of 2E-LRP2L solutions. 79

List of Tables

2.1 Overview of PDPT approaches and transfer details. 9
2.2 TTRP approaches and transfer details. 11
2.3 ILS-SB parameters and values. 31
2.4 MIP and ILS-SB comparison. 33
2.5 ILS-SB results for the *DetB* instance set. 34
2.6 Body transfers analysis. 35
2.7 Solution values when using different set of neighborhoods. . . . 36

3.1 Overview of 2E-LRP approaches and related work in the literature. 45
3.2 Overview of 2L-CVRP approaches. 46
3.3 ILS-LR2L parameters and values. 61
3.4 Parameters and respective values for the ILS-LR. 69
3.5 Comparison of state-of-the-art methods. Best results of ILS-LR are shown in bold. 70
3.6 Pairwise T-test results. 70

Chapter 1

Introduction

Freight transportation is an indispensable component of the modern economy and responsible for supplying businesses as well as delivering goods to homes. Operational research concerning transportation problems dates back several decades. Dantzig and Ramser (1959) formulated the Vehicle Routing Problem (VRP), which proved to be the beginning of what has since become a vast research domain. The VRP involves a fleet of capacitated vehicles and a set of geographically scattered customers, each of which is associated with a certain request of goods. Respecting capacity constraints, each vehicle serves customers during its route which starts and ends at a depot. Each customer must be served by a single vehicle in such a way that minimizes the total cost of servicing all customers. These costs may correspond to total travel distance, total travel duration, the number of vehicles used or a weighted combination of these costs. Despite its simple description, in terms of computational complexity theory the VRP is an $\mathcal{NP}$-hard problem (Lenstra and Kan, 1981). What this means is that proven optimal solutions cannot be obtained in deterministic polynomial time. VRPs thus poses a challenge to practitioners and researchers alike.

Real-world applications found throughout industry are associated with additional problem-specific constraints which extend beyond the classical VRP definition. The need to bridge the gap between the classical vehicle routing definition and such practical cases gives the rise to many variants of the VRP which have been researched extensively over the years (Golden et al., 2008; Laporte, 2009; Toth and Vigo, 2014; Cuda et al., 2015). Some examples of widely addressed VRP variants which are relevant for this thesis are outlined in what follows. (*i*) Multi-depot VRP (MDVRP): the distribution network has multiple capacitated depots from which routes may start/end. (*ii*) VRP

with split delivery (SDVRP): customers may be served by multiple visits of the same or different vehicles, a situation one encounters when customers can have a demand greater than vehicle capacity. (*iii*) Two-echelon VRP (2E-VRP): when the distribution of goods from depot(s) to customers must be managed through intermediate satellite facilities, where each level of distribution is referred to as an echelon. (*iv*) VRP with pickup and delivery (PDP): each customer request comprises of a pickup location and corresponding delivery location, with precedence constraints in effect which require vehicles to visit the pickup location before the delivery location. (*v*) Open VRP (OVRP): vehicle routes can end at a location different from where they began. (*vi*) VRP with time windows (VRPTW): each customer request must be served only within a predefined period of time. Despite significant academic progress, some practical requirements continue to be ignored which restrict the applicability of current methods.

This thesis addresses VRP problems encountered by real-world companies which existing models fail to fully capture the intricacies of. These complex VRP variants generally comprise of a network where batches of requests may be transferred between vehicles at certain nodes. These transfers can take place in either the same distribution level or different distribution levels, with the array of possibilities significantly increasing the solution search space and posing huge logistical challenges to human planners. Besides routing decisions, the VRP variants this thesis introduces include other classes of decisions, namely: location and loading. Location decisions are concerned with where to transfer these batches of requests, while loading decisions aim to validate the feasibility of a certain combination of items being positioned inside vehicles. These additional and complex decisions have a direct impact on the routing part of the problems and thus cannot be ignored.

To provide these companies with methods capable of improving their current solutions, this thesis models and solves these VRP variants considering the companies' unique constraints in an attempt to bring mathematical theory closer to physical reality. When solving these variants the rich VRP literature should not be ignored and thus, whenever possible, algorithms with proven performance should be incorporated into new solution methods. Furthermore, given that the core of these variants is likely to be encountered in other applications, the algorithms developed in this thesis should be general enough so that they can be reused in other similar problem contexts.

The general approach taken in this thesis is to decomposes each problem into distinct levels where different location, loading and routing decisions must be made. These different decision levels give the problems a hierarchical structure whereby a lower-level problem is embedded within an upper-level problem. This structure allows the lower-level problem, the routing, to be generated starting

from a higher-level solution in which some variables are fixed, for instance the locations of transfers. Therefore, a heuristic may iteratively fix part of the solution and solve a simplified version of the problem. The VRP component of each problem is optimized using state-of-the-art VRP algorithms, while tailored methods are developed to handle the newly introduced constraints.

The first problem addressed in this thesis is the Vessel Swap-Body Routing problem (VSBR), a generalization of the vehicle routing problem with pickup and delivery and time windows. A slightly modified version of Chapter 2 has been submitted to the European Journal of Operational Research and is currently under review. The VSBR arises in the context of a maritime transportation company which considers freight distribution between various container terminals located throughout an inland waterway network. Requests are served by bodies: capacitated components that cannot move independently and must be towed by a vessel. Any vessel-body combination is allowed as long as the capacity of vessels is respected. The VSBR is a generalization of the OVRP and therefore vessels and bodies may end their routes at any location. Vessels may tow multiple bodies and, in order to reduce vessel handling times, bodies can be transferred between vessels at customer locations or special transfer points. Given that the VSBR considers a limited fleet of vessels and bodies, it may not always be possible to serve every request due to the time and capacity constraints in effect. When this situation occurs, requests must be outsourced and served by trucks. This outsourcing incurs large operational costs and should be avoided wherever possible. The optimization of routes and efficient use of body transfers have the potential to avoid the need to outsource requests and thus, by extension, significantly reduce the company's operation costs.

Chapter 3 then goes on to introduce the Two-Echelon Location-Routing problem with two-dimensional loading constraints (2E-LRP2L). This problem emerges from a logistics company which rents large items to their customers on a daily basis. The distribution of these items takes place over two echelons. In the first echelon large vehicles begin their routes at depots and transport the requested items to intermediate facilities. In the second echelon, smaller vehicles then transport the items from those intermediate facilities to customer. The location routing problem (LRP) is a generalization of the MDVRP, where each capacitated depot incurs a cost upon its use and where location decisions impact the objective function. Location decisions arise in the 2E-LRP2L since multiple depots and intermediate facilities are available for the company to choose from. Connecting two LRPs operating at distinct levels results in the two-echelon location routing problem (2E-LRP).

The 2E-LRP2L is a generalization of the 2E-LRP since it must also take into account two-dimensional loading restrictions in order to generate routes with

feasible load plans. The need for these loading constraints steams from the fact that solutions which only take into account the area of items often result in load plans which are physically impossible in practice. This requires the use of extra vehicles and incurs significant additional costs. Split deliveries are allowed in the first echelon since intermediate facilities have a much greater capacity than first-echelon vehicles. Figure 1.1 provides a summary of the multiple VRP variants simultaneously present in the 2E-LRP2L. The findings in Chapter 3 are a slightly modified version of a paper which has been submitted to the *Journal of Computers & Industrial Engineering* and is currently under review.

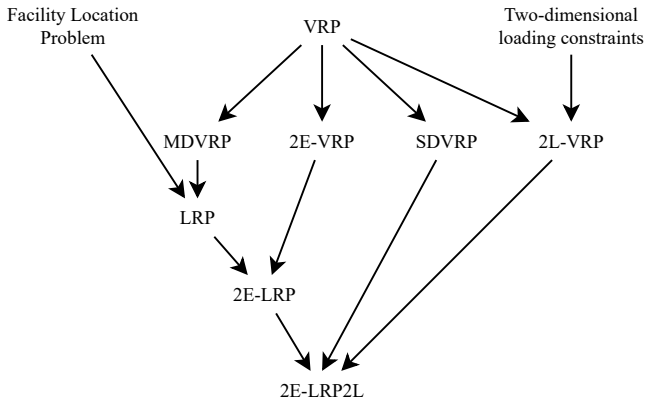


Figure 1.1: VRP variants present in the 2E-LRP2L.

This thesis brings these two very challenging problems together given that they are both VRPs with location decisions for transfers of goods and additional real-world side constraints. For each problem, instances derived from real-world data are used for computational experiments. Careful analyses are performed on the produced results to study the impact of the new constraints on solution quality and to validate the quality of the proposed methods. The approach developed for these problems will follow the same general principle, laying a foundation for future research on related VRP variants.

Chapter 2

Vessel swap-body routing problem

2.1 Introduction

The Pickup and Delivery Problem with Time Windows (PDPTW) is a generalization of the vehicle routing problem (VRP) in which requests comprise of paired pickup and delivery services subject to capacity, time window and precedence constraints. We introduce an extension of the PDPTW corresponding to certain real-world operations present in a shipping company operating in an inland waterway network. The *Vessel Swap-Body Routing problem* (VSBR) employs a fleet of homogeneous vehicles, referred to as *vessels*, which tow one or more *bodies* to serve requests within a given scheduling horizon. Each request comprises of a load volume given in containers which needs to be transported from one customer location to another. Customer locations correspond to container terminals where they retrieve the goods being shipped, thus they are not unique customer locations but locations which may be associated with multiple different customers. Containers associated with requests are loaded into homogeneous bodies: capacitated transports which only move when towed by a vessel. Vessels themselves are incapable of holding containers, but instead have a capacity limit concerning the number of towed bodies. The VSBR considers open routes, where vessels and bodies depart from a given location and may end at any location without having to return to a central depot.

Containers can be loaded into an arbitrary body, but once loaded they cannot be transferred into another. Additionally, all containers associated with a

request must be loaded into the same body. Thus, the body conducting the pickup service of a request must always carry out its corresponding delivery. Nevertheless, vessels may transfer bodies at special transfer points or customer locations. These transfer points typically correspond to central locations of the network. Bodies can be detached at transfer points without any restrictions and may be picked up at any time. However, bodies detached at a customer's location must have a service to perform at that location (either pickup or delivery of containers). When such a transfer occurs, the vessel arrives at the customer location either before or during its time window, detaches the necessary body and is then free to continue on with its route without having to wait for the customer's time window to open or the service to complete. The detached body then serves the customer as soon as its detachment is complete and the service time window is open. After the customer is served, the same or a different vessel may visit the customer to pick up the body.

Figure 2.1 illustrates a vessel-body configuration with three bodies attached to a vessel. Loading/unloading containers into/from bodies takes a certain amount of service time which is assumed to be the same for each container. Similarly, detaching/attaching a body also consumes a fixed service time independent of the configuration of attached bodies.

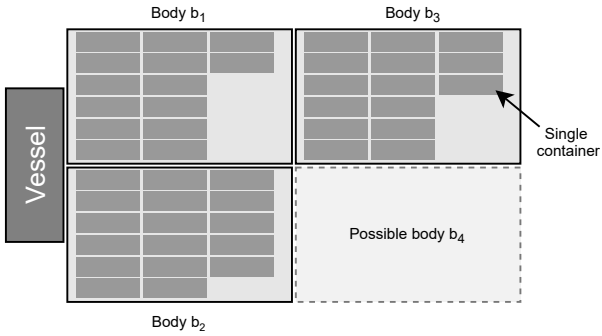


Figure 2.1: Example of a vessel towing three bodies.

Figure 2.2 illustrates a VSBR solution with body transfers in a toy network of 12 customer nodes (the triangles) and one transfer point (the blue diamond). For simplicity, time windows and travel times are omitted from the figure. Customers with pickup services (p_i) are depicted as upward triangles, while downward triangles correspond to delivery services (d_i). The route of each body is illustrated by means of colored directed edges, while nodes serviced by a given body have the same color. The vessels responsible for towing bodies have their routes illustrated with distinct dashed edges and their initial positions are

depicted as gray squares. Bodies b_1 and b_4 begin their routes attached to vessel v_1. Meanwhile, bodies b_2 and b_3 begin attached to vessel v_2 and v_3, respectively. Note that body b_4 serves the pickup and delivery services of requests 2, 3 and 5. To service all of these requests, body b_4 is towed by three separate vessels. Towed by vessel v_1, b_4 serves p_2 and p_3 before being detached at the transfer point and attached to v_2. After serving d_2, body b_4 is detached once again at customer p_5 and serves the request while detached. Finally, b_4 is attached one final time and towed by v_3 in order to deliver the final two requests. Note that if body transfers were not allowed, the vessel that picks up service p_3 would need to traverse almost the entire network to serve its paired service d_3.

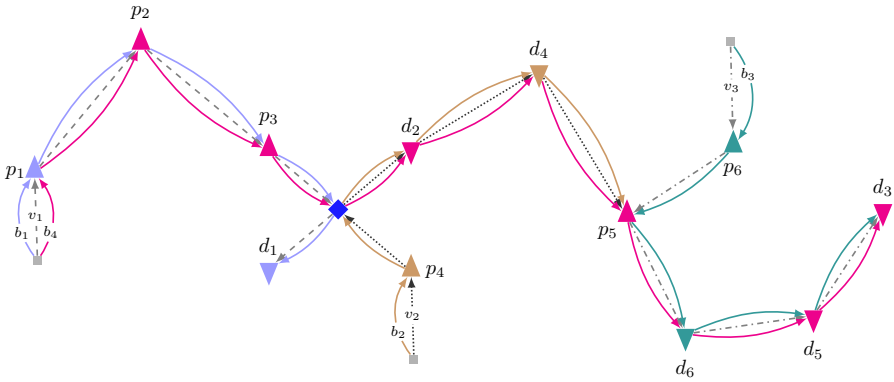


Figure 2.2: VSBR solution with body transfers.

There is no fixed assignment for vessels and bodies, therefore a body may be towed by any vessel. Moreover, customers may be visited by all vessels and bodies. When en route, vessels are assumed to travel at the same constant speed, regardless of the number of bodies connected and containers loaded. Requests may share the same customer location for their pickup and/or delivery services. A single request typically comprises of a paired pickup and delivery service, but within the VSBR's limited scheduling horizon it is possible that a request only corresponds to a single pickup or delivery service. This is the case when a pickup is scheduled towards the end of the scheduling horizon, while its corresponding delivery will take place during the subsequent scheduling horizon. The fleet of vessels and bodies available to serve requests is considered fixed and given a priori. As a result of this hard constraint, it may be impossible to serve every request within its time window given the capacity constraints of the available fleet. In this case, requests are outsourced and assumed to be served by trucks. Outsourcing requests in this manner incurs high costs and should thus be avoided whenever possible. The objective function of the VSBR

is designed to minimize the sum of (i) vessel travel costs and (ii) outsourcing costs.

The following section of this chapter presents an overview of related research and their main differences and similarities to the VSBR. Section 2.3 introduces the necessary notation and presents a mixed integer programming formulation for the VSBR. A heuristic is also introduced in order to generate high-quality solutions within reasonable computational runtimes for instances of realistic size (Section 2.4). A set of tailored neighborhoods is embedded in the proposed heuristic to handle body transfers. Body transfers are an intrinsic element of the VSBR, however their inclusion significantly increases the size of the solution space and poses a significant logistical challenge for human operators who are unable to exploit the full potential of these transfers. In order to assess the impact of body transfers on solution quality, we report the results of computational experiments in Section 2.5. Given the lack of benchmark instances for the VSBR, instance sets are generated based on real-world data. Section 2.6 concludes the chapter and outlines some directions for future research.

2.2 Related work

The combination of multiple starting locations, open routes, numerous vessel-body assignments and body transfers makes the VSBR a unique routing problem. Nevertheless, other routing problems can be identified in which specific requests must be scheduled considering transfers and different vehicle combinations. The most similar problems are the pickup and delivery problem with transshipment (PDPT) and the VRP with trailers and transshipment. Distinctions can be made regarding various characteristics such as what is being transferred, transport combinations and accessibility constraints. This section presents an overview of these related problems in order to identify some similarities and key differences with respect to the VSBR.

2.2.1 Pickup and delivery problems with transshipment

Several variants of pickup-and-delivery problems (PDPs) exist and have been studied extensively. Important surveys of these variants have been compiled by Cordeau et al. (2008), Parragh et al. (2008) and Battarra et al. (2014). A generalization of the PDP arises when items are permitted to be transferred at so-called transfer points in order to minimize vehicle routing costs. Shiri et al. (2020) studied the impact of allowing transfers in pickup-and-delivery systems with different modeling (objective functions), system design (number

of locations and transfer points) and operational parameters (capacity, cost and number of vehicles). This study revealed the gains that are possible in many different scenarios when permitting transfers to take place.

Transfers significantly increase the difficulty of these problems as they introduce precedence and synchronization constraints between multiple routes. If an item is transferred at a transfer point, the vehicle which picks up the item must visit the transfer point after the vehicle which dropped it off. Noting the interdependence of routes in the PDPT and the difficulty of efficiently checking the feasibility of candidate transfers, Masson et al. (2013a) proposed a method which checks the feasibility of solutions in constant time. For a more thorough review of multiple synchronization constraints in VRPs and their applications we refer interested readers to the survey by Drexl (2012).

Table 2.1 provides an overview of recent research concerning PDPT variants which focuses on their transfer characteristics. In the PDPT, a request i given by the pair (r_i^P, r_i^D) and served by a single vehicle may be divided into two requests served by two vehicles given by pairs (r_i^P, t) and (t, r_i^D), where t is a transfer point. Most of the papers listed in Table 2.1 transfer requests (goods or passengers) at most once at pre-determined transfer points, while routes have fixed start and end points. Variants differ with respect to time windows (PDPTWT), split delivery, maximum route duration and hard time windows at transfer points.

Table 2.1: Overview of PDPT approaches and transfer details.

	Problem	Method	What is transferred?	#Transfers	Transfer locations	Start and end locations	Extra constraints
Mitrović-Minić and Laporte (2006)	PDPTWT	H	Request	1/r	TP	F	
Cortés et al. (2010)	PDPTWT	E	Request	1/r	TP	F	Time window at transfer points
Takoudjou et al. (2012)	PDPTWT	E	Request	1/r	TP	F	
Qu and Bard (2012)	PDPTWT	H	Request	1/r	TP	F	
Masson et al. (2013b)	PDPTWT	H	Request	1/r	TP	F	
Masson et al. (2014)	PDPTWT	H	Request	1/r	TP	F	Max route duration and travel time
Rais et al. (2014)	PDPT + PDPTWT	E	Request	1/r	TP	F and NF	Split delivery
Danloup et al. (2018)	PDPTWT	H	Request	1/r	TP	F	
Zhang et al. (2020)	PDPTWT	E and H	Request	1/r	TP	F	
Wolfinger (2021)	PDPTWSLT	E and H	Request	1+/r	TP	F	Split delivery
This chapter	VSBR	E and H	Body containing multiple requests	one or more per body	TP and customer locations	NF	

PDPTWSLT: PDPT with time windows and split deliveries.
(N)F: (Not)Fixed locations to start and end routes.
1(+)/r: one (or more) transfer per request.
TP: transfer points.
E/H: Exact/Heuristic approach.

In contrast to the PDPT, transfers in the VSBR are not conducted at an individual request level. Instead only entire bodies, which contain multiple requests, may be transferred. The VSBR also allows bodies to be transferred multiple times, either at transfer points or customer locations, and their routes

do not need to end at a fixed location. For example, in Figure 2.2, body b_4 is transferred twice. The transfer takes place when b_4 is detached after picking up two requests, which leads to multiple changes in the route of the vessel to which it is subsequently attached. These differences concerning what is being transferred and how many requests are involved in each transfer makes it challenging to incorporate efficient transfer insertion methods proposed for PDPTs into the VSBR.

2.2.2 Vehicle routing problems with trailers and transshipment

The VSBR exhibits some characteristics of truck-and-trailer routing problems (TTRPs). These represent a generalization of the VRP where lorries may extend their capacity by towing at most one trailer: a capacitated component that depends on lorries to move. Customers may be visited by either a lorry or a lorry-trailer combination (LTC). Due to accessibility constraints, some customers can only be visited by lorries and are referred to as lorry customers (LCs), while others can be visited by lorries with or without a trailer and are referred to as trailer customers (TCs). In order to visit LCs, an LTC may detach its trailer at a transfer point and serve LCs with a sub-tour that starts and ends at this transfer point. Loads may be transferred between the lorry and its respective trailer at a transfer point. At the end of the sub-tour, the lorry then picks up the trailer and continues on with its route which must end at a depot. The goal of the TTRP is to determine routes for lorries and LTCs so that every customer is served and routing costs are minimized while respecting loading capacities, accessibility constraints and time windows (if any). The surveys conducted by Prodhon and Prins (2014) and Cuda et al. (2015) contain sections on TTRPs, while Drexl (2013) has shown how several VRPs with different synchronization constraints may be modeled as variants of the TTRP.

Recent research regarding the TTRP is documented in Table 2.2, focusing on transshipment characteristics. A few papers considered the TTRP with time windows (TTRPTW), while Drexl (2020) is the only one to consider the TTRPTW with pickup and delivery. In addition to the constraints already introduced, routes may be subject to maximum duration, lorries may have fixed trailer assignments (meaning trailer transfers between lorries are not allowed), or it may be possible for a trailer to be detached multiple times during an LTC route. The papers documented in Table 2.2 typically consider transfers only at depots and TC locations, while the routing cost does not depend on the vehicle used. However, some papers include dedicated transfer points and edge costs depending on the vehicle used (only lorry or LTC). Considering the TTRP proposed by Chao (2002) as the baseline, variants may consider an unlimited

Table 2.2: TTRP approaches and transfer details.

Reference	Problem	Method	Transfer location	Edge cost	Vehicles fleet	Extra conditions
Chao (2002)	TTRP	E and H	D and TC	Same	L	
Scheuerer (2006)	TTRP	H	D and TC	Same	L	
Lin et al. (2009)	TTRP	H	D and TC	Same	L	
Caramia and Guerriero (2010)	TTRP	E and H	D and TC	Same	L	
Lin et al. (2011)	TTRPTW	H	D and TC	Same	U	
Derigs et al. (2013)	TTRP + TTRPTW	H	D and TC	Same	L	Allow/forbid transfers between lorries and trailers
Villegas et al. (2013)	TTRP	E and H	D and TC	Same	L	
Parragh and Cordeau (2017)	TTRPTW	E and H	D and TC	Same	L	
Rothenbächer et al. (2018)	TTRP + TTRPTW	E	TC and TP	VD	L	Heterogeneous fleet
Toffolo et al. (2018)	TTRP	H	TP	VD	U	No load transfer
Drexl (2020)	PD-TTRPTW	H	TP	VD	U	Heterogeneous fleet
This chapter	VSBR	E and H	TP and any customer location	Same	L	

D: depot.
TC: trailer customer.
VD: vehicle-dependent.
TP: transfer points.
U/L: Unlimited/Limited fleet of vehicles.
E/H: Exact/Heuristic approach.

and/or heterogeneous fleet of vehicles and forbid load transfer between lorries and their corresponding trailers.

The VSBR also considers a vehicle which moves independently while towing a capacitated component that can be detached at special locations. However, unlike TTRPs, the VSBR considers neither customer accessibility constraints nor fixed assignments concerning vessels and bodies. Therefore, any vessel-body combination is valid and may visit any customer location. A vessel may also tow multiple bodies at the same time. For TTRPs, trailers are used to increase overall capacity, minimize total routing costs, and reduce the number of necessary vehicles. Transfers are thus employed as a mechanism to temporarily reduce vehicle size and access customers at restricted locations. In the VSBR by contrast, bodies are the only capacitated vehicles capable of transporting containers and transfers are used to minimize travel and outsourcing costs. When one takes into account these key differences between TTRPs and the VSBR, the development of a dedicated approach for the VSBR is clearly required.

2.3 Notation and problem formulation

The VSBR considers a set of customer locations, a set V of vessels, a set B of bodies and a set R of requests. Each request $r \in R$ is associated with a load

volume ρ_r that must be picked up from one customer and delivered to another. The pickup should be conducted within time window $[t_r^{P-}, t_r^{P+}]$ and the delivery should be conducted within time window $[t_r^{D-}, t_r^{D+}]$. The VSBR considers serving these requests within a predetermined time horizon. Depending on the start and end of this horizon and the time windows of individual requests, three types of request are possible:

R^1 is the set of requests with both pickup and delivery services,
R^2 is the set of requests with only a delivery service,
R^3 is the set of requests with only a pickup service, where $R = R^1 \cup R^2 \cup R^3$.

Requests are expected to be served by vessels using the bodies attached to them. However, when a request cannot be served by a vessel within its time window, it is outsourced and assumed to be served by trucks at a high cost π. The service time of each request r is proportional to ρ_r and is denoted by s_r.

Each vessel $v \in V$ can travel with at most Q^V bodies, each having capacity Q^B. These bodies can be attached to or detached from vessels at any transfer point or a service location to be served by these bodies. The time needed for attaching or detaching a body is T^C. When transferring bodies at a service location, the detaching/attaching of such bodies may occur before/after the start/end of the service's time window. Bodies can therefore end a scheduling horizon detached from vessels. As a result, bodies may also start a scheduling horizon detached.

2.3.1 A modified network construction

In order to formulate the VSBR we introduce a network $G = (N, A)$, as depicted in Figure 2.3. The network comprises of node set $N = I \cup S \cup J^0 \cup \{*\}$ and arc set A. We associate each transfer, pickup and delivery service with two nodes due to the possibility of detaching a body at the corresponding service location and reattaching it later to the same or another vessel. During the timespan between detaching and reattaching a body to the same vessel, this vessel may conduct services associated with other requests.

Let us define $I = P \cup P' \cup D \cup D'$, depicted as circles in the network, such that P and P' denote the sets of nodes associated with pickup services, while D and D' denote the sets of nodes associated with delivery services. Note that if the containers associated with a request are already in a body, then we only need the delivery nodes for this request (see request 2 in Figure 2.3). Similarly, if a request only has a pickup service, then we only need to create pickup nodes for this request (see request n in Figure 2.3).

Set S, illustrated by diamonds, contains the transfer nodes which are not customer nodes. Let us assume that each body b can be detached at a transfer node s at most m times. For each visit of b to s, we create two transfer nodes, say s^1 and s^2, whose physical locations are identical to those of s. Only body b can be detached at s^1, while b then moves from s^1 to s^2 detached from any vessel before it can be picked up again from s^2. Let $S_b = S_b^1 \cup S_b^2$ be the set of such transfer nodes created for body b and let $s_k^2 \in S_b^2$ denote the second copy of node $s_k^1 \in S_b^1$. Then $S = \bigcup_{b \in B} S_b$.

Set J^0 contains a node o_v (o_b) associated with the initial location of each vessel v (body b). Vessels do not need to return to their initial location. In order to simplify the modeling, we introduce a dummy node "$*$" to the network and ensure that every vessel and body ends their route at this dummy node. Initial and final location nodes are represented by squares in Figure 2.3.

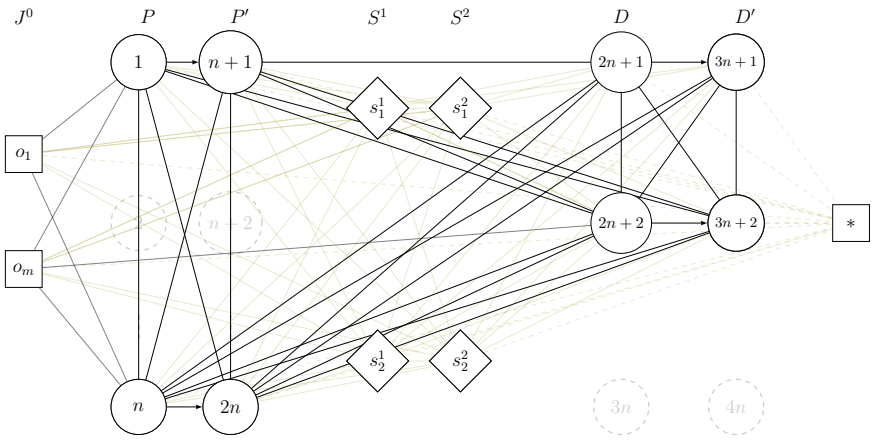


Figure 2.3: A modified network representation for the model

Let $r_i \in R$ denote the request associated with $i \in I$. We then construct arc set $A = A^1 \cup A^2 \cup A^3 \cup A^4 \cup A^5 \cup A^*$ as follows:

$$
\begin{aligned}
A^1 &= \{(i,j) : i \in J^0 \cup S, j \in N, i \neq j\},\\
A^2 &= \{(i,i+n) : i \in P\} \cup \{(i,j) : i \in P, j \in N \setminus \{u \in I : r_u = r_i\}\},\\
A^3 &= \{(i,i+n) : i \in P'\} \cup \{(i,j) : i \in P', j \in N \setminus \{u \in I : r_u = r_i\}\},\\
A^4 &= \{(i,i+n) : i \in D\} \cup \{(i,j) : i \in D, j \in N \setminus \{u \in I : r_u = r_i\}\},\\
A^5 &= \{(i,i+n) : i \in D'\} \cup \{(i,j) : i \in D', j \in N \setminus \{u \in I : r_u = r_i\}\},\\
A^* &= \{(i,*), i \in N \setminus \{*\}\}.
\end{aligned}
$$

The travel time and the travel cost for each arc $(i,j) \in A$ is denoted by e_{ij} and c_{ij}, respectively. The travel cost is calculated as $c_{ij} = e_{ij} * \epsilon$, where ϵ is a coefficient which expresses cost per time unit and is used to convert time into cost. For each dummy arc $(i,j) \in A^*$, we set $e_{ij} = c_{ij} = 0$. Let L denote the set of physical locations and $l_i \in L$ denote the location associated with node $i \in N$. Then $e_{ij} = c_{ij} = 0$ and $\forall (i,j) \in A : l_i = l_j$, whereas $e_{i(i+n)} = s_{r_i}$ for $i \in P \cup D$. We further define $A^D = A^* \cup \{(i,j) \in A : l_i = l_j\}$ as the set of arcs bodies can traverse without being attached to any vessel. If node $i \in D \cup D'$ is associated with a request whose pickup service is already completed (meaning $r_i \in R_2$), then $b_i = b_{r_i} \in B$ denotes the body that holds the containers of this request.

2.3.2 Decision variables

For the introduced model we define the following sets of decision variables:

$z_{ij}^v = 1$ if vessel $v \in V$ traverses arc $(i,j) \in A$, 0 otherwise.
$\beta_{ij}^b = 1$ if body $b \in B$ traverses arc $(i,j) \in A$, 0 otherwise.
$\omega_{ij}^{bv} = 1$ if body $b \in B$ is attached to vessel $v \in V$ and traverses arc $(i,j) \in A$, 0 otherwise.
$x_{ij}^{br} = 1$ if the containers of request $r \in R$ traverse arc $(i,j) \in A$ inside body $b \in B$, 0 otherwise.
$\delta_{ij}^b = 1$ if body $b \in B$ traverses arc $(i,j) \in A^D$ without being attached to any vessel, 0 otherwise.
$\gamma^r = 1$ if request $r \in R$ is served by vessels, 0 otherwise.
$\tau_i^{Av} \geq 0$ is the arrival time of vessel $v \in V$ at node $i \in N$.
$\tau_i^{Dv} \geq 0$ is the departure time of vessel $v \in V$ from node $i \in N$.
$\tau_i^{BAb} \geq 0$ is the arrival time of body $b \in B$ at node $i \in N$.
$\tau_i^{BDb} \geq 0$ is the departure time of body $b \in B$ from node $i \in N$.
$y_{ij}^v = 1$ if $i \in N$ precedes $j \in N$ (not necessarily immediately) during the trip of vessel v, 0 otherwise.
$\varphi_{ij}^b = 1$ if $i \in N$ precedes $j \in N$ (not necessarily immediately) during the trip of body b, 0 otherwise.

2.3.3 A mixed integer programming formulation for the VSBR

The objective function: Equation (2.1) minimizes the total cost of vessel routes plus the outsourcing cost for requests served by trucks.

$$\min \sum_{v \in V} \sum_{(i,j) \in A} c_{ij} z_{ij}^v + \sum_{r \in R} \pi(1 - \gamma^r) \tag{2.1}$$

Routing of vessels and bodies: By Constraints (2.2)-(2.4) and (2.10)-(2.13), each vessel starts its journey from the node associated with its initial location and ends it at the dummy node without any sub-tours. Constraints (2.5)-(2.7) and (2.14)-(2.17) function analogously for bodies. Constraints (2.8) and (2.9) ensure a body either traverses an arc while attached to a single vessel or not attached to any vessel (if possible), thereby disabling the possibility that a body is simultaneously attached to two or more vessels.

$$\sum_{j:(o_v,j)\in A} z^v_{o_v j} = 1, \qquad \forall v \in V \tag{2.2}$$

$$\sum_{j:(j,*)\in A} z^v_{j*} = 1, \qquad \forall v \in V \tag{2.3}$$

$$\sum_{j:(j,i)\in A} z^v_{ji} = \sum_{j:(i,j)\in A} z^v_{ij}, \qquad \forall v \in V, i \in I \cup S \tag{2.4}$$

$$\sum_{j:(o_b,j)\in A} \beta^b_{o_b j} = 1, \qquad \forall b \in B \tag{2.5}$$

$$\sum_{j:(j,*)\in A} \beta^b_{j*} = 1, \qquad \forall b \in B \tag{2.6}$$

$$\sum_{j:(j,i)\in A} \beta^b_{ji} = \sum_{j:(i,j)\in A} \beta^b_{ij}, \qquad \forall b \in B, i \in I \cup S \tag{2.7}$$

$$\beta^b_{ij} = \delta^b_{ij} + \sum_{v\in V} \omega^{vb}_{ij}, \qquad \forall b \in B, (i,j) \in A^D \tag{2.8}$$

$$\beta^b_{ij} = \sum_{v\in V} \omega^{vb}_{ij}, \qquad \forall b \in B, (i,j) \in A \setminus A^D \tag{2.9}$$

Sub-tour elimination for vessel routes: For elimination of sub-tours, we utilize Constraints (2.10) - (2.13), which are shown to provide tight bounds in a comparative study by (Öncan et al., 2009). This group of sub-tour elimination constraints have also been adopted in recent and relevant PDPT and PDPTW

studies (Rais et al., 2014; Zhang et al., 2020; Christiaens et al., 2020).

$$z_{ij}^v \leq y_{ij}^v, \qquad \forall (i,j) \in A, v \in V \tag{2.10}$$

$$y_{ij}^v + y_{ji}^v = 1, \qquad \forall (i,j) \in A : (j,i) \in A, v \in V \tag{2.11}$$

$$y_{ij}^v = 1, \qquad \forall (i,j) \in A : (j,i) \notin A, v \in V \tag{2.12}$$

$$y_{ij}^v + y_{jl}^v + y_{li}^v \leq 2, \qquad \forall (i,j),(j,l),(l,i) \in A, v \in V \tag{2.13}$$

Sub-tour elimination for body routes: Constraints (2.14) - (2.17) are analogous to Constraints (2.10) - (2.13).

$$\beta_{ij}^b \leq \varphi_{ij}^b, \qquad \forall (i,j) \in A, b \in B \tag{2.14}$$

$$\varphi_{ij}^b + \varphi_{ji}^b = 1, \qquad \forall (i,j) \in A : (j,i) \in A, b \in B \tag{2.15}$$

$$\varphi_{ij}^b = 1, \qquad \forall (i,j) \in A : (j,i) \notin A, b \in B \tag{2.16}$$

$$\varphi_{ij}^b + \varphi_{jl}^b + \varphi_{li}^b \leq 2, \qquad \forall (i,j),(j,l),(l,i) \in A, b \in B \tag{2.17}$$

Capacity restrictions: Constraints (2.18) and (2.19) ensure that capacities are not exceeded for bodies and vessels, respectively.

$$\sum_{r \in R} q_r x_{ij}^{br} \leq Q^B \beta_{ij}^b, \qquad \forall b \in B, (i,j) \in A \tag{2.18}$$

$$\sum_{b \in B} \omega_{ij}^{bv} \leq Q^V z_{ij}^v, \qquad \forall v \in V, (i,j) \in A \tag{2.19}$$

Serving requests: If a request is served, Constraints (2.20) ensure that all service arcs associated with that request are traversed by a body holding the containers of the request. Constraints (2.21) and (2.23) ensure that the delivery service is conducted by the same body as the pickup service for each request. Constraints (2.22) ensure that the flow between the first and the last service node associated with each request is preserved for each body. Constraints (2.24) prevent the

containers associated with a request from being split across multiple bodies.

$$\sum_{b\in B} x^{br_i}_{i(i+n)} = \gamma^{r_i}, \qquad \forall i \in P \cup D \tag{2.20}$$

$$x^{br_i}_{i(i+n)} = x^{br_i}_{(i+2n)(i+3n)}, \qquad \forall i \in P : r_i \in R^1, b \in B \tag{2.21}$$

$$\sum_{k\in N: k\neq i,(j,k)\in A} x^{br_i}_{jk} = \sum_{k\in N: k\neq i+3n,(k,j)\in A} x^{br_i}_{kj}, \quad \forall i \in P : r_i \in R^1, \; b \in B, j \in N \setminus \{*, i, i+3n\} \tag{2.22}$$

$$x^{b_{r_i} r_i}_{i(i+n)} = \gamma^{r_i}, \qquad \forall i \in D : r_i \in R^2 \tag{2.23}$$

$$\sum_{j:(j,i)\in A} x^{b_1 r}_{ji} + \sum_{j:(i,j)\in A} x^{b_2 r}_{ij} \leq 1, \qquad \forall i \in N, r \in R, b_1, b_2 \in B : b_1 \neq b_2 \tag{2.24}$$

Updating arrival/departure times: Let $T^{max} = \max_{r\in R} t^{D+}_r$. Constraints (2.25) update the arrival and departure times of a vessel at the end nodes of an arc visited by that vessel. Constraints (2.26) function similarly for bodies.

$$\tau^{Av}_j \geq \tau^{Dv}_i + e_{ij} z^v_{ij} - T^{max}(1 - z^v_{ij}), \qquad \forall (i,j) \in A, v \in V \tag{2.25}$$

$$\tau^{BAb}_j \geq \tau^{BDb}_i + e_{ij} \beta^b_{ij} - T^{max}(1 - \beta^b_{ij}), \qquad \forall (i,j) \in A, b \in B \tag{2.26}$$

Departures after arrivals: Constraints (2.27) and (2.28) ensure that the departure time from a node is not earlier than the arrival time at that node for vessels and bodies, respectively.

$$\tau^{Dv}_i \geq \tau^{Av}_i, \qquad \forall i \in N, v \in V \tag{2.27}$$

$$\tau^{BDb}_i \geq \tau^{BAb}_i, \qquad \forall i \in N, b \in B \tag{2.28}$$

Attaching and detaching times: If a body enters node i attached to a vessel but leaves the node detached from that vessel, Constraints (2.29) add the detaching time to the vessel's departure time label from node i. Similarly, if a body enters the node detached from any vessel but leaves attached to a vessel, Constraints (2.30) add the attaching time to the vessel's departure time label from that node.

$$\tau_i^{Dv} \geq \tau_i^{Av} + T^C + T^{max} \sum_{j:(j,i)\in A} w_{ji}^{vb} - T^{max} \sum_{j:(i,j)\in A} w_{ij}^{vb} - T^{max}, \quad \forall b \in B, v \in V, i \in N : i \neq o_b, * \tag{2.29}$$

$$\tau_i^{Dv} \geq \tau_i^{Av} + T^C - T^{max} \sum_{j:(j,i)\in A} w_{ji}^{vb} + T^{max} \sum_{j:(i,j)\in A} w_{ij}^{vb} - T^{max}, \quad \forall b \in B, v \in V, i \in N : i \neq o_b, * \tag{2.30}$$

Vessel arrives before body: Constraints (2.31)-(2.34) establish the relation between the arrival and departure times of attached vessel-body pairs entering or leaving a node.

$$\tau_i^{BAb} \geq \tau_i^{Av} + T^{max} \sum_{j:(j,i)\in A} w_{ji}^{vb} - T^{max}, \qquad \forall i \in N, v \in V, b \in B \tag{2.31}$$

$$\tau_i^{BDb} \geq \tau_i^{Dv} + T^{max} \sum_{j:(j,i)\in A} w_{ji}^{vb} - T^{max}, \qquad \forall i \in N, v \in V, b \in B \tag{2.32}$$

$$\tau_i^{Av} \geq \tau_i^{BAb} + T^{max} \sum_{j:(i,j)\in A} w_{ij}^{vb} - T^{max}, \qquad \forall i \in N, v \in V, b \in B \tag{2.33}$$

$$\tau_i^{Dv} \geq \tau_i^{BDb} + T^{max} \sum_{j:(i,j)\in A} w_{ij}^{vb} - T^{max}, \qquad \forall i \in N, v \in V, b \in B \tag{2.34}$$

Time windows: The time windows at service nodes are respected via Constraints (2.35)-(2.38).

$$\tau_i^{BDb} \geq t_{r_i}^{P-} \gamma^{r_i}, \qquad \forall i \in P, b \in B \tag{2.35}$$

$$\tau_i^{BDb} \geq t_{r_i}^{D-} \gamma^{r_i}, \qquad \forall i \in D, b \in B \tag{2.36}$$

$$\tau_i^{BAb} \leq t_{r_i}^{P+} \gamma^{r_i}, \qquad \forall i \in P', b \in B \tag{2.37}$$

$$\tau_i^{BAb} \leq t_{r_i}^{D+} \gamma^{r_i}, \qquad \forall i \in D', b \in B \tag{2.38}$$

Transfers at non-client nodes: Constraints (2.39) ensure that a body is only detached at one of its own transfer nodes. Constraints (2.40) and (2.41) require body b to be detached if one of its transfer nodes is visited.

$$\sum_{j:(j,s)\in A} w_{js}^{vb} = \sum_{j:(s,j)\in A} w_{sj}^{vb}, \qquad \forall b \in B, v \in V, s \notin S_b \tag{2.39}$$

$$\sum_{j:(j,s_k^1)\in A} \beta_{js_k^1}^{b} = \delta_{s_k^1 s_k^2}^{b}, \qquad \forall b \in B, s_k^1 \in S_b^1 \tag{2.40}$$

$$\sum_{j:(s_k^2,j)\in A} \beta_{s_k^2 j}^{b} = \delta_{s_k^1 s_k^2}^{b}, \qquad \forall b \in B, s_k^2 \in S_b^2 \tag{2.41}$$

The binary and non-negativity restrictions on the decision variables are expressed via Constraints (2.42) and (2.43).

$$z_{ij}^{v}, y_{ij}^{v}, \beta_{ij}^{b}, \omega_{ij}^{bv}, x_{ij}^{br}, \delta_{ij}^{b}, \gamma^{r}, \varphi_{ij}^{b} \in \{0,1\}, \quad \forall v \in V, b \in B, r \in R, (i,j) \in A \tag{2.42}$$

$$\tau_i^{Av}, \tau_i^{Dv}, \tau_i^{BAb}, \tau_i^{BDb} \geq 0, \qquad \forall v \in V, b \in B, i \in N \tag{2.43}$$

Assumption: In order to keep the model straightforward to follow, we make the following practical assumption. At the beginning of the scheduling horizon, we assume that the attaching or detaching of bodies to/from vessels at their starting point is performed a priori and thus the routing can begin immediately at time zero. This assumption has the following impact: if body b is attached to vessel v_1 but will be taken by vessel v_2 from its start location then the model will not count the time to detach b from v_1, but it will count the time to attach b to v_2.

Valid inequalities: Constraints (2.44) and (2.45) ensure that the containers associated with a request are not inside a body when arriving at the pickup service node or leaving the delivery service, respectively.

$$\sum_{b\in B} \sum_{j:(j,i)\in A} x_{ji}^{br_i} = 0, \qquad \forall i \in P \tag{2.44}$$

$$\sum_{b\in B} \sum_{j:(i,j)\in A} x_{ij}^{br_i} = 0, \qquad \forall i \in D' \tag{2.45}$$

During preliminary experiments using an off-the-shelf solver, Constraints (2.44) and (2.45) were helpful in decreasing the solving time of the MIP model. Moreover in these experiments, we observed that the problem very quickly becomes intractable for the solver even for some small instances. In order to handle larger instances in a reasonable amount of time, we introduce a heuristic algorithm in the following section.

2.4 A heuristic for the VSBR

The purpose of developing a tailored heuristic for the VSBR is to generate high-quality solutions within reasonable runtimes. Preliminary experiments revealed that this could not be achieved by means of the proposed integer formulation. Given the many decisions involved when solving the VSBR and the necessity of efficiently exploring the solution space, a heuristic should be able to improve a solution by applying changes focused on different decision levels of the problem. When generating new solutions one can consider two consecutive decisions: (i) decide and fix at which locations to transfer bodies and then (ii) focus on PDPTW optimization. However, inserting a body transfer into a solution can be very disruptive as the transfer may impact multiple vessels' routes as well as the assignment of containers to bodies.

Given an initial solution, the Iterated Local Search (ILS) method introduced by Lourenço et al. (2003) generates neighbor solutions by applying a perturbation method followed by a local search. This method has proven to be widely applicable to routing and scheduling problems (Lourenço et al., 2019). We therefore adapted ILS for addressing the VSBR, henceforth referred to as ILS-SB, in which the perturbation method consists of a transfer phase where body transfers are inserted or removed. The local search phase of ILS-SB then fixes these transfers before solving a PDPTW. Each newly generated solution is potentially accepted based on the Late Acceptance Hill-Climbing (LAHC) method introduced by Burke and Bykov (2017). LAHC benefits from the fact that there is only one parameter which requires calibration.

The basic framework of ILS-SB is provided by way of Algorithm 4, while details of each individual component are provided in subsequent sections. First, the algorithm receives as input an initial solution s and the maximum length for the LAHC list, denoted by l (line 1). The LAHC list is initialized with the initial solution value (line 3), resulting in solutions generated during the first $l-1$ iterations to be accepted. At each iteration, ILS-SB generates a neighbor solution s' by performing a transfer and routing phase (lines 6-7). Solution s' replaces the current best solution if it improves upon s^* (lines 8-9). While the LAHC list is not full, value $f(s')$ is added to the list and s' replaces the incumbent solution s (lines 10-12). Otherwise, s' only replaces s if its value improves upon the considered entry in the LAHC list, in which case the considered entry is also replaced with $f(s')$ (lines 13-15). The best solution is returned (line 17) when the algorithm reaches either of the two stop criteria: the maximum number of iterations without improvement ($\#maxIt$) or the time limit.

Algorithm 1: ILS-SB framework.

```
Input: initial solution s, list size l;
s* ← s;
AcceptL ← {f(s)};
i ← 0;
while #maxIt or time limit is not reached do
    s_t ← transfer-phase(s);
    s' ← routing-phase(s_t);
    if f(s') < f(s*) then
        s* ← s';
    if length(AcceptL) < l then
        s ← s';
        AcceptL ← AcceptL ∪ f(s');
    else if f(s') < AcceptL[i mod l] then
        s ← s';
        AcceptL[i mod l] ← f(s');
    i++;
return s*;
```

2.4.1 Acceptance criterion

The VSBR's objective function aims to fulfill as many requests as possible with vessels while minimizing fuel consumption. Thus, the VSBR minimizes the travel cost plus the outsourcing costs incurred when requests are served by trucks. When using the objective function in Equation (2.1), referred to as OF^1, it is common to find different solutions with equal objective values. To distinguish between two solutions when this situation occurs, a secondary objective function, referred to as OF^2 and calculated by Equation (2.46), is used every time the first objective results in a tie. This secondary objective function minimizes the total *operational cost* plus outsourcing costs. The operational cost of a vessel is given by the time the last request was served minus the starting time of the vessel multiplied by the cost coefficient per unit time. Therefore, OF^2 takes into account everything already present in OF^1 in addition to request servicing, detaching/attaching and waiting times. As a result, a solution with identical travel cost but lower operational cost is preferable.

$$\min \sum_{v \in V} (\tau_*^{Av} - \tau_{o_v}^{Av})\epsilon + \sum_{r \in R} \pi(1 - \gamma^r) \tag{2.46}$$

2.4.2 Initial solution

At the start of the scheduling horizon, all vessels are located at a container terminal and have either zero or more bodies attached. Similarly, bodies have an initial location, may be initially connected to a vessel and may have zero or more containers already loaded. Since ILS-SB has methods dedicated to handling body transfers, we opt to quickly generate an initial solution which does not include such transfers. Thus, the proposed initial solution method generates a solution in which bodies remain attached to their initially assigned vessel and no transfers take place. This means that bodies which began detached will not be used, while a given vessel which serves a pickup service will also perform the corresponding delivery. For a feasible solution, all capacity and temporal constraints must be satisfied and transportation requests which cannot be served by vessels are placed into the set of unscheduled requests U.

A solution for the VSBR is represented by a set of routes for all vessels $v \in V$, where each route is defined by a sequence of services, the location of each of those, the body used to fulfill the service and any other attached bodies. An initial solution is generated by inserting requests one by one into body routes. Requests are iterated over in a random order and inserted into the best position considering all body routes. The request insertion method designed to construct the initial solution is detailed in Section 2.4.3. To avoid poor quality initial solutions, ι candidates are produced from which the best is selected for ILS-SB.

2.4.3 Best insertion method for requests

The best insertion method attempts to insert every unscheduled request at the best position of a given solution. Given the sequence of visited nodes in a body/vessel route, an insertion position is given by a node n where the request is inserted into the route immediately after n. The list of unscheduled requests U is iterated over in one of the following three ways, which are selected with equal probability: (i) random order, (ii) request r with the earliest t_r^{P-} first or (iii) request r with the earliest t_r^{D-} first.

For each request r_i ($i \in R_1 \cup R_3$), the pickup service r_i^p is first inserted into the best position considering every position in the routes of all available bodies. Once inserted, the corresponding delivery service r_i^d is inserted into the best position of body route $b_{r_i^p}$. If $i \in R_2$, meaning that the pickup service was already loaded into a specific body b_j, only the positions in b_j will be considered for the insertion of r_i^d. When no feasible insertion position can be found for a pickup or delivery service, the complete request r_i is added to the set of unscheduled requests U.

Best insertion methods for PDPTWs are often costly in terms of processing time given the need to evaluate the solution after the insertion of a request into each position. To accelerate the method, we follow the efficient feasibility testing method for request insertion proposed by Savelsbergh (1992). Before the insertion of a request, both forward time slack and forward load slack are calculated in linear time. Forward time slack consists of the maximum amount of time a service in the route can be postponed by without violating any time windows constraints associated with succeeding services. Similarly, forward load slack corresponds to the maximum load a body can take at a certain stop without violating the capacity constraints associated with any of its succeeding nodes. By maintaining these two forms of slack, it is possible to check in constant time whether or not a given insertion is feasible. When feasible positions for both the pickup and delivery service are obtained, the new solution value considering the request insertion is calculated in constant time with a delta evaluation which updates the travel cost based on the removed and added edges. The complete solution is therefore only evaluated once, after the insertion of the request into its best feasible position.

2.4.4 Transfer phase

The transfer phase inserts attaching and detaching operations into the solution. This phase introduces diversity with respect to the vessels' attached bodies and enables different vessel-body combinations to be explored. The rationale behind attaching/detaching a body is that a single vessel is neither obligated to visit both the pickup and delivery location of requests, nor does it need to wait for the entire container service time to elapse before departing. Therefore, by using the vessel time more efficiently, one may reduce travel cost and/or outsourcing costs.

We designed four transfer insertion and two transfer removal neighborhoods for the transfer phase. Each time the transfer phase is called, one of the six neighborhoods is uniformly selected. Transfer insertion neighborhoods are selected with $\nu\%$ probability, while transfer removal neighborhoods are selected with the remaining $100 - \nu\%$ probability. Transfers are performed either at a customer's location, where the detached body has one or more services to attend to, or at transfer points where the body is simply detached before later being reattached to either the same or a different vessel.

Transfer insertion

The four different transfer insertion neighborhoods follow the same basic steps outlined in Algorithm 2. Each neighborhood receives as input a solution, a body to be transferred, a node to perform the transfer and a position in the body's route to insert the transfer (lines 1-2). First, since locations after $prev_t$ will no longer be visited by b, the requests associated with those locations are removed from the body's route and inserted into the unscheduled request set U (line 3). If b was transferred after $prev_t$, the transfer request is also removed. When removing requests from the body's route, they are also removed from the route of the vessel towing body b. Next, the detaching operation is inserted at $node_t$ after $prev_t$ in the body's route (line 4).

Algorithm 2: General method for transfer insertion.

```
1  Input: solution s, body b to be transferred;
2  Input: transfer node node_t, last node prev_t into b's route before transfer;
3  U ← Remove from b's route every served request after prev_t (s);
4  Detachment of b at node_t after prev_t (s);
5  if Detached body is empty then
6  |  Best insertion of all requests in U (s);
7  else
8  |  foreach vessel route v_r ∈ V do
9  |  |  foreach vessel stop v_stop ∈ v_r do
10 |  |  |  s' ← Attaching of body b after vessel stop v_stop (s);
11 |  |  |  Best insertion of all requests in U (s');
12 |  |  |  if f(s') < f(s) then
13 |  |  |  |  s ← s';
14 return s;
```

If the detached body is empty, meaning there is no container loaded in the body, the unscheduled requests are inserted into the solution using the best insertion method and the body is left unattached (lines 5-6). Otherwise, the body must be attached to a vessel in order to be able to continue on its route and deliver the containers that have already been loaded. Note that if a pickup service was served by b before $node_t$, the corresponding delivery service must also be served by b after the transfer. The attaching of a body is conducted in a "best insertion" manner. In other words, every position in each vessel route is checked before selecting the best insertion position. The vessel which will visit $node_t$ to attach body b must respect feasibility constraints such as vessel capacity, the time windows of customers visited after $node_t$ and cross synchronization which avoids cycle dependencies between transfers (Masson et al., 2013a). As

with the best insertion of requests into body routes, the best insertion of bodies into vessel routes calculates the forward time slack and forward load slack to accelerate the feasibility check of each insertion. After each feasible attachment, the unscheduled requests are inserted into the solution (lines 8-13). Finally, the solution yielding the lowest cost is saved and returned (line 14).

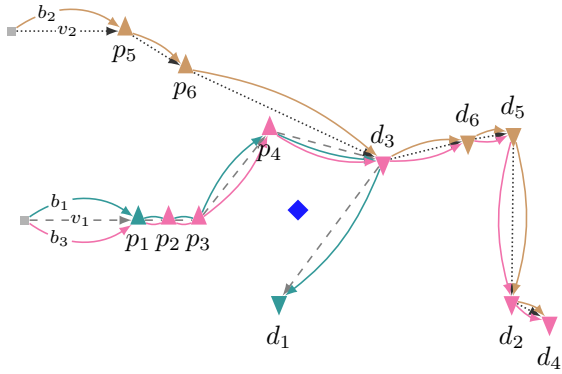
Figure 2.4 details the steps to insert a transfer body request. The network representation follows the same format as that used in Figure 2.2, where the routes of bodies and vessels are depicted by colored and dashed edges, respectively. Figure 2.4(a) shows a solution for the VSBR where bodies b_1 and b_3 start attached to vessel v_1 and body b_2 starts attached to vessel v_2. A transfer occurs at node d_3, where body b_3 is detached by vessel v_1 and attached to vessel v_2.

Consider a transfer request insertion of body b_3 at the transfer point ($prev_t$ is p_3). Figure 2.4(b) shows a destroyed solution after detaching b_3 at the transfer point. Services p_4, d_3, d_2 and d_4 are all successors of p_3 in the route of body b_3 and are therefore removed from the solution and included in the set of unscheduled requests U. Customers removed from a body's route are also removed from the route of the vessel which tows this body. Note that these removed services alter the route of both vessels, with b_2 left at the transfer point with two services already loaded (p_2 and p_3). Figure 2.4(c) shows one possible way of repairing the solution, whereby body b_3 is picked up by vessel v_2. Services d_2 and d_3 must be served by body b_3 since their respective pickup services were previously loaded into this body. By contrast, both the pickup and delivery services of request 4 are unscheduled and can be served by any other body. In Figure 2.4(c), b_3 is picked up by vessel v_2 and is able to serve requests d_2 and d_3. Finally, vessel v_2 serves request 4 by loading it into body b_2.

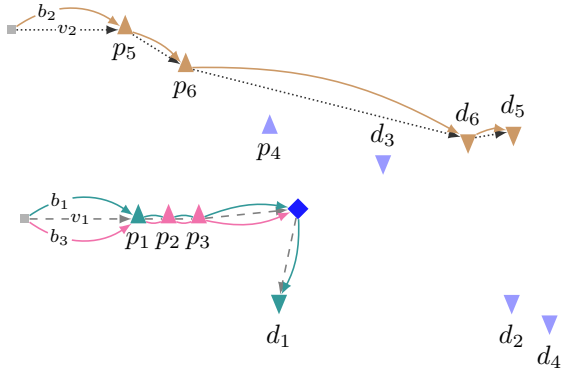
Each transfer insertion neighborhood may vary in terms of how it selects the node to perform the transfer and the body to be transferred. Neighborhoods and their particularities are outlined in what follows.

Score-based transfer insertion at a customer location (TI1). Given the large number of possibilities, performing transfers at randomly selected customers may lead to many poor-quality solutions. This first neighborhood attempts to minimize the chance of this occurring and identify promising customers to perform transfers. The neighborhood begins by selecting a customer which is associated with a request r (either pickup or delivery) and which is not in U. It is assumed that beneficial selections for customers are those with the following characteristics:

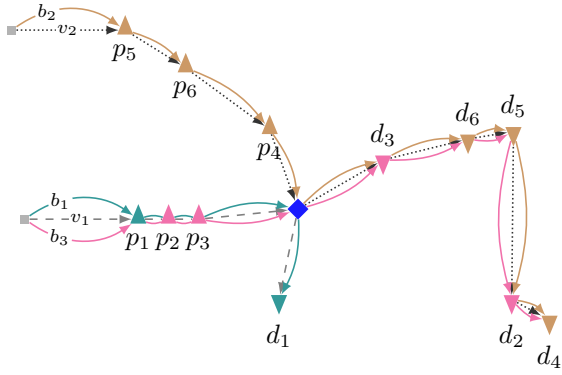
1. *Low travel cost*: customers which are located close to another vessel's route.



(a) A simple VSBR solution.



(b) A destroyed VSBR solution with a new body transfer and unscheduled requests.



(c) A repaired VSBR solution.

Figure 2.4: Body transfer insertion.

Small detours for vessels which attach a body should not significantly affect the objective value.

2. *Large demand of containers*: customers where a significant number of containers are being served. The total service time at a customer is proportional to the number of containers to be (un)loaded. By detaching a body at such a customer, vessels will no longer need to wait for the full service time. For example, instead of waiting at a customer location a vessel may use this time to serve other requests, which can potentially reduce the travel cost of other vessels or reduce outsourcing costs.

Taking into account these two desirable characteristics, we introduce a function which assigns a score to each visited customer $c \in C$:

$$score_c = \alpha * \frac{1}{travel_c} + (1 - \alpha) * demand_c \tag{2.47}$$

In Equation (2.47), parameter α controls the extent to which travel cost and demand contribute to the selection criterion. Parameter $travel_c$ corresponds to the total travel cost between customer c and all the other customers which are visited by other bodies. A low $travel_c$ reflects a customer which is close to many others, while a high value implies that the customer in question is far away from others. The total number of containers (un)loaded at a customer by a body is defined by *demand* (ρ_c), which is directly proportional to the total service time.

A probabilistic selection is employed in the present study to diversify the solution and avoid an overly greedy convergence of the method. Scores are sorted in descending order and a rank γ is assigned to each, which corresponds to their position in the array ($\gamma = 1$ for the first element). A rank-based $bias(\gamma)$ is then assigned to each option, which is calculated by an exponential bias function. The probability $p(\gamma)$ of selecting an option is given by Equation (2.48):

$$p(\gamma) = (\sum_{k=1}^{R} bias(k))^{-1} * bias(\gamma) \tag{2.48}$$

The node where the selected customer is located becomes $node_t$: the node to perform the detachment. Body b_r is the body to be detached and $prev_t$ is the last node visited by b_r before serving request r. Note that r will still be served by b_r, but only after it is detached. After determining the value of these three variables, the transfer insertion method follows the steps outlined in Algorithm 2.

Transfer insertion at transfer points (TI2). Bodies may also be detached/attached at special transfer points where no additional services are performed. Transfer points are usually located in central positions and intersections which connect different clusters of customers. This neighborhood randomly selects a transfer point, which corresponds to $node_t$. Each body is then a candidate to be detached at $node_t$. A score is calculated for each body b_i and corresponds to the average distance between $node_t$ and every customer served by b_i. The exponential bias function used in neighborhood TI1 is also used here to select body b. The final decision to make is where in the body's route to insert the transfer point. The closest customer to $node_t$ visited by b is selected as $prev_t$.

Transfer at first node (TI3). Transferring at the first node enables a vessel to detach a body as its first action within the scheduling horizon, even if the body has no service to perform at this node. This assumes a service was conducted during the preceding scheduling horizon. Thus, this move serves as an opportunity for vessels to detach undesired bodies and free capacity to attach other bodies.

This neighborhood starts by selecting a vessel, with priority given to vessels with many bodies attached. A body b is then selected, with priority given to empty bodies. Note that detaching an empty body does not lead to immediate reattachment. The remaining variables, $node_t$ and $prev_t$, are given by the location where the vessel started and the dummy node at the beginning of the vessel's route, respectively.

Pick up originally detached body (TI4). The scheduling horizon may begin with detached bodies which may or may not be empty. It is advantageous for a vessel to attach an empty body when it needs more capacity, whereas attaching a body which started loaded with customer containers is required to fulfill all requests. First a detached body is selected. For this, the bias function gives priority to selecting bodies with loaded containers. The attachment location ($node_t$) is given by the node where the body is located at the start of the scheduling horizon. Since detaching is not required, $prev_t$ is null and lines 3-6 in Algorithm 2 are ignored.

Transfer removal

Algorithm 3 provides an overview of how to remove transfer services. A transfer service comprises of the delivery t_d and pickup t_p of a body b. Given a solution and a transfer service as input, each customer served by body b after pickup node t_p is removed from b's route and inserted into the unscheduled set (line 2). These nodes are also removed from the route of the vessel attached to body b.

After removing all customers and transfers after t_p, the pickup node is safely removed (line 3).

Delivery node t_d is the last node to be removed, leaving the body attached to the vessel that begun the transfer (line 4). Finally, the removed requests are reinserted using the best insertion method and the solution is returned (lines 5-6). Transfers may comprise of only a delivery or only a pickup node. For the first scenario, the delivery node is removed and the body continues on its route attached to the vessel. In this case, lines 2, 3 and 5 are not executed. When only removing a pickup, line 4 is skipped and the remainder of the algorithm functions as normal.

Algorithm 3: General transfer removal method.

1 **Input:** solution s, transfer service to remove t_p, t_d, transferred body b;
2 $U \leftarrow$ Remove every served request after node t_p from b's route (s);
3 Remove transfer pickup node t_p (s);
4 Remove transfer delivery node t_d (s);
5 Best insertion of all requests in U (s);
6 **return** s;

Removing a transfer may lead to infeasible solutions which violate vessel capacities. For instance, a vessel towing the maximum number of bodies may detach a body b_1 as its first action, leaving a capacity slack of one body. Later in the route, this same vessel may attach to a body b_2, again reaching its capacity limit. In case b_1's detachment would not be conducted, the previously feasible attachment of b_2 becomes infeasible as the vessel's capacity is violated. To remedy this, a removal in cascade is employed whereby transfers which cause infeasibility are also removed. The two removal neighborhoods differ in terms of how exactly they select the transfer service to remove.

Random transfer removal (TR1). This neighborhood selects, with equal probability, a single transfer request to be removed from the solution. If after the removal the solution is infeasible due to vessel capacity, additional transfers are removed until a feasible solution is obtained.

Worst transfer removal (TR2). This neighborhood considers one transfer request at a time. The removal which yields the best solution value is maintained and the solution is returned with at least one fewer transfer requests.

2.4.5 Routing phase

Once transfers are fixed, the routing phase is employed and handles a PDPTW. This method optimizes vessel routes by rescheduling requests while respecting predefined body transfers. A fast and efficient routing algorithm enables ILS-SB to generate high-quality PDPTW solutions for many body transfer configurations. From the many existing routing algorithms, we selected one which is fast and simple to implement with proven performance on PDPTWs. SISRs (Christiaens and Vanden Berghe, 2020) was developed to be a general method for VRPs and experiments have shown that it generates competitive solutions for a wide range of variants, including the PDPTW.

SISRs comprises of a single ruin operator which removes customers based on a novel property called spatial slack. SISRs removes a sufficient number of customers in different yet geographically proximate routes, creating spatial and capacity slack. When reinserting customers back into the solution, multiple routes close to those customers offer a range of efficient options. The recreate operator comprises of a best insertion method (described in Section 2.4.3) which uses rank-based probabilities to avoid always inserting customers into the best position, thereby avoiding premature convergence. Each time the routing phase is invoked, SISRs is executed for a fixed number of iterations and returns an equal quality or improved solution.

2.5 Computational experiments

This section will investigate whether body transfers have a positive impact on solution quality despite the increase it brings to the problem's complexity and search space. Computational experiments using newly generated instances are performed with ILS-SB to study if it can efficiently employ body transfers and obtain high-quality solutions in reasonable runtimes. Additionally, an evaluation on how each proposed transfer neighborhood impacts solution quality will be performed. All experiments were conducted on a computer with an IntelXeon E5-2660 processor at 2.6 GHz and 164 GB of RAM running Ubuntu 18.04 LTS. ILS-SB was implemented in C++ and compiled using gcc 7.4.0 and options -O3.

All parameters required by ILS-SB are detailed in Table 2.3 and were calibrated using *irace* (López-Ibáñez et al., 2016) with the range of values provided in column *Range*. SISRs was implemented with the original parameters calibrated by Christiaens and Vanden Berghe (2020) for its best behavior across a range

of instance sizes. The maximum number of iterations and time limit were set manually considering the trade-off between solution quality and processing time.

Table 2.3: ILS-SB parameters and values.

Parameter	Value	Range	Parameter	Value
ι	10	{5, 10,...,45, 50}	Time limit(s)	600
l	75	{10, 25, 50,..., 175, 200}	SISRs iterations	1000
ν	70	{10, 20, ..., 80, 90}		
α	0.70	{0.0,...,1.0}		
$\#maxIt$	80	{10, 20,...,140, 150}		

2.5.1 Instance sets

Given that the VSBR is a new problem there are no academic instances available and so, in order to perform experiments and encourage future research, instances are generated based on historical data from a shipping company. The data provided contains information concerning vessels and bodies, such as their speed, capacity, detaching/attaching time and container service time. Furthermore, a set of requests was provided along with the corresponding customer locations on a waterway network.

We generated a total of 70 instances, divided into two instance sets. In the first set, *AttB*, each body is attached to a single vessel and each vessel has at least one body attached to it. In the second instance set, *DetB*, bodies may start detached and vessels may start empty (not towing any body). Each instance set has seven groups of five instances containing 10, 50, 100, 150, 200, 250 and 300 requests. All benchmark instances are anonymized and have been made publicly available as supplementary online material. For each instance we assumed a scheduling horizon of 20 days and ϵ equals to one while the other attributes were generated as follows.

Routing network. The employed routing network consists of 14 nodes with each node representing a container terminal where the pickup and/or delivery service of one or more requests must be served. Each of the nodes is connected to at least one other node via edges. Distances between nodes are calculated by employing a routing Application Programming Interface [1] which returns the shortest waterway trajectory between each pair of terminals. To facilitate the reading of instances, a distance matrix is provided for every instance, while the real location of nodes have been anonymized.

[1] `https://routino.grade.de/`

Vessels and bodies. The total number and capacity of available vessels and bodies corresponds to the real-world scenario. Since there is no depot or source/sink node, vessels may start at any location. For each vessel, a location in the network is selected randomly with uniform probability. The scheduling horizon starts given a "partial snapshot" of a solution, thus vessels may be en route between two nodes. To simulate this scenario, instances have a *start time* parameter for each vessel which corresponds to the earliest time it will be available at the first node of its route. Given two uniformly selected nodes n_1 and n_2, the start time of each vessel is generated between 0 and 50% of the travel time from n_1 to n_2. In the MIP model it is sufficient to fix $\tau^{Av}_{o_v}$ to vessel v's start time to consider this particular information. For the *DetB* instance set, bodies have a one-in-three chance to start detached from vessels. Bodies' start locations are selected with uniform probability by first considering only locations or vessels that do not have a body present or attached.

Requests. A single request is identified by a pickup and delivery service — each with their corresponding customer and time windows — and the number of containers ρ to be transported. Using the historical data we calculated the average and standard deviation of the number of containers associated with requests, the length of services time windows and amount of time overlap between pickup and delivery services of the same request. On average, requests involve seven containers with time windows of 4 and 7 days for pickup and delivery services, respectively. Pickup and delivery time windows overlap three days on average. All these attributes were generated based on a normal distribution considering the calculated average and standard deviation.

When creating an instance, first the number of containers ρ associated with each request is generated. Then, the locations of pickup and delivery services are generated uniformly, selecting a different location for each service. Starting with the pickup service, the size of its time window (TW^P_{size}) is generated followed by the opening day ($Open^P_{day}$), which is selected with uniform probability between -5 and 20-TW^P_{size}. If $Open^P_{day} < 0$, the request belongs to R^2 which means no pickup service is needed and the containers are inserted into a body (uniformly selected) with sufficient capacity.

After generating the pickup, the delivery service is generated starting with the size of its time window (TW^D_{size}). The opening day for the delivery service ($Open^D_{day}$) strongly depends on the opening day of the pickup service. If $Open^P_{day} < 0$, then $Open^D_{day}$ is selected with uniform probability between 0 and 20-TW^D_{size}. Otherwise, the considered range for $Open^D_{day}$ is 0-20. If $Open^D_{day} > 20 - TW^D_{size}$, the request belongs to R^3. In this case the delivery service will be left for the next scheduling horizon. For R^1 requests, the amount of time overlap is generated such that the delivery service may start as early as

the pickup service or as late as when the pickup time window ends.

An unserved request incurs an outsourcing cost π of 10,000. This very large coefficient ensures that it is hardly advantageous to leave a request unscheduled. Therefore, a solution which serves more customers, even if it has a larger travel cost, is generally better than a solution serving fewer customers.

2.5.2 Detailed results

We first conduct experiments with the MIP formulation using Java 8 and CPLEX 12.6.3. Given that the model is unable to solve instances of realistic size within reasonable computational runtimes, the instances employed in these experiments have reduced size and were generated specifically for the MIP (with fewer vessels, bodies and requests). For the sake of diversity and to test the different constraints implemented in the model, we generated instances with R^1, R^2 and R^3 requests and instances where at least one request must be served by a truck. Table 2.4 presents the results of the experiments obtained by the MIP formulation and by ILS-SB. We introduce a time limit of five hours and a memory limit of 100 GB when solving the instances with the MIP.

The first columns of Table 2.4 provide details of the instance: the number of requests |R|, vessels |V| and bodies |B|. Regarding the MIP results, the table provides the number of nodes in the model network |N|, the upper bound (UB), lower bound (LB), processing time in seconds (T(s)) and number of unscheduled requests |U|. Given the non-deterministic nature of ILS-SB, each experiment is henceforth performed 10 times with a computational time limit of 10 minutes per run. The final columns of Table 2.4 document the results for ILS-SB and provide the value of the best solution obtained (S^*), the average processing time T(s), number of transfers (BT) and number of unscheduled requests |U|.

Table 2.4: MIP and ILS-SB comparison.

Instance			MIP					ILS-SB			
|R|	|V|	|B|	|N|	UB	LB	T(s)	|U|	S^*	T(s)	BT	|U|
3	2	2	14	10460	10460	30.41	1	10460	0.23	1.90	1
3	2	2	11	290	290	1.29	0	290	0.22	0.00	0
4	2	2	15	560	560	8.23	0	560	0.91	3.20	0
4	2	2	15	10460	10460	1625.73	1	10460	0.95	2.50	1
5	2	2	17	10490	10490	288.33	1	10490	3.50	1.40	1
5	2	2	17	10490	10490	557.88	1	10490	1.89	1.50	1
5	3	3	31	10587.99	350.11	18000.00	1	1448	7.74	2.90	0
6	3	3	35	30796.10	318.25	18000.00	3	1782	9.91	1.90	0

Table 2.4 reveals that although the MIP formulation is able to obtain the optimal solutions rather quickly for instances with two vessels and two bodies with up to five requests, CPLEX already experiences difficulty in solving problems with one more vessel and body. If the number of requests increases to 10 then even five hours of runtime is not enough to find a non-trivial solution (a solution that serves at least one request). Meanwhile, ILS-SB is able to obtain all proven optimal solutions and much better solutions for the final two instances in under 10 seconds.

To evaluate the statistical significance of the differences between methods, the Wilcoxon signed-rank test is performed when comparing results of two methods while the Pairwise T-test is performed in the neighborhood analysis, both with a confidence level of 95%. Table 2.5 details the results of ILS-SB for the *DetB* instance set and reports the value of the best solution found (S^*), the average solution values (S_{avg}), the average solution value considering the additional objective function (OF^2_{avg}), the average number of body transfers (BT) and unscheduled requests |U|, the average processing time in seconds T(s) and the average standard deviation (SD). In order to facilitate the presentation of results, different instances with the same number of requests were aggregated. Thus, each row corresponds to the average results across five instances.

Table 2.5: ILS-SB results for the *DetB* instance set.

Instance	S^*	S_{avg}	OF^2_{avg}	BT	|U|	T(s)	SD
I_10	2053.20	2066.20	46057.52	4.60	0.00	64.62	13.10
I_50	4690.80	4843.24	80532.28	9.64	0.00	605.05	112.17
I_100	6677.50	6980.60	84837.40	8.03	0.00	601.57	256.95
I_150	8765.40	9436.12	91706.00	6.36	0.00	603.41	596.51
I_200	12395.20	13325.08	103928.48	6.28	0.00	604.90	816.82
I_250	17455.80	18663.84	102816.80	5.56	0.00	607.19	917.07
I_300	28441.80	46144.16	120687.28	4.84	1.48	608.77	36374.70
Avg.	11497.10	14494.18	90080.82	6.47	0.21	527.69	5583.90

For this instance set, all instances with up to 250 requests had all of their requests served, while three I_300 instance resulted in a few unscheduled requests given that it becomes more difficult to serve more requests within the same length scheduling horizon. The number of unscheduled requests helps explain the larger standard deviation, as not serving customers leads to large outsourcing costs in the solution value. The maximum number of iterations without improvement was triggered only for instances with 10 requests, while the execution of all others was halted by the time limit. Doubling ILS-SB's time limit does not lead to significant difference concerning solution value. In addition, maintaining a short time limit enables the method to be adapted for

a dynamic version of the problem, where, for example, it is uncertain when exactly requests become available.

Remaining with the results provided in Table 2.5, body transfers were inserted into every solution, with the results showing that a given body is transferred at most three times during the 20 days scheduling horizon. The average number of transfers in a solution considering all bodies is 6.47, meaning that on average 60% of the available bodies were transferred at least once. However, when only considering transfers between two different vessels, the average drops to 3.59, with the most significant differences occurring in the instances with 50 and 100 requests.

In order to evaluate the impact of body transfers on solution quality, a comparison is performed between the results of the complete ILS-SB and the corresponding PDPTW where body transfers are not considered. Thus, for the purposes of this experiment, only the routing phase of Algorithm 4 is used, with the resulting method referred to as ILS-R. To enable a fair comparison this experiment is conducted on the *AttB* instance set, where each body starts attached to a vessel. Since the routing phase will never schedule body transfers, no instances where bodies start detached are used, given that this would provide an unfair advantage to ILS-SB which can attach such bodies. Table 2.6 provides a summary of the results produced by ILS-R and documents the average solution value (S_{avg}), the average solution value considering the secondary objective function (OF^2_{avg}) and the average number of unscheduled requests |U|. The final row of the table provides the gap from the best average solution value and the best average solution value with respect to OF^2. These gaps are calculated for each instance using the best solution generated from both methods.

Table 2.6: Body transfers analysis.

	ILS-R			ILS-SB		
Instances	S_{avg}	OF^2_{avg}	|U|	S_{avg}	OF^2_{avg}	|U|
I_10	1836.80	73433.40	0.00	1775.92	65345.12	0.00
I_50	4932.32	99826.88	0.00	4741.44	90681.80	0.00
I_100	6838.23	108816.53	0.00	6750.93	92951.73	0.00
I_150	9037.20	122971.80	0.00	8685.12	107205.00	0.00
I_200	12105.76	120886.40	0.00	12088.68	108114.40	0.00
I_250	19361.44	126635.40	0.00	18295.16	115457.60	0.00
I_300	60487.16	158526.60	2.84	54219.80	144411.00	2.36
gap	4.79%	13.47%		0.33%	0.63%	

For the considered instance set, average solutions including body transfers are on average 4% better than those which do not. An even larger improvement is

observed when comparing the values of OF^2 (13%), highlighting the even greater impact of body transfers on total operational cost. For best solution values, ILS-SB obtains solutions which are on average 6.5% better than those obtained by ILS-R. Despite the moderate average gap, a few outliers are observed which are worth mentioning. ILS-SB obtained solutions values 14%, 20%, 17% and 27% better than ILS-R for instances I_10_3, I_50_2, I_250_5 and I_300_2, respectively. ILS-R obtained better solutions for only three instances, resulting in the low positive ILS-SB gaps. Nevertheless, when comparing the results obtained by ILS-SB and ILS-R, significant statistical differences are observed regarding best solution, average solution and average OF^2 solution. Together, these results confirm the positive impact of permitting body transfers. Finally, it is worth noting that both algorithms were executed with the same time limit. This means that even with a far larger search space ILS-SB is able, within the same length of time, to significantly improve solution quality.

Additional analysis

ILS-SB comprises of six neighborhoods dedicated to the insertion/removal of body transfers. In order to identify the contribution of each neighborhood to solution quality, a set of experiments is conducted on the *DetB* instance set where a different neighborhood is deactivated in each experiment and the obtained solutions are compared. Table 2.7 provides the results for each neighborhood by dividing the objective function into two parts. Row *Travel cost* corresponds to the average travel cost of vessel routes, given by the number of edges used by vessels. Meanwhile, the average number of unscheduled requests is given by $|\mathrm{U}|_{avg}$. The last row corresponds to the gap between the average solution value of the given neighborhoods versus the best average solution value considering all neighborhoods ($\mathrm{gap}(\mathrm{S}_avg)$). Column ILS-SB corresponds to the complete method with all neighborhoods, while each remaining column corresponds to the method without the labeled neighborhood. For instance, column TI1 corresponds to the results when the first insertion neighborhood is absent.

Table 2.7: Solution values when using different set of neighborhoods.

	ILS-SB	TI1	TI2	TI3	TI4	TR1	TR2
Travel cost	12432.33	12438.00	**12342.86**	12482.09	13706.50	12592.34	12659.30
$\|\mathrm{U}\|_{avg}$	**0.21**	0.49	0.35	0.43	10.25	0.81	0.58
$\mathrm{gap}(S_{avg})$	**2.93**	4.86	3.26	4.78	73.35	8.67	4.89

TI1: transfer at a customer location; TI2: transfer at transfer points; TI3: transfer at first node; TI4: pick up originally detached body; TR1: random removal; TR2: worst removal.

Although average solution quality always deteriorates when deactivating neighborhoods, significant statistical differences are present only between each

method and TI4. The other neighborhoods did not significantly improve the solution, however each one of them is tailored to a unique way of performing body transfers and can prove valuable in different scenarios. Despite larger travel cost, ILS-SB obtains the smallest gap considering average solution value (2.93%). This result derives itself from the low number of unscheduled requests (0.21 on average). Given the nature of the VSBR and the large outsourcing costs associated with unscheduled requests, a solution serving more requests, even if it has longer routes, will often be better than a solution serving fewer.

The large gaps between solutions with different sets of neighborhoods are concentrated in instances with more than 250 requests and primarily derive from the number of unscheduled requests. To demonstrate the impact of outsourcing costs on solution quality, Figure 2.5 provides the average solution value associated with the five instances containing 300 requests. The total value is divided into travel and outsourcing costs. All methods have similar travel costs, which is approximately 32000, but the total solution value varies depending on the number of unscheduled requests. The best solution is obtained by ILS-SB, which on average has 1.48 unscheduled requests. On the other hand, TI4 and TR1 have on average 26 and 4 unscheduled requests which results in solutions that are 84% and 43% worse, respectively.

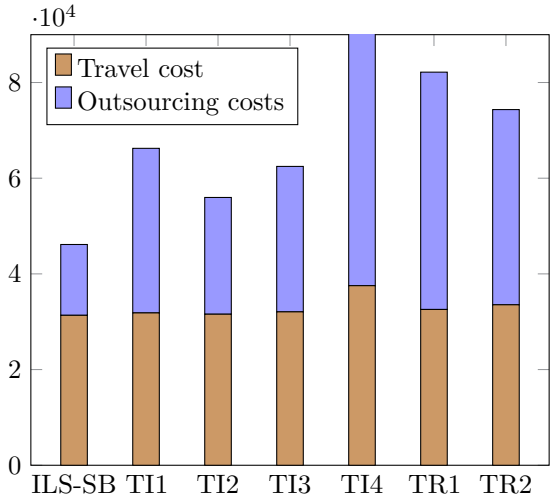


Figure 2.5: Solution values for the I_300 instances.

Note that despite the large number of unserved requests, TI4 still results in a greater average travel cost. TI4 is the only neighborhood capable of attaching bodies which began detached. When this particular neighborhood is deactivated, these bodies can no longer be attached. As a result, not only will requests

which began loaded into these detached bodies remain unserved, but there is also less capacity available to serve requests since the initially detached bodies will not be used. For instances with up to 200 requests, TI4 obtains solutions with a low travel cost (average gap of 2.26). However, although the number of unscheduled requests for larger instances remains high, this does not lead to a decrease in travel cost. Due to the high number of requests to serve along with the reduced fleet of bodies (thus less total available capacity), more trips are necessary to serve requests which in turn leads to travel cost increases.

Figure 2.6 illustrates the relationship between travel cost and the percentage of unscheduled requests |U|% with a parallel coordinates plot. For this experiment, a large number of solutions were generated by uniformly selecting the number of requests to remain unscheduled, while the other requests were randomly inserted into their first feasible position. Travel costs were normalized between 1 and 100 to better visualize the results. These results demonstrate how greater travel costs are associated with low percentages of unscheduled requests, while low travel costs are associated with a high proportion of unscheduled requests.

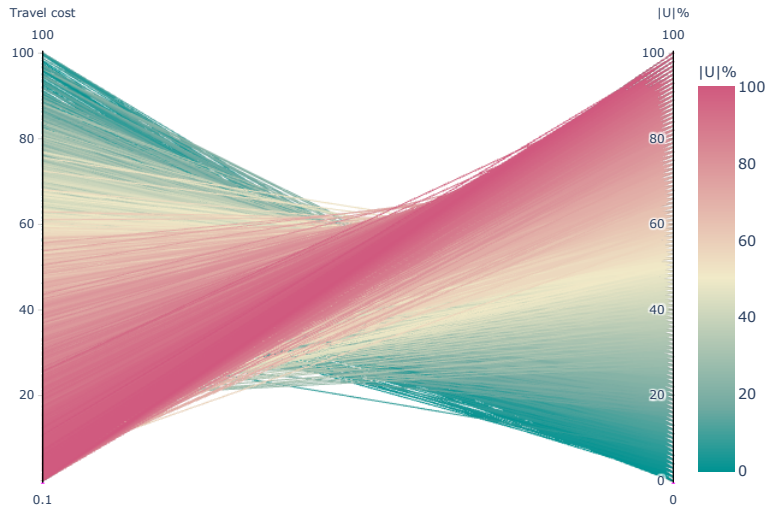


Figure 2.6: Relationship between travel cost and percentage of unserved requests.

Finally, an experiment is performed concerning the impact of employing OF^2. Breaking ties with respect to travel cost and directing the search towards solutions with less total operational cost improves overall solution quality by

5.9%. Although better solutions are obtained on average, statistical experiments show no significant difference between the results with and without the tie break regarding best and average solutions. However, significant differences are observed with respect to the average OF^2 solution values.

2.6 Conclusions

This chapter introduced the vessel swap-body routing problem (VSBR), a generalization of the pickup and delivery problem with time windows. The problem is encountered in a shipping company where body transfers may be employed to reduce overall costs. These transfers significantly increase the search space and difficulty of the problem given how they have a large impact on the number of routing and scheduling possibilities, confronting human operators with a significant challenge. The VSBR therefore represents a significant academic challenge with practical applications. We proposed a solution method for the problem and investigated whether body transfers can bring significant gains to solution quality when runtimes remain the same.

To facilitate the development of a solution method and to better tackle the multiple levels of decisions, we decomposed the problem into location decisions (where to perform body transfers) and routing decisions. Once decomposed, these different decision levels were addressed using dedicated methods. We proposed a heuristic which consists of a state-of-the-art algorithm for the PDPTW and tailored neighborhoods for performing body transfers. Computational experiments on instances derived from real-world data indicated the positive impact of body transfers on both travel and outsourcing costs. The inclusion of body transfers lead to solutions with shorter vessel routes and more served customers, which directly impacted the overall cost of transportation. The significant gains obtained for a scheduling horizon of 20 days could translate into even larger gains for longer scheduling horizons.

This research indicated the potential benefit of transferring batches of requests between vehicles rather than handling the entirety of a request (pickup and delivery) with a single vehicle. These transfers give rise to an additional decision level which can be efficiently managed by decomposing the problem and employing dedicated methods for each decision level. Techniques to accelerate the solution method and better direct the search towards high-quality solutions should be incorporated as the search space significantly increases in size when handling these problems with multiple decisions levels. The foundations laid by this chapter aim to encourage researchers to consider including the transfer of batches of requests in other VRP variants, as doing so has the potential

to significantly improve solution quality in reasonable runtimes, despite the additional complexity.

Chapter 3

Two-echelon location routing problem with loading constraints

3.1 Introduction

This chapter addresses a real-world problem emerging from a logistics company which supplies its customers with large material handling equipment as well as industrial and agricultural machines. The company chooses from the depots (platforms) of several external providers to satisfy the demand of its customers. When a platform is utilized, a fixed rental or contract cost must be paid. Using capacitated vehicles the items dispatched from these platforms are then delivered to parking areas or sorting facilities (satellites), which must also be selected from multiple options. As with platforms, using a satellite incurs a fixed rental cost. The distribution of items to customers is then performed from these intermediate facilities. Both the platforms and satellites have a limited service capacity and thus the total demand allocated to each facility should not exceed its capacity.

In order to immediately ground the problem in physical reality, Figure 3.1 illustrates a toy network involving four candidate platforms, six candidate satellites and 15 customers. The first distribution level (echelon) comprises of the two open platforms (B and D) and the routes connecting these platforms to the satellites (represented with the thicker lines). The second echelon comprises of

three open satellites: g, h and i. Each customer (1-15) is served by a single vehicle whose origin is one of these satellites. The routes connecting the smaller second-echelon vehicles with customers are represented with thinner lines. An example of a load plan, the exact position of items in the vehicle's two-dimensional loading surface, is provided for the vehicle during route $h \rightarrow 7 \rightarrow 8 \rightarrow 9 \rightarrow 10 \rightarrow h$. Items are loaded from the back of the vehicle (rear loading) at satellite h before the trip begins. During the trip, each item will be unloaded from the rear. In this particular load plan, the items belonging to customers 9 and 10 are positioned behind those belonging to customers 8 and 7, respectively. Thanks to this positioning, the unloading of items at customer locations is possible without having to move any other items out of the way.

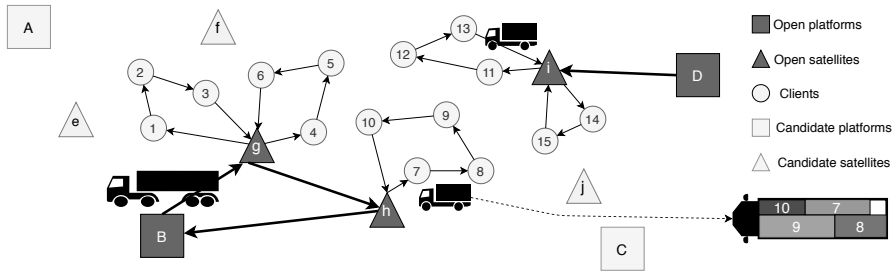


Figure 3.1: Two-echelon routes with load plan.

The literature on multi-echelon location-routing problems dates back to the early study by Jacobsen and Madsen (1980), while the two-echelon location routing problem (2E-LRP) where both platforms and satellites must be selected from multiple candidates first occurs in Boccia et al. (2010). The 2E-LRP, the most closely related problem to the presented scenario, considers only one dimension (either area or volume) when loading items into vehicles. The logistics company that introduced us to the problem we address in this chapter faces a similar situation insofar as they utilize the area of a rectangular-shaped projection of the items (on the ground surface) in their calculations when assigning items to facilities and vehicles. However, this approximation of item dimensions (length and width) does not always provide accurate results when loading said items inside the first- and second-echelon vehicles. As a result of ignoring the items' two-dimensional footprint, the company generates load plans with items which will never fit together. When attempting to implement these infeasible load plans, the company may be forced to outsource the delivery of some items to third-party vehicles. Given that this can prove very expensive, in this chapter we take into account realistic two-dimensional loading constraints when assigning items to vehicles (which are also characterized by their length and width).

The problem we arrive at in order to properly address this situation is a

generalization of the 2E-LRP which we term the *two-echelon location-routing problem with two-dimensional loading constraints* (2E-LRP2L). In the particular application considered in this chapter, the total demand assigned to a satellite may be larger than the capacity of first-echelon vehicles. Therefore, it is permitted to split deliveries to satellites across multiple vehicles in the first echelon. Moreover, besides the standard two-dimensional loading constraints, handling of items (unloading and re-loading of an item at the same location or movements inside the vehicle) is considered highly undesirable, as the free space necessary for such movements is limited. Hence when these movements and item relocations take place, a *sequential loading penalty* is incurred.

The 2E-LRP2L is extremely challenging as it integrates $\mathcal{NP}$-complete loading decisions (Fowler et al., 1981) into the 2E-LRP which, on its own, is already an $\mathcal{NP}$-hard problem. Given that the 2E-LRP is difficult to solve with exact methods, one can logically assume that such methods will also be impractical when it comes to solving medium- and large-scale 2E-LRP2L instances within a reasonable amount of time. Therefore, this chapter introduces a heuristic approach to generate solutions for the 2E-LRP2L.

Besides introducing a heuristic for the 2E-LRP2L, this chapter will also study the effects of incorporating realistic loading constraints into the 2E-LRP and the impact of items' loading order on solution quality. To quantify these effects, different loading strategies are developed for the proposed heuristic. These strategies enable us to arrive at a better understanding of the problem's unique features. A set of instances is derived from real customer locations and requests, as well as real vehicle and item sizes. Furthermore, computational experiments are also conducted on benchmark instances of the standard 2E-LRP to compare the performance of the proposed algorithm against state-of-the-art methods. All instances, solutions and tables with results are available as supplementary online material.

3.2 Related work

This section initially focuses on detailing the main contributions made in relation to the 2E-LRP, which are recent and few in number, while also providing an overview of some closely related problems from the literature in Section 3.2.1. Later, in Section 3.2.2, the most relevant studies on the vehicle routing problem with two-dimensional loading constraint (2L-CVRP) are discussed. The 2E-LRP considers location and routing decisions in two echelons and is a generalization of the two-echelon vehicle routing problem (2E-VRP) first introduced by Perboli et al. (2011). Location and routing have often been studied in an intertwined

manner since tackling such decisions separately may result in excessive overall costs (Salhi and Rand, 1989; Salhi and Nagy, 1999).

The studies reviewed in this section are limited to those which had a direct impact on this chapter. We refer readers interested in more extensive reviews to the following surveys. Cuda et al. (2015) and Drexl and Schneider (2015) review two echelon routing problems that include location routing, vehicle routing, as well as truck and trailer problems. Nagy and Salhi (2007), Prodhon and Prins (2014) and Schneider and Drexl (2017) provide complementary reviews on location-routing problems. Laporte (2009), Kumar and Panneerselvam (2012) and Vidal et al. (2019) discuss the vehicle routing problem and its variants. Finally, Bortfeldt and Wäscher (2013), Iori and Martello (2010) and Pollaris et al. (2015) present a broad overview concerning container loading problems and vehicle routing problems with two- and three-dimensional loading constraints.

3.2.1 Selected studies concerning the 2E-LRP

Boccia et al. (2010), the first paper to formalize the 2E-LRP, decomposed the problem into two location routing problems (LRPs), one per echelon. A bottom-up approach was employed whereby the solution for the first echelon was generated and heuristically improved based on the solution of the second. Each echelon was decomposed into a capacitated facility location problem (the location phase) and a multi-depot vehicle routing problem (the routing phase). This decomposition approach facilitates the development dedicated location and routing heuristics to solve the problem and is the most common approach adopted in the literature.

Assessing and comparing the quality of methods proposed for the 2E-LRP is possible thanks to Boccia et al. (2010) and Nguyen et al. (2012a), who generated benchmark instances and made them publicly available. Nguyen et al. (2012a) proposed instances with a single platform, a special case referred to as the 2E-LRPSD, which has no location decisions in the first echelon. All relevant papers for the 2E-LRP have since employed these instances to assess the performance of their methods.

The two current state-of-the-art methods for the 2E-LRP are Contardo et al. (2012) and Schwengerer et al. (2012). Contardo et al. (2012) proposed a two-index vehicle-flow formulation that serves as the base for a branch-and-cut (B&C) algorithm. This exact B&C method is able to solve instances with up to 50 customers and 10 satellites to optimality. Meanwhile, Contardo et al. (2012) proposed an adaptive large neighborhood search (ALNS) heuristic. Their heuristic decomposes the 2E-LRP into two LRPs and is complemented by five local search operators which improve the second-echelon LRP. This ALNS

method outperformed previous heuristics proposed by Nguyen et al. (2012b) and Nguyen et al. (2012a). The second state-of-the-art method, Schwengerer et al. (2012), addressed the 2E-LRP with an extension of the variable neighborhood search (VNS) originally introduced for the LRP by Pirkwieser and Raidl (2010). Their method was able to generate new best solutions, but no significant difference concerning solution quality is evident when comparing the VNS with the ALNS of Contardo et al. (2012).

Table 3.1 provides an overview of other relevant work for the 2E-LRP and the 2E-LRPSD. The table indicates that high-quality solutions for medium- and large-scale instances of the 2E-LRP are most often obtained with heuristics. Although dedicated fast approaches are typically employed for the first echelon given its small size, Boccia et al. (2010) and Contardo et al. (2012) both apply a single method recursively to both echelons. We observe that applying a dedicated approach to the first echelon is preferable in terms of saving computational time while maintaining solution quality.

When location and routing heuristics are applied interchangeably (without a specific order), facility configurations may be generated without optimizing the routes, since multiple location heuristics may be applied in sequence without a routing heuristic. Therefore, we opted for an approach capable of efficiently exploring the location of facilities while optimizing the routes for each configuration.

Table 3.1: Overview of 2E-LRP approaches and related work in the literature.

Reference	Problems addressed	Method	First/second echelon method	Bottom-up	Order of location and routing heuristics
Boccia et al. (2010)	2E-LRP	H	Identical	✓	Location then routing
Boccia et al. (2011)	2E-LRP	E	-	-	-
Nguyen et al. (2012b)	2E-LRP and 2E-LRPSD	H	Different	✓	Interchangeably
Nguyen et al. (2012a)	2E-LRPSD	H	Different	✓	Interchangeably
Contardo et al. (2012)	2E-LRP and 2E-LRPSD	E and H	Identical	✓	Interchangeably
Schwengerer et al. (2012)	2E-LRP and 2E-LRPSD	H	Different	✓	Location then routing
Breunig et al. (2016)	2E-VRP and 2E-LRPSD	H	Different	✓	Location then routing
Pichka et al. (2018)	2E-OLRPSD and 2E-LRPSD	E and H	Different	-	Location then routing
This chapter	2E-LRP2L and 2E-LRP(SD)	H	Different	✓	Location then routing

E: exact approach;
H: heuristic;
Identical: a single method applied to both echelons;
Different: each echelon solved by a different method;
Interchangeably: heuristics applied without a specific order;
Location then routing: location heuristics applied before the routing heuristics;
2E-OLRPSD: 2E-LRPSD with open routes (vehicles do not need to return to depots).

3.2.2 Selected studies concerning the 2L-CVRP

When it comes to loading restrictions, the most closely related problem to the 2E-LRP2L is the 2L-CVRP. Although a generalization of the 2L-CVRP which

considers a VRP with three-dimensional loading constraints (3L-CVRP) was introduced by Gendreau et al. (2006), the first dedicated method for the 2L-CVRP was provided by Iori et al. (2007), who developed an exact approach. Wei et al. (2018) later introduced a simulated annealing (SA) framework combined with efficient data structures and a heuristic based on free spaces (unpacked regions) for the loading of items. Wei et al. (2018) outperform all previous approaches for the four variants of the problem: with/without sequential loading while allowing/prohibiting item rotation.

Various heuristic approaches with satisfactory results for the CVRP have also been extended to address the 2L-CVRP. Table 3.2 provides the characteristics of relevant work on the 2L-CVRP published beginning from the first academic paper (Iori et al., 2007) through to the current state of the art (Wei et al., 2018). All these papers employ a two-stage approach, where the routing is treated as the main problem and for each generated or candidate route a packing heuristic is called to solve the loading subproblem. Packing methods are generally constructive and the load plans computed are stored together with the routes in a memory structure to avoid re-computation. For each generated route, the memory structure is first checked for the given route's load plan and the packing method is only called if there is no load plan associated with the generated route in memory.

Table 3.2: Overview of 2L-CVRP approaches.

Reference	Method	Packing method	Memory structure for packing
Iori et al. (2007)	E	B&B	
Gendreau et al. (2008)	H	Lodi et al. (1999)	
Zachariadis et al. (2009)	H	Chazelle (1983) and Lodi et al. (1999)	✓
Fuellerer et al. (2009)	H	Martello and Vigo (1998), Lodi et al. (1999) and Burke et al. (2004)	✓
Leung et al. (2010)	H	Zachariadis et al. (2009)	✓
Leung et al. (2011)	H	Zachariadis et al. (2009) and lowest reference line best-fit heuristic	
Zachariadis et al. (2013)	H	Zachariadis et al. (2009)	✓
Wei et al. (2015)	H	Wei et al. (2011)	✓
Wei et al. (2018)	H	Open-space based heuristic	✓

E: exact approach;
H: heuristic.

3.3 Problem description

The 2E-LRP2L can be formally described with a weighted undirected graph $G = (V, E)$. $V = P \cup S \cup C$ is the vertex set where the three disjoint sets P, S and C correspond to the sets of platforms, satellites and customers,

respectively. $E = E^1 \cup E^2$ is the set of all edges, where $E^1 = \{\{i, j\} : i, j \in P \cup S, i \neq j, \{i, j\} \notin P \times P\}$ corresponds to the edges in the first echelon while $E^2 = \{\{i, j\} : i, j \in S \cup C, i \neq j, \{i, j\} \notin S \times S\}$ corresponds to the edges in the second echelon. Each $i \in P \cup S$ is associated with a capacity Q_i (given in square meters) and a fixed cost H_i incurred upon usage. Each edge $\{i, j\} \in E$ is associated with a routing cost $e_{ij} > 0$ proportional to its travel time.

Let M be the set of items requested by all customers and $M_c \subset M$ be the set of items requested by customer $c \in C$. Each item $m \in M$ is defined by its length l_m, width w_m and a service time st_m which is the time needed to (un)load the item. An unlimited fleet of homogeneous vehicles is assumed to be available for each echelon. Each vehicle has a rectangular loading surface where (un)loading operations can only be performed via the vehicle's rear. Vehicles belonging to the first/second echelon have a fixed cost h^1/h^2 and two dimensions: length l_{v1}/l_{v2} and width w_{v1}/w_{v2}. The length and width of vehicles and items are provided in centimeters. Based on the information provided by the logistics company we cooperated with, we can safely assume that physically (un)loading and maneuvering items at facilities does not pose a problem. Therefore, only how much physical space is occupied by items is considered when calculating the capacity usage at facilities. Thus, the loading constraints are only applied to the loading of items into vehicles.

A solution for the 2E-LRP2L is given by a finite set of pairs (R_k, L_k) where R_k is a route performed by some vehicle k and L_k is the full load plan of k before it reaches the first customer of its route. A route is deemed feasible if it adheres to the following constraints defined by Boccia et al. (2010): (C1) first-echelon routes start from a platform, serve one or more satellites and end at the same platform; (C2) second-echelon routes start from a satellite, serve one or more customers and end at the same satellite; and (C3) a customer must be served by a single satellite and a single vehicle. In the particular real-world application which has inspired this work, satellites have a far greater capacity than first-echelon vehicles. Therefore, the total demand allocated to a satellite may require multiple deliveries from a platform. Given this real-world demand we propose a relaxation of the standard 2E-LRP where split deliveries are allowed in the first echelon, thus a satellite may be served by multiple platforms and vehicles.

In order to mathematically represent L_k, we can consider the loading surface of a vehicle as the first quadrant of the Cartesian coordinate system bounded from above on one axis by the width and on the other axis by the length of the vehicle. The items to be delivered during R_k are represented by rectangles which must be positioned on this bounded region and are subject to the following loading constraints: (C5) all items assigned to a vehicle must fit totally within the loading surface of that vehicle; (C6) stacking is strictly forbidden and items

must not overlap with one another; (C7) orthogonal loading is required (items have a fixed orientation and may not be rotated when loaded into a vehicle). Thus, the length edge of items must run parallel to the length edge of vehicles.

The choice of these loading constraints is justified by the practical context studied in this work. In this real-world application, items are large and heavy, which makes stacking and overlapping impractical as it may result in damage or deformation. In addition, most items have wheels with limited maneuvering capacity and must be driven into vehicles, while some others must be (un)loaded using forklifts in a specific direction only, making rotation impossible.

These loading constraints are visualized by way of Figure 3.2, which provides a bird's-eye view of the vehicle loading surface. The orientation of the items to be loaded is shown in Figure 3.2(a). This same figure demonstrates a violation of the loading surface constraint (C5) as one of the items goes over the vehicle's edges. Constraint (C6) is violated in Figure 3.2(b) since two items share a portion of the loading surface (overlap). In Figure 3.2(c) the rightmost item was rotated in order to fit into the vehicle. However, rotation is not allowed and therefore this item violates the third loading constraint (C7). Finally, Figure 3.2(d) illustrates a load plan with one item less where no constraints are violated.

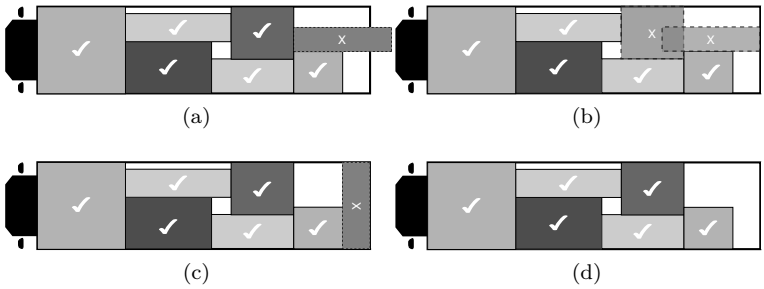


Figure 3.2: Loading constraints and violations.

In addition to the aforementioned loading constraints, the items' loading sequence plays a crucial role in the 2E-LRP2L. The literature defines *sequential loading* as follows: at each customer i, items M_i must be unloaded in a single movement without the need to rearrange any of the items assigned to customers later in the route. In other words, the space between the rear door of the vehicle and all requested items M_i should be free of any item in $M \setminus M_i$. In the 2E-LRP2L, sequential loading is always preferable and when violated it generates a penalty based on the service time of the item(s) that must be rearranged. The penalty for moving item m is twice its service time ($2 \times st_m$). This penalty

is referred to as *sequential loading penalty* or SLP (Iori and Martello, 2010) and is only applied to routes in the second echelon since we may face parking space and parking time restrictions at customer locations. Furthermore, most customers do not have the necessary equipment to (un)load every sort of item. By contrast, the sole purpose of satellites is to provide the necessary (un)loading facilities.

The objective of the 2E-LRP2L is to find a finite vector $(\mathbf{R}, \mathbf{L})$ serving all the customer demand while satisfying all the aforementioned constraints such that the total cost of (*i*) open facilities, (*ii*) routes, (*iii*) employed vehicles and (*iv*) sequential loading penalties is minimized. An example of how to calculate the cost of routes and possible penalties is shown in Figure 3.3. Recall that the items should be unloaded from rear (the rightmost side on the figure). Items 1, 2, 3 and 4 have service times equal to 10, 20, 5 and 14 minutes, respectively. Route (a) employing the load plan shown in (c) (pair a-c) and route (b) employing the load plan in (d) (pair b-d) both execute all their deliveries without needing to rearrange items. Therefore, these pairs do not incur any SLP cost. The routing cost is 105 for route (a) and 87 for route (b). However, if we consider the other two potential pairs, a-d and b-c, the costs will increase. In pair a-d, item 2 requires delivery first, but it is blocked by item 1. Thus, on top of the routing cost 105, pair a-d incurs an SLP cost of 20 due to the need to move item 1. Similarly, pair b-c delivers item 1 first, but it is blocked by item 2. Pair b-c incurs an additional SLP cost of 40 for moving item 2 while its routing cost remains 87.

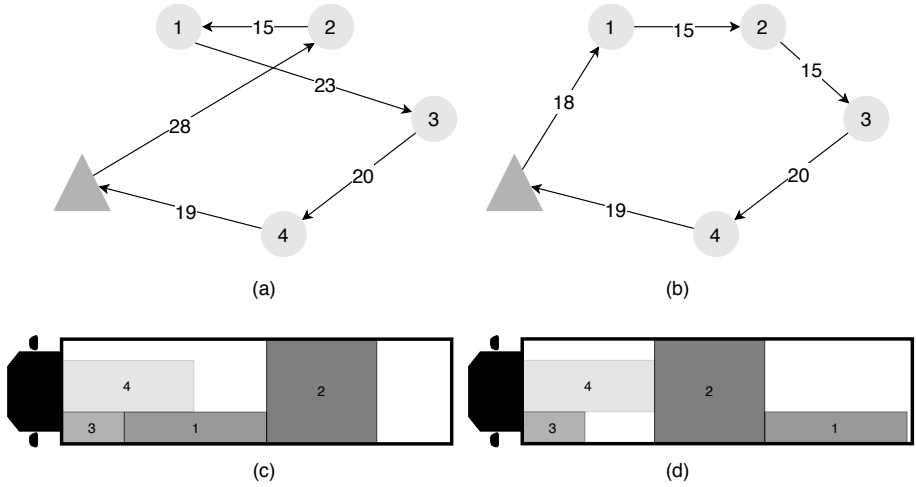


Figure 3.3: Cost for the combinations of two routes and two load plans.

A mathematical formulation for the 2E-LRP2L, although interesting as an academic exercise, would be lengthy and to solve even small-scale instances would require developing very sophisticated exact procedures, which are out of the scope of this work. For such potential future attempts, we refer interested readers to the 2E-LRP formulations proposed by Boccia et al. (2011). Their one-index formulation, provided in Appendix A.1, is a path-based formulation which defines all feasible routes. This formulation could easily be adapted to a simplified version of the 2E-LRP2L which does not consider split delivery in the first echelon. The loading constraints and sequential loading penalty could then be integrated into a pre-processing method generating all feasible routes. These feasible routes could be generated by the two-dimensional orthogonal packing with sequential loading constraint formulation by Côté et al. (2014).

3.4 Methodology

Inspired by the publications reviewed in Section 3.2.1, the method developed in this chapter decomposes the problem into two LRPs, applies a bottom-up approach with a different methodology for each echelon and optimizes the location of satellites and routes with tailored heuristics. We implement an adaptation of Iterated Local Search (Lourenço et al., 2003) for the 2E-LRP2L, henceforth referred to as ILS-LR2L. At each iteration the incumbent solution undergoes a perturbation procedure and a locally optimal solution is obtained by applying a local search operator.

Since location changes are disruptive, the ILS-LR2L perturbation operator consists of a location phase which explores several satellite configurations. For each generated satellite configuration (perturbed solution), the local search operator (routing phase) is employed, which optimizes the assignment of customers and routes. A two-stage approach during the routing phase is adopted where first the candidate route is generated and then this route's feasibility is checked by employing a given loading strategy.

The basic framework of ILS-LR2L is detailed in Algorithm 4. It begins by applying the routing local search to the initial solution and saving the best solution (s^*) found so far (lines 3-4). Next, an iterative procedure generates a neighboring solution s' by perturbing the satellite locations of incumbent solution s, applying a routing heuristic in the second echelon and a dedicated heuristic for solving the first echelon (lines 6-8). This neighbor solution s' replaces the best solution s^* if the value of s' is better than that of s^*. Every η iteration the incumbent solution s restarts from the best solution or is replaced by neighbor solution s' (lines 11-14). This procedure limits how much

an incumbent solution can differ from the best solution. After *maxNumIter* iterations, the best solution is returned (line 15). The following subsections detail the algorithm's components.

Algorithm 4: ILS-LR2L framework.

```
1  Input: initial solution s^0                    // s^0 produced by Algorithm 5
2  Input: loading strategy ls, maxNumIter
3  s ← routing local search(s^0, ls)
4  s* ← s
5  for i ← 0 to maxNumIter do
6  |   s_l ← perturb location (s)
7  |   s_r ← routing local search(s_l, ls)
8  |   s' ← solve the first echelon(s_r)
9  |   if f(s') < f(s*) then
10 |   |   s* ← s'
11 |   if i mod η = 0 then
12 |   |   s ← s*
13 |   else
14 |   |   s ← s'
15 return s*
```

3.4.1 Initial solution

The initial solution method decomposes the problem into two LRPs and builds the solution in a bottom-up manner. Therefore, this method begins by generating a solution for the second-echelon LRP. First, candidate satellites are randomly opened until their cumulative capacity is capable of serving the total demand. Next, customers are selected in a random order and assigned to the closest open satellite with sufficient residual capacity, that is, the original capacity minus the total demand of the customers previously assigned to this satellite. We adapted the Clarke and Wright (1964) *savings algorithm*, initially proposed for VRPs, to produce feasible routes for the 2L-CVRP. Given a loading strategy, the adapted *savings algorithm* is employed to generate routes for each satellite. To avoid poor initial solutions, ι solutions are produced from which the best is selected.

Using the second-echelon solution as input, the first-echelon solution can then be generated. This order is necessary since the first echelon depends on the satellite configuration and the demand associated with each satellite. In order to accelerate the optimization for the first-echelon LRP, we reduce the search space

of platform locations to explore as follows. Let $\overline{S}$ be the set of satellites opened and $demand(j)$ denote the total demand (area) allocated to satellite $j \in \overline{S}$. Let N_P^{min} denote the minimum number of platforms needed to serve $\overline{S}$. We then only consider the subsets $\overline{P} \subset P$ with cardinality $|\overline{P}| \in \{N_P^{min}, N_P^{min} + 1\}$. Consider an instance with five potential platform locations. If a minimum of two platforms is necessary to meet the total demand, then all combinations of two and three platforms will be considered. The motivation behind considering subsets of P with cardinality $N_P^{min} + 1$ is the likelihood of compensation for the cost of an additional platform with shorter routes. Preliminary experiments showed no improvements when adding more than one extra platform.

Given a platform configuration, the remainder of the initial solution method is similar to that used for the second echelon. The $demand(j)$ of each satellite $j \in \overline{S}$ is assigned to the closest platform with sufficient residual capacity. The route is then generated using the *savings algorithm*. Finally, a 2-SWAP local search operator is applied to each first-echelon solution and the best is incorporated into the complete initial solution. Algorithm 5 provides an overview of how the initial solution is generated. In this algorithm, for any set of facilities U, $demand(U)$ denotes the total customer demand in or associated with U (denoted $demand(j)$ if j is the only element in U) and $cumulativeCap(U)$ denotes the cumulative capacity of the facilities in U.

The initial solution method for the first echelon builds near-optimal solutions and is used every time a change is made to the second-echelon solution. In an attempt to save processing time, first-echelon solutions are cached together with the set of open satellites and the items required per satellite. Whenever a first-echelon solution requires the same satellite configuration, the solution in the cache is retrieved. Either the entire solution is reused if the satellite configuration and the items per satellite are the same, or only its platform configuration is reused to construct a new solution. In the latter case, the method builds a new first-echelon solution for only one platform configuration instead of all possible configurations. This cache therefore considerably decreases processing time without any loss in solution quality.

The time complexity of generating all combinations of n platforms in subsets of size u, where u is a constant, is given by $O(n^{min(u,n-u)})$. Although this operation may be costly, it is important to note that in the specific problem context of this chapter, the first echelon contains a maximum of five candidate platforms, enabling a complete search of these locations in reasonable time. To compare how much processing time is used in each echelon, we perform a preliminary test to ascertain the total time spent building first-echelon solutions throughout all iterations of the ILS-LR2L. The experiment show that at most 20% of the total running time is dedicated to generating solutions for the first echelon. Finally, since in the 2E-LRP variants the dimension of the second

Algorithm 5: Initial solution framework.

Input: empty solution $s*$, loading strategy ls
$C \leftarrow$ Customers in random order
$P_c \leftarrow$ Configurations with N_P^{min} and $N_P^{min} + 1$ platforms and sufficient cumulative capacity to serve $demand(C)$
while *number of solutions generated* $< \iota$ **do**
    Randomly open satellites until $cumulativeCap(\overline{S}) \geq demand(C)$
    foreach $c \in C$ **do**
        Assign c to the closest open satellite with sufficient residual capacity
    foreach $j \in \overline{S}$ **do**
        savings algorithm(j, ls)
    foreach $p_c \in P_c$ **do**
        Open platforms in p_c
        foreach *Pair(item, satellite)* **do**
            Assign item to the closest open platform with sufficient residual capacity
        foreach $p \in p_c$ **do**
            savings algorithm(p, ls)
        Apply the 2-SWAP operator on the first-echelon routes
        Save the current best solution
    Update the best solution $s*$
return s^*

echelon is much greater than the first, the proposed first-echelon algorithm remains relevant for larger instances as the bottleneck would continue to be the second echelon.

3.4.2 Location phase

The location phase perturbs the solution by changing satellite locations. This phase utilizes three neighborhoods: (i) close an open satellite, (ii) open a closed satellite and (iii) close an open satellite while opening one or more closed satellites. Each time the location phase is invoked, one of these three neighborhood is randomly selected. The new satellite configuration in a neighborhood is only maintained if its lower bound (see Section 3.4.3) is not worse than the current best solution. In other words, any satellite configuration which leads to a lower bound greater than the cost of the best known solution is discarded from the search space and never visited again in further iterations of any location neighborhood. As a result, the search will not spend unnecessary

processing time optimizing the assignment and routing of customers in a solution whose satellite configuration can never improve upon the current best solution. Each neighborhood works as follows.

The first neighborhood opens a random satellite and selects a subset of customers to be assigned to this satellite. The customers are selected in one of two ways: (1) customers that are closer to the new satellite than to their current one or (2) the closest customer to the new satellite along with customers in the same route that are closer to the new satellite than to their current one. Once selected, the assignment of customers is performed in a random order and ends when the satellite is full or all customers have been assigned. The routes associated with the new satellite are then created by means of the savings algorithm. This neighborhood disallows closing satellites. Thus, a customer can only be removed from a satellite that is serving two or more customers.

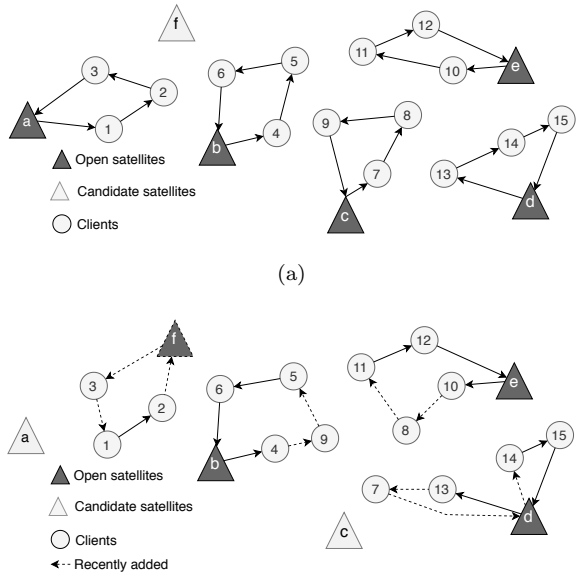
The second neighborhood closes a random satellite in such a way that the cumulative capacity of those remaining is still sufficient to meet the demand of all customers. This neighborhood tries to modify the existing routes as little as possible. First, the algorithm attempts to re-allocate entire routes to the best possible satellite while respecting capacity constraints. Whenever a complete route is unable to be reassigned, single customers are randomly inserted into the best position in their best possible route considering all satellites. An insertion position can be in a new or existing route.

The third neighborhood closes a random satellite and opens one or more satellites (depending on capacity requirements) that have the minimum average travel time from all removed customers. To avoid cycles, it is prohibited to reopen the most recently closed satellite. The insertion of customers is first conducted with respect to entire routes and then any remaining unassigned customers are inserted one by one into their best position.

Figure 3.4 shows two second-echelon solutions to illustrate changes in satellite configuration. Figure 3.4(a) employs five satellites (a, b, c, d and e) to serve all customers. Figure 3.4(b) presents an alternative solution after two moves have been applied. First, Satellite-c has been closed and its customers (7, 8 and 9) have been reassigned to nearby satellites (b, d and e). The second move swapped the status of satellites a and f, with the complete route of Satellite-a reassigned to Satellite-f.

3.4.3 Lower bound

The location of satellites is of major importance when it comes to obtaining good solutions. One can imagine that while certain configurations of open



(a)

(b)

Figure 3.4: Second-echelon routes with different satellite configurations.

satellites might result in high-quality solutions, others will consistently result in poor solutions. Thus, we design a mechanism that utilizes a lower bound function and is capable of identifying and avoiding poor satellite configurations.

Given a configuration of open satellites, the lower bound is calculated by summing together the cost of opening those satellites, the minimum vehicle cost and the minimum routing cost needed to serve satellites and customers. The minimum vehicle cost is given by the minimum number of vehicles needed in each echelon multiplied by their cost. The minimum number of vehicles is determined by the minimum length required to load all orders divided by the length of the vehicles.

Figure 3.5 illustrates how the minimum routing cost in the second echelon is calculated. A minimum spanning tree of all customers is first generated, see Figure 3.5(a). In Figure 3.5(b), this tree is then partitioned into clusters by removing the ℓ longest edges, where ℓ is the number of open satellites minus one. Each satellite is then connected to its two nearest customers (Figure 3.5(c)). The same procedure is conducted for the first echelon, where a single platform is opened and connected to the clusters of open satellites. The opened platform is the one which results in the lowest total cost considering platform opening

cost and routing cost from the platform to satellites.

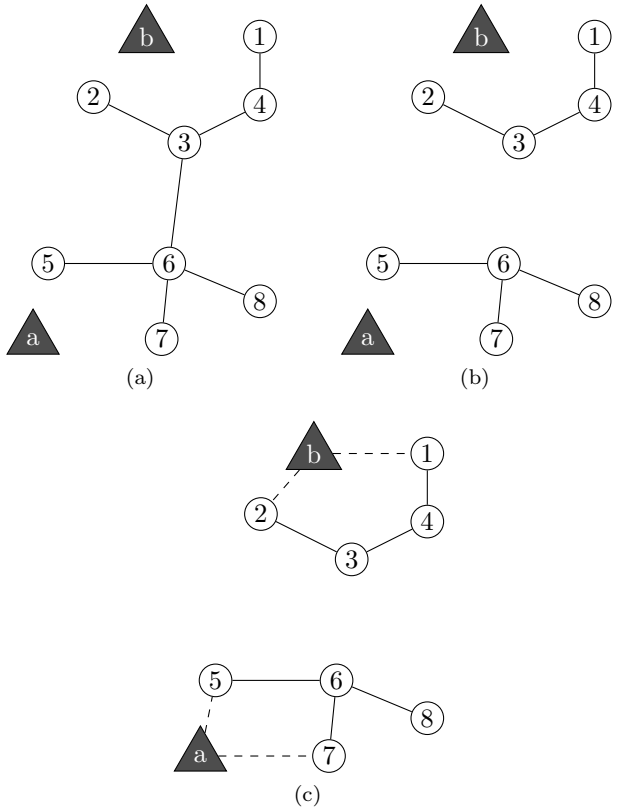


Figure 3.5: Second echelon minimum routing cost construction.

3.4.4 Routing phase

Given that the routing phase is only one of many components comprising ILS-LR2L, it must quickly produce high-quality solutions. One of the state-of-the-art heuristics for VRPs, referred to as SISRs, was introduced by Christiaens and Vanden Berghe (2020) and is adopted in this work. The ruin and recreate heuristic was chosen given its speed, simplicity to implement and proven efficiency in terms of the quality of generated solutions for a range of VRPs.

Christiaens and Vanden Berghe (2020) introduced a novel property, referred to as spatial slack, which removes a sufficient number of customers from different

geographically proximate routes, creating spatial and capacity slack in each route. Introducing such slack provides a range of options when reassigning a customer, since multiple routes close to that customer might be available for insertion. Regarding the recreate operator, insertion positions are selected using rank-based probabilities, which prevents the best insertion method from converging in an excessively greedy manner.

This chapter proposes a multi-depot adaptation of SISRs, henceforth referred to as SISRs-MD, which removes customers from different, yet geographically proximate, routes. Note that these routes may be served by different satellites. When reinserting customers, the best insertion procedure considers all satellites and routes. Each time SISRs-MD is called it performs a number of iterations and newly generated solutions are evaluated based on a simulated annealing acceptance criterion. The initial temperature value for the simulated annealing algorithm is solution-dependent and computed as $-wp \times S' / \log 0.5$ (Ropke and Pisinger, 2006), where S' is the value of the solution passed to the routing phase and wp is the worsening percentage which controls how much an accepted solution may be worse than the incumbent solution. This initial temperature states that a solution $(wp \times 100)\%$ worse than S' is accepted with 50% probability. With the exception of wp, SISRs-MD employs all the original parameter values by Christiaens and Vanden Berghe (2020) and returns an improved or equal quality solution.

Since the ruin method works based on geographic distances, it may happen that all customers removed in one iteration belong to the same satellite. Although improvements may still occur, the search would probably be too localized. Therefore, it is worthwhile developing additional operators to enforce the assignment of customers across different satellites.

The ruin and recreate methods for the SISRs-MD were extended to force, with a certain probability, the removal/insertion of customers from/into different satellites. In other words, during $\sigma_{ruin}\%$ of all SISRs-MD iterations a small set of customers are removed from every open satellite. In the same way, there is a $\sigma_{recreate}\%$ chance customers are inserted into their best position considering all satellites and a (100-$\sigma_{recreate}$)% chance they are inserted into their best position excluding the satellite from which they were just removed.

To further improve routes, at the end of the routing phase SISRs is applied to each satellite followed by a descent heuristic. We consider two descent heuristics: MOVE and 2-OPT*. The MOVE operator relocates a sequence of 1-3 customers to its best position in any route. The operator works intra- and inter-route as well as intra- and inter-satellite. Meanwhile, the 2-OPT* operator (Potvin and Rousseau, 1995) is only performed intra-satellite for every pair of routes u and v and every customer $i \in u$ and $j \in v$. Both the MOVE and 2-OPT* operators

are performed in a random order and in a first-improvement manner. The two operators are applied repeatedly and the search ends when both operators return a solution with no improvement. These additional operators are commonly used for VRPs, cheap in computational cost and bring an improvement of 0.03% on average for the considered instance sets.

3.4.5 Load plan method

In order to create feasible load plans we apply an algorithm to determine the placement (position) of items inside vehicles. This load plan method (LPM) uses a placement heuristic and attempts to build a feasible load plan for a given vehicle size and set of items. We apply the best-fit placement heuristic proposed by Burke et al. (2004) and follow the efficient implementation by Imahori and Yagiura (2010). We choose this fast approach since it produces high-quality solutions and is simple to implement. The LPM can be called when a new customer or item is inserted into a route (by any of the local search operators) or when the visited order of customers is altered.

In order to produce a higher number of feasible load plans, the method by Burke et al. (2004) is extended to include additional strategies and insertion policies. The implementation in this chapter sorts items based on their width, length and area. Ties between items with the same dimension are broken by either length or width. Items are sorted in reverse order of delivery with a view to achieving a low SLP value. Thus, the last item to be delivered is placed inside the vehicle first. When placing an item inside the vehicle, one of two insertion policies can be followed: insert the item next to the longest neighbor or insert the item next to the neighbor with the most similar length. The LPM combines each item order with an insertion policy and returns the first feasible load plan found.

Every time a load plan is generated, feasible or not, it is stored together with the route in a memory structure entitled Trie (Leung et al., 2010). Trie avoids evaluating routes that have already had their load plan checked. Thus, given a route, the method first searches for it in the Trie structure and, if it is not present there, only then is the LPM called and the structure updated with the new route and corresponding load plan.

3.4.6 Loading strategies

Different loading strategies are proposed to study the best way to handle the loading constraints associated with the 2E-LRP2L. By employing different

strategies, this chapter aims to investigate the impact a dedicated load plan method has on 2E-LRP2L solutions and to study how item loading sequences and SLP influences solution quality.

This first strategy is referred to as *LazyLoading* and attempts to mimic what is currently done in practice by the company that inspired this study. *LazyLoading* assumes that the loading of large items is simple and that a dedicated load plan method is unnecessary. Therefore, when constructing vehicle routes throughout ILS-LR2L iterations, only one-dimensional capacity constraints are enforced. In this case, items are loaded into vehicles as long as their accumulated area is less than the vehicle's total area. The LPM described in Section 3.4.5 is then applied only a single time in the final solution to generate a load plan and calculate the SLP.

During the preliminary tests, we observed that the ILS-LR2L with *LazyLoading* without any further adjustments resulted in infeasible load plans for 50% of the routes. More specifically, although the vehicle capacities were respected area-wise, in reality the items could not fit together inside the vehicle. Consider Figure 3.6, where the total area of the vehicle is 17.5 and the three items which require loading have a combined total area of 14.18. If one only considers area one would expect a feasible load plan, yet in reality a feasible load plan is impossible due to the dimensions of the items.

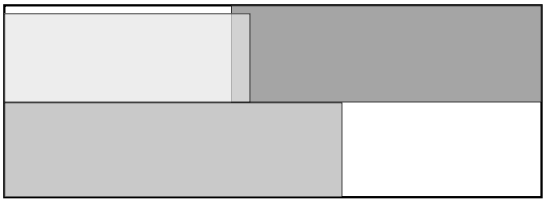


Figure 3.6: An infeasible load plan for items whose total area is less than that of the vehicle.

If a vehicle turns out to have an infeasible load plan, items will need to be unloaded and reloaded into other vehicles or additional vehicles may need to be employed. This may lead to additional costs and delays. As a quick remedy for this issue, route planners in the company artificially modify the vehicle capacities. More specifically: they employ reduced capacities in their *LazyLoading*-like strategy.

Such a quick fix increases the chance of generating feasible solutions without needing to validate the challenging two-dimensional loading constraints. Therefore, we impose a *maximum occupancy rate* on the *LazyLoading* strategy. $MaxL_{v1}$ and $MaxL_{v2}$ denote these rates for the first- and second-echelon

vehicles, respectively. Although these restrictions significantly reduce the number of infeasible solutions, 3% of the routes *LazyLoading* generates using these maximum occupancy rates continue to be infeasible.

To guarantee route feasibility and evaluate the impact of the loading constraints on the final solution a different approach is taken. *ActiveLoading* employs the LPM for every generated route. This approach is necessary since, as demonstrated, capacity constraints do not guarantee the generation of feasible routes for the 2E-LRP2L. Additionally, to investigate the impact of the loading sequence of items in the solution, the *ActiveLoading* approach is divided into three loading strategies, where each is defined in accordance with how exactly they handle the SLP. The three strategies, ordered from least to most restrictive, can be summarized as follows:

- *LateSLP*: route feasibility is checked by the LPM but no SLP is applied until the final solution.
- *OnTimeSLP*: the SLP is calculated and added to the route cost on the fly at each load plan feasibility check.
- *StrictSLP*: the SLP is calculated at each load plan feasibility check and routes with non-zero SLP are considered infeasible. Note that this strategy is capable of providing guaranteed feasible solutions for an eventual 2E-LRP2L variant where sequential loading is mandatory.

These three *ActiveLoading* strategies result in lengthy computational times given that the LPM (calculating or not the SLP) is invoked for every generated route. However, it is unlikely that infeasible load plans are generated at the beginning of route construction since vehicles are largely empty and there is plenty of space to place items. Therefore, to reduce the processing time of the *ActiveLoading* strategies, we propose a *minimum occupancy rate* so that the LPM is only called if the occupancy rate is above $MinL_{v1}$ and $MinL_{v2}$ for first- and second-echelon vehicles, respectively.

3.5 Computational study

All experiments were conducted on a computer with an Intel Xeon E5-2660 processor at 2.6 GHz, with 164 GB of RAM running Ubuntu 18.04 LTS. ILS-LR2L was implemented in C++ and compiled using gcc 7.4.0 and option -O3.

Table 3.3 details all necessary parameters for implementing ILS-LR2L and their respective values. Parameters concerning the number of iterations were calibrated manually with a compromise between solution quality and processing time. The other parameters were calibrated using *irace* (López-Ibáñez et al., 2016), with the range of values provided to irace also shown in Table 3.3.

Table 3.3: ILS-LR2L parameters and values.

Parameter	Value	Parameter	Value	Range
$maxNumIter$	50	wp	0.003	{0.001, 0.003, 0.005, 0.01, 0.03}
SISRs-MD iterations	1000	η	2	{1, 2, 4, 6, 8, 10}
SISRs iterations	500	$\sigma_{recreate}$	60	{20, ...,80}
		σ_{ruin}	50	{20, ...,80}
		ι	10	{1, 5, 10, 25, 50, 75, 100}
		$MaxL_{v1}$	80%	{50, ..., 100}
		$MaxL_{v2}$	70%	{50, ..., 100}
		$MinL_{v1}$	70%	{50, ..., 95}
		$MinL_{v2}$	60%	{50, ..., 95}

During preliminary experiments, the maximum occupancy rate parameters were calibrated in such way that no infeasible route is generated when the occupancy rate is under $MinL_{v1}$ and $MinL_{v2}$. When employing these minimum occupancy rates, the processing times for the *ActiveLoading* strategies are eight times faster, with the average processing time plummeting from 3768 seconds to 470 seconds without any loss in solution quality. It is important to note that for all the experiments in this section, $MaxL_{v1}$ and $MaxL_{v2}$ are always enforced when using the *LazyLoading* strategy, while $MinL_{v1}$ and $MinL_{v2}$ are always enforced when using one of the three *ActiveLoading* strategies.

3.5.1 New instance set

To run experiments with ILS-LR2L, and in order to stimulate further research regarding the 2E-LRP2L, instances were generated based on real-world data. A transportation company of large equipment provided locations of facilities (platforms and satellites) and customers, vehicle and items sizes (l_v, w_v and l_m, w_m), and approximately two months worth of customer demands and their respective day of delivery. There are up to four platforms and eight satellites which can be operational depending on the total customer demand. Vehicles have equal width (2.5 meters) but three different lengths (17.6, 8.5 and 7 meters), while items are at most 5 meters long and 2.5 meters wide.

A travel time matrix provides realistic travel times between all pairs of locations. Meanwhile, service time st_m is based on item m's dimensions and may range

from 5 to 20 minutes. Requests were divided into 32 instances according to their delivery date and range from 81 to 274 customer requests. Due to privacy issues, all identifying features such as names and physical locations have been omitted.

The cost and capacity of facilities and the cost associated with vehicles were not provided by the company and were generated for this chapter in accordance with the 2E-LRP benchmark instances. We first analyzed how the cost is distributed among the many components present in a 2E-LRP solution. Next, costs were calibrated to generate 2E-LRP2L instances with similar proportions. The cost breakdown is illustrated in Appendix A.2 (Figure A.1), where the total cost of an initial solution is divided in percentages considering first and second echelons, the cost of open facilities and routes, as well as traveling time and vehicle cost. The reported percentages are an average across all instances.

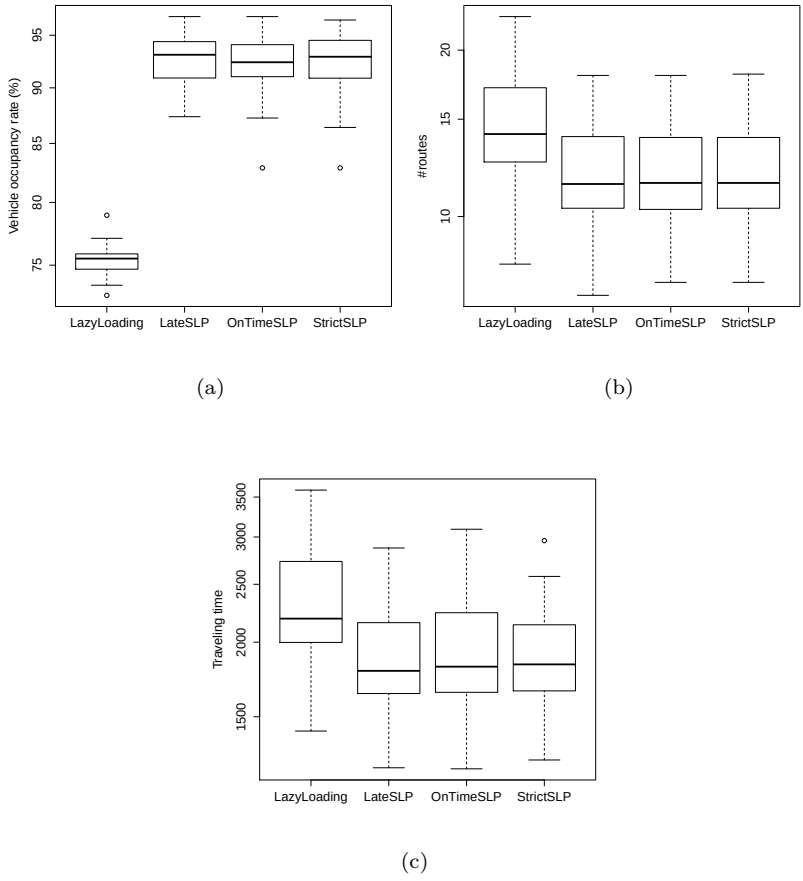
Tuning facility capacities correctly is important in order to create instances that represent the range of different sizes observed in practice. The idea is to assign capacities to facilities in such a way that all customers may be served without needing to open all facilities. For each instance, given a set of customers and their respective demand, the total area of all requested items was taken into account when generating the capacity of facilities. Satellite capacities were randomly generated between 17% and 30% of total item area. In this way, solutions must open at least four satellites out of the eight available. Platform capacities were selected between 40% and 70% of total item area, which results in solutions with two or three open platforms out of four.

3.5.2 Detailed results

This section presents computational experiments and compares the results produced by ILS-LR2L using the four different loading strategies. Figures 3.7(a)-3.10(a) provide boxplot charts which compare the four loading strategies, namely: *LazyLoading*, *LateSLP*, *OnTimeSLP* and *StrictSLP*. The values in the charts correspond to the average values of 10 runs across all 32 instances. To evaluate statistically significant differences, the pairwise T-test was performed with a confidence level of 95%.

Figures 3.7 provides computational results concerning the first echelon. The *LazyLoading* strategy, which employs a maximum occupancy rate of 80% for the first-echelon vehicles, uses less vehicle capacity than the other strategies. As a result, one can observe that more routes and longer traveling times are needed. More specifically: *LazyLoading* employs 22.8% more routes on average and travels 20.3% longer. The pairwise T-test shows that while all three *ActiveLoading* strategies perform similarly, given that no SLP is applied in

the first echelon, statistically significant differences are observed between each *ActiveLoading* strategy and *LazyLoading*.



(a) (b)

(c)

Figure 3.7: Different loading strategies for the first echelon.

Figure 3.8 provides the results of the second echelon, where the SLP is applied. For the *ActiveLoading* strategies, Figure 3.8(a) shows a decrease concerning vehicle occupancy rates as the loading constraints become more restrictive. However, *LazyLoading* is a clear outlier here due to the aforementioned maximum occupancy rate restrictions (which are set to 70% for second-echelon vehicles). A similar discrepancy can also be observed in Figure 3.8(b), where *LazyLoading* uses 34.92% more routes than *StrictSLP*. Statistically significant differences are

observed for every pair of strategies with the exception of pair *OnTimeSLP*-*StrictSLP*. In other words, for the *ActiveLoading* strategies, *LateSLP* uses significantly fewer routes with more load per vehicle than *OnTimeSLP* and *StrictSLP*.

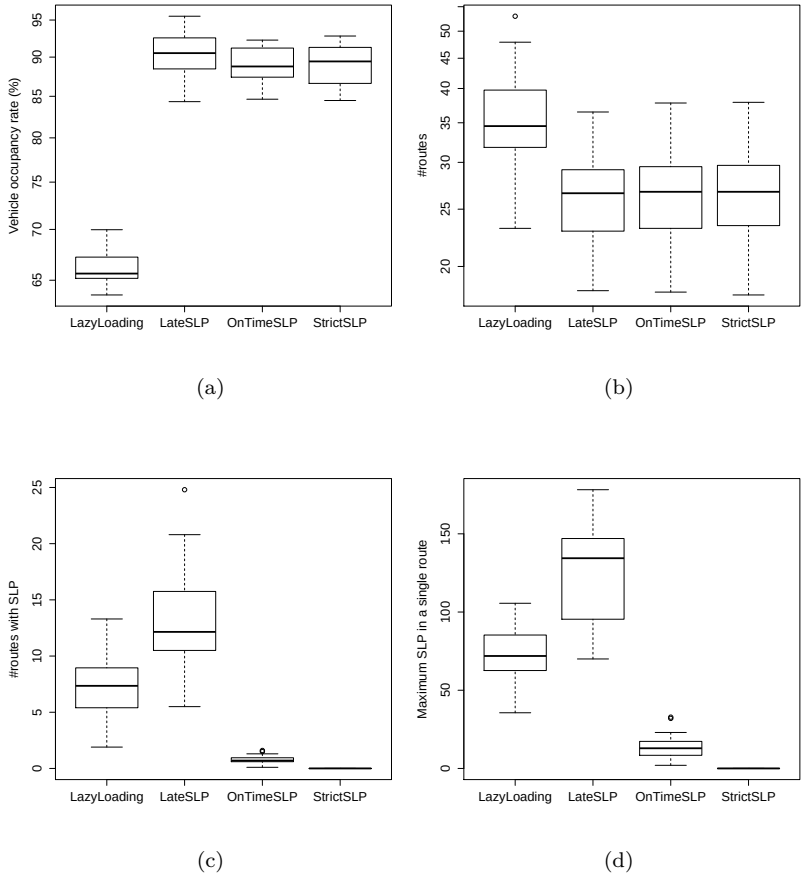


Figure 3.8: Different loading strategies for the second echelon.

Figure 3.8(c) shows the proportion of routes which incur penalties per loading strategy. Only 2.9% of the routes produced by *OnTimeSLP* incur SLP penalties, while *StrictSLP* forbids them totally. A significant increase can be observed when the SLP is not applied during the routing phase, as 19% and 48% of the routes produced by *LazyLoading* and *LateSLP*, respectively, incur a penalty.

More penalties are observed in *LateSLP*'s routes than in *LazyLoading*'s due to the fact that the vehicles of *LateSLP* have much higher occupancy rates. Thus, the more items a load plan has, the greater the chance sequential loading is violated when it is not explicitly checked for. The total penalty associated with a single route may reach as high as 178 minutes (see Figure 3.8(d)), which is a very long idle time spent at the customer's location when one considers that the average driving time for a route is only 115 minutes. Such results imply that poor loading sequences can result in vehicles spending upwards of half their service time parked at customer locations rearranging items.

Figure 3.9 provides the total traveling time and the total penalty incurred by the second echelon. The *LazyLoading* and *LateSLP* strategies take up to 20% longer to complete routes. This additional time is strongly related to the penalties their routes incur. When comparing only the traveling time, statistical tests demonstrate significant differences between every pair of strategies, with the exception of *OnTimeSLP-StrictSLP*. With respect to the total time spent, the traveling time with SLP (Figure 3.9(c)) results show that all pairs are significantly different, with the exception of *LazyLoading-LateSLP*.

Finally, Figure 3.10 illustrates the average processing time and the best solution found across all 10 runs. Since *LazyLoading* does not employ the LPM, it is considerably faster than the *ActiveLoading* strategies. As for solution quality, *StrictSLP* performs best, generating 21 of the best-known solutions with only a 0.12% gap with respect to the best solutions found by the other strategies. *OnTimeSLP* obtained the remaining 11 best-known solutions and exhibits a gap of 0.29%. *LateSLP* and *LazyLoading* clearly have the worst average solutions and exhibit gaps of 5.45% and 15.42%, respectively. For both processing time and best solution values, there are statistically significant differences between all pairs of strategies, with the exception of *OnTimeSLP-StrictSLP*. The detailed results of each variant are available as supplementary online material.

3.5.3 Result discussion

Even though *LazyLoading* is considerably faster than the three *ActiveLoading* strategies, it is associated with more routes and substantial penalties which translate into far higher total costs. The poor performance of the *LazyLoading* strategy in terms of solution quality indicates the necessity of applying a dedicated load plan method during the optimization process.

When dealing with large items, the time spent moving (loading and unloading) such items may be significant when compared to the average travel time. Considering the three *ActiveLoading* strategies, *LateSLP* results in slightly shorter travel times than *OnTimeSLP* and *StrictSLP*. However, *LateSLP*'s

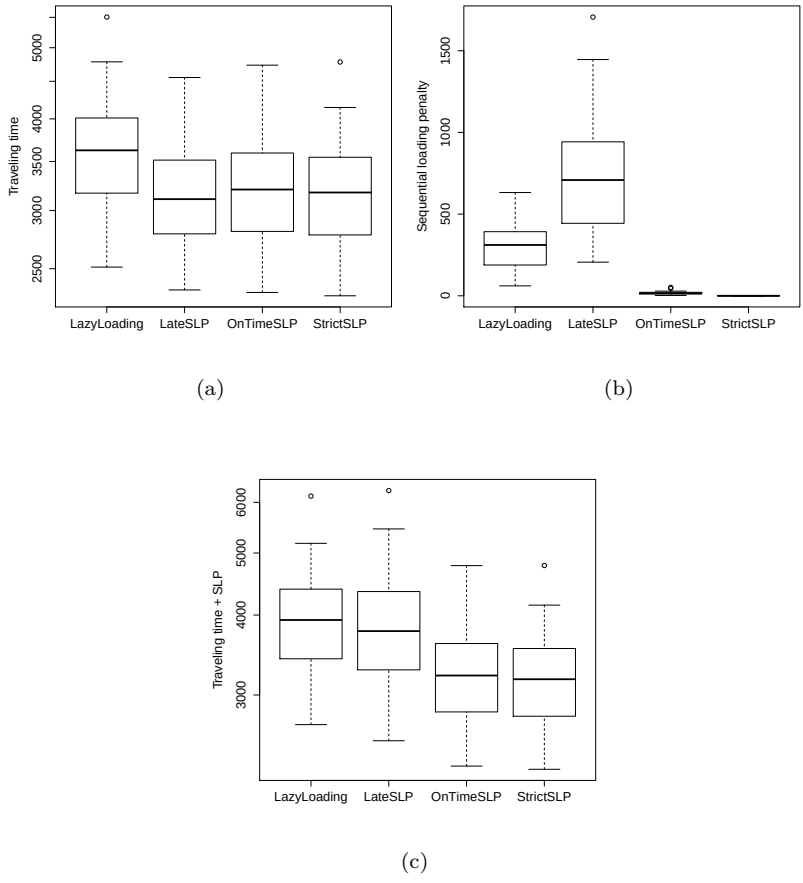
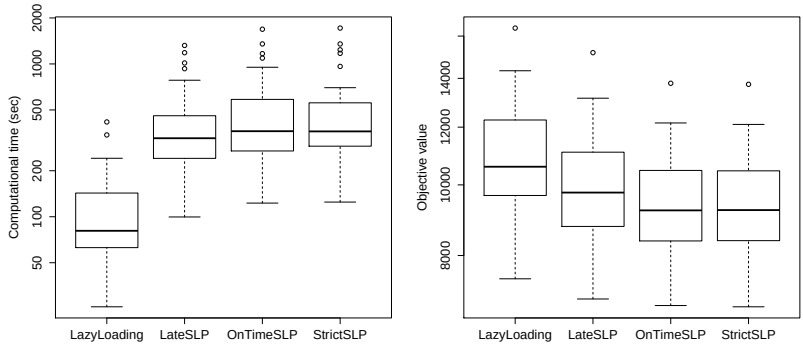


Figure 3.9: Different loading strategies for the second echelon. Traveling time and penalty analysis.

poor final solution quality demonstrates how SLPs may negatively influence solution values when they are ignored. *OnTimeSLP* and *StrictSLP* consider the sequential loading penalty at all times and, as a result, produced the best solutions. *OnTimeSLP* efficiently keeps penalties at a low level and generates good quality solutions. The similarity between *OnTimeSLP* and *StrictSLP* is noticeable, highlighting the impact of the SLP on the final solution in the studied context: that of large items whose rearrangements are time-consuming compared to travel times.



(a) (b)

Figure 3.10: Best solution comparison and run time of different loading strategies.

StrictSLP not only produces high-quality solutions for the present problem, but it can also be used for problem environments in which sequential loading is mandatory. This situation occurs when customers do not possess the unloading equipment necessary to move other customers' items, when physical space at the customer locations is limited or when parking time limits are in effect.

3.6 The 2E-LRP special case

This section is dedicated to strengthen the computational experiments by using benchmark instances available in the literature. Given the absence of instances for the 2E-LRP2L, the quality of our heuristic is evaluated by solving the 2E-LRP special case. In order to eliminate unnecessary loading constraints and split-delivery procedures and improve the efficiency of the algorithm when solving this special case problem, we initially present a few modifications of ILS-LR2L. The results obtained from this modified algorithm, which we refer to as ILS-LR, are then compared against those generated by state-of-the-art methods and statistical tests are conducted to evaluate and verify the competitiveness and quality of ILS-LR.

In the 2E-LRP, the capacity of platforms and satellites is given by Q_i ($i \in P \cup S$). A demand $D_c > 0$ is associated with each customer $c \in C$. Vehicles in the first and second echelons have capacity q^1 and q^2 and a fixed cost h^1 and h^2 is

incurred when such vehicles are routed. The feasibility of facilities and routes relies only on the total capacity being respected. The 2E-LRP does not allow split deliveries to take place in either echelon and therefore customers and open satellites must both be served by a single vehicle.

3.6.1 Benchmark instances

This computational study employs three benchmark sets, which are referred to as *Prodhon*, *Nguyen*, and *Sterle* and contain a total of 147 instances. The first two instance sets were introduced by Nguyen et al. (2012a) and contain only one platform in the first echelon. *Prodhon* comprises of 30 instances arising from the CLRP with the addition of a single platform at coordinates (0,0). The instances range in size from 20-200 customers and 5-10 satellites. The second set, *Nguyen*, comprises of 24 instances containing 25-200 customers and 5-10 satellites. The third and final instance set, *Sterle*, was generated following the specifications outlined by Boccia et al. (2010) and made available by Contardo et al. (2012). The instance set is composed of 93 instances divided into three groups (I_1, I_2 and I_3) with different spatial distributions of satellites. Instances contain 8-200 customers, 3-20 satellites and 2-5 platforms.

3.6.2 ILS-LR for the 2E-LRP

The initial solution for both echelons employs the heuristic described in Section 3.4.1, with the exception that split delivery is not allowed. As a result, a platform must serve all items requested by a given satellite. The location and routing phases are also conducted as before. Meanwhile, the lower bound is calculated slightly differently: the minimum number of required vehicles is given by the total demand of goods divided by vehicle capacity.

The final solution of the 2E-LRP undergoes a post-processing step consisting of an exact approach for the first echelon. Since first-echelon dimensions are generally small, with few platforms and routes the formulation can be solved within short computational runtimes. For instances with only one platform, an optimal solution is generated using the Capacitated Vehicle Routing Problem formulation introduced by Kulkarni and Bhave (1985). For first-echelon problems with multiple platforms, we propose the Capacitated Multi-Depot Location Routing Problem (CMLRP) formulation in Appendix A.3. This CMLRP model uses load conservation constraints for sub-tour elimination (routes disjoint from P) and determines both the platform locations and the routes from those platforms to the given satellites.

For the 2E-LRP benchmark considered in this work, the mathematical formulations are on average responsible for a marginal increase in time of 23 seconds per instance set while providing an improvement of 0.02% in the solution value. This post-processing step guarantees optimal CVRP/CMLRP solutions for the first echelon with short processing time and can be valuable for future sets of instances of a similar size. Since these mathematical formulations are a new component in the methodology for the 2E-LRP, which was not used in the 2E-LRP2L method, we performed statistical tests to analyze their influence on solution value. These tests revealed no significant difference between ILS-LR approaches with and without the mathematical formulations on the current sets of instances. Therefore, we remain confident of the quality and competitiveness of ILS-LR2L, even without an exact approach as post-processing for the first echelon.

3.6.3 Parameters and setup

Experiments were performed using the same computational architecture and setup detailed in Section 3.5. The values of all the calibrated parameters were maintained. The only difference lies in the algorithmic adaptations detailed in Section 3.6.2 and the number of iterations, which was increased given that the ILS-LR is much faster without the loading constraints (see Table 3.4).

Table 3.4: Parameters and respective values for the ILS-LR.

Parameter	Value
$maxNumIter$	150
SISRs-MD iterations	15000
SISRs iterations	5000

3.6.4 Comparison with the state-of-the-art 2E-LRP methods

Table 3.5 compares ILS-LR against two state-of-the-art algorithms: ALNS (Contardo et al., 2012) and VNS (Schwengerer et al., 2012). Due to the algorithms' stochastic component, 10 runs were performed for each instance. Gap_{min} from Table 3.5 refers to the percentage difference between the best value s obtained from all runs and the best value published $s*$. $Gap_{s,s*}$ is calculated as $100 \times \frac{s-s*}{s*}$. Meanwhile, Gap_{avg} corresponds to the percentage difference between the average best solution, considering all the runs, and the best-known solution. The average CPU time in seconds is provided in column t_{avg}.

According to Passmark (accessed May 11, 2020), the hardware we employed is 1.37 and 1.50 times faster than the hardware employed by Contardo et al. (2012) and Schwengerer et al. (2012), respectively. Therefore, runtimes have been converted so as to enable a fair comparison (original runtime from ALNS and VNS divided by 1.37 and 1.50, respectively). The number of best-known solutions found by each algorithm is detailed in the last row of the table. More detailed results are available as supplementary online material.

Table 3.5: Comparison of state-of-the-art methods. Best results of ILS-LR are shown in bold.

	ALNS			VNS			ILS-LR		
	Gap_{min}	Gap_{avg}	t_{avg}	Gap_{min}	Gap_{avg}	t_{avg}	Gap_{min}	Gap_{avg}	t_{avg}
Prodhon	0.32	0.83	339.17	0.08	0.50	208.69	0.08	0.27	134.30
Nguyen	0.16	0.49	139.78	0.27	0.90	183.08	**-0.03**	0.29	161.12
Sterle I1	0.24	0.41	223.36	0.05	0.60	222.98	**-0.02**	0.20	134.05
Sterle I2	0.25	0.45	241.01	0.25	0.72	202.65	**0.11**	0.32	197.40
Sterle I3	0.05	0.22	240.17	0.04	0.35	191.70	**0.02**	0.25	192.67
# of BKS(147)	115			115			127		

ILS-LR obtains 75.51% of the best-known solutions and improves upon an additional 16 instances (10.88%). Regarding the average gap, ILS-LR outperforms the previous algorithms across almost all instance sets in terms of both best solutions and average solution values. The average gap produced by the ILS-LR is the lowest of all methods, except for the best solution for the Prodhon instance set and the average solution for the Sterle I3 instance set.

A pairwise T-test was performed to assess the results of the three algorithms for each instance set. Table 3.6 reports these results, with statistically significant differences highlighted in bold. With a confidence level of 95%, ILS-LR dominates previous state-of-the-art methods for the Sterle instance set and is clearly competitive for the other two.

Table 3.6: Pairwise T-test results.

	Prodhon		Nguyen		Sterle	
	ALNS	ILS-LR	ALNS	ILS-LR	ALNS	ILS-LR
ILS-LR	**0.011**	-	0.12	-	**0.036**	-
VNS	**0.031**	1.00	1.00	0.08	0.426	**0.042**

3.7 Conclusions

The two-echelon location-routing problem (2E-LRP) considers freight distribution at two levels, where decisions include choosing the location of facilities

and routing from facilities to customers. This challenging problem has been widely studied and a number of methods are available. However, in practice, transportation companies are faced with additional complex decisions regarding the loading of items.

In order to supply such companies with feasible solutions, this chapter introduced the two-echelon location-routing problem with two-dimensional loading constraints (2E-LRP2L) and a heuristic optimization method named ILS-LR2L. With this heuristic, several loading strategies, concerning how to handle the loading of items, were investigated using a set of instances based on real-world data. These instances have been made publicly available online to stimulate future research on the 2E-LRP2L.

This chapter revealed that a loading strategy which considers only one-dimensional capacity constraints during optimization may lead to many infeasible routes, even when considering small numbers of large items. Results indicate that a dedicated load plan method, which considers two-dimensional loading constraints, considerably increases the occupancy rate of vehicles and thus reduces the number of necessary routes. Moreover, if moving items is time-consuming, optimizing the loading sequence of such items helps reduce route duration by as much as 20%.

With minor modifications, ILS-LR2L was also able to address the 2E-LRP. Computational experiments demonstrated that this modified version of the algorithm is competitive, producing a higher number of best-known solutions than previous state-of-the-art methods, while also generating some new best solutions. Statistical tests show how the ILS-LR2L dominates previous approaches for at least one instance set.

In terms of future research, this study provides a basic methodology for investigating the impact of real-world constraints on vehicle routing solutions. For approaches considering two-dimensional loading constraints, it is strongly recommended to define the minimum vehicle occupancy rate presented in this chapter, as it is able to drastically reduce the processing time. The loading constraints could further be generalized to handle three-dimensional stacking.

Chapter 4

Conclusion

This thesis introduced new VRP generalizations arising in logistic companies and investigated the impact of new operational constraints on solution quality. The intricate and very complex VRPs studied in this thesis arise from intertwined routing and loading decisions, synchronization constraints and different forms of location decisions. The general approach adopted in this thesis consists of a heuristic which decomposes each problem into distinct levels where different location and routing decisions are made. The heuristic search is carried out by first applying a method which decides (and then fixes) the location of intermediate facilities or transfer locations and then, for each of these partial solutions, a state-of-the-art VRP algorithm is used to optimize the routing component of the problems.

The location phase of each algorithm comprises of multiple neighborhoods. In the VSBR the location phase is responsible for inserting or removing body transfers, which may take place at either customer locations or special transfer points. This same phase in the 2E-LRP2L is responsible for opening, closing or swapping the status of satellites. These location decisions significantly increase the problems' search space and require the creation of better ways to explore it. In the VSBR, a score-based selection was employed which attempts to schedule body transfers at more advantageous customer locations. By assigning scores to each customer, transfers at a customer which is located very far from the others and which would generate poor-quality solutions are selected with a far lower probability. Following the same general idea, a lower bound was developed for the 2E-LRP2L which discards solutions with satellite configurations that are unable to improve upon the current best solution. These mechanisms filter location possibilities and help guide the search towards high-quality solutions

in shorter computational runtimes.

During the location phase in both algorithms emphasis is placed on keeping the disturbance in already optimized routes to a minimum. When removing a transfer in the VSBR, only routes associated with the transferred body are altered. For the insertion of body transfers, the pickup of a given body is scheduled in a route while the others remain intact. When a satellite is closed in the 2E-LRP2L, the method first attempts to insert entire routes into already open or newly opened satellites. The idea is to create a neighbor solution with different location configurations while maintaining high-quality routes as much as possible. This is important since the perturbation phase in ILS frameworks should not be too disruptive, otherwise the local search which follows would be unable to obtain significant improvements.

After the location phase, the routing phase of each algorithm solves a simplified version of their corresponding problem. By fixing body transfers, the routing phase in the VSBR solves a pickup and delivery problem with time windows and open routes, while a multi-depot VRP is solved in the 2E-LRP2L by fixing satellite locations. The routing phase can also be computationally intensive as best insertion methods often require many iterations and route evaluations. Therefore, techniques were applied to speed up route evaluation in both algorithms. By using forward time slack and forward load slack, the feasibility of a route can be checked in constant time during the VSBR routing phase. The route feasibility check in the 2E-LRP2L requires checking several two-dimensional loading constraints. To accelerate this process a minimum occupancy rate is employed where only routes above this threshold are evaluated, while the others are assumed to be feasible. Additionally, a memory structure holds the already calculated load plans to avoid unnecessary recomputations.

Computational experiments were performed using instances generated based on real-world data. The newly introduced constraints in both problems were analyzed to assess their impact on solution quality. These experiments demonstrated that body transfers lead to shorter vessel routes and more customers served in the VSBR, while taking into account the two-dimensions layout and the loading sequence of items in the 2E-LRP2L results in more feasible routes and helps reduce the total duration of those routes. With some minor modification, the algorithm developed in Chapter 3 for the 2E-LRP2L was also compared against state-of-the-art methods for the special case 2E-LRP. The results of these experiments confirmed the competitiveness of the proposed heuristic as it was able to obtain most best-known solutions and even improve some. A similar comparison, however, was not performed in Chapter 2 since solving special cases related to the VSBR would require a fundamental restructuring of the proposed heuristic, leading to an unfair comparison and inconclusive results.

Researchers interested in studying VRPs with diverse location decisions in addition to other levels of complex decisions should consider developing algorithms that are capable of iteratively addressing each decision level. Components associated with each decision level should be efficient as the search will jump back and forth between them to explore the search space. Despite the specificities of each multilevel VRP variants, the large search space associated with them requires methods that direct the search towards high-quality solutions, identifying and discarding poor solutions as quickly as possible. The results of this thesis emphasize the importance of paying necessary attention to efficient data structure and pre-processing methods when implement an algorithm, which save significant time when evaluating the many neighbor solutions generated.

4.1 Future research for this thesis

Local-search-based methods often rely on performing many iterations, with the computational time associated with each iteration varying depending on the size of local search neighborhoods. Each neighboring solution must first have its objective value evaluated and only then can it be decided whether or not this solution should be accepted before continuing on with the search. This kind of method may expend a large portion of its computational resources exploring parts of the search space which consist only of poor solutions.

As researchers move towards extending their models by including real-world constraints to bridge the gap between theoretical problems and practice, models tend to become slower as the search space grows in size. This thesis should not only focus on extending its models with these realistic constraints, but also exploiting problem-specific structures in order to develop and incorporate techniques to reduce the search space by avoiding certain regions and discarding poor solutions. Ideally, these poor-quality solutions would be identified and discarded before they are even evaluated, thereby redirecting the search towards higher quality regions. One example of such a technique is the lower bound developed for the 2E-LRP2L, which discards solutions with poor satellite configurations. Other possibilities which this thesis aims to experiment with include incorporating exact optimization methodologies such as valid inequalities generation and constraint relaxation in local-search algorithms.

Appendix A

A.1 2E-LRP formulation

We present the one-index formulation for the 2E-LRP proposed by Boccia et al. (2011) for a clear understanding of the constraints. This path-based formulation could easily be adapted to a simplified version of the 2E-LRP2L and comprises of the following variables:

$D_c > 0$ demand of each customer $c \in C$;

$y_i = \{0,1\}$, $i \in P \cup S$: 1, if a facility is open at node i, 0 otherwise;

$\mathcal{T}^1$ and $\mathcal{T}^2$ are the sets of routes for the first and second echelons;

$\mathcal{T}_p^1$ is the set of routes starting from platform p ($\mathcal{T}_p^1 \subseteq \mathcal{T}^1$);

R_i is the total cost of route $i \in \mathcal{T}^1 \cup \mathcal{T}^2$;

$x_i = \{0,1\}$, $i \in \mathcal{T}^2$: 1 if a second-echelon route is selected, 0 otherwise;

$r_i = \{0,1\}$, $i \in \mathcal{T}^1$: 1 if a first-echelon route is selected, 0 otherwise;

$f_i \geq 0$, $i \in \mathcal{T}^1$: flow traveling on first-echelon route i from platforms to satellites;

$\alpha_{is} = \{0,1\}$: 1 if a satellite $s \in S$ is covered by first-echelon route $i \in \mathcal{T}^1$, 0 otherwise;

$\beta_{ic} = \{0,1\}$: 1 if a customer $c \in C$ is covered by second-echelon route $i \in \mathcal{T}^2$, 0 otherwise;

$\epsilon_{ps} = \{0,1\}$: 1 if a satellite $s \in S$ may be covered by a platform $p \in P$, 0 otherwise;

$\varphi_{sc} = \{0,1\}$: 1 if a customer $c \in C$ may be covered by satellite $s \in S$, 0 otherwise.

$$Minimize : \sum_{p \in P} H_p y_p + \sum_{s \in S} H_s y_s + h^1 \sum_{i \in \mathcal{T}^1} r_i + h^2 \sum_{i \in \mathcal{T}^2} x_i + \sum_{i \in \mathcal{T}^1} R_i r_i + \sum_{i \in \mathcal{T}^2} R_i x_i. \tag{A.1}$$

which are subject to

$$\sum_{s \in S} \varphi_{sc} y_s = 1, \qquad \forall c \in C \tag{A.2}$$

$$\sum_{i \in \mathcal{T}^2} \beta_{ic} x_i = \sum_{j \in S} \varphi_{jc} y_j, \qquad \forall c \in C \tag{A.3}$$

$$\sum_{p \in P} \epsilon_{ps} y_p = y_s, \qquad \forall j \in S \tag{A.4}$$

$$\sum_{i \in \mathcal{T}^1} \alpha_{is} r_i = \sum_{p \in P} \epsilon_{ps} y_p, \qquad \forall s \in S \tag{A.5}$$

$$\sum_{i \in \mathcal{T}^1} f_i \alpha_{is} - \sum_{c \in C} D_c \varphi_{sc} y_s = 0, \qquad \forall p \in P \tag{A.6}$$

$$\sum_{c \in C} D_c \varphi_{sc} y_s \leq Q_s y_s, \qquad \forall s \in S \tag{A.7}$$

$$\sum_{i \in \mathcal{T}_p^1} f_i \leq Q_p y_s, \qquad \forall p \in P \tag{A.8}$$

$$q^1 r_i - f_i \geq 0, \qquad \forall i \in \mathcal{T}^1 \tag{A.9}$$

$$r_i = \{0,1\}, \qquad \forall i \in \mathcal{T}^1 \tag{A.10}$$

$$x_i = \{0,1\}, \qquad \forall i \in \mathcal{T}^2 \tag{A.11}$$

$$y_p = \{0,1\}, \qquad \forall p \in P \tag{A.12}$$

$$y_s = \{0,1\}, \qquad \forall s \in S \tag{A.13}$$

$$f_i \geq = 0, \qquad \forall s \in \mathcal{T}^1 \tag{A.14}$$

Objective function (A.1) minimizes the total cost. For the second echelon, Constraints (A.2) guarantee that each customer is served by a single satellite while Constraints (A.3) ensure that a customer served by a satellite must be visited by only one vehicle originating from that satellite. Constraints (A.4) and (A.5) ensure the same routing conditions for the first echelon. Constraints (A.6) are flow balance constraints for satellites. Constraints (A.7) and (A.8) enforce capacity restrictions for satellites and platforms, respectively. Constraints (A.9) are consistency constraints between flow and routing variables. Finally, Constraints (A.10)-(A.14) are binary constraints and non-negativity restrictions.

A.2 2E-LRP2L solution cost breakdown

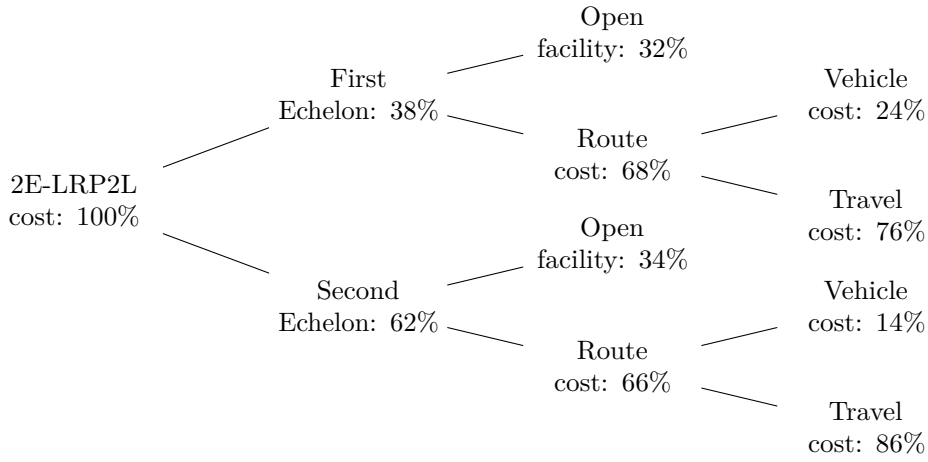


Figure A.1: The average cost breakdown of 2E-LRP2L solutions.

A.3 CMLRP formulation

The Multi-Depot Capacitated Location Routing Problem formulation proposed for the first echelon with multiple platforms comprises of four decision variables:

$x_{ij}^k = 1$ if vehicle k travels from node i to node j, 0 otherwise.

$y_j = 1$ if depot j is open.

$l_{ij}^k \in \mathbb{R}$ corresponds to the load being transported by vehicle k from node i to node j.

$v_k = 1$ if vehicle k is being used, 0 otherwise.

Assume we have a set S^o of open satellites and a demand D'_s for each satellite $s \in S^o$ which equals the sum of the demand of all customers served by s. The following mixed integer programming formulation corresponds to the first-echelon CLRP:

$$min : \sum_{j \in P} H_j y_j + \sum_{k \in K} h^1 v_k + \sum_{(i,j) \in E} \sum_{k \in K} e_{ij} x_{ij}^k. \tag{A.15}$$

$$\sum_{k \in K} \sum_{(j,i) \in E} x_{ji}^k = 1, \qquad \forall i \in S^o \tag{A.16}$$

$$\sum_{j \in P} \sum_{i \in S^o} x_{ji}^k \leq v_k, \qquad \forall k \in K \tag{A.17}$$

$$\sum_{k \in K} \sum_{j \in V} (x_{ij}^k - x_{ji}^k) = 0, \qquad \forall i \in V \tag{A.18}$$

$$\sum_{i \in S^o} x_{ji}^k \leq y_j, \qquad \forall j \in P, k \in K \tag{A.19}$$

$$\sum_{j \in V} l_{ij}^k = \sum_{j \in V} (l_{ji}^k - D'_i x_{ij}^k), \qquad \forall i \in S^o, k \in K \tag{A.20}$$

$$l_{ij}^k \leq q^1 x_{ij}^k, \qquad \forall (i,j) \in E, k \in K \tag{A.21}$$

$$\sum_{k \in K} \sum_{i \in S^o} l_{ji}^k \leq Q_j y_j, \qquad \forall j \in P \tag{A.22}$$

$$x_{ji}^k \in \{0,1\}, \qquad \forall i,j \in V, k \in K \tag{A.23}$$

$$y_j \in \{0,1\}, \qquad \forall j \in P \tag{A.24}$$

$$v_k \in \{0,1\}, \qquad \forall k \in K \tag{A.25}$$

$$l_{ji}^k \geq 0, \qquad \forall i,j \in V, k \in K \tag{A.26}$$

Objective function (A.15) minimizes the cost of open platforms, employed vehicles and travel time. Constraints (A.16) ensure each satellite is served once

and once only by a platform. Constraints (A.17) guarantee each vehicle is used at most once. Constraints (A.18) are flow conservation constraints which ensure that the number of edges entering and leaving a node is equal. Constraints (A.19) guarantee that if an edge is leaving a platform, this platform must be open. Constraints (A.20) concern load conservation and avoid the occurrence of sub-routes by specifying that the load entering node i must be equal to the load leaving node i minus the demand of node i. Finally, Constraints (A.21) and (A.22) are the vehicle and platform capacity constraints.

Bibliography

George B Dantzig and John H Ramser. The truck dispatching problem. *Management science*, 6(1):80–91, 1959.

Jan Karel Lenstra and AHG Rinnooy Kan. Complexity of vehicle routing and scheduling problems. *Networks*, 11(2):221–227, 1981.

Bruce L Golden, Subramanian Raghavan, and Edward A Wasil. *The vehicle routing problem: latest advances and new challenges*, volume 43. Springer Science & Business Media, 2008.

Gilbert Laporte. Fifty years of vehicle routing. *Transportation Science*, 43(4): 408–416, 2009.

Paolo Toth and Daniele Vigo. *Vehicle routing: problems, methods, and applications*. SIAM, 2014.

Rosario Cuda, Gianfranco Guastaroba, and Maria Grazia Speranza. A survey on two-echelon routing problems. *Computers & Operations Research*, 55: 185–199, 2015.

Jean-Françcois Cordeau, Gilbert Laporte, and Stefan Ropke. Recent models and algorithms for one-to-one pickup and delivery problems. In *The vehicle routing problem: latest advances and new challenges*, pages 327–357. Springer, 2008.

Sophie N Parragh, Karl F Doerner, and Richard F Hartl. A survey on pickup and delivery problems (part ii: Transportation between pickup and delivery locations). *Journal für Betriebswirtschaft*, 58(2):81–117, 2008.

Maria Battarra, Jean-François Cordeau, and Manuel Iori. Chapter 6: pickup-and-delivery problems for goods transportation. In *Vehicle Routing: Problems, Methods, and Applications, Second Edition*, pages 161–191. SIAM, 2014.

H Shiri, M Rahmani, and M Bafruei. Examining the impact of transfers in pickup and delivery systems. *Uncertain Supply Chain Management*, 8(1): 207–224, 2020.

Renaud Masson, Fabien Lehuédé, and Olivier Péton. Efficient feasibility testing for request insertion in the pickup and delivery problem with transfers. *Operations Research Letters*, 41(3):211–215, 2013a.

Michael Drexl. Synchronization in vehicle routing—a survey of VRPs with multiple synchronization constraints. *Transportation Science*, 46(3):297–316, 2012.

Snežana Mitrović-Minić and Gilbert Laporte. The pickup and delivery problem with time windows and transshipment. *INFOR: Information Systems and Operational Research*, 44(3):217–227, 2006.

Cristián E Cortés, Martín Matamala, and Claudio 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.

Rodrigue Tchapnga Takoudjou, Jean-Christophe Deschamps, and Rémy Dupas. A mip formulation for the pickup and delivery problem with time window and transshipment. *IFAC Proceedings Volumes*, 45(6):333–338, 2012.

Yuan Qu and Jonathan F Bard. A grasp with adaptive large neighborhood search for pickup and delivery problems with transshipment. *Computers & Operations Research*, 39(10):2439–2456, 2012.

Renaud Masson, Fabien Lehuédé, and Olivier Péton. An adaptive large neighborhood search for the pickup and delivery problem with transfers. *Transportation Science*, 47(3):344–355, 2013b.

Renaud Masson, Fabien Lehuédé, and Olivier Péton. The dial-a-ride problem with transfers. *Computers & Operations Research*, 41:12–23, 2014.

Abdur Rais, Filipe Alvelos, and Maria Sameiro Carvalho. New mixed integer-programming model for the pickup-and-delivery problem with transshipment. *European Journal of Operational Research*, 235(3):530–539, 2014.

Nicolas Danloup, Hamid Allaoui, and Gilles Goncalves. A comparison of two meta-heuristics for the pickup and delivery problem with transshipment. *Computers & Operations Research*, 100:155–171, 2018.

Yimeng Zhang, Bilge Atasoy, Dimitris Souravlias, and Rudy R Negenborn. Pickup and delivery problem with transshipment for inland waterway transport. In *International Conference on Computational Logistics*, pages 18–35. Springer, 2020.

David Wolfinger. A large neighborhood search for the pickup and delivery problem with time windows, split loads and transshipments. *Computers & Operations Research*, 126:105110, 2021.

Caroline Prodhon and Christian Prins. A survey of recent research on location-routing problems. *European Journal of Operational Research*, 238(1):1–17, 2014.

Michael Drexl. Applications of the vehicle routing problem with trailers and transshipments. *European Journal of Operational Research*, 227(2):275–283, 2013.

I-Ming Chao. A tabu search method for the truck and trailer routing problem. *Computers & Operations Research*, 29(1):33–51, 2002.

Stephan Scheuerer. A tabu search heuristic for the truck and trailer routing problem. *Computers & Operations Research*, 33(4):894–909, 2006.

Shih-Wei Lin, F Yu Vincent, and Shuo-Yan Chou. Solving the truck and trailer routing problem based on a simulated annealing heuristic. *Computers & Operations Research*, 36(5):1683–1692, 2009.

Massimiliano Caramia and Francesca Guerriero. A heuristic approach for the truck and trailer routing problem. *Journal of the Operational Research Society*, 61(7):1168–1180, 2010.

Shih-Wei Lin, F Yu Vincent, and Chung-Cheng Lu. A simulated annealing heuristic for the truck and trailer routing problem with time windows. *Expert Systems with Applications*, 38(12):15244–15252, 2011.

Ulrich Derigs, Markus Pullmann, and Ulrich Vogel. Truck and trailer routing—problems, heuristics and computational experience. *Computers & Operations Research*, 40(2):536–546, 2013.

Juan G Villegas, Christian Prins, Caroline Prodhon, Andrés L Medaglia, and Nubia Velasco. A matheuristic for the truck and trailer routing problem. *European Journal of Operational Research*, 230(2):231–244, 2013.

Sophie N Parragh and Jean-François Cordeau. Branch-and-price and adaptive large neighborhood search for the truck and trailer routing problem with time windows. *Computers & Operations Research*, 83:28–44, 2017.

Ann-Kathrin Rothenbächer, Michael Drexl, and Stefan Irnich. Branch-and-price-and-cut for the truck-and-trailer routing problem with time windows. *Transportation Science*, 52(5):1174–1190, 2018.

Túlio AM Toffolo, Jan Christiaens, Sam Van Malderen, Tony Wauters, and Greet Vanden Berghe. Stochastic local search with learning automaton for the swap-body vehicle routing problem. *Computers & Operations Research*, 89:68–81, 2018.

Michael Drexl. On the one-to-one pickup-and-delivery problem with time windows and trailers. *Central European Journal of Operations Research*, pages 1–48, 2020.

Temel Öncan, I Kuban Altınel, and Gilbert Laporte. A comparative analysis of several asymmetric traveling salesman problem formulations. *Computers & Operations Research*, 36(3):637–654, 2009.

Jan Christiaens, Hatice Çalik, Tony Wauters, Reshma Chirayil Chandrasekharan, and Greet Vanden Berghe. The prisoner transportation problem. *European Journal of Operational Research*, 284(3):1058–1073, 2020.

Helena R Lourenço, Olivier C Martin, and Thomas Stützle. Iterated local search. In *Handbook of Metaheuristics*, pages 320–353. Springer, 2003.

Helena Ramalhinho Lourenço, Olivier C Martin, and Thomas Stützle. Iterated local search: Framework and applications. In *Handbook of metaheuristics*, pages 129–168. Springer, 2019.

Edmund K Burke and Yuri Bykov. The late acceptance hill-climbing heuristic. *European Journal of Operational Research*, 258(1):70–78, 2017.

Martin WP Savelsbergh. The vehicle routing problem with time windows: Minimizing route duration. *ORSA journal on computing*, 4(2):146–154, 1992.

Jan Christiaens and Greet Vanden Berghe. Slack induction by string removals for vehicle routing problems. *Transportation Science*, 54(2):417–433, 2020.

Manuel López-Ibáñez, Jérémie Dubois-Lacoste, Leslie Pérez Cáceres, Mauro Birattari, and Thomas Stützle. The irace package: Iterated racing for automatic algorithm configuration. *Operations Research Perspectives*, 3: 43–58, 2016.

S Kruse Jacobsen and Oli BG Madsen. A comparative study of heuristics for a two-level routing-location problem. *European Journal of Operational Research*, 5(6):378–387, 1980.

Maurizio Boccia, Teodor G Crainic, Antonio Sforza, and Claudio Sterle. A metaheuristic for a two echelon location-routing problem. In *International Symposium on Experimental Algorithms*, pages 288–301. Springer, 2010.

Robert J Fowler, Michael S Paterson, and Steven L Tanimoto. Optimal packing and covering in the plane are np-complete. *Information Processing Letters*, 12(3):133–137, 1981.

Guido Perboli, Roberto Tadei, and Daniele Vigo. The two-echelon capacitated vehicle routing problem: Models and math-based heuristics. *Transportation Science*, 45(3):364–380, 2011.

Said Salhi and Graham K Rand. The effect of ignoring routes when locating depots. *European Journal of Operational Research*, 39(2):150–156, 1989.

Said Salhi and Gábor Nagy. Consistency and robustness in location-routing. *Studies in Locational Analysis*, (13):3–19, 1999.

Michael Drexl and Michael Schneider. A survey of variants and extensions of the location-routing problem. *European Journal of Operational Research*, 241 (2):283–308, 2015.

Gábor Nagy and Saïd Salhi. Location-routing: Issues, models and methods. *European Journal of Operational Research*, 177(2):649–672, 2007.

Michael Schneider and Michael Drexl. A survey of the standard location-routing problem. *Annals of Operations Research*, 259(1-2):389–414, 2017.

Suresh Nanda Kumar and Ramasamy Panneerselvam. A survey on the vehicle routing problem and its variants. *Intelligent Information Management*, 4(03): 66, 2012.

Thibaut Vidal, Gilbert Laporte, and Piotr Matl. A concise guide to existing and emerging vehicle routing problem variants. *European Journal of Operational Research*, 2019.

Andreas Bortfeldt and Gerhard Wäscher. Constraints in container loading–a state-of-the-art review. *European Journal of Operational Research*, 229(1): 1–20, 2013.

Manuel Iori and Silvano Martello. Routing problems with loading constraints. *Top*, 18(1):4–27, 2010.

Hanne Pollaris, Kris Braekers, An Caris, Gerrit K Janssens, and Sabine Limbourg. Vehicle routing problems with loading constraints: state-of-the-art and future directions. *OR Spectrum*, 37(2):297–330, 2015.

Viet-Phuong Nguyen, Christian Prins, and Caroline Prodhon. Solving the two-echelon location routing problem by a grasp reinforced by a learning process and path relinking. *European Journal of Operational Research*, 216 (1):113–126, 2012a.

Claudio Contardo, Vera Hemmelmayr, and Teodor Gabriel Crainic. Lower and upper bounds for the two-echelon capacitated location-routing problem. *Computers & operations research*, 39(12):3185–3199, 2012.

Martin Schwengerer, Sandro Pirkwieser, and Günther R Raidl. A variable neighborhood search approach for the two-echelon location-routing problem. In *European Conference on Evolutionary Computation in Combinatorial Optimization*, pages 13–24. Springer, 2012.

Viet-Phuong Nguyen, Christian Prins, and Caroline Prodhon. A multi-start iterated local search with tabu list and path relinking for the two-echelon location-routing problem. *Engineering Applications of Artificial Intelligence*, 25(1):56–71, 2012b.

Sandro Pirkwieser and Günther R Raidl. Variable neighborhood search coupled with ilp-based very large neighborhood searches for the (periodic) location-routing problem. In *International Workshop on Hybrid Metaheuristics*, pages 174–189. Springer, 2010.

Maurizio Boccia, Teodor Gabriel Crainic, Antonio Sforza, and Claudio Sterle. Location-routing models for designing a two-echelon freight distribution system. *Rapport technique, CIRRELT, Université de Montréal*, page 91, 2011.

Ulrich Breunig, Verena Schmid, Richard F Hartl, and Thibaut Vidal. A large neighbourhood based heuristic for two-echelon routing problems. *Computers & Operations Research*, 76:208–225, 2016.

Khosro Pichka, Amirsaman H Bajgiran, Matthew EH Petering, Jaejin Jang, and Xiaohang Yue. The two echelon open location routing problem: Mathematical model and hybrid heuristic. *Computers & Industrial Engineering*, 121:97–112, 2018.

Michel Gendreau, Manuel Iori, Gilbert Laporte, and Silvano Martello. A tabu search algorithm for a routing and container loading problem. *Transportation Science*, 40(3):342–350, 2006.

Manuel Iori, Juan-José Salazar-González, and Daniele Vigo. An exact approach for the vehicle routing problem with two-dimensional loading constraints. *Transportation Science*, 41(2):253–264, 2007.

Lijun Wei, Zhenzhen Zhang, Defu Zhang, and Stephen CH Leung. A simulated annealing algorithm for the capacitated vehicle routing problem with two-dimensional loading constraints. *European Journal of Operational Research*, 265(3):843–859, 2018.

Michel Gendreau, Manuel Iori, Gilbert Laporte, and Silvaro Martello. A tabu search heuristic for the vehicle routing problem with two-dimensional loading constraints. *Networks: An International Journal*, 51(1):4–18, 2008.

Andrea Lodi, Silvano Martello, and Daniele Vigo. Heuristic and metaheuristic approaches for a class of two-dimensional bin packing problems. *INFORMS Journal on Computing*, 11(4):345–357, 1999.

Emmanouil E Zachariadis, Christos D Tarantilis, and Christos T Kiranoudis. A guided tabu search for the vehicle routing problem with two-dimensional loading constraints. *European Journal of Operational Research*, 195(3):729–743, 2009.

Bernard Chazelle. The bottomn-left bin-packing heuristic: An efficient implementation. *IEEE Transactions on Computers*, C-32(8):697–707, 1983.

Guenther Fuellerer, Karl F Doerner, Richard F Hartl, and Manuel Iori. Ant colony optimization for the two-dimensional loading vehicle routing problem. *Computers & Operations Research*, 36(3):655–673, 2009.

Silvano Martello and Daniele Vigo. Exact solution of the two-dimensional finite bin packing problem. *Management Science*, 44(3):388–399, 1998.

Edmund K Burke, Graham Kendall, and Glenn Whitwell. A new placement heuristic for the orthogonal stock-cutting problem. *Operations Research*, 52 (4):655–671, 2004.

Stephen CH Leung, Jiemin Zheng, Defu Zhang, and Xiyue Zhou. Simulated annealing for the vehicle routing problem with two-dimensional loading constraints. *Flexible Services and Manufacturing Journal*, 22(1-2):61–82, 2010.

Stephen CH Leung, Xiyue Zhou, Defu Zhang, and Jiemin Zheng. Extended guided tabu search and a new packing algorithm for the two-dimensional loading vehicle routing problem. *Computers & Operations Research*, 38(1): 205–215, 2011.

Emmanouil E Zachariadis, Christos D Tarantilis, and Chris T Kiranoudis. Integrated distribution and loading planning via a compact metaheuristic algorithm. *European Journal of Operational Research*, 228(1):56–71, 2013.

Lijun Wei, Zhenzhen Zhang, Defu Zhang, and Andrew Lim. A variable neighborhood search for the capacitated vehicle routing problem with two-dimensional loading constraints. *European Journal of Operational Research*, 243(3):798–814, 2015.

Lijun Wei, Wee-Chong Oon, Wenbin Zhu, and Andrew Lim. A skyline heuristic for the 2d rectangular packing and strip packing problems. *European Journal of Operational Research*, 215(2):337–346, 2011.

Jean-François Côté, Michel Gendreau, and Jean-Yves Potvin. An exact algorithm for the two-dimensional orthogonal packing problem with unloading constraints. *Operations Research*, 62(5):1126–1141, 2014.

Geoff Clarke and John W Wright. Scheduling of vehicles from a central depot to a number of delivery points. *Operations Research*, 12(4):568–581, 1964.

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

Jean-Yves Potvin and Jean-Marc Rousseau. An exchange heuristic for routeing problems with time windows. *Journal of the Operational Research Society*, 46(12):1433–1446, 1995.

Shinji Imahori and Mutsunori Yagiura. The best-fit heuristic for the rectangular strip packing problem: An efficient implementation and the worst-case approximation ratio. *Computers & Operations Research*, 37(2):325–333, 2010.

RV Kulkarni and Pramod R Bhave. Integer programming formulations of vehicle routing problems. *European Journal of Operational Research*, 20(1):58–67, 1985.

Passmark. *CPU benchmarks*, accessed May 11, 2020. `http://www.cpubenchmark.net/cpu_list.php`.

List of publications during PhD

Articles in internationally reviewed academic journals

Under review

Vinicius S. M. Gandra, Hatice Çalik, Tony Wauters, Túlio A. M. Toffolo, Marco Antonio M. Carvalho and Greet Vanden Berghe. The impact of loading restrictions on the two-echelon location routing problem. *Computers & Industrial Engineering*.

Vinicius S. M. Gandra, Hatice Çalik, Túlio A. M. Toffolo, Marco Antonio M. Carvalho and Greet Vanden Berghe. The vessel swap-body routing problem. *European Journal of Operational Research*.

Papers at international conferences, published in full in proceedings

Vinicius Gandra, Hatice Çalik and Greet Vanden Berghe. A heuristic approach to feasibility verification for truck loading. *In 2021 Genetic and Evolutionary Computation Conference Companion (GECCO '21 Companion), July 10–14, 2021, Lille, France.* ACM, New York, NY, USA, 6 pages. https://doi.org/10.1145/3449726.3463184.

Under review

Carlo S. Sartori, Vinicius Gandra, Hatice Çalik and Pieter Smet. Selective production scheduling with stock- and staff-related restrictions. *In 2021 International Conference on Computational Logistics (ICCL2021)*.

Accepted abstracts at conferences and scientific meetings

Vinicius S. M. Gandra, Hatice Çalik, Túlio A. M. Toffolo, Marco Antonio M. Carvalho and Greet Vanden Berghe. A metaheuristic approach for the two-echelon location routing problem, *ORBEL 34*, 30-31 Jan, 2020.

FACULTY OF ENGINEERING TECHNOLOGY
DEPARTMENT OF COMPUTER SCIENCE
COMBINATORIAL OPTIMISATION DECISION SUPPORT (CODES)
Celestijnenlaan 200A box 2402
B-3001 Leuven
vinicius.gandramartinssantos@kuleuven.be