KU LEUVEN

ARENBERG DOCTORAL SCHOOL
Faculty of Engineering Technology

Optimization algorithms for vehicle routing problems with multiple decision levels

Vinícius Gandra Martins Santos

Supervisors:
Prof. dr. Greet Vanden Berghe
Prof. dr. Marco A. M. de Carvalho
Dr. Hatice Çalık
Prof. dr. Túlio A. M. Toffolo

Dissertation presented in partial fulfillment of the requirements for the degree of Doctor of Engineering Technology (PhD) by KU Leuven and the degree of Doctor of Computer Science by UFOP

July 2023

Optimization algorithms for vehicle routing problems with multiple decision levels

Vinícius GANDRA MARTINS SANTOS

Dissertation presented in partial fulfillment of the requirements for the degree of Doctor of Engineering Technology (PhD) by KU Leuven and the degree of Doctor of Computer Science by UFOP

July 2023

Acknowledgments

This thesis would not have been possible without the invaluable advice, guidance and support of the many people who played a crucial role in this achievement. I am deeply grateful to each and every one of them.

First and foremost, I would like to extend my heartfelt thanks to my supervisor Greet Vanden Berghe. You not only provided me with the unique opportunity to pursue research at KU Leuven, but also offered constant support, guidance and encouragement. The knowledge I gained under your supervision has helped shape my present and future, and I am truly grateful.

I would also like to express my gratitude to Marco Carvalho, who not only supervised and tutored me since the second year of my undergraduate, but also became a dear friend. You are a true inspiration to me as both a professor and researcher, and I have learned so much from you.

I am also thankful to my PhD co-supervisors: Hatice and Túlio. Hatice, thank you for your support, encouragement, friendship, and for always being available to brainstorm with me. Túlio, I am grateful to you for providing me with the guidance to embark on this PhD journey and for your continued support.

Throughout my PhD, I had the privilege of working on several projects and collaborating with numerous colleagues from academia and industry. I am thankful to all of my colleagues for the experiences we shared. I would like to give a special shout-out to my colleagues at CODeS: Tony, Carlo, Pieter, Jonas, Jeroen, Luke, Manos, Michiel, Bryan, and Kelvin. Thank you all for creating a positive and enjoyable work atmosphere, for the engaging lunch conversations, and for the camaraderie we shared. I feel truly lucky to have met you all and it would not have been the same without you.

I would like to extend a special thanks to Luke and Carlo, who played a significant role in my PhD journey. Luke, thank you for your revisions and editorial advice concerning both my work at CODeS and my personal endeavors.

Your support and insights have gone above and beyond what I could have asked for. To my dear friend Carlo: you were always there to help and provide brilliant ideas, a compatriot who gave me some feeling of home, and a friend with whom I shared memorable moments in Ghent. I cannot express enough gratitude to you and I will miss our time together.

A warm thank you also goes out to my friends in Belgium - Tainara, Nieves, Max, Aya, Abdel, Dennis, and others - who made this foreign country feel a little more like home. I would also like to thank my beloved mother Claudilene, as well as my family and friends in Brazil. You all provided me with support and comfort even from across the ocean. Thank you also to my extended family in the United States, who always believed in me, provided essential support and celebrated each little step.

Finally, I would like to express my deepest gratitude to Rachel, my partner and safe haven. You were there with me every step of the way with unwavering support. Your strength and love have been instrumental in my success, and I cannot thank you enough.

Abstract

In their day-to-day operations, logistics companies typically need to address multiple decision levels in addition to a vast number of realistic problem features. These decisions are not only related to transportation, but can also involve determining the location of facilities, arranging items for shipment or scheduling personnel. Addressing all of these decision levels is crucial given that they have direct impact on resource usage, costs and customer satisfaction. Meanwhile, some examples of realistic problem features include time windows, multiple depot options and working regulations.

Developing solution approaches for these extremely complex problems is often time-consuming and can easily lead to a complexity explosion. In an attempt to help orient researchers and practitioners who face this challenging problem context, this thesis introduces a general methodology for addressing problems that feature multiple interconnected decision levels with an underlying vehicle routing problem. This methodology should provide those confronted with such problems a general set of steps they can take to simplify and accelerate the problem-solving process.

In order to arrive at such a general methodology, we investigated challenges faced by a diverse set of logistics companies and developed approaches to solve each problem. We investigated how the different decision levels interact with one another, how we can take advantage of such interactions, what methods can reduce and effectively explore the search space of the problem, and how we can minimize the risk of a complexity explosion in order to quickly produce high-quality solutions. Ultimately, rather than beginning by focusing on what sets these multi-level problems apart, we hope this thesis will encourage other researchers to first exploit what they share in common.

Beknopte samenvatting

Logistieke bedrijven nemen dagelijks diverse beslissingen die zich niet beperken tot transportaangelegenheden. De beslissingen behelzen ook het kiezen van geschikte locaties voor faciliteiten, het plaatsen van producten op transportpalletten en het inplannen van taken voor de medewerkers. Al deze beslissingen dienen aangepakt omdat ze een directe invloed hebben op het gebruik van middelen, op transportkosten en op klantentevredenheid. Logistieke bedrijven houden daarbij rekening met operationele beperkingen zoals bijvoorbeeld tijdsvensters, depots op verschillende locaties en collectieve arbeidsovereenkomsten.

De ontwikkeling van oplossingsmethoden voor deze extreem complexe problemen vergt veel ontwikkeltijd. Eens ze klaar zijn gebruiken ze vaak ook onaanvaardbaar veel rekentijd. Deze thesis introduceert daarom een algemene methodologie om onderzoekers en praktijkmensen te ondersteunen bij het ontwikkelen van adequate beslissingsondersteuning. Het onderzoek richt zich hoofdzakelijk op problemen die uit verschillende, afhankelijke deelproblemen bestaan, waaronder een rittenplanningscomponent. De methodologie beoogt een stappenplan aan te bieden om het globale optimalisatieprobleem te vereenvoudigen en het oplossingsproces te versnellen.

Om tot deze algemene methodologie te komen onderzochten we de uitdagingen van een diverse groep van logistieke bedrijven en ontwikkelden algoritmen voor elk probleem. We onderzochten hoe de verschillende beslissingsniveaus met elkaar interageren, hoe we deze interacties kunnen benutten, welke methoden de zoekruimte kunnen verkleinen en exploreren, en hoe de complexiteit te beheersen om tot kwalitatieve oplossingen te komen. In plaats van te focussen op wat deze problemen met verschillende beslissingsniveaus onderscheidt, hopen we dat deze thesis andere onderzoekers zal stimuleren om in de eerste plaats de gelijkenissen te exploiteren.

Resumo

Em suas operações diárias, empresas de logística geralmente precisam resolver problemas com múltiplos níveis de decisão, além de uma vasta quantidade de aspectos práticos. Essas decisões não estão apenas relacionadas ao transporte, mas também podem envolver a determinação da localização de instalações, empacotamento de itens para envio ou escalonamento de pessoal. Incorporar todos esses níveis de decisão é crucial, dado que eles têm impacto direto no uso de recursos, custos e satisfação do cliente. Enquanto isso, alguns exemplos de aspectos práticos incluem janelas de tempo, várias opções de depósito e regulamentações de trabalho.

Desenvolver métodos de solução para esses problemas extremamente complexos é frequentemente demorado e pode facilmente levar a uma explosão de complexidade. Em uma tentativa de ajudar a orientar pesquisadores e profissionais da indústria que enfrentam esse contexto de problema desafiador, nesta tese desenvolveremos uma metodologia geral para abordar problemas que apresentam múltiplos níveis de decisão interconectados com um problema de roteamento de veículos. Essa metodologia deve fornecer àqueles confrontados com tais problemas um conjunto geral de etapas que podem ser tomadas para simplificar e acelerar o processo de desenvolvimento de métodos de solução.

Para chegar a essa metodologia geral, investigaremos os desafios enfrentados por um conjunto diversificado de empresas de logística e desenvolveremos métodos para resolver cada problema. Investigaremos como os diferentes níveis de decisão interagem entre si, como podemos aproveitar essas interações, que métodos podem reduzir e explorar efetivamente o espaço de busca do problema e como podemos minimizar o risco de uma explosão de complexidade para produzir rapidamente soluções de alta qualidade. Em última análise, em vez de começar focando no que diferencia esses problemas de múltiplos níveis, esperamos que esta tese incentive outros pesquisadores a primeiro explorar o que eles têm em comum.

List of Abbreviations

2E-LRP	Two-Echelon Location Routing Problem
2E-LRP2L	Two-Echelon Location-Routing Problem with Two-Dimensional Loading Constraints
2L-VRP	Vehicle Routing Problem with Two-Dimensional Loading Constraints
FVMP	Feasibility Verification and Machine Packing Problem
ILS	Iterated Local Search
LAHC	Late Acceptance Hill-Climbing
LPM	Load Plan Method
LRP	Location Routing Problem
MDPVRPTW	Multi-Depot Periodic Vehicle Routing Problem with Time Windows
MDPVRPVP	Multi-Depot Periodic Vehicle Routing Problem with time windows and Variable-Pattern customers
MTRSP	Multi-period Team Routing and Scheduling Problem
PDP	Pickup and Delivery Problem
PDPT	Pickup and Delivery Problem with Transshipment
PDPTW	Pickup and Delivery Problem with Time Windows
PDPTWT	Pickup and Delivery Problem with Time Windows and Transshipment
PPM	Predicted Percentage Method
PVRP	Periodic Vehicle Routing Problem
SLP	Sequential Loading Penalty
TTRP	Truck-and-Trailer Routing Problem
VRP	Vehicle Routing Problem
VRPTW	Vehicle Routing Problem with Time Windows
VSBR	Vessel Swap-Body Routing Problem

Contents

List of Figures

List of Tables

Chapter 1

Introduction

One consequence of the increasingly globalized world in which we live today is the interdependence of the world's economies. This has in turn resulted in the widespread international movement of goods and services. Transportation plays a crucial role in this intensification of cross-border movement, giving rise to logistics companies of many sizes and specialties. Each of these companies must solve complex problems in order to provide efficient transportation solutions, with their strategic and operational decisions having a direct impact on resource usage, costs, safety and customer satisfaction.

Most logistics companies will either directly or indirectly need to address an underlying *Vehicle Routing Problem* (VRP). The VRP in its most basic form involves a fleet of vehicles and a set of geographically dispersed customers, each of which requires a certain request of goods to be delivered (Dantzig and Ramser, 1959). Routes should start and end at the same depot while also respecting certain feasibility constraints, such as capacity. The goal is to minimize the total cost of servicing all customers. That cost may correspond to total travel distance, total travel duration, the number of vehicles used or a weighted combination of these costs.

There are numerous logistics applications in which underlying VRP plays an important role. Some examples include postal services, healthcare, transportation of goods, waste collection and shared mobility. In many of these use cases the underlying VRP is intertwined with other challenging decisions leading to problems with multiple decision levels. These additional decisions can include determining transfer locations, packing items into trucks, scheduling customer visits, or assigning qualified personnel to routes.

Given the many multilevel problems that involve an underlying VRP, our aim in this thesis is to develop a *general methodology* capable of offering direction to researchers and practitioners who need to combine an underlying VRP with multiple decision levels into an integrated problem. Rather than offering a singular algorithmic implementation, this flexible methodology should instead provide some initial questions and guidelines to consider before commencing the development of a tailored solution method. The ultimate aim of such a methodology is to improve efficiency and help developers arrive at an initial solution method more quickly. Moreover, it should result in a more standardized approach to such problems, ultimately leading to more standardized methods.

In order to arrive at such general methodology we will investigate challenges faced by a diverse set of logistics companies whose operations can be categorized as follows: (i) lengthy international journeys where goods must be transferred between vehicles, (ii) transporting large machines with challenging loading restrictions, (iii) periodically transporting small valuable goods and (iv) scheduling tasks where qualified personnel must visit customers in teams.

Figure 1.1 showcases how additional decision levels can interact with an underlying VRP. The set of used and unused depots in Figure 1.1 represent location decisions, which arise when one needs to select a subset of depots to be operational. These location decisions will ultimately impact the quality of routes. Such decisions might be long term in the case where a company is trying to decide where to build a new depot or short term when companies only need to rent depots for a few days or weeks. Other decision levels are illustrated in Figure 1.1 by the different sets of routes in (a), (b) and (c).

The set of routes depicted in (a) illustrates the transfer of goods between vehicles. This problem context might involve detaching a trailer loaded with many customer requests at a pre-determined transfer point, which another vehicle then attaches before continuing on with its route to conduct the deliveries. Figure 1.1(a) illustrates two different vehicles (A and B) that transfer a trailer at a transfer point TP. Planning transfers such as these requires making decisions regarding the transfer location, the specific trailer and vehicles involved, as well as ensuring the routes are properly synchronized.

The route depicted in (b) illustrates a VRP with packing, where a route's feasibility also depends on loading items inside the vehicle in accordance with certain multi-dimensional constraints. Common applications might involve packing large items which cannot overlap, loading items in such a way that those delivered first are located closer to the vehicle's rear, or constraints concerning a maximum height when stacking items.

For the routes illustrated in (c), each customer requires a service to be performed

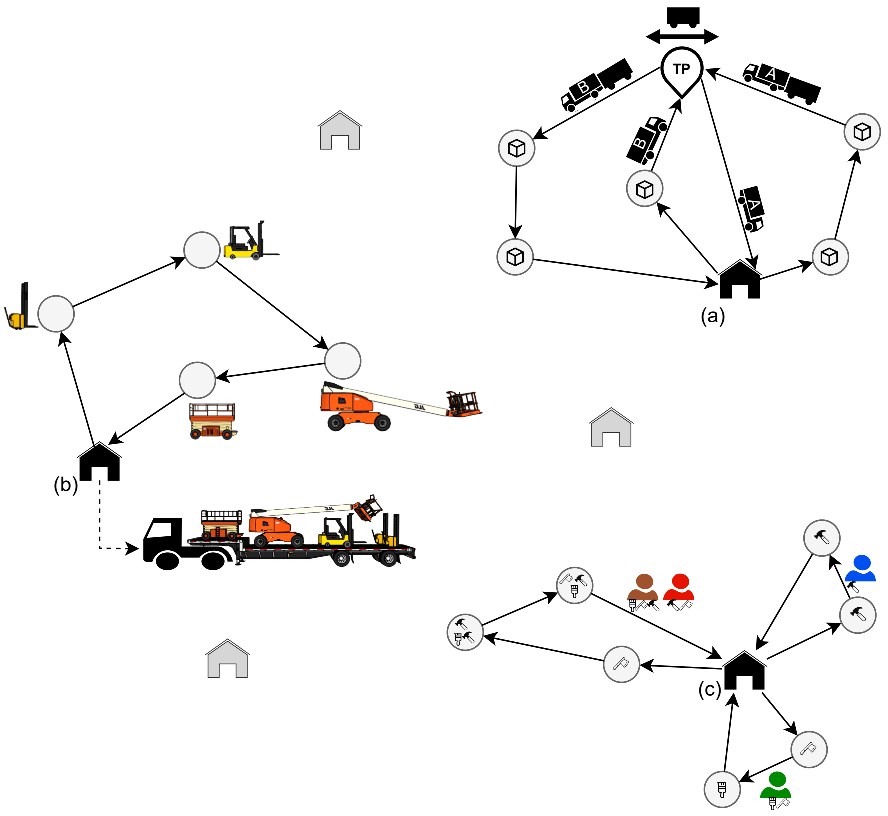


Figure 1.1: Vehicle routing problems with different additional decision levels: (a) transfers, (b) packing and (c) personnel scheduling.

rather than goods to be delivered. These services are associated with a set of skills and are performed by qualified technicians. Thus, the feasibility of a route depends on the team of technicians assigned to it and whether they have all of the necessary skills to serve customer requests. Moreover, there may be a minimum number of technicians to perform certain services.

Problems such as those visualized in Figure 1.1 are often solved just for a single day of operations. However, another decision level emerges when considering these problems across a time horizon spanning multiple days. In such cases a crucial aspect of the problem involves determining the best day to visit each customer in order to optimize routes over the entire time horizon.

When logistics companies are confronted with such multilevel problems they often respond to the complexity of the situation by decomposing their problem and solving it in distinct parts. For example, one level may be solved manually with simple rules of thumb. This partial solution might then be fed as input to an algorithm which solves the other level, ultimately returning a complete solution. This methodology fails to fully quantify the positive or negative impact that one level exerts on another since each is solved in isolation. However, integrating all decision levels into a single problem can prove very challenging, leading to complex methods with slow performance. This is not practical for companies under intense pressure to produce solutions on a daily basis and often update those solutions in response to unexpected disruptions.

A suitable trade-off between high quality solutions and reasonable computation time is typically desired for the dynamic context in which real-world logistics problems necessarily exist. Therefore, in this thesis we will investigate how different decision levels interact with one another and what type of tailored solution approaches can be implemented to take advantage of such integrations. Additionally, we will also investigate methods that can help reduce and effectively explore the search space of the problem, thereby minimizing the risks of a complexity explosion. This could easily occur if one simply combines the multiple decision levels without constraining the search space and without knowing the large impact small changes can have on solutions.

By keeping the following questions in mind when addressing each of the problems in this thesis, we hope to gradually arrive at a general methodology for approaching problems involving multiple decision levels with an underlying VRP. What is the dependency between different levels of the problem, how do they interact and how does one level react to changes in the other? Is the impact significant and requires starting the optimization from scratch, or is it more limited and can elements of the existing solution be reused? Can potential solutions for one level be eliminated by calculating and exploiting bounds? Is it possible to preprocess all possible solutions for one of the decision levels? Do fast evaluations, like those used to calculate the impact of solution modifications (delta evaluation), remain valid for the integrated problem? Are there any optimization techniques that are capable of changing a solution at multiple levels without having to completely re-optimize any of them?

The following chapters of this thesis will explore four independent problems which integrate vehicle routing with other decision levels. In order to ground our research in real operational challenges rather than academic assumptions, these problems correspond to those faced by shipping, machine rental, security and software companies currently operating in Belgium. While catering to the unique logistics problem faced by each company, we will also investigate the potential of using bounds, decomposition, and preprocessing methods to reduce the search space and compute better quality solutions within a reasonable amount of time.

Experiments will help us understand the impact of solution quality, processing time and resource usage when integrating additional constraints and including characteristics which bring our problem definitions closer to reality.

Chapter 2

The vessel swap-body routing problem

In this chapter we investigate a VRP integrated with transfer decisions. This transportation problem considers freight distribution between ports located throughout an inland waterway network, where batches of containers can be transferred between vessels at designated transfer locations. Provided that operational costs must be minimized, determining where to locate these transfer points and which requests should be transferred are exceptionally difficult decisions for human operators and automated methods alike.

Transfers represent a decision level which one ought to consider given the potential to improve solution quality. However, for the considered transportation problem transfers are not essential in terms of producing feasible solutions. This provides an opportunity to determine whether the added complexity of transfers is worth the returns by comparing solution approaches with and without transfers. Therefore, this chapter represents a natural starting point for the thesis to begin exploring our ideas concerning decision level integration. More specifically, we will investigate whether it is possible to limit the impact on already computed routes when adding/removing transfers, and whether a given solution has information which can help identify promising transfer locations.

This chapter is a minimally altered version of the peer-reviewed publication Gandra et al. (2022b)[1]. For this publication, Hatice Çalık contributed to the development of the MIP formulation, Túlio Toffolo and Marco Carvalho gave

[1]V. S. Gandra et al. (2022b). The vessel swap-body routing problem. In: *European Journal of Operational Research* 303.1, pp. 354–369.

suggestions concerning the design of the heuristic, while Greet Vanden Berghe supervised the research.

2.1 Introduction

This chapter investigates a routing problem introduced by a shipping company operating in an inland waterway network. Given a scheduling horizon, the company must complete hundreds of *requests*. Each of these requests is associated with a number of containers which must be picked up from one customer location and delivered to another while respecting time windows. Customer locations correspond to container terminals where they retrieve the goods being shipped, thus different customers may be associated with the same location. Containers associated with requests are loaded into *bodies*: capacitated transports which can only move when towed by a *vessel*. Vessels themselves are incapable of holding containers, but instead have a capacity limit concerning the number of bodies they can tow. All containers associated with a single request must be loaded into the same body and once loaded they cannot be transferred into another. Figure 2.1 illustrates a vessel-body configuration where two bodies are attached to a vessel.

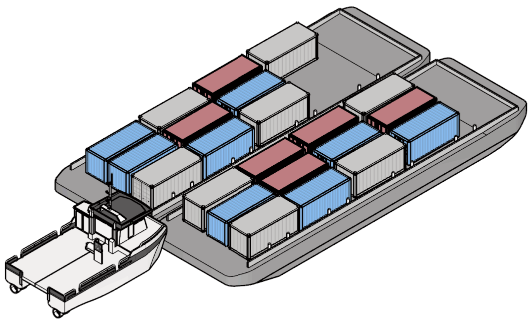


Figure 2.1: Example of a vessel towing two bodies.

Bodies are not fixed to vessels and the transportation network contains locations where these bodies can be parked. This enables the possibility of conducting *body transfers*, where a body is *detached* from one vessel at a special transfer point or customer location and then later *attached* to either the same vessel or a different one. 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. Body transfers significantly increase the solution space. They introduce a new decision level to the problem (where and when to perform these transfers) and require route synchronization: a vessel must first detach a body before another can visit the transfer point or customer location to attach it.

Human operators are unable to exploit the full potential of body transfers given the added logistical challenges they bring to the problem. Thus they are often ignored in practice or used in a very limited way. We believe that despite the extra computational effort needed, employing transfers has the potential to significantly reduce overall costs. Therefore, in this chapter we introduce a formulation for this transportation problem and propose a solution approach which enables us to study the impact of body transfers on solution quality. We refer to this new transportation problem as the *Vessel Swap-Body Routing problem* (VSBR).

The VSBR is a generalization of the Pickup and Delivery Problem (PDP) with time windows. Important insights concerning how to model and solve the VSBR can therefore be drawn from other PDP generalizations. One such generalization is the Pickup and Delivery Problem with Transshipment (PDPT), which also involves the transfer of goods between routes and complex route synchronization. However, unlike how it is possible to transfer individual requests from one vehicle to another in the PDPT, the entire set of requests held by a body must be transferred from one vessel to another in the VSBR. Another important generalization worth considering is the Truck-and-Trailer Routing Problem (TTRP) where detachable trailers are towed by trucks and each trailer is assigned to a specific truck. By contrast, bodies in the VSBR are treated independently from vessels and therefore a given body can be towed by any vessel. Thus, all vessel-body combinations are permitted as long as vessel capacities (which often exceed two bodies) are respected. Moreover, it is also worth noting that in the VSBR all customers can be visited by all possible vessel-body combinations. Body transfers are optional and only employed in an attempt to reduce overall costs. Conversely, detaching trailers in the TTRP is performed to ensure feasibility: certain customers can only be reached by trucks without any trailer attached. Although related to the VSBR and worth reviewing, these other PDP generalizations are unable to address the important characteristics of the inland waterway distribution scenario we investigate in this chapter.

In order to clarify the remaining details of the VSBR, Figure 2.2 illustrates a solution with body transfers in a toy network of 12 customer locations (the triangles) and one transfer point (the blue diamond). 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 three vessels responsible for towing bodies have their routes illustrated with distinct dashed edges and their initial positions are depicted by way of gray squares. 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. When en route, vessels are assumed to travel at the same constant speed, regardless of the number of bodies connected and containers loaded. Loading/unloading containers into/from bodies takes a certain length of time which is assumed to be the same for each container. Similarly, detaching/attaching a body also takes a fixed length of time independent of the configuration of attached bodies. For simplicity, time windows as well as travel times are omitted from Figure 2.2 and each customer location corresponds to a single service.

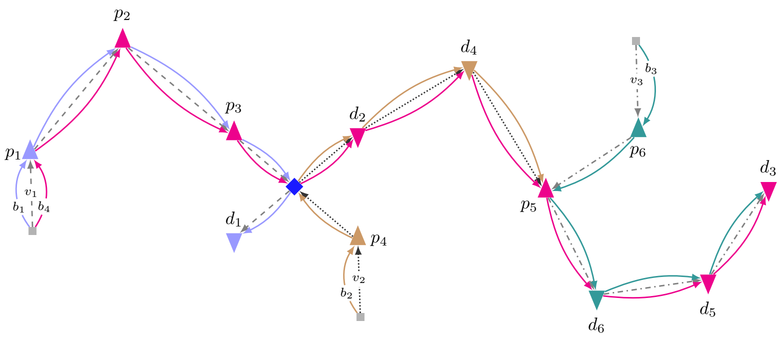


Figure 2.2: VSBR solution with body transfers.

Bodies b_1 and b_4 begin their routes attached to vessel v_1. Meanwhile, bodies b_2 and b_3 begin attached to vessels v_2 and v_3, respectively. Note that body b_4 serves the pickup and delivery services of requests 2, 3 and 5. To serve all of these requests, body b_4 is transferred between all three 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 for the second time at customer p_5 and serves the request while detached. Note that 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 to which the body is attached arrives at the customer's location either before or during its time window. The vessel then detaches the necessary body and is 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 serves the customer as soon as 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. Finally, b_4 is attached one final time and towed by v_3 in order to deliver its 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.

Despite the fact that all requests were served in this small example, this might not always be the case. The homogeneous fleet of vessels and bodies available to serve requests is considered fixed and given a priori. Therefore, given the capacity constraints of the available fleet it may prove impossible to serve every request within its time window, in which case some requests must be outsourced and 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 therefore designed to minimize the sum of (i) vessel travel costs and (ii) outsourcing costs.

Figure 2.2 not only makes it clear why human operators currently avoid dealing with the complexity of body transfers, but also highlights the potential for those transfers to improve solutions. Transferring requests in batches is not necessarily an exclusive feature of the VSBR and could occur in the context of waterway, rail or road transportation. The challenging new decision level the VSBR involves, in addition to the range of transportation systems within which it may occur, makes us confident that it is worthwhile introducing this new problem to the academic community. The following section presents an overview of related research, comparing and contrasting them with the VSBR. Section 2.3 introduces the necessary notation and presents a mixed integer programming formulation for the problem. To tackle instances of realistic size, a solution approach is introduced and described in Section 2.4. 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. Finally, 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 vehicle routing problem. Nevertheless, other routing problems can be identified in which specific requests must be served 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 conducted by Cordeau et al. (2008), Parragh et al. (2008) and Battarra et al. (2014). A generalization of the PDP arises when requests 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 a request is transferred at a transfer point, the vehicle which picks up this request 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. (2013b) proposed a method which checks the feasibility of solutions in constant time. This method can be used in the VSBR to efficiently avoid cross synchronization between body transfers. 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.

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

Table 2.1: Overview of PDPT methods 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. (2013a)	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.

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.

A variant of the TTRP is the Swap-Body Vehicle Routing Problem (SB-VRP), which was introduced in the first Vehicle Routing and Logistics Optimization (VeRoLog[2]) solver challenge in 2014. Similar to the TTRP, the SB-VRP also considers vehicles with either one or two swap-bodies in addition to customers with accessibility constraints. A truck towing two swap-bodies is called a train in the SB-VRP. Besides parking and picking up the swap-body at a transfer point, a train may detach the front swap-body instead of the one at the back. A third class of customers is also considered which has a higher demand than the capacity of a single swap-body and must therefore be visited by a train. The objective function of the SB-VRP also incorporates more terms such as fixed usage cost for trucks and trailers as well as additional costs for conducting operations at transfer points. However, the SB-VRP permits neither swap operations at TCs nor transferring loads between swap-bodies. Given the similarities between the TTRP and SB-VRP, we believe the swap-body terminology was coined given its vehicle-independent phrasing which may denote trailers, railroad cars or barges.

Table 2.2: TTRP methods and transfer details.

Reference	Problem	Method	Transfer location	Edge cost	Fleet size	Fixed vehicle-body combination
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	✓
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	✓
Toffolo et al. (2018)	SB-VRP	H	TP	VD	U	✓
Drexl (2020)	PD-TTRPTW	H	TP	VD	U	✓
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.

Recent research regarding the TTRP is documented in Table 2.2, focusing on transshipment characteristics. A few authors 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

[2]EURO Working Group on Vehicle Routing and Logistics Optimization (`http://www.verolog.eu/`)

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 and/or heterogeneous fleet of vehicles and forbid load transfer between lorries and their corresponding trailers. Given that Toffolo et al. (2018)'s algorithm won first place in the VeRoLog competition, their paper is included in Table 2.2 as representative of the SB-VRP.

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. Consider a hypothetical generalized problem HYP where V is the set of all vehicles (vessels or trucks), B is the set of bodies/trailers, N is the set of nodes, $V_b \subset V$ is the subset of vehicles that can tow a body/trailer b, and $N_b \subset N$ is the subset of nodes that b can access. The TTRP is then a special case of HYP where an arbitrary trailer b can only be towed by a specific vehicle v_b and therefore $V_b = \{v_b\}$. By contrast, the VSBR would correspond to a second special case where $V_b = V$ and $N_b = N$. One can further conclude that neither of these two special cases can be reduced to the other.

Another crucial difference between the two problems concerns how vessels in the VSBR may tow multiple bodies at the same time. Moreover, (un)loading operations can be performed without the presence of the vessel when a body is detached at a customer location. 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 number of containers ρ_r that must be picked up from one customer and delivered to another by a single body in a single trip. 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 any vessel within its time window, it is outsourced and assumed to be served by trucks at a high cost π. Outsourced requests are therefore not routed in this problem formulation. 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. Therefore, at the end of a scheduling horizon bodies may be left unattached to a vessel. As a result, bodies may also start a scheduling horizon unattached.

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.

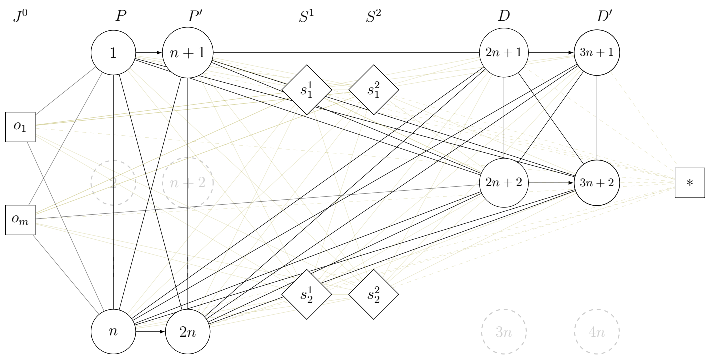


Figure 2.3: A modified network representation for the model

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:

$A^1 = \{(i,j) : i \in J^0 \cup S, j \in N, i \neq j\}$ is the set of arcs from vessel/body starting nodes and swap nodes to any other node,

$A^2 = \{(i,i+n) : i \in P\} \cup \{(i,j) : i \in P, j \in N \setminus \{u \in I : r_u = r_i\}\}$ is the set of arcs from each node $i \in P$ to every other node except the delivery nodes associated with r_i,

$A^3 = \{(i,i+n) : i \in P'\} \cup \{(i,j) : i \in P', j \in N \setminus \{u \in I : r_u = r_i\}\}$ is the set of arcs from each node $i \in P'$ to its corresponding delivery node in D and to every other node except those associated with r_i,

$A^4 = \{(i, i+n) : i \in D\} \cup \{(i,j) : i \in D, j \in N \setminus \{u \in I : r_u = r_i\}\}$ is the set of arcs from each node $i \in D$ to its copy in D' and to every other node except the pickup nodes associated with r_i,

$A^5 = \{(i,j) : i \in D', j \in N \setminus \{u \in I : r_u = r_i\}\}$ is the set of arcs from each node $i \in D'$ to every other node except those associated with r_i,

$A^* = \{(i,*), i \in N \setminus \{*\}\}$ is the set of arcs from every node to the sink node.

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:

$z^v_{ij} = 1$ if vessel $v \in V$ traverses arc $(i,j) \in A$, 0 otherwise.

$\beta^b_{ij} = 1$ if body $b \in B$ traverses arc $(i,j) \in A$, 0 otherwise.

$\omega^{bv}_{ij} = 1$ if body $b \in B$ is attached to vessel $v \in V$ and traverses arc $(i,j) \in A$, 0 otherwise.

$x^{br}_{ij} = 1$ if the containers of request $r \in R$ traverse arc $(i,j) \in A$ inside body $b \in B$, 0 otherwise.

$\delta^b_{ij} = 1$ 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 a vessel-body combination, 0 otherwise.

$\tau^{Av}_i \geq 0$ is the arrival time of vessel $v \in V$ at node $i \in N$.

$\tau^{Dv}_i \geq 0$ is the departure time of vessel $v \in V$ from node $i \in N$.

$\tau^{BAb}_i \geq 0$ is the arrival time of body $b \in B$ at node $i \in N$.

$\tau^{BDb}_i \geq 0$ is the departure time of body $b \in B$ from node $i \in N$.

$y^v_{ij} = 1$ if $i \in N$ precedes $j \in N$ (not necessarily immediately) during the trip of vessel v, 0 otherwise.

$\varphi^b_{ij} = 1$ 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.

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

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 constraints: 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 (2.22) ensure that the delivery service is conducted by the same body as the pickup service for each request. Constraints (2.23) 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.

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

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

$$x^{b_{r_i} r_i}_{i(i+n)} = \gamma^{r_i}, \qquad \forall i \in D : r_i \in R^2 \quad (2.22)$$

$$\sum_{k \in N : k \neq i, (j,k) \in A} x^{br_i}_{jk} = \sum_{k \in N : k \neq i+3n, (k,j) \in A} x^{br_i}_{kj}, \qquad \forall i \in P : r_i \in R^1, b \in B, \; j \in N \setminus \{*, i, i+3n\} \quad (2.23)$$

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

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.

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

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

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.

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

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

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.

$$\tau^{Dv}_i \geq \tau^{Av}_i + T^C + T^{max} \textstyle\sum_{j:(j,i) \in A} w^{vb}_{ji} - T^{max} \textstyle\sum_{j:(i,j) \in A} w^{vb}_{ij} - T^{max}, \qquad \forall b \in B, v \in V, \; i \in N : i \neq o_b, * \quad (2.29)$$

$$\tau^{Dv}_i \geq \tau^{Av}_i + T^C - T^{max} \textstyle\sum_{j:(j,i) \in A} w^{vb}_{ji} + T^{max} \textstyle\sum_{j:(i,j) \in A} w^{vb}_{ij} - T^{max}, \qquad \forall b \in B, v \in V, \; i \in N : i \neq o_b, * \quad (2.30)$$

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).

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

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

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

Preliminary experiments revealed that the proposed integer programming formulation was unable to generate high-quality solutions within reasonable runtimes. Therefore, in order to provide time-constrained industry with a decision support tool they can use in practice, this chapter will introduce a tailored heuristic for the VSBR. Given the many problem components involved and the necessity of efficiently exploring the solution space, we will focus on the following two decision levels: (i) deciding when and at which locations to transfer bodies and (ii) PDPTW optimization.

The first decision level requires one to efficiently explore the vast number of possible body transfers given that each body can be transferred multiple times in many different places throughout the scheduling horizon. This decision level can be very disruptive since removing or inserting body transfers may impact multiple vessels' routes as well as the assignment of containers to bodies. However, once decided, body transfers can be fixed. What this means is that vessels have mandatory stops in their routes. The next decision level can then improve these routes by employing an efficient PDPTW algorithm which takes into account these fixed transfers.

Given the need to search with respect to two decision levels, the Iterated Local Search (ILS) method (Lourenço et al., 2003) seems an intuitive approach. Given an initial solution, ILS generates neighboring solutions by applying a perturbation method followed by a local search. Besides featuring an iterative search which is subdivided into levels, ILS has also demonstrated good results when applied to routing and scheduling problems (Lourenço et al., 2019). For instance, the SB-VRP was successfully addressed by Toffolo et al. (2018), who developed a hybrid algorithm combining ILS and the Late Acceptance Hill-Climbing (LAHC) metaheuristic (Burke and Bykov, 2017).

We therefore decided to adapt ILS in order to address 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 in the vessels' routes before solving a PDPTW where vessels with fixed bodies must be routed to serve requests. Each newly generated solution is potentially accepted based on a list inspired by LAHC. This acceptance criterion benefits from the fact that there is only one parameter which requires calibration: the length of the list. Although the high-level framework of our heuristic shares some similarities with Toffolo et al. (2018), we only employ the late acceptance list embedded into ILS and not the entire LAHC metaheuristic framework. All of the other components of our algorithm, which will be described in the following sections, are completely different and tailored to the VSBR.

The basic framework of ILS-SB is provided by way of Algorithm 1. First, the algorithm receives as input an initial solution s and the maximum length for the late acceptance list, denoted by l (line 1). The late acceptance list ($AcceptL$) is then initialized with the initial solution value (line 3), meaning that solutions generated during the first $l-1$ iterations will 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). Similarly, solution s' replaces incumbent solution s if *AcceptL* is not full or if $f(s')$ improves upon the considered entry in the late acceptance list (lines 10-11). Next, *AcceptL* is updated with the cost of the incumbent solution (line 13). The best solution is returned (line 14) when the algorithm reaches either of its two stopping criteria: the maximum number of iterations without improvement (*#maxIt*) or the time limit.

Algorithm 1: ILS-SB framework.

```
1  Input: initial solution $s$, list size $l$
2  $s^* \leftarrow s$
3  $AcceptL[0] \leftarrow f(s)$
4  $i \leftarrow 0$
5  while #maxIt or time limit is not reached do
6      $s_t \leftarrow$ Transfer phase($s$) // Section 2.4.4
7      $s' \leftarrow$ Routing phase($s_t$) // Section 2.4.5
8      if $f(s') < f(s^*)$ then
9          $s^* \leftarrow s'$;
10     if $length(AcceptL) < l$ or $f(s') < AcceptL[i \ mod \ l]$ then
11         $s \leftarrow s'$
12     i++
13     $AcceptL[i \text{ mod } l] \leftarrow f(s)$
14 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, we propose a constructive method that generates a solution in which bodies remain attached to their initially assigned vessel and no transfers take place. This means that any body which is detached at the beginning of the scheduling horizon will not be used, while each vessel which serves a pickup service must 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 unserved 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 unserved 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 unserved 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 unserved 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 number of containers a body can load 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.

Body-transfer insertion

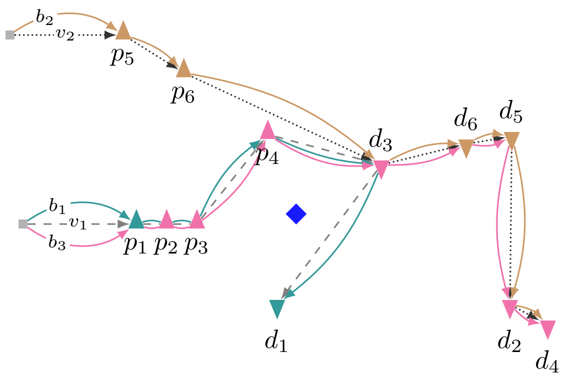
The four different transfer insertion neighborhoods follow the same basic steps outlined in Algorithm 2. Each neighborhood receives as input a solution, a body to be transferred, a node to perform the transfer and a position in the body's route to insert the transfer (lines 1-2). First, since locations 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 unserved request set U (line 3). If b was transferred after $prev_t$, the body transfer is also removed. When removing requests and/or transfers 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: Body-transfer insertion.

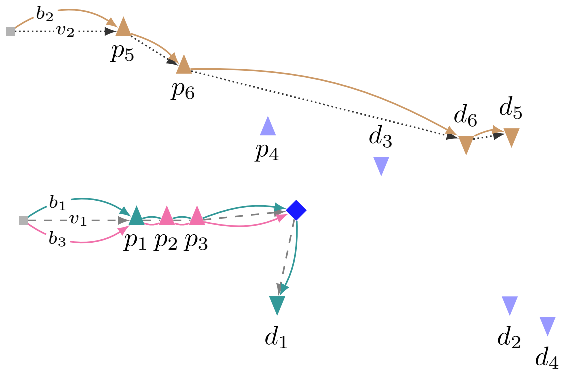
```
1  Input: solution $s$, body $b$ to be transferred
2  Input: transfer node $node_t$, last node $prev_t$ into $b$'s route before
     transfer
3  $U \leftarrow$ Remove from $b$'s route every served request after $prev_t$ (s)
4  Detachment of $b$ at $node_t$ after $prev_t$ (s)
5  if Detached body is empty then
6      Best insertion of all requests in $U$ (s)
7  else
8      foreach vessel route $v_r \in V$ do
9          foreach vessel stop $v_{stop} \in v_r$ do
10             $s' \leftarrow$ 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                 $s \leftarrow s'$
14 return $s$
```

If the detached body is empty, meaning there is no container loaded in the body, the unserved 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., 2013b). As with the best insertion of requests into body routes, the best insertion of body transfers 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 unserved requests are inserted into the solution (lines 8-13). Finally, the solution yielding the lowest cost is saved and returned (line 14).

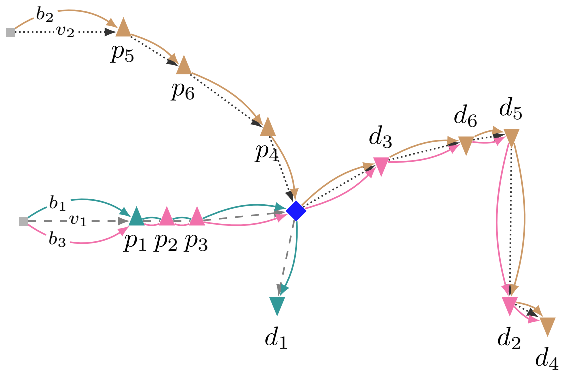
Figure 2.4 details the steps to insert a body transfer. The network representation follows the same format as that used in Figure 2.2, where the routes of bodies and vessels are depicted by colored and dashed edges, respectively. Figure 2.4(a) shows a solution for the VSBR where bodies b_1 and b_3 start 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.



(a)



(b)



(c)

Figure 2.4: Body transfer insertion.

Consider a body-transfer 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 unserved 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 unserved 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 c_r which is associated with either a pickup or delivery service of a request r not in U, in other words, a served customer. 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. Small detours for vessels which attach a body should not significantly affect the objective value.

2. *Large number of containers*: customers where a significant number of containers ρ_r are being served. The total service time of 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 served customer c_r:

$$score_{c_r} = \alpha * \frac{1}{travel_{c_r}} + (1 - \alpha) * \rho_r \tag{2.47}$$

In Equation (2.47), parameter α controls the extent to which travel cost and number of requested containers contribute to the selection criterion. Parameter $travel_{c_r}$ corresponds to the total travel cost between customer c_r and all the other customers which are visited by other bodies. A low $travel_{c_r}$ 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 ρ_r, 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 customer c_r. Note that c_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.

Body-transfer removal

Algorithm 3 provides an overview of how to remove body transfers. A body transfer comprises of the delivery t_d and pickup t_p of a body b. Given a solution and a body transfer as input, each customer served by body b after pickup node t_p is removed from b's route and inserted into the unserved 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: Body-transfer removal.

```
1 Input: solution $s$, body transfer to remove ($t_p$ and $t_d$), 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 body transfer to remove.

Random transfer removal (TR1). This neighborhood selects, with equal probability, a single body transfer 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 body transfer at a time. The removal which yields the best solution value is maintained and the solution is returned with at least one fewer body transfer.

2.4.5 Routing phase

Once body transfers are fixed, vessel-body combinations may be considered as a single vehicle which have fixed visits scheduled in their routes. Note that the capacity of a vessel-body combination may change during the execution of a route given that body transfers may take place. The VSBR with fixed body transfers is therefore reduced to a PDPTW which must be addressed by the routing phase. This method optimizes vessel routes by rescheduling requests while respecting predefined body transfers. 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 option -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. The data provided to us is limited given the confidentiality terms agreed upon. For example, the company's executed schedules and routes were not disclosed. Thus a direct comparison with their results is unfortunately not possible. However, the instances we generated employ as much of the real data provided to us as possible in order to approximate the real scenario and provide valuable insights concerning the problem's characteristics. All benchmark instances are anonymized and have been made publicly available[3].

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. 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[4] 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.

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

[3]`https://doi.org/10.1016/j.ejor.2022.02.015`
[4]`https://www.routino.org/`

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 $\tau_{o_v}^{Av}$ 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_{size}^{P}) is generated followed by the opening day ($Open_{day}^{P}$), which is selected with uniform probability between -5 and 20-TW_{size}^{P}. If $Open_{day}^{P} < 0$, 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_{size}^{D}). The opening day for the delivery service ($Open_{day}^{D}$) strongly depends on the opening day of the pickup service. If $Open_{day}^{P} < 0$, then $Open_{day}^{D}$ is selected with uniform probability between 0 and 20-TW_{size}^{D}. Otherwise, the considered range for $Open_{day}^{D}$ is 0-20. If $Open_{day}^{D} > 20 - TW_{size}^{D}$, 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.

In the considered scenario, requests are associated with multiple containers and vessels can tow bodies capable of holding hundreds of such containers. Therefore, due to economies of scale, serving requests with vessels will always be considerably cheaper than serving them with trucks. This is because trucks typically only carry one or two containers and therefore either multiple trips or multiple trucks would be required to serve a single request. As a result, an unserved request incurs a significant outsourcing cost π which ensures that it is extremely unlikely to have an advantageous solution by leaving a request unserved. For the introduced benchmark, π was artificially set to 10,000 in order to simulate an expensive outsourcing cost. Thus, a solution which serves more customers, even if it has a greater travel cost, is almost always going to be better than a solution which serves fewer customers. Note that the value of π may be adjusted depending on the problem context. Take, for example, a small-scale problem where bodies have less capacity and requests are associated with fewer containers. In this case π may require a lower value in order to capture the fact that outsourcing will make financial sense more frequently.

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 unserved requests |U|. Given the non-deterministic nature of ILS-SB, each experiment is 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 unserved requests |U|.

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.

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

ILS-SB results (with exception of S^*) correspond to the average over 10 runs.

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 unserved 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 unserved requests given that it becomes more difficult to serve more requests within the same length scheduling horizon. The number of unserved 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 1 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 unserved 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.

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,

Table 2.6: Body transfers analysis.

Instances	ILS-R			ILS-SB		
	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%	

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 unserved requests is given by $|\text{U}|_{avg}$. 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 (gap($\text{S}_{a}vg$)). 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.

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

Table 2.7: Solution values when using different sets 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
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.

travel cost, ILS-SB obtains the smallest gap considering average solution value (2.93%). This result derives itself from the low number of unserved requests (0.21 on average). Given the nature of the VSBR and the large outsourcing costs associated with unserved 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 unserved 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 unserved requests. The best solution is obtained by ILS-SB, which on average has 1.48 unserved requests. On the other hand, TI4 and TR1 have on average 26 and 4 unserved requests which results in solutions that are 84% and 43% worse, respectively.

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 unserved 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 is a parallel-coordinates plot which illustrates the relationship between travel cost, the percentage of unserved requests $|\mathrm{U}|\%$ and the total cost. For this experiment, a large number of solutions were generated by uniformly selecting

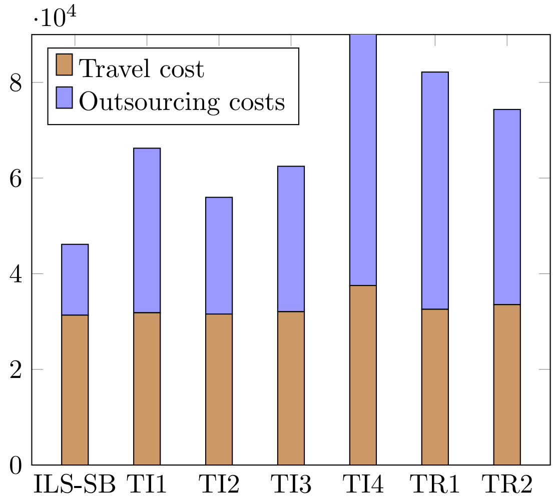


Figure 2.5: Solution values for the I_300 instances.

the number of requests to remain unserved, while the other requests were randomly inserted into their first feasible position. Travel costs and total costs were normalized from 1-100 to better visualize the results. Note that the left-hand side of the figure has crossing edges, while the right-hand side demonstrates a near one-to-one direct correspondence. These results demonstrate that while a trade-off between travel cost and the percentage of unserved requests may occur, the proportion of unserved requests is directly connected to the total cost. This demonstrates the significant impact outsourcing costs have on the solution value and how more unserved requests will never result in lower total costs given the adopted value of π in the proposed instance set. Thus, it is always beneficial to serve more requests, despite the increase in travel cost.

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.

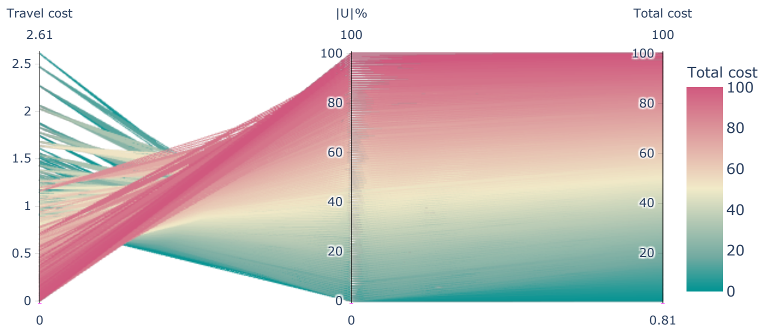


Figure 2.6: Relationship between travel cost, percentage of unserved requests and total cost.

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 solution 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 transfer decisions (when and at which locations 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

The impact of loading restrictions on the two-echelon location routing problem

This chapter integrates a location routing problem spread across two distribution levels (*echelons*) together with two-dimensional truck loading restrictions. This problem arises in freight distribution when goods available at different origins are delivered to their respective destinations via intermediate facilities. The first echelon corresponds to delivery of goods by large vehicles from depots to smaller intermediary facilities located in the outskirts of cities, while the second echelon involves the transportation of these goods from the intermediary facilities to final customer locations with smaller vehicles. Both depots and intermediary facilities must be selected from many possible options and they incur a fixed rental cost when used. Moreover, goods being transported are large industrial machines that require additional packing restrictions and special attention during loading to ensure their safe transportation.

In contrast to the previous chapter, the extra decision levels in this chapter (location and packing) are not optional but crucial for ensuring the feasibility of solutions. Nevertheless, we are once again interested in determining whether it is possible to minimize the impact on routes when varying location decisions and the impact on the solution (route and location) of one echelon when changing the other. Bounds concerning the routing component of the problem are generated in an attempt to reduce location possibilities. Furthermore, we will also investigate whether it is possible to design a load plan method that is capable of speeding up the packing feasibility check.

This chapter is a minimally altered version of the peer-reviewed publication Gandra et al. (2021b)[1]. For this publication Tony Wauters gave suggestions concerning the design of the load plan method, Hatice Çalık contributed to the design of the MIP formulation for the first-echelon solution and the design of the lower bound method, Túlio Toffolo and Marco Carvalho guided the heuristic development stage, while Greet Vanden Berghe supervised the research.

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 (p_2 and p_4) and the routes connecting these platforms to the satellites (represented with the thicker lines). The second echelon comprises of three open satellites: s_3, s_4 and s_5. 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, which indicates the exact position of items in the vehicle's two-dimensional loading surface, is provided for the vehicle during route $\mathrm{s}_4 \to 7 \to 8 \to 9 \to 10 \to \mathrm{s}_4$. Items are loaded from the back of the vehicle (rear loading) at satellite s_4 before the trip begins. During the trip, each item will be unloaded from the rear. In this particular load plan, the items belonging

[1]V. M. S. Gandra et al. (2021b). The impact of loading restrictions on the two-echelon location routing problem. In: *Computers & Industrial Engineering* 160, p. 107609.

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.

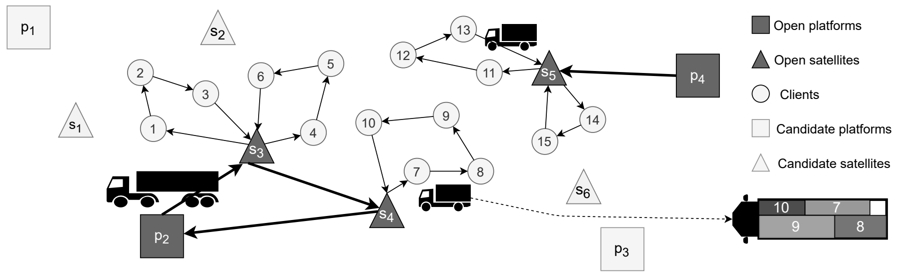


Figure 3.1: Two-echelon routes with load plan.

The literature on multi-echelon location-routing problems dates back to the early study by Jacobsen and Madsen (1980), while Boccia et al. (2010) first described the two-echelon location routing problem (2E-LRP) where both platforms and satellites must be selected from multiple candidates. 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 choice of two-dimensional loading constraints is justified by the real-world context which motivated this research. Given that items are extremely large and heavy machinery products, they cannot overlap with one another or be stacked without causing significant damage or deformation. Furthermore, items have wheels with limited maneuvering capacity and are either driven into vehicles or (un)loaded in a specific direction using forklifts. Either way, this (un)loading procedure means that rotating items is impossible.Therefore, it is perfectly realistic to ignore the height of these items and consider only their two-dimensional projection (bottom surface).

The problem thus becomes 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, unloading and re-loading an item at the same location or movements inside the vehicle are considered highly undesirable. This is due to the fact that the space required 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, four loading strategies are developed for the proposed heuristic. These different strategies should 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.

The remainder of the chapter is organized as follows. Section 3.2 presents a literature review concerning related problems and indicates relevant elements for the present study. A detailed description of the problem and notation is provided in Section 3.3, while the proposed method for the 2E-LRP2L is introduced in Section 3.4. Computational experiments and a study on the impact of loading constraints are conducted in Section 3.5. Section 3.6 makes a few minor modifications to the proposed algorithm so as to make it applicable to 2E-LRP instances. Finally, Section 3.7 concludes this chapter and outlines some future research directions.

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 in Section 3.2.1. Later, in Section 3.2.2, the most relevant studies on the vehicle routing problem with two-dimensional loading constraints (2L-VRP) 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 assessing 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. Table 3.1 provides an overview of those studies and details the problem(s) each addresses along with the decisions involved when doing so. The general observations made in the following subsections are supported by the papers from this table. 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.

Table 3.1: Overview of 2E-LRP and 2L-VRP approaches and related literature.

Reference	Problem(s) addressed	Method	Two-echelon	Location	Routing	Multi platform	Split delivery	Packing
Boccia et al. (2010)	2E-LRP	H	✓	✓	✓	✓	-	-
Boccia et al. (2011)	2E-LRP	E	✓	✓	✓	✓	-	-
Nguyen et al. (2012a)	2E-LRP and 2E-LRPSD	H	✓	✓	✓	✓	-	-
Nguyen et al. (2012b)	2E-LRPSD	H	✓	✓	✓	-	-	-
Contardo et al. (2012)	2E-LRP and 2E-LRPSD	E and H	✓	✓	✓	✓	-	-
Schwengerer et al. (2012)	2E-LRP and 2E-LRPSD	H	✓	✓	✓	✓	-	-
Breunig et al. (2016)	2E-VRP and 2E-LRPSD	H	✓	✓	✓	-	-	-
Pichka et al. (2018)	2E-OLRPSD and 2E-LRPSD	E and H	✓	✓	✓	-	-	-
Iori et al. (2007)	2L-VRP	E	-	-	✓	-	-	✓
Gendreau et al. (2008)	2L-VRP	H	-	-	✓	-	-	✓
Zachariadis et al. (2009)	2L-VRP	H	-	-	✓	-	-	✓
Fuellerer et al. (2009)	2L-VRP	H	-	-	✓	-	-	✓
Leung et al. (2010)	2L-VRP	H	-	-	✓	-	-	✓
Leung et al. (2011)	2L-VRP	H	-	-	✓	-	-	✓
Zachariadis et al. (2013)	2L-VRP	H	-	-	✓	-	-	✓
Wei et al. (2015)	2L-VRP	H	-	-	✓	-	–	✓
Wei et al. (2018)	2L-VRP	H	-	-	✓	-	-	✓
This chapter	2E-LRP2L and 2E-LRP(SD)	H	✓	✓	✓	✓	✓	✓

E: exact approach. H: heuristic. 2E-OLRPSD: 2E-LRPSD with open routes (vehicles do not need to return to depots).

3.2.1 Selected studies concerning the 2E-LRP

Boccia et al. (2010) were the first to formalize the 2E-LRP. They 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 of 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. (2012b), who generated benchmark instances and made them publicly available. Nguyen et al. (2012b) 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 were introduced by 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. Moreover, 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. (2012a) and Nguyen et al. (2012b). 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).

There are a few observations worth stressing concerning the research related to the 2E-LRP and the 2E-LRPSD documented in Table 3.1. The table indicates that high-quality solutions for medium- and large-scale instances of the 2E-LRP are most often obtained with heuristics. Fast, dedicated approaches are typically employed for the first echelon given its small size relative to the second echelon. However, it is worth noting that Boccia et al. (2010) and Contardo et al. (2012) recursively address both echelons using a single method.

3.2.2 Selected studies concerning the 2L-VRP

When it comes to loading restrictions, the most closely related problem to the 2E-LRP2L is the 2L-VRP. Although a generalization of the 2L-VRP which considers a VRP with three-dimensional loading constraints (3L-VRP) was introduced by Gendreau et al. (2006), the first dedicated method for the 2L-VRP 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 VRP have also been extended to address the 2L-VRP. Studies concerning the 2L-VRP documented in Table 3.1 range from the paper which first introduced the problem (Iori et al., 2007) through to the 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.

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 length of time required to either load or unload the item. An unlimited fleet of homogeneous vehicles is assumed to be available for each echelon. Without loss of generality, let us consider a finite set of vehicles to be $K = K^1 \cup K^2$, where K^1 and K^2 are the sets of vehicles associated with the first and second echelons, respectively. The number of items which must be delivered in each echelon serve as upper bounds for the number of vehicles $|K^1|$ and $|K^2|$, which will always be sufficient to deliver all items. 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 can safely assume that physically (un)loading and maneuvering items at facilities does not pose a problem. Therefore, only the area 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 vector (R, L), where $R = \{R_1, ...R_{|K|}\}$ and $L = \{L_1, ...L_{|K|}\}$. For each vehicle $k \in K$, R_k is a route performed by k and L_k is the full load plan of k as it departs from the platform/satellite. A route is deemed feasible if it adheres to the following constraints defined by Boccia et al. (2010): first-echelon routes start from a platform, serve one or more satellites and end at the same platform; second-echelon routes start from a satellite, serve one or more customers and end at the same satellite; and 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: all items assigned to a vehicle must fit totally within the loading surface of that vehicle; stacking is strictly forbidden and items must not overlap with one another; 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.

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 figure demonstrates a violation given that one of the items breaches the limits of the vehicle's loading surface. Figure 3.2(b) demonstrates an overlapping violation, with two items sharing a portion of the loading surface. 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 orthogonal loading constraint. Finally, Figure 3.2(d) illustrates a load plan where no loading constraints are violated.

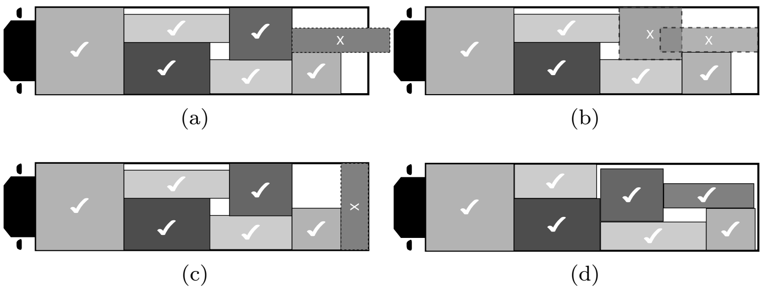


Figure 3.2: Loading constraints and violations.

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. Violations of this constraint mean that a certain item $m \in M \setminus M_i$ is blocking the unloading of another item $m' \in M_i$, which incurs a penalty proportional to the time it takes to rearrange m. Since one must not only unload but also reload an item which is blocking another, the penalty for moving item m is twice its service time $(2 \times st_m)$. This penalty is referred to as a *sequential loading penalty* (Iori and Martello, 2010), or SLP for short, 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 (R, 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.

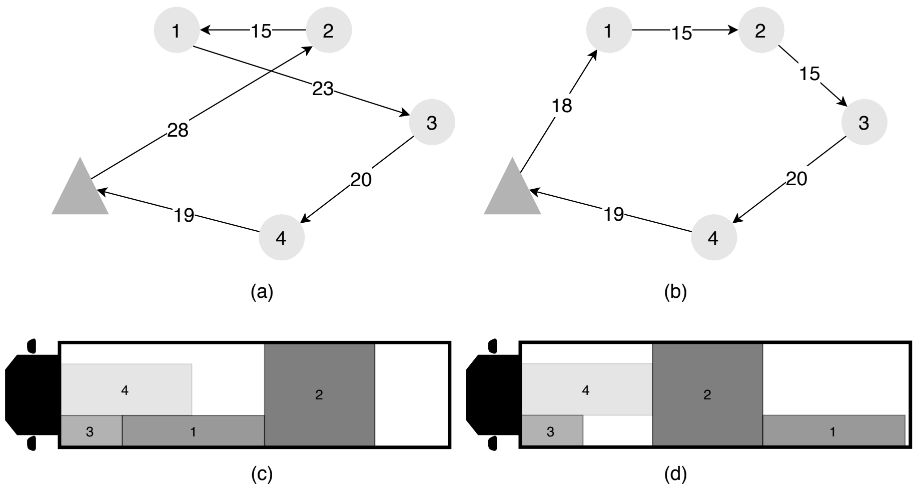


Figure 3.3: Cost for the combinations of two routes and two load plans.

Modeling challenges: In order to develop a mathematical formulation for the 2E-LRP2L one could extend the 2E-LRP formulation proposed by Boccia et al. (2011). This one-index formulation, provided in Appendix A.1, is a path-based formulation which defines a variable for each feasible route. In the 2E-LRP2L, routes must deliver a set of unstackable items M_c to each customer $c \in C$. Thus, all feasible routes must be generated considering two-dimensional loading constraints and sequential loading penalties. Moreover, split deliveries must be included, but are only allowed for first-echelon routes. Given these nuances, there are two main adaptations one must make to the formulation by Boccia et al. (2011) in order to arrive at a formulation for the 2E-LRP2L.

First, when generating routes, the feasibility of each could be checked by employing the two-dimensional orthogonal packing with sequential loading constraint formulation by Côté et al. (2014). Since violations of sequential loading are penalized rather than strictly forbidden in the 2E-LRP2L, the constraints which enforce sequential loading in the formulation proposed by Côté et al. (2014) should be relaxed and included in the objective function with the corresponding penalties. Second, instead of customer requests (given by a set of items), first-echelon routes must deliver individual items $m \in M$ in order to facilitate split deliveries. Constraints which enforce a satellite to be served by only one vehicle should therefore be removed from the formulation by Boccia et al. (2011). However, since split delivery is not allowed in the second echelon, a single vehicle carrying all items $m \in M_c$ of customer c must depart from one satellite to serve the customer in a single trip. Therefore, satellites can be visited multiple times by vehicles departing from different platforms so long as all items $m \in M_c$ are delivered to the same satellite. To model such constraints one could introduce an additional variable ω_{mst} which is equal to 1 if item $m \in M$ is delivered to satellite $s \in S$ by first-echelon route $t \in \mathcal{T}^1$, and 0 otherwise.

This strategy for developing a mathematical formulation for the 2E-LRP2L represents an extension of existing methods. Although intuitive, the resulting formulation would be lengthy since, as we have shown, it requires one to combine multiple formulations and would involve a large number of variables and constraints. Indeed, given that the formulations by Boccia et al. (2011) already face challenges to solve even small-scale instances it is logical to assume that an extended version would experience even greater difficulties. Therefore, we will focus on developing a heuristic approach for solving the 2E-LRP2L.

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. When location and routing heuristics are applied interchangeably (without a specific order), facility configurations may be generated without optimizing the routes. This is because multiple location heuristics may be applied one after another without a routing heuristic. In this chapter, 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  $s \leftarrow$ routing local search($s^0$, $ls$)
4  $s^* \leftarrow s$
5  **for** $i \leftarrow 0$ **to** $maxNumIter$ **do**
6      $s_l \leftarrow$ perturb location ($s$)
7      $s_r \leftarrow$ routing local search($s_l$, $ls$)
8      $s' \leftarrow$ solve the first echelon($s_r$)
9      **if** $f(s') < f(s^*)$ **then**
10         $s^* \leftarrow s'$
11     **if** $i \; mod \; \eta = 0$ **then**
12         $s \leftarrow s^*$
13     **else**
14         $s \leftarrow s'$
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-VRP. 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.

Algorithm 5: Initial solution framework.

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

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 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 (s_1, s_2, s_3, s_4 and s_5) to serve all customers. Figure 3.4(b) presents an alternative solution after two moves have been applied. First, satellite s_3 has been closed and its customers (7, 8 and 9) have been reassigned to nearby satellites (s_2, s_4 and s_5). The second move swapped the status of satellites s_3 and s_6, with the complete route of s_1 reassigned to s_6.

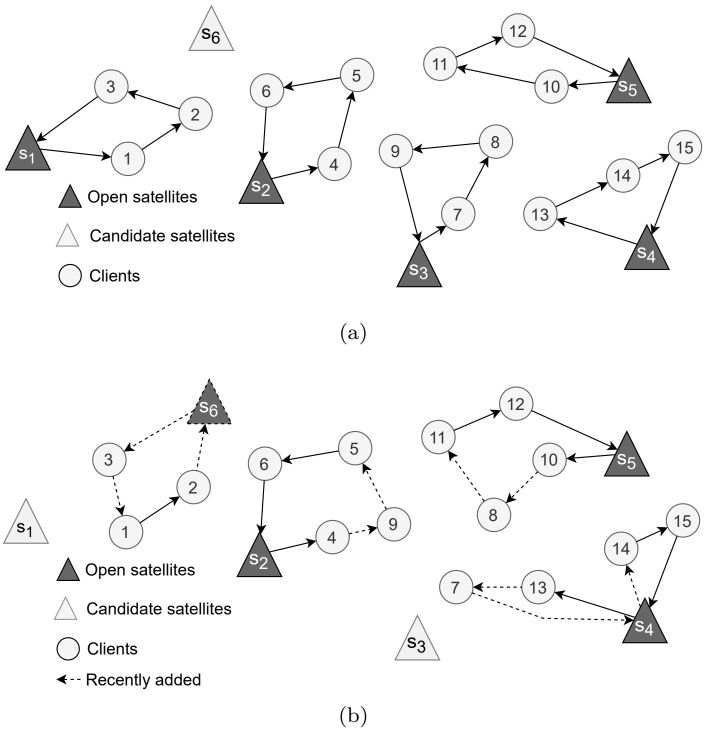


Figure 3.4: Second-echelon routes with different satellite configurations.

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 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.

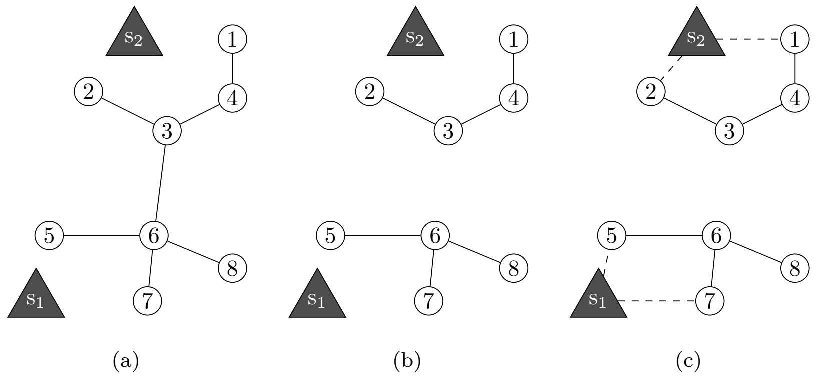


Figure 3.5: Second echelon minimum routing cost construction.

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. For interested readers we refer to Appendix B which includes an extension of this load plan method. In this extension the feasibility of a load plan for trucks must be guaranteed in accordance with several additional packing restrictions and flexibility, such as minimum distance between items and partial overlapping of items.

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.

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.

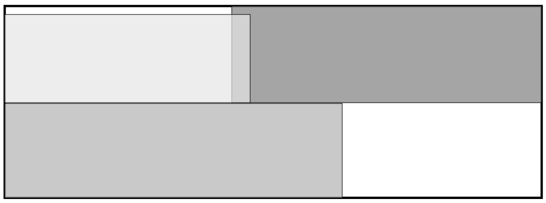


Figure 3.6: An infeasible load plan for items whose total area is less than that of the vehicle.

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. All instances, solutions and tables with results are available in a public repository[2].

Table 3.2 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 3.2.

Table 3.2: ILS-LR2L parameters and values.

Parameter	Value	Range
$maxNumIter$	50	-
SISRs-MD iterations	1000	-
SISRs iterations	500	-
wp	0.003	{0.001, 0.003, 0.005, 0.01, 0.03}
η	2	{1, 2, 4, 6, 8, 10}
$\sigma_{recreate}$	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,

[2]`https://doi.org/10.17632/hkxchx5sxp.1`

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 Figure 3.7, 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

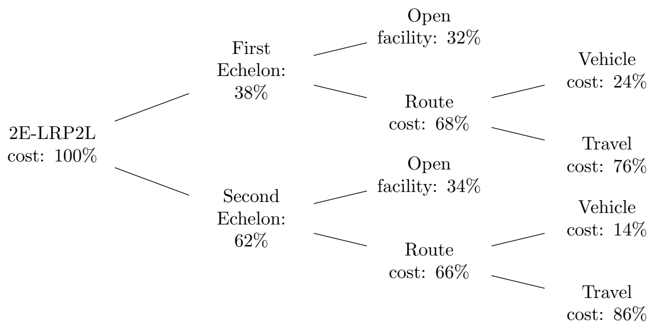


Figure 3.7: The average cost breakdown of 2E-LRP2L solutions.

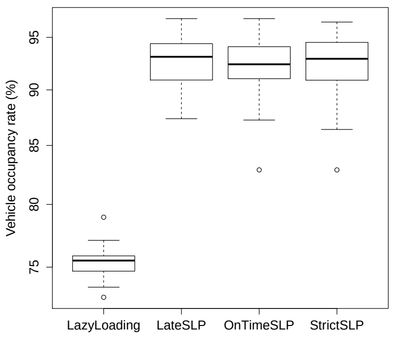
capacities were selected between 40% and 70% of total item area, which results in solutions with two or three open platforms out of four.

3.5.2 Detailed results

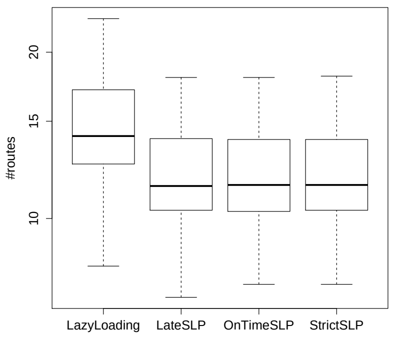
This section presents computational experiments and compares the results produced by ILS-LR2L using the four different loading strategies. Figures 3.8(a)-3.11(a) provide boxplot charts which compare the four loading strategies, namely: *LazyLoading*, *LateSLP*, *OnTimeSLP* and *StrictSLP*. The values 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 3.8 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*.

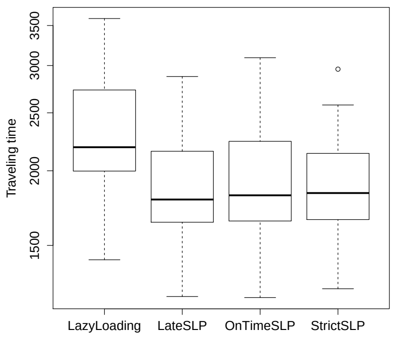
Figure 3.9 provides the results of the second echelon, where the SLP is applied. For the *ActiveLoading* strategies, Figure 3.9(a) shows a decrease concerning vehicle occupancy rates as the loading constraints become more restrictive. However, *LazyLoading* is a clear outlier here due to the aforementioned maximum



(a)



(b)

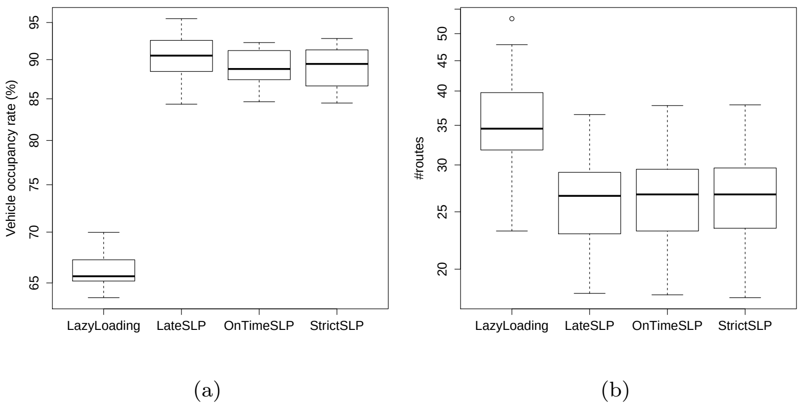


(c)

Figure 3.8: Different loading strategies for the first echelon.

occupancy rate restrictions (which are set to 70% for second-echelon vehicles). A similar discrepancy can also be observed in Figure 3.9(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 3.9(c) shows the proportion of routes which incur penalties per loading strategy. Only 2.9% of the routes produced by *OnTimeSLP* incur SLP penalties,



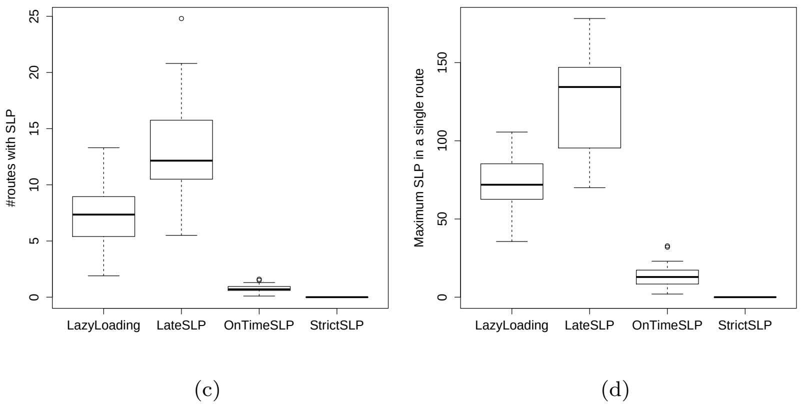


Figure 3.9: Different loading strategies for the second echelon.

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 3.9(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.

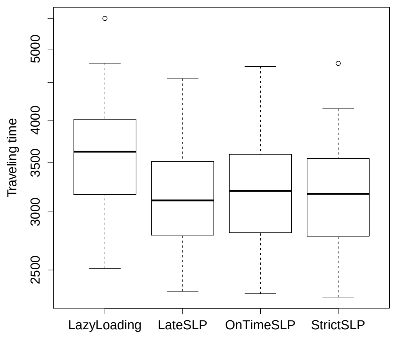
Figure 3.10 provides the total traveling time and the total penalty incurred by the second echelon. The *LazyLoading* and *LateSLP* strategies take up to 20% longer to complete routes. This additional time is strongly related to the penalties their routes incur. When comparing only the traveling time, 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 3.10(c)) results show that all pairs are significantly different, with the exception of *LazyLoading*-*LateSLP*.

Finally, Figure 3.11 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*.

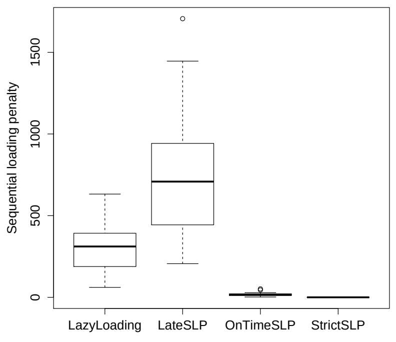
3.5.3 Result discussion

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.

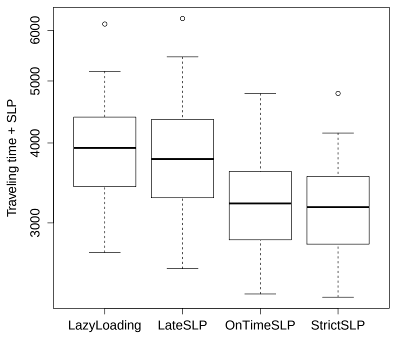
When dealing with large items, the time spent moving (loading and unloading) such items may be significant when compared to the average travel time. Considering the three *ActiveLoading* strategies, *LateSLP* results in slightly shorter travel times than *OnTimeSLP* and *StrictSLP*. However, *LateSLP*'s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



(a)



(b)

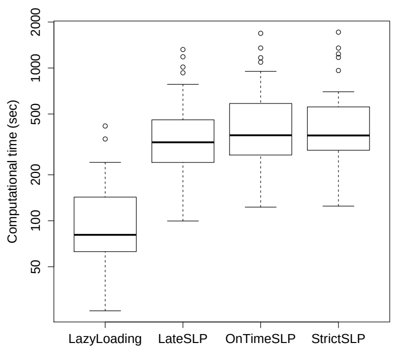


(c)

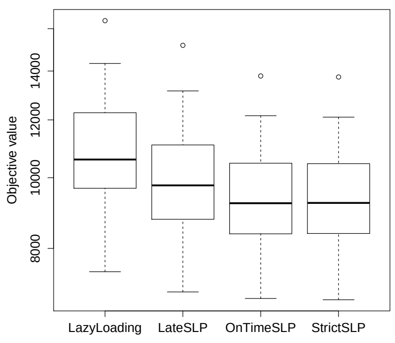
Figure 3.10: Different loading strategies for the second echelon. Traveling time and penalty analysis.

studied context: that of large items whose rearrangements are time-consuming compared to travel times.

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.



(a)



(b)

Figure 3.11: Best solution comparison and run time of different loading strategies.

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. (2012b) and contain only one platform in the first echelon. *Prodhon* comprises of 30 instances arising from the LRP 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.2. 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 VRP/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 2.5. 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 3.3).

Table 3.3: 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 3.4 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 3.4 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 April 26, 2022), 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.

Table 3.4: 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 3.5 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 3.5: 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

Multi-depot periodic vehicle routing with variable visit patterns

This chapter addresses a VRP in which customers must be served periodically over a time horizon of multiple days. For customer satisfaction and security reasons, two visits to a given customer cannot be scheduled longer than a given number of days apart, although scheduling visits sooner than required is allowed. This flexibility concerning customer schedules results in many possible visiting patterns which considerably expands the size of the search space, while simultaneously providing opportunities to improve the solution. Given that we have already studied the characteristics of solving a single-day VRP, in this chapter we can focus on the multi-day aspect of the problem. Our core aim is to find a way of manipulating customer schedules in a timely fashion and produce high-quality routes for each day in order to optimize the solution across the entire time horizon. Given that throughout the search process the set of customers to be served on a given day may change, we aim to investigate mechanisms to limit the number of possible visiting patterns and to accelerate the routing optimization process for each day.

This chapter is based on the peer-reviewed publication Gandra et al. (2022a)[1]. For this publication, Hatice Çalık and Carlo Sartori contributed to the

[1] V. Gandra et al. (2022a). Multi-depot periodic vehicle routing with variable visit patterns. In: *Proceedings of the Genetic and Evolutionary Computation Conference Companion*, pp. 2019–2026.

development of the MIP formulation, with Carlo Sartori contributing equally to the development of the heuristic and writing of the text. Pieter Smet together with Hatice Çalık supervised the research.

4.1 Introduction

The periodic vehicle routing problem (PVRP) is a generalization of the VRP where each customer is also associated with a service periodicity and must be served one or more times within a given time horizon. The goal is to schedule visits for each customer in order to reduce the total routing cost over the entire time horizon while respecting the periodicity of customers' services and additional routing constraints. The PVRP has extensive real-world applications such as the collection of waste, repair and maintenance work, the routing of home healthcare nurses and inventory routing problems (Campbell and Wilson, 2014).

In this chapter, we investigate a multi-depot PVRP with time windows which was introduced to us by a logistics company, which specializes in the transportation of small valuable items. The periodicity associated with each of their customers can assume one of two forms: fixed or variable. We will refer to this problem as the *Multi-Depot Periodic Vehicle Routing Problem with time windows and Variable-Pattern customers* (MDPVRPVP). In practice, the time horizon corresponds to one month and each day (period) can contain hundreds of visits. Therefore, solving just a single instance requires one to solve many medium- or large-sized VRPs.

The MDPVRPVP takes place over a time horizon P of multiple periods. A graph $G = (V, A)$ represents the transportation network, where $V = V_O \cup V_F \cup V_D$ is the set of nodes corresponding to either depots or customers, and A is the set of arcs (i, j, t_{ij}), $i, j \in V$ for which traveling from i to j takes time t_{ij}. Set V_O contains the depots where vehicles start and end their journey in each period. Meanwhile, sets V_F and V_D represent the set of *fixed-pattern* (FP) and *variable-pattern* (VP) customers, respectively. We associate each $i \in V_F$ with a set Φ_i of potential fixed visit patterns, of which exactly one must be selected for i. Each pattern $\Delta_i^c \in \Phi_i : \Delta_i^c \subset P$ defines a subset of periods during which customer i should be visited. FP customers are the most common type found in the PVRP literature. Meanwhile, we associate each $i \in V_D$ with a maximum number f_i of periods before the first visit to i in any solution and a number m_i of elapsed periods between subsequent visits to i $(f_i \leq m_i)$.

While a set Φ of visit patterns with fixed periods are provided for FP customers, the visit periods of VP customers become an additional decision of the problem bounded by a given frequency of visits. Even though the increased solution space makes the problem harder, exploiting the flexibility of VP customers could lead to reduced expenses while still maintaining quality of service.

To illustrate how incorporating variable pattern customers can impact solutions, consider the very simple MDPVRPVP example outlined in Figure 4.1 with one depot, four periods and four customers. The two gray circles correspond to FP customers who must be visited in periods 1 and 3, while the pink circles correspond to two VP customers who must be visited at most once every two periods ($m = 2$). In this figure we illustrate two solutions for this example in which we exploit the flexibility of VP customers by visiting them in different periods. Figure 4.1(a) depicts a solution where VP and FP customers are not visited during the same periods, which creates routes in all four periods. By contrast, the solution outlined in Figure 4.1(b) shifts VP customer visits one period earlier. As a result, VP customer visits coincide with those of FP customers and operational costs are reduced while ensuring that the visit frequency of VP customers is still respected.

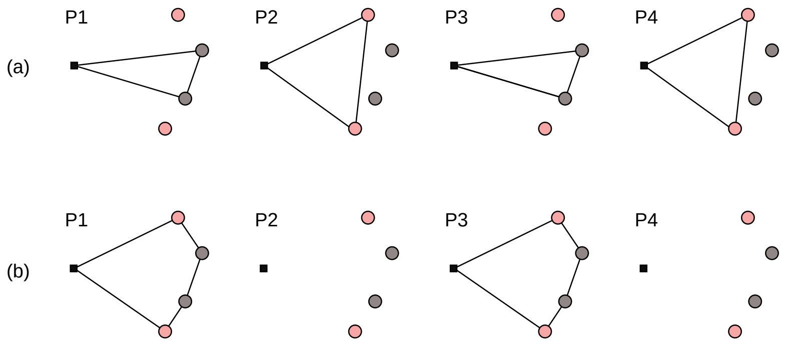


Figure 4.1: Two solutions for an MDPVRPVP instance with four periods and two VP customers (pink).

Returning to the formal definition of the problem at hand, each customer node $i \in V_F \cup V_D$ is associated with a demand d_i, service time s_i and time window $[e_i^p, l_i^p]$ for each period $p \in P$. We assume that $s_i = 0$ and $d_i = 0$ for $i \in V_O$. For $i \in V_O$ the time window denotes the earliest departure time from i and the latest return time to i. For $i \in V_F \cup V_D$ a vehicle is allowed to arrive at i in period $p \in P$ before e_i^p and before beginning service, but it cannot arrive later than l_i^p.

For each period $p \in P$ a fleet K of identical vehicles each of which has a maximum capacity of q is available to perform routes. Vehicles that leave a depot to visit customers must return to the same depot at the end of the period. Moreover, routes in a period $p \in P$ cannot exceed a maximum duration R (from departure to arrival back at the depot) which denotes the working hours of the drivers. In other words, the departure time of routes must be decided in such a way that maximum route durations are respected.

A solution of the MDPVRPVP consists of a single fixed pattern $\Delta_i^c \in \Phi_i$ for each customer $i \in V_F$ and a sequence of visits to customers $i \in V_D$ that complies with the visit frequency constraints. Then, for each period $p \in P$, vehicle routes are created to visit customers assigned to p while respecting vehicle capacity, customer time windows and maximum route duration. The objective is to arrive at a solution that minimizes fuel consumption (driving time) and wage costs (total route duration). We assume a cost γ per minute of driving time and α per minute of route duration.

In this chapter we will investigate the impact of the VP customers in practice. More specifically, do they really allow for cost savings or more effective use of available resources? Is it possible to apply simple state-of-the-art techniques to improve the operational planning of the company? From a purely academic point of view, this chapter also represents the first attempt to combine fixed- and variable-pattern customers within a single PVRP. We also assume longer time horizons compared to typical PVRP instances available in the literature, which impose additional challenges when solving the problem. Instances based on real-world data and results for them have all been made publicly available in order to stimulate further research.

The remainder of this chapter is organized as follows. Section 4.2 provides a mathematical formulation of the problem. Related work is presented in Section 4.3. Section 4.4 introduces a metaheuristic algorithm to solve real, large-scale instances of the company's problem. A computational study is conducted within Section 4.5 where we answer some key research questions and extract key managerial insights. Section 4.6 concludes the chapter and outlines some possible future research directions.

4.2 Mathematical formulation

The MDPVRPVP can be formulated as a Mixed-Integer Linear Program (MILP). In order to do so, let us introduce the following additional notation. Nodes o_k, o'_k denote the start and end depots for vehicle $k \in K$ (o_k and o'_k may denote the same physical depot location). The set of all nodes is $N = \{o_k, o'_k : k \in K\} \cup V_F \cup V_D$. Meanwhile, $A^k = \{(i,j), (o_k, j), (i, o'_k) : i, j \in V_F \cup V_D\}$ is the set of available arcs for vehicle $k \in K$. Additionally, for every $i \in V_F$ we associate an index set Π_i for i and $c \in \Pi_i$ refers to visit pattern $\Delta_i^c \in \Phi_i$ which contains the periods during which customer i must be visited if Δ_i^c is selected. We further define $T_p^{\max} = \max_{k \in K}(l_{o_k}^p)$ for all $p \in P$.

Decision variable $x_{ij}^{kp} = 1$ if arc $(i,j) \in A^k$ is traversed by vehicle $k \in K$ in period $p \in P$, and 0 otherwise. Variable $\pi_i^c = 1$ if for customer $i \in V_F$ the pattern of index $c \in \Pi_i$ is selected, and 0 otherwise. Finally, τ_i^{kp} is the departure time of vehicle $k \in K$ from node $i \in N$ in period $p \in P$. Then, the MILP corresponding to the MDPVRPVP is:

$$\min \ \gamma \sum_{p \in P} \sum_{k \in K} \sum_{(i,j) \in A^k} t_{ij} x_{ij}^{kp} + \alpha \sum_{p \in P} \sum_{k \in K} (\tau_{o'_k}^{kp} - \tau_{o_k}^{kp}) \tag{4.1}$$

$$\sum_{i \in V_F \cup V_D} x_{o_k i}^{kp} \leq 1, \qquad \forall p \in P, k \in K \tag{4.2}$$

$$\sum_{i \in V_F \cup V_D} x_{i o'_k}^{kp} \leq 1, \qquad \forall p \in P, k \in K \tag{4.3}$$

$$\sum_{c \in \Pi_i} \pi_i^c = 1, \qquad \forall i \in V_F \tag{4.4}$$

$$\sum_{k \in K} \sum_{j:(i,j) \in A^k} x_{ij}^{kp} \geq \sum_{c \in \Pi_i \ : \ p \in \Delta_i^c} \pi_i^c, \qquad \forall i \in V_F, p \in P \tag{4.5}$$

$$\sum_{k \in K} \sum_{j:(j,i) \in A^k} x_{ji}^{kp} \geq \sum_{c \in \Pi_i \ : \ p \in \Delta_i^c} \pi_i^c, \qquad \forall i \in V_F, p \in P \tag{4.6}$$

$$\sum_{k \in K} \sum_{j:(i,j) \in A^k} x_{ij}^{kp} \leq 1, \qquad \forall i \in V_D, p \in P \tag{4.7}$$

$$\sum_{k \in K} \sum_{j:(j,i) \in A^k} x_{ji}^{kp} \leq 1, \qquad \forall i \in V_D, p \in P \tag{4.8}$$

$$\sum_{\substack{p \leq f_i \\ p \in P}} \sum_{k \in K} \sum_{j:(j,i) \in A^k} x_{ji}^{kp} \geq 1, \qquad \forall i \in V_D \tag{4.9}$$

$$\sum_{\substack{p'=p-m_i \\ p' \in P}}^{p} \sum_{k \in K} \sum_{j:(j,i) \in A^k} x_{ji}^{kp'} \geq 1, \qquad \forall i \in V_D, p \in P : p \geq m_i \tag{4.10}$$

$$\sum_{j:(i,j)\in A^k} x_{ij}^{kp} - \sum_{j:(j,i)\in A^k} x_{ji}^{kp} = 0, \qquad \forall i \in V_F \cup V_D, p \in P, k \in K \tag{4.11}$$

$$\sum_{i\in V_F\cup V_D} d_i \sum_{j:(i,j)\in A^k} x_{ij}^{kp} \leq q, \qquad \forall p \in P, k \in K \tag{4.12}$$

$$\tau_j^{kp} \geq \tau_i^{kp} + s_j + t_{ij} - T_p^{max}(1 - x_{ij}^{kp}), \qquad \forall p \in P, k \in K, (i,j) \in A^k \tag{4.13}$$

$$\tau_i^{kp} \geq (e_i^p + s_i), \qquad \forall i \in N, p \in P, k \in K \tag{4.14}$$

$$\tau_i^{kp} \leq (l_i^p + s_i), \qquad \forall i \in N, p \in P, k \in K \tag{4.15}$$

$$\tau_{o'_k}^{kp} - \tau_{o_k}^{kp} \leq R, \qquad \forall p \in P, k \in K \tag{4.16}$$

$$\tau_{o'_k}^{kp} \geq \tau_{o_k}^{kp}, \qquad \forall p \in P, k \in K \tag{4.17}$$

$$x_{ij}^{kp} \in \{0,1\}, \qquad \forall k \in K, p \in P, (i,j) \in A^k \tag{4.18}$$

$$\pi_i^c \in \{0,1\}, \qquad \forall i \in V_F \cup V_D, c \in \Pi_i \tag{4.19}$$

Objective function (4.1) minimizes the sum of fuel and wage costs. Constraints (6.1a) and (6.1b) ensure that a vehicle can leave and enter its depots at most once per period. Constraints (4.4) ensure that exactly one visit pattern is selected for each FP customer. Meanwhile, Constraints (4.5) and (4.6) ensure that FP customers are visited in periods that belong to their selected visit pattern. Constraints (4.7) and (4.8) restrict visits to VP customers to at most once per period. Constraints (4.9) require that VP customers are visited before their first visit limit, while the maximum number of elapsed days between two successive visits to a VP customer is restricted by Constraints (4.10). Flow conservation for each vehicle, period and customer node combination is ensured by Constraints (4.11). Vehicle capacity is never exceeded thanks to Constraint (4.12). Time variables τ are updated via Constraints (4.13), while time window restrictions are respected by Constraints (4.14) and (4.15). Constraints (4.16) enforce the maximum route duration restriction. Constraints (4.17) require that the time at which a vehicle completes its journey is not earlier than the time it begins its journey. Finally, Constraints (4.18) and (4.19) set decision variables to be binary.

4.3 Related work

Beltrami and Bodin (1974) first introduced the PVRP in the context of garbage collection at large industrial sites. For each customer, one visit pattern should be selected from a set of available patterns, thereby introducing periodicity into the vehicle routing problem by way of FP customers. State-of-the-art metaheuristics for the basic PVRP are credited to Cordeau and Maischberger (2012) as well as Vidal et al. (2014), while Baldacci et al. (2011) proposed an exact algorithm capable of solving instances with up to 100 customers.

Since the first introduction of the PVRP, a large number of papers with variants has been published creating a rich literature that includes many specialized algorithms for real-world applications. For a broader analysis of this literature, interested readers are referred to the surveys of Francis et al. (2008) and Campbell and Wilson (2014). The latter includes a section focused on variants and extensions arising in real-world applications. For the remainder of this section, we will focus on previous studies closely related to the MDPVRPVP.

Multi-depots were included in the PVRP (MDPVRP) by Hadjiconstantinou and Baldacci (1998) for resource planning in preventive maintenance. They employed a heuristic algorithm with dedicated neighborhoods and local search procedures to tackle the assignment of customers to periods and depots while using classic VRP methods for route optimization. Following Hadjiconstantinou and Baldacci (1998), later researchers focused on the MDPVRP including the current state-of-the-art methods proposed by Vidal et al. (2012) and Rahimi-Vahed et al. (2013).

Cordeau et al. (2001) introduced the PVRP with time windows. Their goal was to develop a single heuristic that was capable of successfully addressing the PVRP, VRP and MDVRP with time windows (PVRPTW, VRPTW and MDVRPTW, respectively) without requiring significant changes. They exploited the fact that both the VRPTW and MDVRPTW are special cases of the PVRPTW and used a tabu search metaheuristic to achieve their goal. State-of-the-art results are currently achieved by Nguyen et al. (2014) and Vidal et al. (2014). In contrast to the MDPVRPVP, most PVRPTWs studied in the literature assume that customers have the same time window in each period. However, this is not necessarily the case in practice given that working hours can differ from day to day.

The full combination of multi-depots, periodicity and time windows which gives rise to the MDPVRPTW was first considered by Vidal et al. (2014). They propose the well-known unified metaheuristic *Hybrid Genetic Search* for multi-attribute VRPs. This metaheuristic was capable of solving 29 VRP variants and obtained competitive or even state-of-the-art results. For the MDPVRPTW, Vidal et al. (2014) introduced a new benchmark set and reported initial upper bounds for each instance. Both the MDPVRPTW and the new benchmarks have received little attention in the literature since then. Nevertheless, since the MDPVRPVP is more closely related to the MDPVRPTW we therefore grant it additional focus in the experiments section as a base case for comparison.

It is worth mentioning that Gaudioso and Paletta (1992) have considered a PVRP where at least n_i and at most m_i periods must have elapsed between successive visits to the same customer i. By setting $n_i = 1$, it is possible to simulate VP customers in the MDPVRPVP. Despite this similarity, Gaudioso and Paletta (1992) did not consider FP customers, time windows and multi-depots. Moreover, the time-horizons considered by Gaudioso and Paletta (1992) was restricted to 12 periods.

Cantu-Funes et al. (2017) introduced the MDPVRPTW with due dates, where the demand of customers should be fulfilled before a given day. Due to high customer demand, routes in their problem were composed only of single-customer trips, rather than multi-customer routes as is typically found in VRPs. Nevertheless, vehicles were allowed to perform multiple trips to compensate for the single-customer trips. This somewhat reduces the importance of routing decisions. In contrast to MDPVRPVP, Cantu-Funes et al. (2017) did not account for fixed delivery frequency or delivery patterns.

Even though there are many papers which have studied PVRPs, the majority consider short time horizons with 4-12 days. In these cases, route optimization is often more important than scheduling decisions concerning which days customers should be visited. One notable exception is Alegre et al. (2007), who studied a PVRP where the time horizon could be as long as 90 days and stressed that in their application scheduling decisions had priority over routing decisions. Our study lies somewhere between these two extremes by considering a time horizon of 28 days (or 2-7 times longer than typical ones in the literature) and requiring a balance of scheduling and routing decisions in order to produce high-quality solutions.

4.4 A heuristic approach

Our literature review highlighted the challenges of PVRPs and the frequent use of heuristic methods to produce solutions in reasonable computation times. Indeed, the MILP introduced in Section 4.2 is unable to handle large real-world instances, especially those with long time horizons, which is precisely the case studied in this work. Therefore, in order to provide industry with high-quality solutions within the time frame necessary for them to be implementable in practice, this section will introduce a tailored heuristic for the MDPVRPVP.

Our heuristic for solving the MDPVRPVP is an adaptation of *Late Acceptance Hill-Climbing* (LAHC) proposed by Burke and Bykov (2017). Algorithm 6 outlines our LAHC approach. Note that the procedures detailed in lines 6-7 are the core of our algorithm. Although one could opt for another metaheuristic framework, we chose LAHC since it is a simple-to-implement yet powerful framework that has been successfully used for a range of scheduling and vehicle routing problems. Besides the standard list length parameter given by L_s, our implementation also requires a maximum number of consecutive rejections (M_{cr}) and maximum running time (M_{time}), both of which are used to halt the algorithm's execution. In addition, ω corresponds to the maximum number of customers to be removed from the scheduling in each iteration.

Algorithm 6: Late Acceptance Hill-Climbing for the MDPVRPVP.

```
1  Input: $L_s$, $M_{\text{cr}}$, $M_{\text{time}}$, $\omega$
2  $s, s^* \leftarrow$ initialSolution()
3  $F[k] \leftarrow +\infty,\ k = 1, \ldots, L_s$
4  cr, time $\leftarrow 0$
5  while cr $< M_{\text{cr}}$ and time $< M_{time}$ do
6      $s'' \leftarrow$ reassignCustomers($s$, $\omega$)
7      $s' \leftarrow$ routingPhase(s")
8      if $f(s') < f(s)$ then cr $\leftarrow 0$
9      else cr $\leftarrow$ cr $+ 1$
10     if $f(s') \leq f(s)$ or $f(s') < F[k]$ then
11         $s \leftarrow s'$
12         if $f(s) < f(s^*)$ then $s^* \leftarrow s$
13     if $f(s) < F[k]$ then $F[k] \leftarrow f(s)$
14     $k \leftarrow$ (k + 1) mod $L_s$
15 return $s^*$
```

Given all the required parameters, Algorithm 6 begins by constructing an initial solution based on the procedure presented in Section 4.4.1, followed by initializing LAHC's fitness array F and counters (lines 2–4). The main loop of the algorithm (lines 7–14) is iterated over until one of the counters, either *cr* or *time*, reaches its limit. In each iteration, a new solution s' is generated (lines 6–7) by first changing the visit pattern of customers using the procedure outlined in Section 4.4.2. Routes of each changed period undergo a routing phase, as described in Section 4.4.3. Depending on the cost of the solution, the number of consecutive rejections may be reset or incremented (lines 8–9). Lines 10–14 update the current solution s, best solution s^* and the fitness array F in accordance with the original LAHC strategy introduced by Burke and Bykov (2017). Finally, the best solution s^* generated over all iterations is returned (line 15).

A limit concerning the total running time is given as input for the proposed algorithm, the LAHC procedure then runs for 60% of this total time. Given the best solution returned by LAHC, the remaining 40% is dedicated to further optimizing the routes in each period. The routing phase is invoked for each period for an equal amount of time. This phase is necessary since the routing phase invoked within LAHC only runs for a very limited number of iterations.

4.4.1 Initial solution

An initial solution is constructed by first assigning customers to periods while respecting their visit frequencies. For each FP customer, one visit pattern is randomly selected and the customer is assigned to the corresponding periods. By contrast, VP customers are assigned to one period at a time. Given a VP customer i and the maximum number of elapsed periods without visit m_i, i is assigned to period $b_i + m_i$, where b_i corresponds to the last period in which i was visited. The assignment for each customer stops when $b_i + m_i$ is greater than the time horizon. If the selected period is invalid due to time windows, the preceding period is selected. The assignment of VP customers can be considered greedy, as visits to the same VP customer are always assigned as far apart as possible for the initial solution. After completing the assignment of customers to periods, a routing method is invoked for each period in which routes are constructed and optimized as described in Section 4.4.3.

4.4.2 Customer assignment strategies

In periodic VRPs it is possible to improve the overall routing costs of the time horizon by changing the assignment of customers to periods. To create neighbor solutions, customer visits are removed from either some or all periods and reinserted into different ones. Given a current solution, the assignment strategies remove a number rem_c of customers (either VP or FP), where rem_c is selected at random from the uniform distribution $U[1, \omega]$. The chance of selecting a specific type of customer to be removed is directly proportional to the quantity of that customer type in the instance. For example, an instance in which 30% of the customers are FP will have such customers removed 30% of the time.

When removing FP customers, we select them at random until rem_c customers have been chosen. Each selected customer i is removed from all their corresponding periods and reassigned to a randomly selected pattern among those available to them in set Φ_i.

By contrast, instead of having predefined visit patterns, VP customers are more flexible with regard to period assignment. Although several visit patterns can be constructed for a VP customer, the most intuitive option is to maximize the length of time between visits. This is in fact the approach adopted by our industry partner, which considers VP customers as though they were FP customers with a single visit pattern. Moreover, they also assume each customer as always being served by their nearest depot. To mimic what is currently enacted in practice, we refer to our first customer assignment strategy as *Base* strategy. This strategy fixes customers to their closest depot and schedule VP customer visits as far apart as possible.

In addition to this base strategy, we also propose other strategies in order to identify potential improvements concerning solution cost, resource usage and the impact of different visit patterns for VP customers. Our strategies vary with respect to three main aspects. The first concerns how VP customer visits are scheduled, which is either as late as possible or using the VP customer neighborhoods described below. The second aspect concerns how to assign customers to depots. The assignment can be fixed, where a given customer is assigned to the closest depot in every period, or flexible, where the assignment is decided by the routing algorithm in each period (Section 4.4.3). The third and final aspect concerns fleet size minimization. Fleet size is defined as $\varphi = \max_{p \in P} v_p$, where v_p is the number of vehicles required in period $p \in P$. To guide the search towards solutions with reduced φ, a sufficiently large penalty is multiplied by φ and added to the objective function. Considering *Base* as the standard strategy, the other five strategies can be summarized as follows:

- *LFlexD*: extends *Base* by allowing flexible assignment of customers to depots.
- *LMinV*: extends *LFlexD* by minimizing the fleet size.
- *FFixD*: fixes assignment of customers to depots and employs VP customer neighborhoods to generate different visit patterns.
- *FFlexD*: combines the neighborhoods used in *FFixD* with flexible assignment of customers to depots.
- *FMinV*: extends *FFlexD* by minimizing the fleet size.

These strategies may be subdivided into two groups with respect to how they handle VP customers: Base, LFlexD and LMinV are referred to as *Latest* *strategies* while FFixD, FFlexD and FMinV can be categorized as *Flexible strategies*.

To create a new sequence of visits and exploit the flexibility of VP customers, we introduce two *Variable-pattern customer neighborhoods*. The first removes a VP customer from all periods at once. The reassignment starts at the beginning of the time horizon and visits are inserted one at a time. The second neighborhood removes 5% to 60% of the customer's visits. This process potentially creates lengthy gaps without a visit, which results in an infeasible solution. To repair the solution, the method proceeds through the time horizon one period at a time in reverse, starting from the final period until the first. Despite the different order when repairing solutions (either forwards or backwards), both methods follow the same rationale to insert visits.

Let us consider for customer $i \in V_D$, l_i their last visited period, m_i the maximum number of elapsed periods between subsequent visits and $l_i + m_i$ the latest possible period to visit customer i in the case of forwards insertion (and $l_i - m_i$ in the case of backwards insertion). Most visits are assigned to the latest possible period, $l_i + m_i$. However, in order to diversify visit patterns, a visit may be assigned to a randomly selected period between $l_i + \lceil \frac{m_i}{2} \rceil$ and $l_i + m_i$ with a probability $r\%$. In other words: the number of periods between subsequent visits may be as low as half of the maximum permitted. Considering the forward insertion method, Figure 4.2 illustrates potential periods to insert a visit during the first 10 days of a given time horizon, where $l_i = 1$ and $m_i = 9$. Preliminary experiments revealed that inserting visits before period $l_i + \lceil \frac{m_i}{2} \rceil$ (2-5) results in slower convergence of the algorithm, while considering only one day before $(l_i + m_i - 1)$ is insufficient to achieve good solutions.

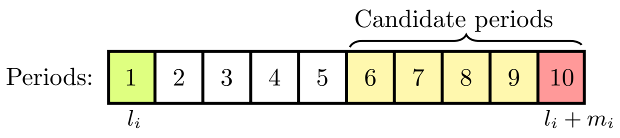


Figure 4.2: Illustration of candidate periods (6-10) which are considered by the heuristic for insertion of a new visit.

4.4.3 Routing phase

In a single iteration of LAHC, multiple periods undergo changes as customers are removed from certain periods and reinserted into others. In order to properly guide the search and evaluate the benefit of reassigning customers it is crucial to use a routing method which is able to produce good quality solutions in a short amount of time. The routing phase is then responsible for optimizing the routes of a single period and is invoked every time customers are removed/inserted from/into a particular period.

From the many options that are available, we chose to implement a version of SISRs which was proposed by Christiaens and Vanden Berghe (2020) and has been shown to generate competitive solutions for a wide range of VRP variants, including the MDVRPTW. This fast and simple-to-implement method comprises of a single ruin operator which removes customers in different yet geographically proximate routes, thereby creating a range of efficient options for reinsertion. A best insertion method is employed as the single recreate operator which uses rank-based probabilities to avoid always inserting customers into their best position. We have used the parameters for SISRs defined by Christiaens and Vanden Berghe (2020) as it has shown good results across a range of problem sizes. SISRs is executed for a fixed number of iterations and returns an equal quality or improved solution.

4.5 Computational study

This section evaluates the performance of LAHC and also derives insights concerning MDPVRPVP characteristics. 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. LAHC was implemented in C++ and compiled using gcc 7.4.0 and option -O3. Executions were limited to a single thread. All instances, solutions and results have been made publicly available[2]. All parameters required by LAHC were calibrated based on the results of empirical tests and set as follows: list length L_s was set to $150000/(|V_F|+|V_D|)$, maximum number of customers to be removed $\omega = 5$, visits are assigned to a random period (instead of the latest) with probability of $r = 2\%$, the first and second VP customer neighborhood are used 40% and 60% of the time and the maximum number of consecutive solution rejections was set to $M_{\text{cr}} = 500$. Throughout this section we report the gaps between one of the 5 strategies we introduced (A) and either our base method or a method from the literature (B). Gaps are then calculated as $\frac{f(A)-f(B)}{f(B)} \times 100$.

[2]`http://dx.doi.org/10.17632/s47bj3x9gm.1`

4.5.1 Literature instances

Given the absence of instances for the MDPVRPVP, we evaluate the quality of our heuristic using instances for the most closely related problem: the MDPVRPTW. More specifically, we consider the 40 instances proposed by Vidal et al. (2014) which are available at VRP-REP (Mendoza et al., 2014). All results for this dataset can be found in the supplementary material.

Some changes to LAHC are needed since MDPVRPTW considers there to be a maximum number of vehicles available at each depot. Thus, LAHC was adapted in such a way that each vehicle above the limit incurs a high penalty in order to incorporate this additional constraint. Although simple, this mechanism proved sufficient to comply with the new constraints as all produced solutions respect the imposed limit. All experiments were conducted with 10 runs per instance and a time limit of 30 minutes, which is a reasonable time limit that has also been used by other multi-period scheduling methods from the literature. One should be cautious when comparing the runtime between our method and that proposed by Vidal et al. (2014), since according to Passmark (accessed April 26, 2022) the hardware we employed is approximately four times faster.

Results show an average gap of -0.14% when compared to average solution values reported by Vidal et al. (2014). When considering best solutions, LAHC found 22 improved solutions for the same benchmark set. These results indicate the competitiveness of LAHC against established methods in the literature. Moreover, the Wilcoxon signed-rank test was performed and no significant statistical difference was observed regarding the results obtained by LAHC and those obtained by Vidal et al. (2014).

In addition to testing our LAHC implementation, we have also run experiments with the MILP formulation using Gurobi 9.2 solver. For these experiments, a time limit of 24h with 20 parallel threads is enforced and the solution produced by LAHC is given to MILP as an initial solution. Under this configuration, MILP was able to find gaps of 2-10% for the smallest instances of the Vidal et al. (2014) set, namely instances with 48 customers and 4 periods (`pr01` and `pr01b`) and with 96 customers and 4 periods (`pr02` and `pr02b`). For a simple, compact model this is quite promising and implies that with more effort, optimality of these instances could likely be proved.

4.5.2 New instance set

In order to stimulate further research regarding the MDPVRPVP, new instances were generated based on real-world data. The company which inspired this work provided us with a set of depots and customers, maximum route duration and vehicle capacity. Realistic travel times between all pairs of locations are calculated using the *Open Source Routing Machine* (Luxen and Vetter, 2011) which computes shortest paths using OpenStreetMap (2022) data. To calculate fuel consumption, the cost per minute of driving time γ was set to 0.25 while for wage cost the cost per minute of route duration α was set to 0.30. Historical data was also provided from which we were able to derive average customer demand and visit frequencies.

Based on the provided data, we generated twelve instances. Each customer is associated with a demand, service time, a time window per day and visit frequency for the given time horizon. Time windows are uniformly selected from morning, afternoon, evening and entire day. For FP customers, visit frequencies range from 1-5 times per week where the distribution follows 40%, 40%, 10%, 5% and 5%, respectively. By contrast, visit frequencies of VP customers are uniformly distributed and range anywhere from 3 to 14 days. Instances contain 300, 700 or 1500 customers which must be served during a time horizon of 4 weeks. Another important instance feature is the proportion of VP and FP customers. VP customers may correspond to 30, 50, 70 or 100% of the total number of customers. Instances are named in the format I_C_D, where C corresponds to the total number of customers and D to the proportion of VP customers, thus I_1500_70 has 1500 customers where 70% of them are VP customers. The new instances as well as a solution validator are available in the supplementary material.

4.5.3 Result discussion

Average results for the new instance set are provided in Table 4.1. Given the stochasticity of the method, each line associated with a different customer assignment strategy corresponds to the average value of ten independent runs over all 12 instances. Table 4.1 provides the average solution cost (Avg.) which is a sum of both the fuel (Fuel) and wage (Wage) costs, the maximum number of vehicles used in a single period (Veh) and the total number of visits to VP customers (Visits). Note that the number of visits to FP customers is always the same. Table 4.1 gives us the costs' order of magnitude and will be used as a reference point for the discussion in this section. Complete results can be found in the supplementary material.

The first thing we observed from results was the dominance of wage cost over the fuel consumption cost. Since wage is the dominant cost component, solutions with shorter route durations and less idle waiting times at customer locations are preferred over those with shorter distance traveled. In fact, if wage costs are

Table 4.1: Results for different customer assignment strategies.

Strategy	Avg.	Fuel	Wage	Veh	Visits
Base	73491.95	18843.32	54648.63	32.65	2306.58
LFlexD	73130.47	18692.58	54437.89	32.28	2306.58
LMinV	73287.47	18761.56	54525.92	31.39	2306.58
FFixD	73019.96	18603.88	54416.08	32.18	2316.07
FFlexD	72640.85	18449.42	54191.43	31.70	2315.53
FMinV	73028.00	18617.01	54410.99	30.45	2316.77

ignored, which is the case in most models from the literature, the overall cost increases by almost 5% on average despite the average 2% of fuel cost savings. This difference may increase even more when one takes into account that two workers (a driver and a security worker) may potentially be assigned to a single vehicle, thus significantly increasing wage costs.

The number of visits made to VP customers is also worth exploring. By always inserting VP customer visits as far apart as possible, *Latest* strategies all have the same number of visits within a given time horizon. By contrast, *Flexible* strategies can visit a VP customer sooner. This potentially leads to a greater number of visits considering the same time horizon. Figure 4.3 demonstrates how many more visits are scheduled for *Flexible* strategies. Indeed, solutions may have up to 27 more visits than the *Latest* strategies.

Given that the number of visits increases, one might logically expect higher costs for solutions produced by the *Flexible* strategies. However, while increasing the number of visits in a single day may increase the routing cost of that day, in a multi-period time horizon more visits not always lead to higher overall costs. Figure 4.4 visualizes the average solution cost and shows the percentage gap from the Base strategy to all others. The most significant improvement can be observed for the FFlexD strategy. Despite the much larger search space, this strategy obtained the best solution cost for all instances, reaching gaps as low as -4% and average gap of -1.62%. Even when compared against LFlexD, the best *Latest* strategy, *Flexible* strategies are clearly advantageous as all three outperform LFlexD. These results demonstrate that despite the additional complexity our proposed algorithm is able to explore the search space and find the right trade-off between inserting more visits and reducing overall routing costs.

Figure 4.5 shows the percentage gap concerning the fleet size needed to operate over the time horizon compared to the Base strategy. This graph readily demonstrates that both strategies to minimize fleet size (LMinV and FMinV) have a positive effect, reducing the number of vehicles required in any given

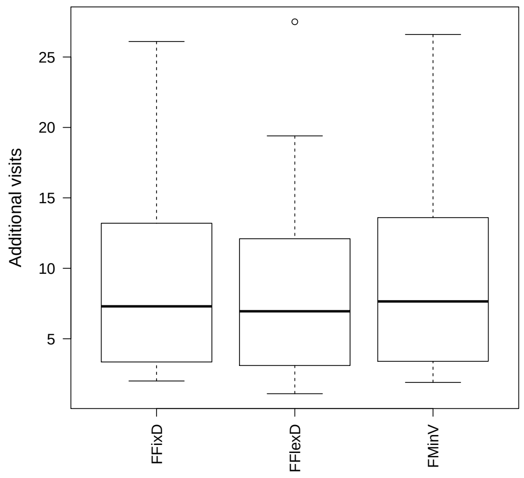


Figure 4.3: Additional visits to VP customers.

period by up to 15%. In absolute terms, these gaps correspond to 1-3 fewer vehicles in the fleet needed for the whole time horizon compared to the Base strategy. This reduction in fleet size can have significant long-term benefits with respect to fuel, personnel and maintenance costs.

Despite the fact that all strategies provide a reduction in fleet size, it is clear that the flexibility of VP customers allows for a greater decrease of fleet size. Meanwhile, even when fleet minimization is not considered a priority, *Flexible* strategies typically lead to solutions with fewer vehicles on average, particularly when depot assignment is not fixed (FFlexD). This might seem like a somewhat counterintuitive result. We know from Figure 4.3 that the flexibility of VP customers increases the number of visits in the time horizon and indeed when minimizing the number of vehicles this increase is even larger. Thus, one would expect that increasing the number of visits could require more vehicles in the fleet. However, *Flexible* strategies enable a better distribution of visits among periods thereby leading to fleet size reduction in the busiest periods. Hence, allowing VP customers can be seen as a way of improving resource utilization and planning.

Additionally, we wish to call attention to the fact that these real-world-based instances lead to solutions with significantly more visits than instances in the literature. When compared to MDPVRPTW instances (Vidal et al., 2014), for example, solutions for our new set required 2-4 times more visits per period.

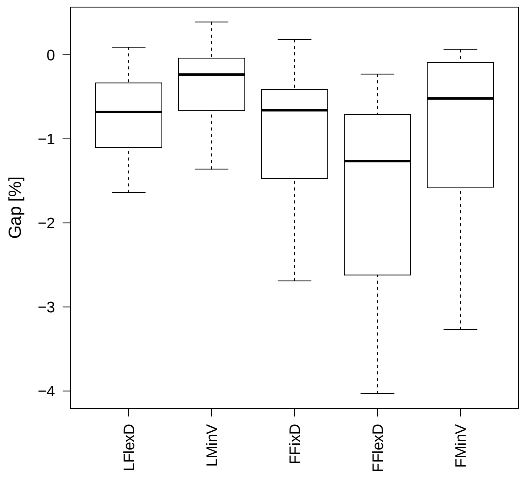


Figure 4.4: Gap against Base strategy with respect to average solution cost.

Essentially, this means that the VRPs solved for each period were 2-4 times larger, thereby creating both methodological and implementation challenges for solving these new benchmark instances.

4.6 Conclusion

The combination of fixed- and variable-pattern customers had not previously been thoroughly studied in the PVRP literature. In this chapter we not only bridged this gap, but also took into account additional constraints that are applicable to real-world scenarios such as multiple depots, time windows, maximum route duration, lengthy time horizons and an objective function that depends on the amount of work done in addition to distance traveled.

We have demonstrated that in certain contexts variable-pattern customers can bring cost reduction by allowing great flexibility and improved resource utilization. These results were achieved in a rather counterintuitive fashion, namely by *increasing* the number of visits to customers over the time horizon, albeit not by a large margin. However, incorporating these variable-pattern customers introduced additional computational complexity and requires specialized local search moves in order to produce high-quality solutions.

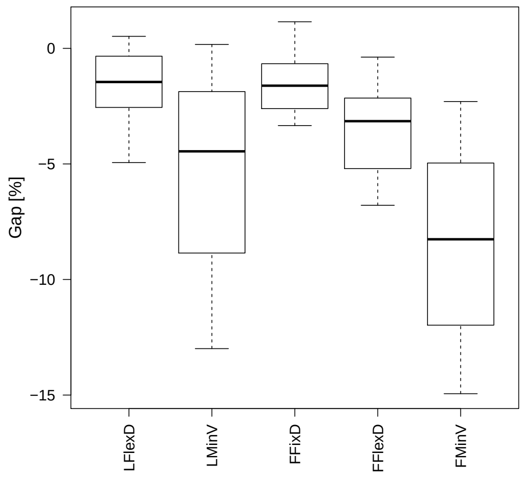


Figure 4.5: Gap against Base strategy with respect to number of vehicles.

While including variable-pattern customers required special local search moves, the remainder of the proposed metaheuristic was implemented using well-known state-of-the-art elements from the PVRP and VRP literature. This allowed the implementation of a simple, yet highly effective, method that could not only solve the real case at hand but also remain competitive against methods from the literature for standard benchmark instances.

For future research, it is worth exploring exact algorithms for this kind of problem in order to determine precisely how many additional visits should scheduled to variable-pattern customers in optimal solutions. Additionally, it might be worth exploring whether cost savings can also be obtained by creating multiple patterns for variable-pattern customers and treating them as fixed-pattern customers. One can expect that only a subset of the feasible patterns can be created in order to avoid high computational overhead, in which case there is also the question concerning how to select a suitable subset of patterns beforehand.

Chapter 5

The team routing and scheduling problem

This chapter integrates the multi-day scheduling problem presented in the previous chapter with a VRP that contains additional real-world features. A team routing and scheduling problem is addressed where tasks associated with time windows, product demands, due dates and required skills must be completed during a time horizon of multiple days. Technicians, each of whom is specialized in a certain set of skills, must be grouped together into teams and assigned to routes in order to perform the required tasks. In this chapter we investigate whether it is possible to accelerate the assignment of technicians using preprocessing methods and whether we can bound the solution space without a significant loss of quality. Like the other chapters, this chapter is grounded in real operational challenges faced by a Belgian company.

For this chapter, Pieter Smet helped guide the design of the heuristic while supervising the research together with Hatice Çalık, Marco Carvalho and Greet Vanden Berghe.

5.1 Introduction

Grouping workers with different skill sets into teams is necessary in a range of business environments in order to serve tasks at different locations. For example, in the construction industry, landscaping companies and home healthcare service providers, teams are often formed and dispatched to different locations.

These examples can be considered resource-constrained problems, where there exist compatibility constraints between workers and tasks. In these problem environments each worker is qualified in one or more skills while each task has certain skill requirements that must be met.

Most previous research associates tasks with certain *skill coverage* requirements. Skill coverage implies that there is a set of skills required by each task and any team of workers that can cover all of those skill requirements is considered feasible, regardless of its size. For example, if there is a task that requires skills A and B, it could potentially be serviced by a single worker qualified in both those skills. However, there are many cases where skill coverage is insufficient and there is instead a need for a minimum number of individuals each with a specific profile.

Take for example the task of placing steel beams during the construction of a building. This complex task requires a worker operating a crane to lift the beam and position it into place. Meanwhile, workers on the ground must bolt the beam into place, an activity which may also require the assistance of an engineer who will read blueprints and indicate where exactly to place the beam and bolts. Finally, another worker may also be needed who has the knowledge to operate the tools required to tighten the bolts. In many scenarios such as this, skill coverage alone is not sufficient since it is impossible to control the number of workers deployed for a task and the precise set of skills associated with each worker.

To address such problems, we will investigate a resource-constrained problem and model the minimal skill requirements for tasks in terms of *profile coverage* requirements. A profile is simply a set of skills which a single worker must be qualified in. The profile coverage approach then considers each task to be characterized by one or more of these profiles, where each profile must be associated with a different worker.

Figure 5.1 provides a very simple example to help highlight the key difference between skill coverage and profile coverage. Technicians are proficient in different skills represented by the hammer, axe and paintbrush symbols. Task A is modeled with skill coverage requirements, meaning that any set of technicians which in combination have all the skills needed to serve the task are considered feasible. Thus, either Technician 2 in isolation or Technicians 1 and 3 in combination can serve task A. On the other hand, task B is modeled with profile coverage requirements and requires exactly two technicians, one with a profile that covers at least the hammer skill and another whose profile covers at least the axe skill. Task B can therefore be served by three possible teams of technicians: (1, 2), (1,3) or (2, 3). Note that unlike for task A, we cannot simply assign Technician 2 in isolation. Despite the fact Technician 2 covers all

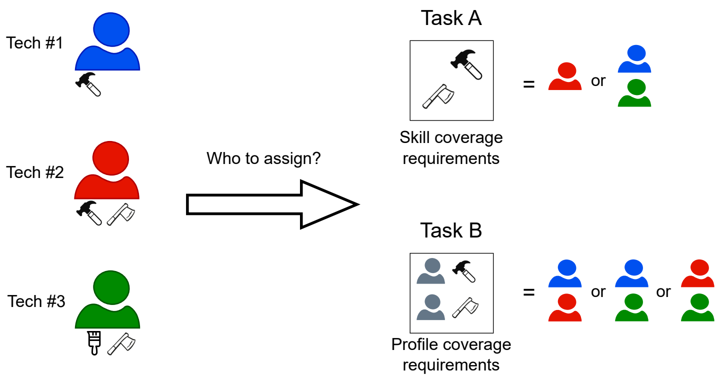


Figure 5.1: Assigning tasks to technicians based on skill coverage and profile coverage requirements.

of the skills needed, task B still requires another technician to be present.

The resource-constrained problem addressed in this chapter was inspired by a company operating in Belgium that provides scheduling solutions for multiple service provider companies (SPC). Each of these SPCs employ many technicians specialized in different skills who must perform services at customer locations around the country. These services include preventive and corrective maintenance, failure diagnoses, equipment replacement, as well as (de)installations. The SPCs share the common goals of reducing their costs and improving service levels, which involves servicing customers on time and quickly responding to emergency tasks.

Our aim is to develop a solution approach for optimizing the schedule and route of technicians over multiple days, where the tasks involved are defined in terms of profile coverage requirements. We will term this problem the *Multi-period Team Routing and Scheduling Problem* (MTRSP), which we will fully define in Section 5.2. The operational environment of the SPCs associated with the problem is complex and includes multiple depots, a limited and heterogeneous fleet of vehicles, technician and task availability, and technician break requirements. The objective is to minimize total costs, which include travel costs, costs associated with delayed or unserved tasks, and personnel cots.

Given the multiple interacting decision levels present in the MTRSP – assigning tasks to days, creating daily routes, grouping technicians into daily teams and assigning them to routes – manual planners typically adopt simplifications to help them deal with the complexity of the problem. For example, they may opt to schedule tasks as soon as possible to avoid the difficulty of determining what the best date and time would actually be. They also often use simple rules of thumb to generate feasible teams for routes, which may result in overqualified teams and higher personnel costs.

To tackle the MTRSP and develop a suitable algorithm, several questions must be addressed. Can bounding methods be used to reduce the search space of the problem without impairing solution quality? Given a set of available technicians, is there a way to quickly determine the best team for a given route? Additionally, the algorithm developed in this chapter should be capable of providing insights concerning various management policies. For example, we will investigate the cost-effectiveness of different contractual arrangements, such as hiring technicians per hour versus a fixed contract.

The remainder of this chapter is structured as follows. A detailed problem description for the MTRSP is given in Section 5.2. An overview of related literature is provided in Section 5.3, which compares published problems to the MTRSP. A solution approach is proposed to tackle instances of realistic size in Section 5.4, while in Section 5.5 we report the results of various computational experiments. These results not only concern solution quality, but also the utilization rate of teams in scenarios featuring a high and low demand concerning technician profiles. Finally, Section 5.6 concludes the chapter and outlines directions for future research.

5.2 Problem definition

The MTRSP considers a time horizon $\mathcal{P} = \{1, ..., P\}$ of P days during which a set $\mathcal{N}$ of tasks must be served. Various resources are needed to service tasks: a set $\mathcal{T}$ of technicians and a set $\mathcal{V}$ of vehicles. Vehicles are initially located at a set of depots $\mathcal{D}$. A complete graph is given where nodes represent the locations of tasks, technician homes and depots.

We consider a limited fleet of heterogeneous vehicles where each vehicle $v \in \mathcal{V}$ is associated with a capacity t_v for accommodating technicians. Each vehicle is linked to a depot $d \in \mathcal{D}$ and has a fuel consumption cost c_v per kilometer traveled. A travel time τ_{mn} and travel distance δ_{mn} are associated with each arc (m, n) in the complete graph. The travel cost of vehicle v is calculated as $c_v \times \delta_{mn}$. Since vehicles may range from small vans to large trucks, at least one technician assigned to the vehicle must have the skill necessary to drive it.

There is a set $\mathcal{L}$ of skills and each technician $t \in \mathcal{T}$ is proficient in a set of skills

$\overline{L}_t \subset \mathcal{L}$, which defines their profile. Each technician t is associated with a cost c_t per normal working hour and a cost c_t^o per hour of overtime per day. These costs are bounded by the maximum working hours and maximum overtime. To serve more demanding tasks, technicians may need to be grouped into teams which stay together during an entire day. Thus, the maximum size of a team is bounded by the maximum capacity of the largest available vehicle. Technicians start/end their route at their home unless they are part of a team, in which case they all meet at a designated depot, beginning and ending their route there. Starting from a depot location may incur a cost c_t^{depo}, which is related to the distance from the technician's home. To prevent technicians from being assigned to certain depots, they can be associated with a set of prohibited depots. While a technician can be part of different teams on different days, when assigned to a team they must remain with it for the duration of the day in question.

Tasks are the final component of this problem formulation. To be completed, a task i requires a set of technician profiles $\mathcal{K}_i$. Profile requirements are given in such way that each $K \in \mathcal{K}_i$ corresponds to a set of skills. To fulfill a task i, a different technician t must be assigned to each profile $K \in \mathcal{K}_i$ and must have all required skills in that profile K ($\overline{L}_t \supseteq K$). Consequently, task i will always be served by a specific number $|\mathcal{K}_i|$ of technicians. In addition to skill requirements, each task $i \in \mathcal{N}$ is associated with a duration , a set π_i of prerequisite tasks that must be completed before i, a due date p_i (which may be after the last day of the time horizon), and a delay cost c_i incurred for each day after the due date until the actual service date of task i. If task i is not served, the following penalty is applied: $max\{0, (|P| - p_i) \times c_i\}$.

Throughout the time horizon, all technicians, vehicles and tasks are subject to unavailable days $\overline{P}_{t,v,i} \subset \mathcal{P}$. This is because technicians may take sick days or holidays, vehicles may require maintenance, while it is common for a task's associated location to not be accessible on certain days. This introduces another level of scheduling difficulty. Indeed, although task availability remains the most constraining restriction, it is not the only one in effect. Moreover, tasks and technicians have a restricted time window during which they are available on any given day. The earliest and latest starting time for task i on day p is given by $[a_{ip}, b_{ip}]$, while $[a_{tp}, b_{tp}]$ is the time window during which technician t is available to work on day p. A short break with a defined duration, which simulates a lunch break, must be scheduled for technicians within a time interval given by $[a_l, b_l]$ for every route that starts before a_l and ends after b_l.

A solution for the MTRSP is given by a set of routes, each of which is associated with a vehicle, a day in the time horizon and a team capable of serving all of its tasks. The objective is to minimize the sum of (i) the travel costs associated with the fuel consumption c_v of each vehicle, (ii) the delay cost c_i and penalty for unserved tasks, and (iii) the personnel costs given by the cost c_t^{depo} of starting a route from a given depot and the cost of hours worked. The cost of a technician t's work on any given day p is determined by the formula $(c_t \cdot w_{t,p}) + (c_t \cdot (w_t^{max} - w_{t,p})) \cdot \mu + (c_t^o \cdot w_{t,p}^o)$, where w_t^{max} is the maximum working hours of technician t, and where $w_{t,p}$ and $w_{t,p}^o$ corresponds to the total working time and overtime, respectively. The binary parameter μ facilitates different management policies. When $\mu = 0$, technicians are paid only for the hours they have worked. However, when $\mu = 1$, technicians are paid for the duration of their entire shift (given by w^{max} hours), regardless of the number of hours they actually work. This provides flexibility in determining a technician's pay based on the company's policies.

5.3 Related literature

The MTRSP combines multiple decision levels and incorporates features from both personnel scheduling and vehicle routing problems. This section will provide an overview of the most closely related problems and classify them according to the extent to which they incorporate the set of decisions encountered in the MTRSP. Table 5.1 details previous research that had a direct impact on this chapter, highlighting what exactly each paper incorporated into its problem formulation with respect to routing as well as the scheduling of technicians and tasks. The general observations made throughout this section draw on the papers listed in Table 5.1. For a more thorough review of related problems, we refer interested readers to the surveys by Castillo-Salazar et al. (2016) and Paraskevopoulos et al. (2017) that focus on resource-constrained routing and scheduling problems.

Most of the papers in Table 5.1 consider a VRPTW with a single depot and a homogeneous fleet of vehicles bounded by the number of technicians. However, in our problem we consider a more general VRPTW with multiple depots, a limited number of heterogeneous vehicles and multiple starting points (either depots or a technician's home). Hashimoto et al. (2011) and Cordeau et al. (2010) study a technician and task scheduling problem where the distances between tasks are negligible and therefore no routing is needed.

The papers in Table 5.1 consider either a single day on which all tasks must be scheduled or they feature a time horizon spanning multiple days. A multi-day time horizon introduces an additional decision level to the problem and greatly increases the size of the search space. A significant number of papers also include the possibility of outsourcing tasks, which may be constrained by a

Table 5.1: Overview of related team routing and scheduling problems

	MTRSP	Zamorano and Stolletz (2017)	Tricoire et al. (2013)	Pillac et al. (2013)	Barrera et al. (2012)	Kovacs et al. (2012)	Hashimoto et al. (2011)	Cordeau et al. (2010)	Zäpfel and Bögl (2008)	Bostel et al. (2008)
Routing										
Start	D/TH	D	D/TH	TH	D	D	NA	NA	D	TH
Heterogeneous fleet	✓	-	-	-	-	-	NA	NA	✓	-
Multiple depots	✓	-	✓	-	-	-	NA	NA	-	-
Scheduling										
Multiple periods	✓	✓	✓	-	✓	-	✓	✓	✓	✓
Different time windows	✓	✓	✓	-	-	-	✓*	✓*	-	✓*
Outsourcing	✓	-	-	-	-	✓	✓	✓	✓	✓
Maximum hours	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
Technicians										
Teams	✓	✓**	-	-	-	✓	✓	✓	-	-
Breaks	✓	-	✓	-	-	✓	-	-	✓	✓
Overtime	✓	✓	-	-	-	✓	-	-	✓	✓
Multiple skills	✓	✓	-	✓	-	✓	✓	✓	-	-
Proficiency level	-	✓	-	-	-	✓	✓	✓	-	-
Tasks										
Prerequisite tasks	✓	-	-	-	-	-	✓	✓	-	-
Time windows	✓	✓	✓	✓	-	✓	-	-	-	✓
Required skills	✓	✓	-	✓	-	✓	✓	✓	-	-
Profile requirements	✓	-	-	-	-	-	-	-	-	-
Method	H	E	E & H	H	E & H	H	H	H	HY	H

D: depot. TH: technician's home. E: exact approach. H: heuristic. HY: hybrid approach.
NA: not applicable. *: different time windows per technician (and not task) per day.
**: teams of fixed size.

certain budget. Finally, all of the papers listed limit technicians to a maximum working time per day.

Technicians may work alone or be grouped together into teams. Papers that do not involve teams assume that every task can be served by a single technician, which makes the problem much easier to solve. Only Zamorano and Stolletz (2017) and Kovacs et al. (2012) consider teams and routing in combination. However, Zamorano and Stolletz (2017) assume that every team is composed of exactly τ technicians. Although their model allows for different values of τ, they only considered $\tau = 2$ in their experiments. While papers are divided in terms of whether or not they schedule breaks and allow overtime for technicians, this chapter and all previous research share the property that a technician is only assigned to a single route/team per day. All of the papers listed in Table 5.1 that consider multiple periods allow the assignment of a technician to different teams on different days.

Some papers assume that technicians have an identical set of skills, meaning each technician can perform every task. Tricoire et al. (2013) and Barrera et al. (2012) assume technicians have the same skills and their assignment is constrained only by availability. Considering technicians with different skill sets greatly increases the difficulty of assigning technicians to tasks. Some papers also consider a proficiency level for each skill such as beginner, intermediate or expert. Although we do not explicitly model different proficiency levels for skills in our problem formulation, these could easily be incorporated in the form of a unique skill per level. In contrast to all existing papers in the literature, the MTRSP takes into account that certain technicians are not permitted to work from certain depots. These conflicts further constrain the assignment of technicians to routes.

Regarding task restrictions, although most previous papers have considered time windows, only those that do not consider routes have taken into account precedence constraints between tasks. These constraints refer to when a task can only be served after another has already finished. Additionally, only papers that consider multiple skills per technician have also taken into account skill requirements for serving tasks. De Bruecker et al. (2015) present a literature review of workforce planning problems with special emphasis on the impact that different skill types have on the problem's formulation, the complex dynamics of incorporating such skills into the problem, as well as the different classes of skills.

The problem we address assumes that a certain task requires a minimum number of technicians each of whom is qualified in a particular set of skills. Although Pillac et al. (2013) do not consider the formation of teams, they follow a similar approach to skill requirements where a task requires multiple skills and only a single technician who covers all skills can service the task. They do not consider solutions in which all team members in combination cover all skills. Li et al. (2005) and Franses and Post (2003) use profile coverage to model the compatibility constraints between tasks and teams of workers for slightly different scenarios: manpower allocation and personnel scheduling. They also consider tasks which require a specific number of workers with certain profiles.

Aside from these exceptions, most papers in Table 5.1 have followed the skill requirement approach presented by Cordeau et al. (2010). They consider each technician to be specialized in different skills with different proficiency levels, while each task is associated with a certain skill coverage requirement. Tasks are defined by a skill coverage matrix where each matrix element defines how many technicians with a certain skill at a certain level are required. The objective is then to find the lowest cost team which together covers all required skills, irrespective of the team's size.

5.3.1 Skill requirement approaches

Let us introduce an example to help highlight the main differences between the two approaches to modeling skill requirements for tasks: skill coverage and profile coverage requirements. Given a set of technicians (T1-T4) and their skills, Table 5.2 provides the teams that can be created for tasks I1-I6 under these two skill requirement approaches. The requirements of tasks I1 and I2 follow the format of Cordeau et al. (2010), which uses a list of tuples where each corresponds to the skill required and the number of technicians who must be qualified in that skill. In order to keep the example as simple as possible we have omitted the proficiency levels of skills. For example, task I1 requires one skill A and one skill B. These two skills could be covered by a single technician who is qualified in both skills or by two technicians where each covers one. Meanwhile, tasks I3-I6 are given in the format we introduce in this chapter, which is a list of profiles. Each profile corresponds to a set of skills that must be served by a unique technician. For instance, task I3 requires one technician with skill A and another technician with skill B. By contrast, task I4 requires a single technician with skills A and B.

Table 5.2: Task personnel demand examples and possible teams

Technician skills	Task skill requirements	Possible teams
T1: A, B, C T2: B, D T3: A, C T4: A, D	Skill coverage requirement: [(skill, #tech)]	
	I1: [(A,1), (B,1)]	(T1), (T2, T3), (T2, T4)
	I2: [(A,2), (D,1)]	(T1, T4), (T3, T4), (T1, T2, T3)
	Profile requirement: [profile]	
	I3: [[A], [B]]	(T1, T2), (T1, T3), (T1, T4), (T2, T3), (T2, T4)
	I4: [[A, B]]	(T1)
	I5: [[A, D], [A]]	(T1, T4), (T3, T4)
	I6: [[A], [A], [D]]	(T1, T2, T3), (T1, T2, T4), (T2, T3, T4)

Table 5.2 shows how tasks with skill coverage requirements can result in teams of different sizes. For example, task I1 could either be served by just technician T1 who has all required skills or through the combination of two technicians where each is qualified in one of the required skills. A similar situation occurs for task I2, which requires two skills A and one skill D. In this case a single technician could cover skills A and D while a second covers the remaining skill A, or a team of three could be formed where each technician covers just one of the three skills. By contrast, in our profile coverage format one must define the skill set required by each individual technician. Consequently, task I1 in the

previous format could be defined in two different ways: either I3 or I4. In I3, there must be at least two technicians, one of whom is qualified in skill A and another qualified in skill B. However, task I4 shows how we could also specify that we need just a single technician who is qualified in both skill A and skill B. In the same way, task I2 could be represented by either task I5 or I6.

Our profile coverage approach is adopted given the need of manual planners and customers to have greater control over the precise team size and skill composition of its individual members. However, a disadvantage arises when one must check the feasibility of a given team to serve a task. While Cordeau et al. (2010) can transform the skill coverage requirement of a task/team into a skill vector and check its feasibility in linear time, our format requires a one-to-one assignment of technicians to each profile required by each task. This requires much more computational effort. Indeed, one of our goals in this chapter is to be able to handle this increase in complexity in an efficient manner so that we can quickly find feasible teams capable of serving an entire route.

5.4 Algorithm

The literature review showcased how the majority of papers addressing problems similar to the MTRSP rely on heuristics to solve instances of realistic size. In this chapter, we aim to develop an algorithm which can produce high-quality technician schedules spanning a week to a few months within just a few minutes. For these reasons, we will develop a heuristic approach for solving the MTRSP.

A commonly used approach for problems with large search spaces is a ruin and recreate method. The approach we adopted in this chapter relies on ruin procedures where each iteration a few tasks are removed from the solution and later reinserted in order to produce better solutions and efficiently explore the search space. To tackle specific parts of the problem in an efficient manner we will introduce a set of tailored neighborhoods to address the optimization of routes and technician assignment. The proposed neighborhoods are embedded into an adaptation of the *Late Acceptance Hill-Climbing* (LAHC) framework introduced by Burke and Bykov (2017). We choose the LAHC framework given that it has been used extensively for both vehicle routing and scheduling problems, it is easy to implement and requires only a single parameter to be calibrated.

Algorithm 7 outlines our LAHC approach. Three parameters are required as input: the list length (L_s), maximum number of consecutive rejections (M_{cr}) and maximum run time (M_{time}). The first of these parameters controls the length of LAHC's fitness array, while the other two are used to halt the algorithm's execution. Given these parameters, the algorithm starts by producing an initial solution (line 2) followed by initializing LAHC's fitness array F and auxiliary variables which keep track of rejections and time (lines 2-4). The initial solution is created by inserting one task at a time, ordered by the earliest due date and based on the procedure that will be described in Section 5.4.2. The main loop of the algorithm (lines 5-14) is iterated over until one of the stopping criteria, either time or consecutive rejections, is reached. At each iteration, a new solution s' is created (lines 6-7) by first removing tasks with the neighborhoods that will be introduced in Section 5.4.3 and reinserting them with the insertion method. The number of consecutive rejections c_r is updated depending on the solution cost (lines 8-9). The current solution s, best solution $s*$, as well as fitness array F are updated (lines 10-14) in accordance with LAHC's original strategy (Burke and Bykov, 2017). Finally, the best solution $s*$ generated over all iterations is returned (line 15).

Algorithm 7: Late Acceptance Hill-Climbing for the MPTRSP.

```
1  **Input:** $L_s$, $M_{\text{cr}}$, $M_{\text{time}}$
2  $s, s^* \leftarrow$ initialSolution()
3  $F[k] \leftarrow +\infty,\ k = 1, \dots, L_s$
4  cr, time $\leftarrow 0$
5  **while** cr $< M_{\text{cr}}$ **and** time $< M_{time}$ **do**
6      $s'' \leftarrow$ ruinSolution($s$, $\omega$)
7      $s' \leftarrow$ recreateSolution(s”)
8      **if** $f(s') < f(s)$ **then** cr $\leftarrow 0$
9      **else** cr $\leftarrow$ cr $+ 1$
10     **if** $f(s') \leq f(s)$ ***or*** $f(s') < F[k]$ **then**
11         $s \leftarrow s'$
12         **if** $f(s) < f(s^*)$ **then** $s^* \leftarrow s$
13     **if** $f(s) < F[k]$ **then** $F[k] \leftarrow f(s)$
14     $k \leftarrow$ (k + 1) **mod** $L_s$
15 **return** $s^*$
```

5.4.1 Assigning teams

An important decision level of the MTRSP concerns how to assign teams of technicians to routes. The team assigned to a route is directly connected to the tasks served on that route. For this reason, changes concerning the tasks served on a route may require updating the assigned team. For example, when a task is removed one of the team's technicians could become redundant. On the other hand, the insertion of an additional task may require a new technician to be assigned. Even in cases where the existing team is sufficient to serve its modified route, a different team configuration may be more desirable in terms of cost reduction. Therefore, whenever changes are made to a route, the team assigned to it should also be reevaluated. To avoid slow algorithm performance, one must have an efficient way of generating teams that are capable of serving all of a route's tasks. In this section we will present two such methods.

Constructive assignment method

One straightforward way of generating a team capable of serving a given route is to develop a *constructive method* for assigning technicians, which builds the team as the route grows. If a new task is inserted into a given route and the team's current members cannot serve this task, the constructive method will add technicians to the route until the new task has all of its profiles satisfied.

The proposed constructive method works as follows. Consider a route r on day p, a set T_r of technicians already assigned to r who can serve all its current tasks, a set T_{free} of technicians currently not assigned to any route on day p and a new task i. The goal is to determine whether T_r is sufficient to serve task i or if any additional technicians from T_{free} are required. Note that for an initially empty route, T_r will also be empty and a team from T_{free} must be selected to serve task i. We use the Hungarian method (Kuhn, 1955) to create an assignment between technicians and required profiles for task i. Note that the Hungarian method can solve the assignment problem optimally and in polynomial time.

To use the Hungarian method we define the assignment problem of technician profiles to profile requirements as follows. Each technician $t \in T_r \cup T_{free}$ is connected to each profile requirement $K \in \mathcal{K}_i$ they can serve by an edge. The technicians in T_r are already assigned to the route and will therefore not incur extra cost if selected. These corresponding edges have cost zero. However, for a technician in T_{free} these edge costs are equal to the hourly cost of that technician. The aim is to select the cheapest set of additional technicians. If there are more technicians than profiles, dummy profiles are created to match the number of technicians and they are connected with zero cost to all technicians. This is required since the Hungarian method expects a balanced assignment, meaning the number of technicians equals the number of profile requirements.

The method is executed only once and attempts to create a feasible assignment of minimum cost between both sets of technicians ($T_r \cup T_{free}$) and the set $\mathcal{K}_i$ of profile requirements for task i. Given the resulting assignment, if only technicians from T_r are selected then the current team is sufficient to serve task i. Otherwise, technicians from T_{free} are added to the team serving r. If no feasible assignment exists, then task i will not be inserted into route r.

Although quick and simple, one downside of the constructive method is that it can potentially generate teams with obsolete members or generate teams which do not respect the vehicle's maximum capacity. Furthermore, this method cannot efficiently handle the reevaluation of a team when a task is removed from the route. This is because even though a technician might become obsolete upon the removal of a task, one must confirm that this technician is not required for any of the other tasks still remaining in the route.

Team matrix assignment method

As an alternative to the preceding constructive method, we decided to enumerate all feasible teams during a preprocessing step and determine which of them are capable of serving each task. This results in a matrix of teams by tasks. The total number of teams depends on the number of technicians and the maximum vehicle capacity, with the smallest team comprising of just a single technician and where the largest team has $\max_{v \in \mathcal{V}} t_v$ technicians. The number of teams is given by the sum of all possible teams of each size $\sum_{k=1}^{\max_{v \in \mathcal{V}} t_v} \binom{|\mathcal{T}|}{k}$.

For each task-team combination we employ the Hungarian method to ensure that there exists a feasible assignment. Once the matrix is created, one can efficiently retrieve all teams that are capable of serving a given route by retrieving all of the teams that are common to all tasks in the route. Although this approach can experience some limitations depending on the number of technicians, it is sufficiently fast for the limited team sizes in practice.

5.4.2 Task insertion method

The task insertion method attempts to insert every unserved task into its best position, considering a maximum number of days and every route within each day. For team assignments, every available team is iterated over and the one that yields the lowest cost is selected. Note that although all of the technicians associated with a team may be available, the working hours or depot restrictions of just one member can make assigning the team as a whole infeasible.

Algorithm 8 provides an overview of the task insertion method, which requires the following input: a partial solution s, the task i to be inserted and an upper bound concerning the number of feasible days being considered max_d (line 1). A day is feasible when the task can be successfully inserted into at least one of its routes. The best insertion method could attempt to insert task i into every single day and return the best option. This would result in $max_d = |\mathcal{P}|$. However, given that the solution may span multiple weeks or months, one can potentially save processing time by limiting the number of days considered for task insertion. In Algorithm 8, parameter max_d limits the number of feasible days taken into consideration.

Algorithm 8: Task insertion method.

```
1  Input: $s$, $i$, $max_d$
2  $s* \leftarrow s$, $d \leftarrow 0$
3  foreach $p \in \mathcal{P}$ do
4      if $isDayFeasible(s, i, p)$ $! = true$ then
5          go to next $p$
6      $foundFeasibleRoute \leftarrow false$
7      $T_{free} \leftarrow$ getFreeTechnicians(s, p)
8      foreach $v \in \mathcal{V}$ do
9          $Teams \leftarrow$ generateTeams($T_{free}$, i, p, v)
10         foreach $t' \in Teams$ do
11             $bool \leftarrow$ isVehicleFeasible(s, i, p, v, $t'$)
12             if $bool = true$ then
13                 $foundFeasibleRoute \leftarrow true$
14                 $s' \leftarrow$ insertInBestPosition(s, i, p, v, $t'$)
15                 if $f(s') < f(s*)$ then
16                     $s* \leftarrow s'$
17     if $foundFeasibleRoute = true$ then
18         $d \mathrel{+}= 1$
19     if $d \geq max_d$ then
20         return $s*$
21 return $s*$
```

After initializing the best solution ($s*$) and the counter for feasible days d (line 2), the method then iterates over each day in the time horizon (lines 3-20). The first check concerns whether the day in question is available for insertion (line 4-5). This entails verifying whether task i is available to be served on that day, ensuring all predecessor tasks π_i have been scheduled before task i begins. Furthermore, each task that has i as its predecessor must be scheduled after i finishes or else it remains unscheduled.

For each feasible day, the insertion method begins by retrieving all available technicians: those who are available on that day and are currently not assigned to any team (line 7). The method then iterates over all available vehicles (lines 8-16). Considering the available technicians together with those already assigned to vehicle v, teams are generated for the given vehicle (line 9). To be considered, a team must be able to serve all of the tasks already assigned to route v in addition to the new task i. Only a single team is returned when the constructive method in Section 5.4.1 is employed, while the team matrix method may return multiple possible teams. Note that team selection not only impacts the route's cost, but also its feasibility as different teams may have different time windows which in turn determines the earliest/latest time the route can start/end. Teams also have a maximum working time and may have a fixed starting position, which can differ from the depot associated with the vehicle used. As a consequence, it is crucial to first define the team before attempting to find a suitable position for the task insertion.

Given a feasible team, the method finds the best position within the route to insert task i while respecting vehicle capacity, the time windows of tasks and the need to schedule breaks for technicians (line 11). Note that all of the tasks with precedence relations that are scheduled to be served in the same day are always inserted into the same route. We impose this restriction in order to handle the additional complexity of route synchronization, which creates dependencies between routes and additional challenges for ruin and recreate neighborhoods. The task is inserted if a feasible position is found and results in a new solution s' (line 14). The best solution may then be updated (lines 15-16), and the number of days with feasible insertion is incremented (lines 17-18). If no feasible insertion is found, the task is left unserved.

When the maximum number of feasible days is reached, the best solution is returned (lines 19-20). Otherwise the method ends once all days have been checked at which point the best solution is returned (line 21). Note that d is not incremented when a task cannot be inserted on a given day. This is to prevent the method from stopping prematurely by counting days that are full and therefore cannot accommodate any additional tasks.

5.4.3 Removal neighborhoods

The tailored neighborhoods proposed in this chapter address two decision levels: the optimization of routes and team assignment. Each route must have a suitably qualified team assigned to it that is capable of serving all of its associated tasks. Given that a route comprises of many tasks with different requirements and since teams must stay together throughout the entire route, it is possible that teams may be overqualified for some of their route's tasks. Although this is not prohibited, it often leads to underutilization and wasted service time of some members in the team. This means that a technician, who is paid for the duration of the entire route, is not working during some tasks.

Let us consider the wasted service time of a team assigned to task i to be the service time of i multiplied by the number of technicians in the team who are idle during the task's execution. The wasted service time of a route is then the sum of the wasted service times of all tasks in that route. This section will address this problematic situation by developing neighborhood operators focused on routes with wasted service time. This situation arises when a route has a team with two or more technicians and at least one task requiring fewer technician profiles than the team's size. By using this operators we aim to improve the utilization of the team members and reduce wasted service time.

Release technicians neighborhood

Given an ordered set of routes, this neighborhood operator randomly selects 1-4 routes with the highest wasted service time and removes as few tasks as possible in order to release at least one technician. Consider the two routes depicted in Figure 5.2 with two available technicians and tasks which each require one hour of service time. In Figure 5.2(a) both technicians are assigned to the route, but the green technician only serves Task 2. Since the green technician is idle during the service of Tasks 1, 3 and 4 and the red technician is idle during the service of Task 2, route (a) has a wasted service time of four hours. Similarly, the route in Figure 5.2(b) also has a wasted service time of four hours: the green technician is idle during Tasks 1, 2 and 3 and the red technician is idle during Task 2. In order to release the green technician in each of these routes the method should remove Task 2 in (a), while Tasks 2 and 4 should be removed in (b).

After removing tasks and releasing a technician, slack with respect to both routes and technicians is created. This means that the route now has space for new tasks to be inserted and there is an additional technician who is available to be assigned to other routes where they may be scheduled with less wasted service time.

Maximize utilization rate of technicians

The goal of the second neighborhood operator is to increase team utilization rates, rather than release technicians. Figure 5.3 provides an example of how the method works considering four available technicians, two routes and eight tasks. Like the previous neighborhood, it begins by also randomly selecting 1-4 routes with the highest wasted service time. Then, all tasks which require fewer technicians than the team's size are removed. Consider the top-left route in Figure 5.3(a) as the candidate route. Tasks 3 and 4 should be removed since they only require a single technician and are resulting in wasted service time. This move creates slack in the candidate route, providing opportunities for other tasks to be inserted. Since high team utilization rates are likely to improve

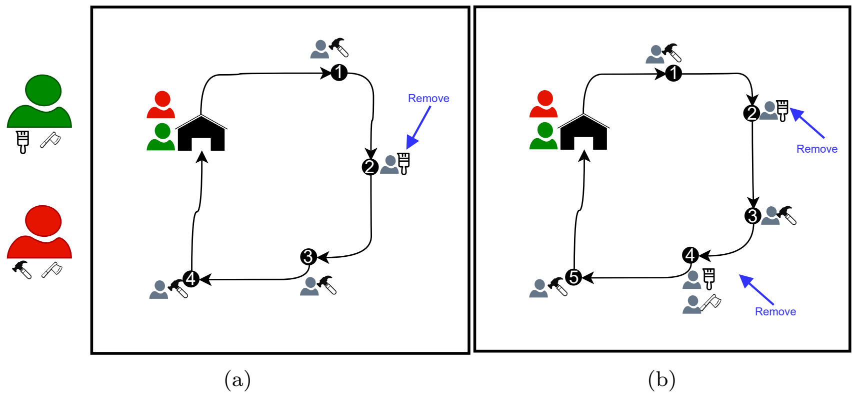


Figure 5.2: Removing tasks to release technicians.

the overall objective, one should search for potential tasks that require the full team and that could be inserted into the candidate route.

The team matrix provides information to identify tasks that are not in the candidate route and can be served by the current team without producing wasted service time. In Figure 5.3(b), Tasks 6 and 7 are identified as potential insertion tasks as they require two technician profiles compatible with the technicians already assigned. These tasks are removed from their previously assigned route, resulting in Figure 5.3(c). Figure 5.3(d) provides a potential solution returned by the task insertion method considering the removed tasks and the partial solution in Figure 5.3(c). The candidate route in the updated solution now only serves tasks which require the full team, thus maximizing its utilization. Moreover, removing tasks which require multiple technicians from other routes may lead to some technicians being released, which is the case for the blue technician in Figure 5.3(d).

Minimize overall distance

While the two previous neighborhood operators focused on increasing the utilization of teams serving routes, the last neighborhood focuses on minimizing travel distance of the routes. While many different methods could be employed for this purpose, we implemented the string removal method introduced by Christiaens and Vanden Berghe (2020). This heuristic provides good results for a variety of VRP problems over a range of instance sizes. The string removal

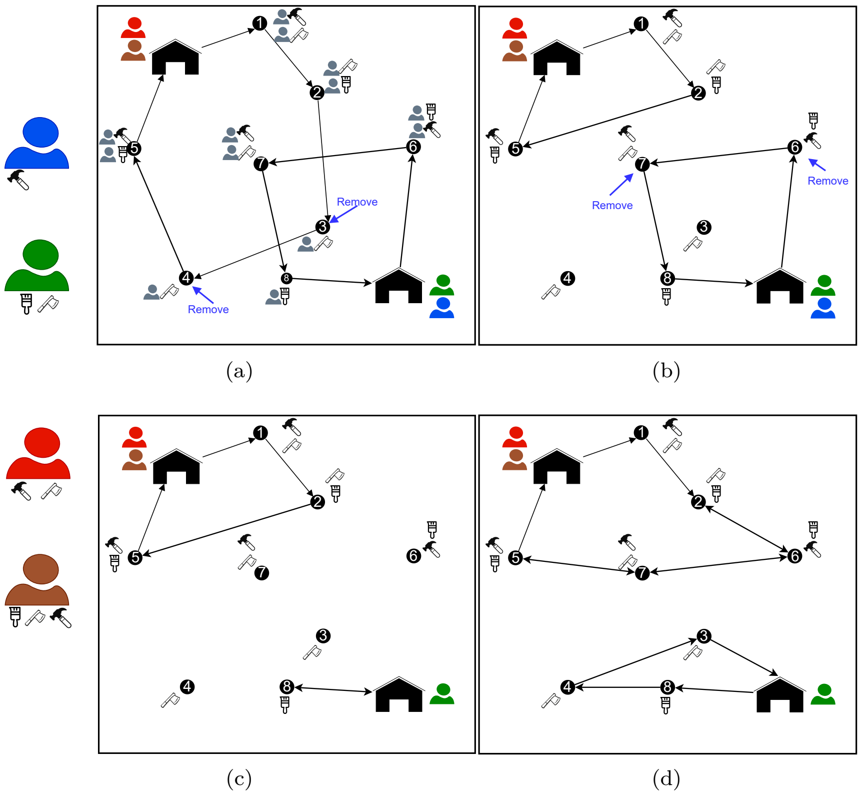


Figure 5.3: Removing tasks to maximize team utilization

method removes geographically proximate tasks in different routes and days, creating slack in many routes across the time horizon. This increases the possibility for tasks to be inserted into another route on either the same or different days.

5.4.4 Routing emergency tasks

During the daily operations of an SPC, unexpected events can occur which cause serious damage to critical equipment and can result in safety hazard. These unexpected events result in emergency tasks which should be dealt with as soon as possible, preferably on the same day. For the algorithm to handle such cases, the current solution is given as input with all its routes and technician assignments, the current time and an indication of which tasks have already been completed.

On the first day of the time horizon (the current day), teams are fixed to routes and the tasks that have already been served are also fixed. Teams cannot be changed since they have already been deployed and the tasks already served are kept in the solution in order to take into account the current location of vehicles and time already spent executing the route. Tasks on future days may also be fixed to ensure that customers who have already been informed of a planned visit do not have this date changed. Tasks that have not been fixed can be rescheduled without any additional restrictions. The algorithm is executed with teams and tasks already fixed, with the emergency task inserted at the earliest possible opportunity and then also fixed. The algorithm's remaining computational time is used to reoptimize the solution while respecting fixed teams and tasks.

5.5 Computational experiments

In this section we aim to evaluate the performance of the algorithm. We will also evaluate how bounding methods, team assignment strategies and different neighborhoods impact solution quality. Solutions will be compared with respect to the cost (route, personnel and delay) as well as the utilization rate of teams. We also employ the solution approach to draw managerial insights regarding different contractual arrangements. Experiments will consider instances where tasks require teams of either a single technician or multiple technicians.

All experiments were conducted on a computer with an Intel Xeon E5-2660 processor at 2.6 GHz and 164 GB of RAM running Ubuntu 18.04 LTS. LAHC was implemented in C++ and compiled using gcc 9.3.0 with option -O3. Given the non-deterministic nature of the algorithm, results presented in this section are the average of 10 independent runs per instance. Complete tables are available as supplementary material and hosted in a publicly available repository[1].

The string removal method from Christiaens and Vanden Berghe (2020) was implemented with its original parameters for best behavior across a range of instance sizes. The list length for LAHC (L_s), maximum number of consecutive rejections (M_{cr}) and maximum run time (M_{time}) were calibrated based on the results of empirical tests and set to 100, 3000 and 600, respectively. These values were selected in order to strike a suitable balance between solution quality and processing time. Additionally, a time limit of 600 seconds was considered

[1]`http://dx.doi.org/10.17632/7fy8zsy5r8.1`

reasonable by our industrial partner to produce routes and schedules for a time horizon of several weeks.

5.5.1 Instance set

In order to not only evaluate the algorithm's performance but also facilitate future research into the MTRSP, instances are generated based on data provided to us by a software developer for various SPCs. The data provided contains information concerning depot and task locations, the number of vehicles and technicians available per depot, wages, as well as regulations concerning working time and overtime. Although instances are generated using as much real data as possible, access to data is still limited. This means that some aspects of the instances are generated based on observations provided by an industrial partner during project meetings. The lack of additional historical data prevents us from making a direct comparison with the schedules manually prepared. However, we are still able to draw a range of important insights concerning both the algorithm's behavior and the problem's characteristics.

Instances are generated with 8 vehicles and 13 technicians that are distributed across 3 depots located throughout Belgium. Vehicles are associated with one of two profiles: a small and cheap pickup truck capable of holding two passengers or a larger and more expensive van capable of holding three technicians. There are a total of six skills, with each technician qualified in 2-3 of these. The hourly cost of technicians qualified in three skills is slightly higher than the cost of those qualified in two. Tasks have their due date uniformly distributed over the time horizon, with higher delay costs associated with tasks with earlier due dates. The service time of tasks is 40, 60 or 80 minutes. Additionally, the time window associated with a task can either cover the whole day or only the morning/afternoon.

Two sets of instances were created, one where every task requires only a single profile (S) and another where tasks require 1-3 profiles (M). Tasks require profiles with one or two skills and there always exists at least one technician who covers that specific combination of skills. Each set has 6 instances, with a varying time horizon and number of tasks to be served, namely: time horizons of 15, 30 or 60 days, and 100, 200, 400 or 800 tasks. The name of each instance conveys all of its relevant properties. For example, instance *I_30_200_M* has a time horizon of 30 days, 200 tasks to be scheduled with these tasks requiring multiple profiles.

5.5.2 Removal neighborhoods analysis

In this section we investigate the contribution of the removal neighborhood operators proposed in Section 5.4.3 to the final solution quality. In order to do so, experiments were designed in such a way that either all three operators are available to be selected during the ruin solution phase (line 6 of Algorithm 7) or only the minimize overall distance neighborhood is considered. When all neighborhood operators are available to be selected, at each LAHC iteration only one randomly selected neighborhood is used. Empirical experiments demonstrated that our algorithm performs best when each of the two neighborhoods associated with technician utilization is selected 10% of the time, while the neighborhood for minimizing overall distance is selected the remaining 80% of the time.

These findings indicate that deactivating the neighborhoods related to technician utilization results in worse average solutions: solution quality worsens by 1.66% on average, with a maximum negative impact of 4% for instances with 15 days and 200 tasks. Utilizing these neighborhoods led to solutions with fewer routes (80 versus 84 on average) and significantly lower delay costs. Although the neighborhood focused on routing is used more frequently, incorporating the other removal neighborhoods enhances the algorithm's ability to find better solutions in the same amount of time.

5.5.3 Number of days to evaluate

Considering a time horizon of multiple days, the day on which a task is scheduled to be served can have a large impact on the overall solution quality. Serving tasks on the soonest possible day may reduce the problem's search space, while evaluating every day in the time horizon could prove very time-consuming. Our goal is to find a suitable balance between processing time and solution quality. In this section we evaluate the role of the parameter max_d on the task insertion method, as outlined in Algorithm 8. This parameter controls, for a given task, the minimum number of feasible days to be considered during the task insertion procedure. The experiments consider max_d set to 0%, 25%, 50% and 100% of the total number of days in the time horizon. When max_d is set to 0% a first feasible insertion approach is employed where tasks are served as soon as possible, while max_d set to 100% means that every day will be considered for insertion and the one that ultimately yields the lowest cost will be selected.

Table 5.3 provides computational results when varying max_d. Each row corresponds to the average result for 12 instances and shows the average solution S_{avg}, the gap to the best average solution cost obtained over all considered values of max_d (gap) and breaks down the average solution cost in terms of delay, personnel and route costs. Additionally, #Iterations provides the number of LAHC iterations, #Routes provides the average number of routes needed to serve all tasks and #Days corresponds to the average number of days considered when inserting a task.

Table 5.3: Impact of the number of considered days for task insertion.

max_d	S_{avg}	gap	Delay	Personnel	Route	#Iterations	#Routes	#Days
0	168287.30	22.23	5320.70	85856.13	77110.47	11674.43	91.57	7.89
25	135920.76	3.71	4259.27	72467.99	59193.50	6515.80	80.83	14.14
50	130883.23	0.00	4337.52	70565.83	55979.88	5214.70	80.18	20.49
100	131552.31	0.51	4353.92	70995.65	56202.75	4332.57	80.53	24.67

Table 5.3 shows that $max_d = 50\%$ results in the lowest average solution cost. Although one would expect that considering every day should perform better, given the limited run time $max_d = 50\%$ achieves a better trade-off between solution quality and performance. Given that considering a larger number of days requires more time, LAHC employing $max_d = 100\%$ conducted 16% fewer iterations and checked 20% more days during task insertion. This resulted in solutions that were on average 0.51% worse than when $max_d = 50\%$. On the other hand, reducing the number of considered days to 25% increases the number of iterations performed by the algorithm. While this reduces the average delay cost, it also results in higher personnel and routing costs. This ultimately produces solutions that are on average 3.71% worse than when $max_d = 50\%$. Figure 5.4 illustrates the number of days considered for task insertion for each value of max_d, where the average results are 8, 14, 20 and 25 days.

For the first feasible insertion approach ($max_d = 0\%$), the number of days considered per task is the lowest and less than half the number of days when $max_d = 50\%$. This has a direct impact on the number of iterations, which correspondingly doubles during the same 10 minutes of execution time. Additionally, for instances with a single profile requirement (instance set S), the first feasible insertion approach obtained the lowest average delay cost. Although achieving much lower delay costs for specific instances, this method produces more routes and is associated with far higher routing and personnel costs.

5.5.4 Analysis of team assignment methods

The next set of experiments concerns the methods for team assignment presented in Section 5.4.1. We compare the results of the team matrix method against the results when using the constructive team method. We also present the results

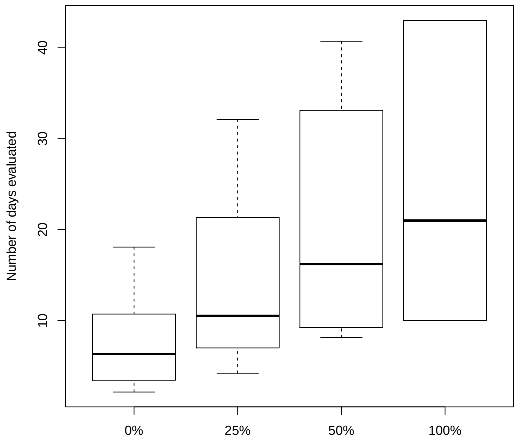


Figure 5.4: Number of days considered when inserting a task for varying percentages of max_d.

for an adapted version of the team matrix method that prohibits wasted service time. In this method, routes are formed in such a way that all tasks visited by a given route r require the same number of profiles ($|\mathcal{K}_i| = k\ \forall i \in r$) and the team assigned to the route must have exactly k technicians. Table 5.4 provides an overview of results for both single-profile (type S) and multiple-profile (type M) instances. The final column corresponds to the average wasted service time per route considering all assigned technicians.

Table 5.4: Evaluation of the number of considered days for task insertion.

Instance set	Technician assignment	S_{avg}	Gap	Delay	Personnel	Route	#Iterations	#Routes	Wasted service time
S	Team matrix	99293.61	0.00	3188.37	42039.24	54066.01	2968.90	79.53	11.43
	Team matrix*	90539.50	-9.67	2370.67	39595.03	48573.80	8151.57	75.60	0.00
	Constructive	96815.58	-2.56	3225.53	41016.91	52573.13	7578.23	81.07	28.74
M	Team matrix	162472.85	0.00	5486.67	99092.42	57893.76	7460.50	80.83	103.44
	Team matrix*	165037.80	1.55	5058.97	96443.24	63535.60	9360.80	90.33	0.00
	Constructive	172435.85	5.78	6597.20	101749.65	64089.00	8058.17	88.80	114.70

*: No wasted service time permitted.

For single-profile instances, the constructive method produces solutions that are on average 2.56% better than the team matrix method. The team matrix method evaluates multiple teams per route and, therefore, for the single-profile instances the number of possible teams to evaluate is much higher which in turn makes the method slower. For example, when all technicians are available (13 technicians and 377 possible teams), on average 276 teams are capable of serving a given task for instance set S, while for instance set M on average only 174 teams are capable of serving a given task. Therefore, the constructive method is much faster in this scenario and able to perform twice as many iterations since it does not require verifying multiple teams for each route. However, both of these methods allow teams with multiple technicians, even though all tasks in this instance set require just one profile. The team matrix method with no wasted service time produces solutions where all routes are performed by a single technician. This method considerably reduces the number of possible teams to evaluate, has a similar performance in term of iterations compared to the constructive method and is the best performing method, producing solutions that are on average 9.97% cheaper than the same method when allowing wasted service time.

For instance set M the team matrix method produces solutions that are on average 5.78% cheaper than the constructive method. This is mainly due to the team matrix method's ability to explore a larger number of teams, increasing the chances to serve more tasks with fewer routes. In instance set M, team configuration plays a crucial role in reducing personnel costs, which are dominant in the objective function for this set. As the constructive method is incapable of generating all possible teams, it cannot find the same solutions as the team matrix method. Thus, by generating and evaluating multiple teams per route, the team matrix method not only produces cheaper solutions with 8% fewer iterations, but also generates fewer routes, less wasted service time, and fewer unscheduled tasks (3.9 versus 6.7 on average). Additionally, the team matrix method also outperforms its adapted version, producing solutions that are 1.55% cheaper on average. Although the team matrix method with no wasted service time generates solutions with lower personnel costs, prohibiting routes with wasted service time results in a higher number of routes and higher routing costs. This method could be more suitable in situations where personnel costs significantly exceed routing costs, or if certain SCP operations prohibit technicians from being idle during the service of any task within a route.

Figure 5.5 illustrates the average wasted service time per instance for both methods with wasted service time. The team matrix method results in the least wasted service time for all instances, with notable differences for large instances featuring 60 days and 800 tasks. We believe that large instances require a particularly careful balance between high-quality routes and team assignment, which gives the team matrix method a distinct advantage as it explores more team configuration options. However, SPCs would benefit from the constructive method under the following circumstances: they only have tasks with a single profile requirement, have a large number of technicians available with similar skills or need to produce high-quality solutions very quickly.

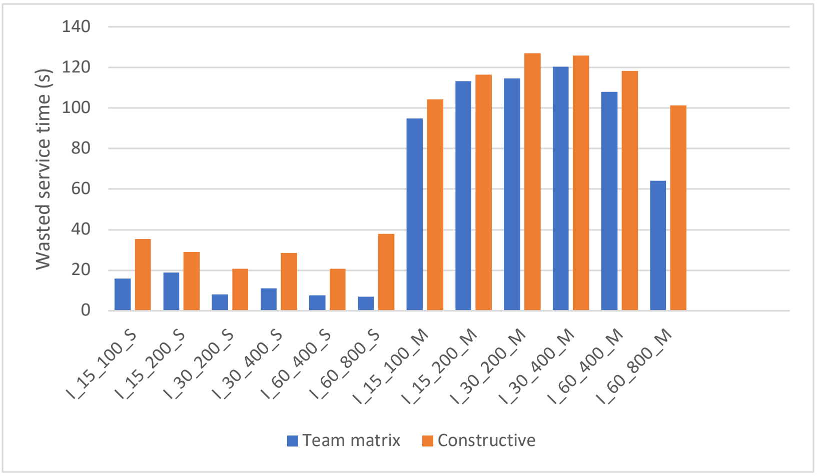


Figure 5.5: Wasted service time of technicians per route for each instance.

5.5.5 Managerial insights

Sections 5.5.2 to 5.5.4 focused on analyzing the algorithm's components and parameters. In this final set of experiments, we aim to explore the algorithm's potential for providing valuable managerial insights. More specifically, we are interested in investigating how different types of contractual arrangements with technicians impact solution quality. The motivation behind these experiments comes from our observations of companies using different contract types and how these decisions not only impact operational costs but also employees' satisfaction and sense of security.

For this set of experiments we altered the value of the binary parameter μ to generate computational results for different contract types. Table 5.5 provides computational results for both single-profile (type S) and multiple-profile (type M) instances for different contract types. The hourly contract type pays technicians only for the hours they work ($\mu = 0$), whereas the daily contract type ($\mu = 1$) pays technicians for an entire shift (of w_t^{max} hours) for each day they work. The full-time contract pays every technician for an entire shift every busy day (Monday to Friday), even if they are not assigned to any route on certain days. For both the daily and full-time contracts, hours worked as overtime are paid separately and are computed accordingly for each day. The last two columns of Table 5.5 correspond to the average duration of routes (in minutes) and the number of technicians employed per instance. For each instance, the latter is calculated as the sum of technicians employed per day for every day in the time horizon. Since daily contracts were the most commonly observed type of contract in our collaborations, we consider it as the basis for comparison.

Table 5.5: Evaluation of different contract types for technicians.

Instance set	Contract type	S_{avg}	Gap	Delay	Personnel	Route	#Routes	Duration	#Tech used
	Daily	100134.75	0.00	3300.73	42824.11	54009.90	78.57	543.25	81.83
S	Hourly	100209.99	0.08	3145.63	42973.80	54090.57	79.70	533.51	82.73
	Full time	191983.89	47.84	2635.53	139394.04	49954.33	79.50	516.21	115.77
	Full time*	159819.71	37.35	2168.50	110796.24	46854.97	77.53	511.94	112.90
	Daily	165766.16	0.00	5543.40	102231.68	57991.08	78.73	538.68	184.10
M	Hourly	165694.65	-0.04	5547.27	102167.27	57980.11	80.87	517.96	186.63
	Full time	213884.34	22.50	3977.70	153088.91	56817.74	84.43	496.85	206.90
	Full time*	185241.48	10.51	8217.10	123575.72	53448.66	80.03	496.83	189.03

*: Workforce with less technicians.

When comparing hourly and daily contracts, results indicate that solution costs are similar. This similarity can be attributed to the high workload associated with the instances under consideration. Although not all technicians work every day, those who do already work long hours for both hourly and daily contracts. As a result, considering hourly contracts instead of daily contracts has little impact on overall costs. Beyond average solution costs, our analysis revealed that for instances of type S and M, solutions with daily contracts resulted in an average of 1.4% and 2.6% fewer routes, respectively. Moreover, routes are on average 1.8% and 4% longer and the solutions require fewer technicians over the time horizon. Given that the costs are similar, managers should be encouraged to explore more options of daily contracts if the benefits exceed the slightly higher costs.

The results also show that full-time contracts significantly increase overall costs compared to daily contracts. Full-time contract solutions cost an average of 47% and 22% more for instances of type S and M, respectively. This is because full-time contract solutions use only 85% of the available technicians on average, resulting in increased personnel costs compared to the daily contract solutions that only pay technicians when they work. However, full-time contract solutions show a decrease in delay costs of 25% and 39% as well as a reduction in route costs by 8.1% and 2.1% for instances of type S and M, respectively. Full-time contract solutions also required more technicians, had a greater number of routes and those routes had shorter durations. Moreover, one could also argue that having extra staff available could improve service levels, as technicians would be available in case of emergency tasks or to replace other technicians who might call in sick.

Given the number of idle technicians in full-time contract solutions, one can also investigate the impact of a reduced workforce. We did this by performing experiments with a workforce of 10 technicians instead of the original 13, eliminating the technicians with the fewest number of skills from each instance. Results show that full-time contract solutions with a smaller workforce experienced a decrease in personnel costs, but a slight increase in delay costs. Compared to daily contract solutions, full-time contract solutions with a workforce of 10 technicians cost on average 37% and 10% more for instances of type S and M, respectively. Additionally, it resulted in an increase of up to 32% in delay costs compared to daily contract solutions.

When comparing different full-time contract scenarios, results should be interpreted carefully. Although a reduced workforce may incur less overall cost, the increased incidence of delays should be taken into consideration. Managers and planners should carefully evaluate the extent to which delaying or not serving tasks may impact customer satisfaction and therefore customer retention in the long term.

Although the ambition of these experiments is not to quantify the precise impact on operational costs that different contractual polices incur, it should provide managers with an general overview of how the system behaves under different polices. For example, the experiments showed that daily contracts can be offered with little to no additional cost when compared to hourly contracts, which has the potential to improve the satisfaction of workers in terms of guaranteed working hours. While these experiments are of course limited to the data we have available, we believe managers and planners can use more detailed historical data to run experiments on various scenarios and understand the impact different managerial polices might have on their operational costs.

5.6 Conclusions

This chapter introduced a team routing and scheduling problem featuring a multi-period time horizon to address the operational problems faced by various service provider companies (SPCs). The goal was to capture all their requirements and develop a solution approach capable of providing high-quality solutions within reasonable computational run time. A key academic challenge involved addressing how tasks require certain technician profiles to be met in order to be served. Indeed, skill coverage alone is not enough given that certain tasks require multiple technicians to be present, each with their own set of skills. Although our focus is on routing and scheduling technicians, the underlying problem has more general practical relevance since it can be applied to activities involving other workforces. For example: the routing and scheduling of sales representatives, consultants, cleaners and healthcare personnel.

Even though the operations of each SPC share many similarities, it is natural to assume that each will also have slightly different goals and preferences. For example, one SPC might prioritize personnel costs over other expenses, while another might be focused on minimizing costs associated with delays rather than routes or personnel. As a result, our algorithm's components should be adjusted in accordance with the SPC's priorities: whether routes should start as early as possible or as late as possible, the minimum number of days considered for task insertion, the maximum number of teams considered per route and whether wasted service time is allowed.

For future research it would be worth considering how one can improve the robustness of solutions, this would not only accommodate unexpected events but also improve the response to emergency tasks. For such scenarios, solutions that have sufficient slack would enable additional tasks to be inserted if needed and hard constraints, such as maximum working time, could be made more flexible. Our algorithm could also be used to provide insights concerning more long-term strategic decisions. Managers could run multiple scenarios to identify the impact on solution cost when a technician is set to retire, a technician is trained in new skills, or hiring a new technician with a specific set of skills.

Chapter 6

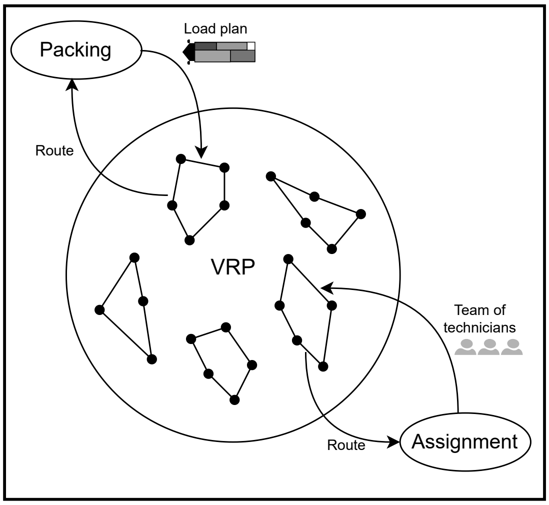
Conclusion

This chapter concludes the thesis and is divided into two sections. First, we present a general methodology for addressing problems with multiple decision levels integrated with an underlying VRP. This methodology was gradually developed throughout the thesis, synthesizing the common features adopted in each chapter to form a high-level framework. The second section ends the thesis by reflecting on the goals we set out to accomplish, some observations regarding our experience collaborating with logistics companies and some of the possible avenues for future research.

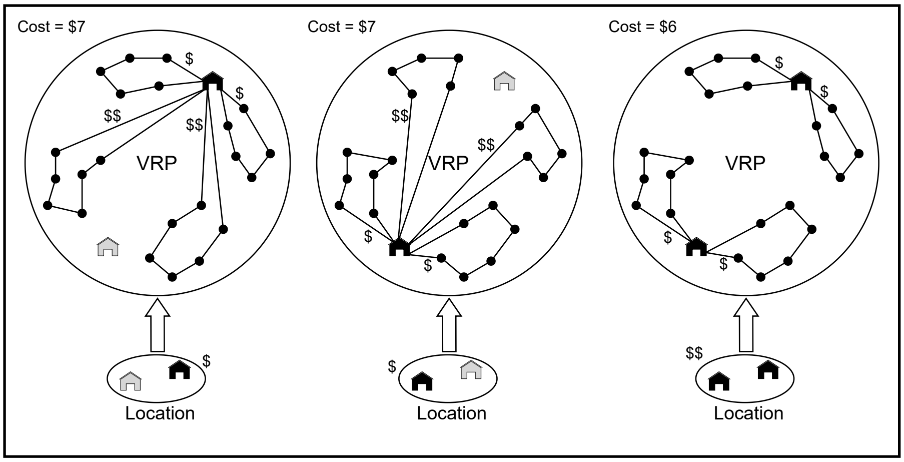
6.1 The general methodology

Our general methodology consists of five steps which should provide researchers with some initial direction and questions to consider when addressing multi-level problems with an underlying VRP. First and foremost one should identify the nature of the problem at hand and what its distinct decision levels involve. These decision levels will likely interact with each other in such a way that the decisions made in one level respond to those made in another. In this thesis we identified that the VRP decision is more influential in terms of having a larger search space and dominating the costs in the objective function. Therefore, the VRP consistently exerted a larger impact on the overall solution quality. In most cases we expect that the decision levels at hand can consequently be classified as one *primary* decision level (VRP) integrated with one or more *secondary* decision levels (packing, team assignment, scheduling and location).

Figure 6.1 illustrates some of the ways in which these levels can interact with one another.



(a)



(b)

Figure 6.1: Some possible interactions between primary and secondary decision levels.

One way in which these levels can interact is by the primary level relying on a secondary decision level to ensure solutions are feasible. Figure 6.1(a) illustrates how the routes produced by the VRP method utilize the packing method to create feasible load plans or the assignment method to find feasible teams of technicians. Despite the importance of these secondary decision levels, the overall quality of the solution primarily derives from how capable the VRP method is of producing high-quality routes. Another form of interaction between decision levels is illustrated by way of Figure 6.1(b) and concerns how changes in the secondary level have the potential to direct the search of the primary level towards better solutions. In this case, the solution of the secondary decision level is evaluated in terms of how positively it can impact the solutions produced by the primary level.

In Figure 6.1(b), despite the impact of the location configuration on the routes, the primary decision level remains dominant in terms of cost and search space. Note how although the rightmost solution in this example maximizes the location cost, the corresponding minimization of VRP cost results in the solution with the lowest total cost. Beyond the examples illustrated, the VRP also remains the primary decision level when customer visits must be scheduled over multiple days in a multi-day VRP or when determining transfer points in a VRP with transfers. Despite the smaller search spaces associated with these secondary levels, it is worth noting that the decision making process for them nevertheless remains a challenging combinatorial problem.

The second step of our general methodology is to take advantage of state-of-the-art approaches for the primary decision level. While the selected method will need to be adapted for the integrated problem, using an established state-of-the-art method will provide strong foundations for the overall approach. At this point, developing a deeper understanding of the primary decision level is extremely advantageous. This is because one needs to determine how the problem's different decision levels will interact, the impact they have on one another and the required adaptations on the state-of-the-art method to accommodate all other decisions. For example, the VRP approaches developed in this thesis required adaptations to handle multiple depot locations and time horizons of multiple days. The VRP approaches were also adapted to integrate packing, transfer and team assignment methods.

The third step of our general methodology involves designing fast approaches for the secondary decision levels. The goal is to produce high-quality solutions within reasonable computation time. This is advantageous because during the optimization process, the methods associated with the secondary decision levels are executed regularly. For example, the packing and team-assignment methods developed in this thesis are executed every time a new route is created. Fast approaches can also involve preprocessing methods that calculate and store all possible solutions for a certain decision level at the start of the algorithm's execution. These solutions can then simply be retrieved later on during execution, without any further processing being needed. One could also save solutions during runtime to be re-used later. Finally, bounding methods can be used to avoid wasting effort optimizing a certain level based on the bound of another. In this thesis we employed a preprocessing method to derive all possible teams of technicians for the MTRSP. In Chapter 3 we have stored both load plans for vehicles as well as solutions for the first echelon which could be reused and have calculated bounds for the location routing problem in order to be able to discard poor location configurations.

The fourth step of our general methodology is to create tailored local search neighborhoods in order to address the unique characteristics of the integrated problem which are not taken into account by the method initially designed with only the primary decision level in mind. Given the large search space of the integrated problem, it is necessary to design neighborhoods that are capable of interfacing between decision levels. This fluid integration of distinct levels will help guide the search more quickly to promising regions. These neighborhoods should take into account how the multiple levels interact and generate changes in one level that will lead to positive impacts in the others. For example, in this thesis we employed neighborhoods in Chapter 2 that evaluate the configuration of routes to derive good transfer locations while neighborhoods employed in Chapter 5 change the configuration of routes to better utilize the team of technicians already assigned.

Integrating all these methods into a metaheuristic to guide the search represents the fifth and final step of our general methodology. Preferably, one should opt for a simple metaheuristic that has been used extensively across a range of problems and requires only a few parameters to be calibrated. In this thesis we employed metaheristic frameworks such as Iterated Local Search and Late Acceptance Hill Climbing. An advantage of these methods is how they allow worse solutions to be accepted. This mechanism is important since although a change in one decision level may initially result in a worse overall solution, after subsequently optimizing the other decision levels an overall better solution may eventually be reached.

6.2 Final remarks

This thesis investigated and addressed four problems arising in logistics companies operating in Belgium. The daily operations of these companies spanned various functions: shipping, machine rental, security services and technical services. In order to produce feasible solutions that they are capable of using in practice, multiple decision levels and characteristics had to be considered. Many of these characteristics went well beyond the classical definition of problems already studied in the literature, resulting in challenging new problem variants. An appropriate trade-off between the completeness of our models and reasonable processing time was achieved by carefully integrating the decision levels that were unique to each problem. All of the problems addressed in this thesis were approached in a similar fashion, with satisfactory results produced in accordance with the required operational demands without exploding algorithmic complexity.

Besides successfully addressing multiple problems, we also arrived at a general methodology which should help orient future researchers who find themselves facing similar problems with multiple decision levels that are best addressed in an integrated manner. Such problems are frequently encountered in logistics, production planning and supply chain management. Although the VRP was the common denominator and primary decision level across our set of problems, this methodology could be applied to other contexts in which there is a primary decision level interconnected with one or more secondary levels. Take for example integrated location and scheduling problems. One common problem is where to locate machines for a certain period of time given that they can only be moved from one location to another at fixed intervals. Given a known customer demand during certain periods, the aim is to determine the location of these machines and the customer schedule such that some scheduling objective function is minimized. Another example of location and scheduling problems could involve an integrated yard management scenario where the goal is to minimize the transfer distance of containers. In this scenario one needs to find the best position at which to berth the vessel and then schedule the transfer of containers to and from the yard. Although these are clear examples of problems that integrate multiple decision levels, executing the first step of our methodology would involve investigating further and determining what are the primary and secondary decision levels at hand.

Future research directions could be to investigate the many opportunities concerning the third step of our methodology. For example, developing more accurate bounding methods can reduce the problem's search space and enable us to handle larger instances in a more efficient manner. Additionally, one could even attempt to integrate all of the different decision levels presented in this thesis into a single unified and configurable algorithm. While this represents a very ambitious undertaking, doing so would enable us to solve VRP problems that involve packing, team assignment, scheduling and location decisions with a single algorithm. The valorisation plan which follows this chapter will elaborate on this idea.

When reflecting on our experience collaborating with company representatives to model and solve their problems, it is clear to us that there are several additional challenges that should be taken into consideration. Especially when addressing the technician routing and scheduling problem in Chapter 5, we were confronted with challenges on how to integrate into our algorithms all of the soft constraints and human preferences. These might include preferences concerning vehicles, work colleagues or departure times. Such preferences are difficult to quantify in terms of cost, but often have an undeniable impact on the solution and how workers as well as customers will ultimately interpret its quality. Indeed, in many operational environments a solution can be deemed unfit for no reason other than the fact it clashes with these preferences. It would therefore be interesting to investigate whether such preferences should in fact represent a decision level all of its own. One could then experiment with how to integrate such a decision level in existing algorithms and deal with it efficiently in order to produce solutions that are deemed satisfactory by all the stakeholders involved: employers, employees and customers.

Another challenge we encountered is that a company will likely already have a method in place for solving their problem, which is built upon a set of underlying assumptions. Companies often adopt these assumptions because they do not possess the full picture of their problem or to simplify matters so that they can tackle it with their available resources. One should therefore begin an industrial cooperation by gaining an understanding of the problem, the approach currently used to solve it, the problem's true constraints and the simplifications artificially enforced by the company in order to make solving their problem easier. To showcase what such simplifications can look like in reality, let us draw on a real example we encountered during the multi-depot transportation problem addressed in Chapter 4. In this problem customers were previously associated with specific depots and only served by that depot throughout the entirety of the multi-day scheduling horizon. This helped operators, as they first decided customer-depot assignments and then concentrated on the route of each depot separately. Their logic behind enforcing fixed assignments is that customers require keys that are located at the predetermined depot. However, we discovered that those keys can have copies present at multiple depots. Exploiting this fact and allowing customers to be served from multiple depots across the time horizon resulted in far better routes with much lower costs. Therefore, one should try to identify such simplifications and then determine whether they help or hinder the optimization process, as there are often significant gains that can be realized by challenging them.

Chapter 7

Valorization Plan

De Jong et al. (2015) define valorisation as “the process of creating value from knowledge, by making knowledge suitable and/or available for economic and/or societal use and translating it into competitive products, services, processes and entrepreneurial activity.” This brief final chapter aims to create a valorisation plan which outlines the potential future activities that could be undertaken in order to transform the research presented in this thesis into a socially valuable resource.

The following sections will explore how the knowledge and research acquired through this thesis can be transferred to an open source software library (SL). This SL would take the form of a decision supporting tool for vehicle routing problems which feature additional decision levels such as location, scheduling or packing. Therefore, the SL which could logically follow from this PhD thesis should target users who cannot afford established commercial software, who face significant operational challenges and are often solving them manually, and that positive social impact would be obtained by using the library. While some software similar to this already exists[1][2][3], they are often very expensive and thus inaccessible to our target audience.

[1]`https://company.ptvgroup.com/en/`

[2]`https://www.aimsun.com/`

[3]`https://ortec.com/en/business-areas/routing-loading`

7.1 Target audience and added value

First, it is important to identify the potential users of our SL. Small and medium-sized businesses could be our first target. In 2019, the logistics industry alone employed 126 thousand workers in the Flemish Region, which accounts for 4.4% of the total number of workers [4]. Aside from businesses, other potential users are NGOs, divisions of local government (universities, hospitals), resident associations and any other low profit organization that requires routing and logistical solutions to provide better services to their respective communities. These users may have routing as the whole or a part of their operations.

For example, disaster management groups often need to make quick decisions concerning the location of depots and design of routes for the distribution of food, medicine and other resources for regions affected by natural disasters. Meanwhile, NGOs may require routes to distribute goods to struggling communities in developing countries. Another example concerns rural villages which often rely on transportation services to connect them with larger cities for medical appointments. For example, residents may be transported to the city for treatment, or healthcare professionals may travel from the city to provide care in rural areas.

All of these potential users of our SL have tight budgetary constraints and could use some decision support tool for their process in order to serve more people with fewer resource and less time. Although our SL may not cover all their needs, it would still offer possibilities for automating at least some of their processes. This project would bring crucial technological solutions to the often struggling small business and overlooked social institutions that are vital for the functioning of our societies.

7.2 Technical aspects and distribution

The SL requires no special hardware or advanced infrastructure and would be able to run on standard, mid-range desktop computers. The code should be hosted in a code versioning platform, such as github, as a private project, while access would be granted to users upon request. The library should have some level of configuration so that different users can take advantage of the SL in the way which helps them most. Most users should have customer information, vehicles and depots, but could configure whether they need to set up personnel skills, potential depot location or item dimensions and packing restrictions. All

[4]`https://www.vlaanderen.be/statistiek-vlaanderen/mobiliteit/werkgelegenheid-in-de-logistieke-sector`

of these data should be provided as a well-defined input to the SL which will rely on proper documentation to provide the user with accessible guidance. A common user with minimum programming skills should be able to have the SL working without much effort. The SL will then return the produced solution in a well-defined output where route information and personnel scheduling will be easy to retrieve.

We do not intend to profit with the SL, rather we would like to create a community of users. Thus, all users granted access to the SL will have full access to the source code. Maintenance, support and further development of features should be open for the user community, eventual changes and additions can be screened and validated by experienced contributors. In the future, depending on the demand, a small specialized team with a background in programming and operations research could be allocated periodically in order to implement extensions requested by users as well as to maintain the code. Finally, to make the SL more accessible, a graphical user interface could be developed on top of the core code where all configurable input would be provided without requiring any programming skills from the users.

7.3 Validation

The SL proposed would represent a direct consequence of the research presented in this thesis, which was developed in collaboration with company representatives with real applications in mind and making use of real-world data to test our methods. Therefore, the core of our proposed SL has already been tested and validated in real-world cases. Quality tests were performed against other methods from the academic literature and also the company's own historical solutions. Further validation will be necessary during an integration phase when new users begin using our SL with different input configurations.

Appendix A

2E-LRP2L related formulations

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 parameters and variables:

Parameters:

D_c is the demand of customer $c \in C$.

Q_i is the capacity of depot $i \in P \cup S$.

q^1 and q^2 are the capacities of first- and second-echelon vehicles, respectively.

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

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

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

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

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

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

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

Decision variables:

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

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

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

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

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

which are subject to

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

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

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

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

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

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

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

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

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

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

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

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

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

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 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:

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

which are subject to

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

$$\sum_{j\in P}\sum_{i\in S^o} x_{ji}^k \le v_k, \qquad \forall k \in K^1 \tag{A.17}$$

$$\sum_{k\in K^1}\sum_{j\in V}(x_{ij}^k - x_{ji}^k) = 0, \qquad \forall i \in V \tag{A.18}$$

$$\sum_{i\in S^o} x_{ji}^k \le y_j, \qquad \forall j \in P, k \in K^1 \tag{A.19}$$

$$\sum_{j\in V} l_{ij}^k = \sum_{j\in V}(l_{ji}^k - D_i' x_{ij}^k), \qquad \forall i \in S^o, k \in K^1 \tag{A.20}$$

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

$$\sum_{k\in K^1}\sum_{i\in S^o} l_{ji}^k \le Q_j y_j, \qquad \forall j \in P \tag{A.22}$$

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

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

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

$$l_{ji}^k \ge 0, \qquad \forall i,j \in V, k \in K^1 \tag{A.26}$$

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.

Appendix B

Load plan method for truck loading with partial overlapping

This appendix extends the load plan method proposed in Chapter 3. For each route generated, one must verify whether or not it is possible to load the corresponding machines onto the truck without violating certain restrictions. The load plan introduced in this appendix considers not only the two-dimensional shape of machines but also the possibility to partially overlap some machines and the non-convex shape of the loading surface. The aim of this appendix is to develop a competitive and automated algorithm capable of efficiently classifying load plans in a very restrictive scenario.

This appendix is based on the peer-reviewed publication Gandra et al. (2021a)[1]. For this publication, Tony Wauters provided insights concerning the design of the load plan method, Hatice Çalık contributed to the text and figures while Greet Vanden Berghe supervised the research.

[1] V. Gandra et al. (2021a). A heuristic approach to feasibility verification for truck loading. In: *Proceedings of the Genetic and Evolutionary Computation Conference Companion*, pp. 1564–1569.

B.1 Introduction

The feasibility verification and machine packing problem (FVMP) introduced in this appendix emerges from a necessity to automate decisions concerning whether or not a given set of machines can be loaded onto a non-convex truck loading surface before physically performing those loading operations in reality. To better appreciate how and why this problem is relevant, we will first provide a broader description of the operations of the company in question.

The company which inspired this research owns a set of non-identical machine types which they rent to their customers. A fleet of identical trucks is employed by the company to pickup and deliver machines from/to the depots or customer locations on a daily basis. Each customer uses the requested machines for a number of days before requiring those machines to be picked up once the rental period ends. On a given day, it is possible to have both pickup and delivery requests from different customers. A machine to be delivered to a customer does not necessarily depart from a depot, but can instead be picked up from another customer on the same day. In order to minimize transportation costs, it is desirable to transport multiple machines in the same truck whenever possible and profitable. However, this requires ensuring the feasibility of loading multiple machines in a truck before it departs on its journey and, ideally, during the planning phase of the routes.

Most of the considered machines have irregular shapes with arms or forks, as illustrated in Figure B.1. The total length of these machines is divided into its wheelbase and these special parts. These special parts allow the vertical projection of machines to overlap with others when loading them onto trucks. Other machines have their length completely defined by their wheelbase, as illustrated in Figure B.2. During transportation, the machines may assume different forms by (un)folding these arms or raising their forks to better position themselves on the truck's loading surface.

Trucks have a regular surface on which any machine can be loaded. An additional loading surface can be occupied over the gooseneck by only a subset of machines and by respecting the relevant restriction which will be detailed in Section B.2.

The company which introduced this problem employs a routing algorithm for scheduling the pickup and delivery of machines. To validate the loading feasibility at each pickup location, the routing algorithm considers only a scalar value to represent the area occupied by each machine on the truck loading surfaces. These values, referred to as the truck occupation percentages, are calculated based on the observations and knowledge of truck drivers. It is thus far from being an automated process at present. Henceforward we address to this method as the predicted percentage method (PPM). Although the

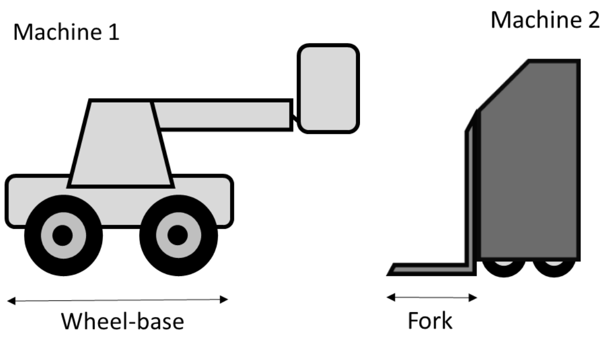


Figure B.1: Machines with an arm (machine 1) and with a fork (machine 2).

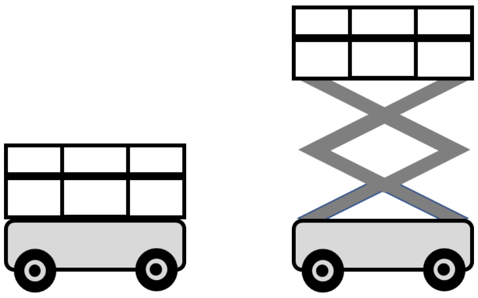


Figure B.2: Machines with a lift which can be folded and unfolded.

PPM percentages are calculated by taking into account the dimensions of the machines, these scalar values are not sufficiently accurate to separate feasible and infeasible loading combinations. As a result, the routing algorithm with the PPM ends up scheduling many routes which are infeasible with respect to loading constraints which results in delayed deliveries and the use of additional trucks.

The company in question owns nearly 1700 different types of machine and new machine types are constantly being added to their inventory. Therefore, it is impractical to physically enumerate all possible feasible loading configurations (with all subsets of machines) in a pre-processing step prior to the routing algorithm. In fact, since routing algorithms often perform hundreds of thousands iterations, see for instance the genetic algorithm by Vidal et al. (2012), any routing feasibility verification should be carried out as fast as possible to avoid excessive runtimes. The challenge is thus to develop a more effective yet fast way of checking the feasibility of truck load plans.

The algorithm introduced in Section B.4 considers machines as two-dimensional (width, length) items and a truck with two different loading surfaces. The foundation for the heuristic algorithm to solve this very challenging problem is the two-dimensional *best-fit heuristic* of Burke et al. (2004). However, given the problem's special restrictions, the best-fit heuristic alone is not enough to serve as an accurate feasibility check. In this appendix we propose an extension of the best-fit heuristic which is able to handle tailored constraints to better adapt to the company's scenario.

In order to maintain the developed heuristic's efficiency, we make a series of assumptions and simplifications regarding the real-world problem. Although rarely, due to these assumptions, the algorithm can yield load plans which are infeasible in reality. Therefore, as a means of evaluating the effectiveness of the algorithm, we conduct a classification study. The motivation for this classification study is briefly outlined in what follows.

The company provided us with real-world data concerning their machines and potential load plans. Prior to executing any packing method, we do not have any knowledge on the feasibility of a potential load plan in reality. Therefore, there are two possible outcomes in reality: (i) a positive outcome, that is, there exists a feasible loading scheme and (ii) a negative outcome, that is, there exists no feasible loading scheme. In theory, an exact method for solving the problem with all its given elements and without any simplifications or assumptions always returns the actual reality. In that case, the algorithmic outcomes can be classified into two types: true positive (the method finds a feasible loading scheme and a feasible scheme exits in reality) or true negative (the method terminates with no feasible schemes and there exists no feasible scheme in reality). The outcomes of a heuristic algorithm for the same problem include an additional classification type: false negative (the method terminates with no feasible schemes while there exits a feasible scheme in reality).

Finally, a heuristic method solving a simplified version of the problem with certain assumptions can provide a fourth type of classification: false positive (the method terminates with a feasible loading scheme while there exists no feasible scheme in reality). The algorithm we introduce in this appendix and the two other algorithms used for computational comparison all fall into this last group of heuristic methods. In these experiments, our approach significantly outperforms the PPM correctly classifying 90% of the potential load plans (the percentage of true positives and true negatives), which is considered highly satisfactory by the company.

The remainder of this appendix is organized as follows. Section B.2 describes the problem in detail. Section B.3 briefly reviews the most relevant studies in the literature. Section B.4 details the algorithm we develop, while Section B.5

presents the results of the computational study we conducted. Finally, Section B.6 concludes the appendix and outlines some future research directions.

B.2 Problem description

The feasibility verification and machine packing problem (FVMP) consists of two main components: a truck and a set of machines. Trucks have two areas onto which machines can be loaded. The first is simply the regular loading area, which is 10 meters long and 2.5 meters wide. The second loading area, which is referred to as the *gooseneck*, extends the length of the truck by an additional 3 meters. As illustrated in Figure B.3, the gooseneck is not at the same height as the regular loading surface and therefore not every machine can occupy this part of the truck.

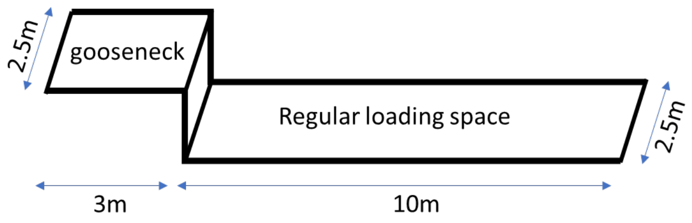


Figure B.3: The gooseneck and the regular loading surface of the truck.

Certain machines can occupy the gooseneck while positioned in the regular loading surface, as illustrated in Figure B.4. Other machines can be entirely placed on the gooseneck if their length is not greater than that of the gooseneck and if they have wheels enabling them to climb onto the gooseneck with the assistance of a portable or build-in ramp (see Figure B.5).

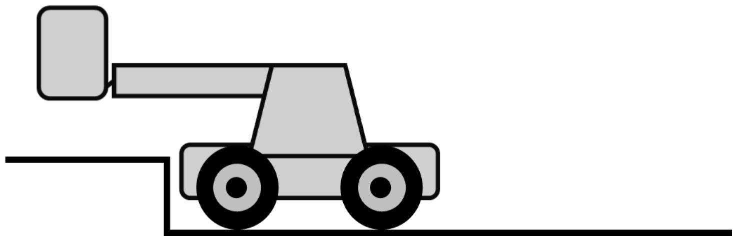


Figure B.4: Illustration of a machine partially occupying the gooseneck.

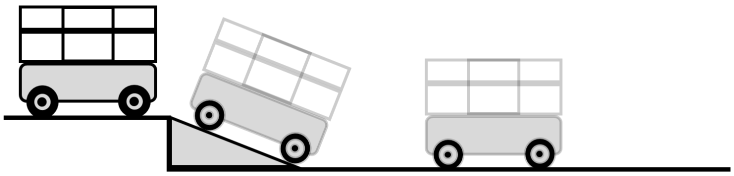


Figure B.5: Illustration of a machine placed entirely on the gooseneck.

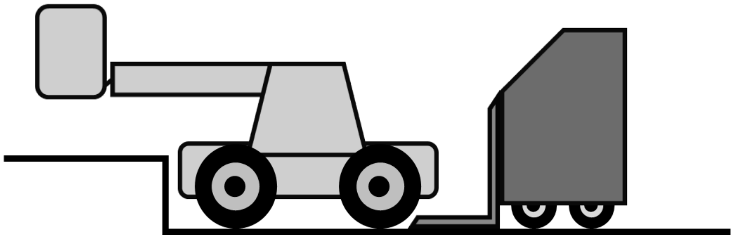


Figure B.6: Illustration of two machines partially overlapping, where the fork of one machine goes under the wheel base of the other.

Most machines do not have a fixed shape. In fact, many machines have articulated arms which can be folded into several forms, some of which can be cuboid. Meanwhile, some machines have forks which can partially go under the arms or wheel base of other machines (see Figure B.6). Two length values are considered per machine: the length of the wheelbase (WBL) and the total length (TL). Therefore, a partial overlapping of the projection of the machines is sometimes possible.

Figure B.7 illustrates just one example of such overlap. In this example machine 1 has a small wheelbase when compared to its total length. However, not all of its extra length can be overlapping another machine. While part of the arm is suspended above the loading surface allowing overlap with shorter machines, the cage (on the end of its articulated arm) is located at ground level forbidding any overlap. Thus, only a portion of the difference between the machine's wheelbase length and total length may overlap with another machine. In Figure B.7, by placing machine 2 over the gooseneck a feasible solution is possible without overlapping. On the other hand, the gooseneck in Figure B.8 is occupied and only by overlapping machines a feasible load plan would be possible.

A feasible load plan for the FVMP must take into account the following constraints:

- All machines must fit onto the loading surface of the truck.

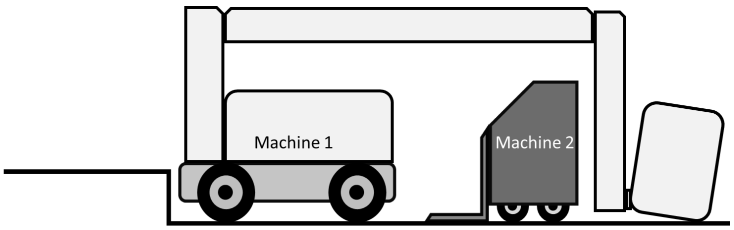


Figure B.7: Illustration of an overlap where one machine is placed entirely under the arm of another.

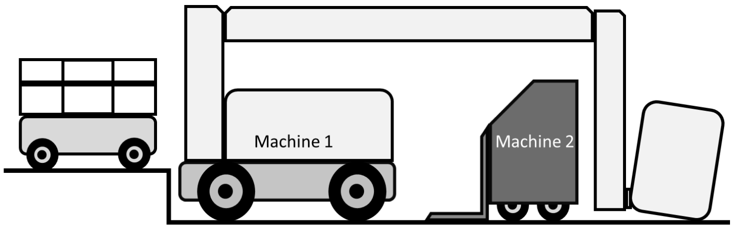


Figure B.8: A loading scheme with overlap and use of the gooseneck.

- For safety reasons there must be open space between two machines that are positioned side by side.
- All load plans are subjected to maximum weight limits.
- The wheelbase of machines should never overlap.
- Machines must not be stacked.
- Given the need to drive the machines onto the truck, a fixed orientation is defined. Machines either drive forward onto the truck or reverse onto the truck, but never sideways.

B.3 Related work

The FVMP combines several components from different *Cutting and Packing* problems (Wäscher et al., 2007). However, it is difficult to fit the FVMP into a single problem category without over-generalizing. In relation to the typology provided by Wäscher et al. (2007), the FVMP is a three-dimensional (3D) packing problem with a single large object (all dimensions fixed) and a few small heterogeneous items of irregular shapes. The FVMP is a feasibility problem where the goal is to decide whether or not a feasible packing of the small items into the larger object is possible. Therefore, the FVMP does not aim to minimize or maximize any objective function.

The FVMP can be considered as a variant of the container loading problem (CLP) where rectangular-shaped items, which are often referred to as boxes, are placed in a cuboid container (Bortfeldt and Wäscher, 2013). In the 2D case of the CLP, items can neither be stacked on top of each other nor are they allowed to overlap. A typical example of the 2D-CLP is the pallet loading problem (Ram, 1992). Stacking is allowed in the 3D version of the problem (Zhao et al., 2016).

The irregular shape of machines in the FVMP resembles the 3D irregular packing problem where items of irregular shapes are packed into a cuboid (Litvinchev et al., 2019). The machines can also be treated as tetris-like items (Fasano, 2014). However, the FVMP differs from all the aforementioned problems as its loading surface is not convex. The machines cannot be suspended in the air and should be standing on their wheels with a fixed orientation. Additionally, the structure of the truck's loading surface introduces additional restrictions which do not exist in the aforementioned studies. Although this structure resembles a multi-compartment CLP variant (Júnior et al., 2019), the fact that some machines can partially occupy both loading surfaces distinguishes the FVMP from the traditional multi-compartment CLP problems.

Despite the distinguishing properties mentioned earlier, it is clearly possible to utilize or adapt some of the methods from the irregular 3D packing literature to address the FVMP. However, we foresee that these methods would require much more processing time which is undesirable as it makes the overall routing and feasibility checking process highly inefficient.

The company is using an off-the-shelf vehicle routing algorithm for the pickup and delivery of machines while the FVMP is solved for each route to ensure its feasibility or to detect infeasibilities and re-route accordingly. However, one might consider tackling these two problems, namely the routing and the loading problems, with the help of integrated methods. When we examine the literature, we see examples of such methods for combined vehicle routing and container loading problems with 2D or 3D rectangular-shaped items. To the best of our knowledge, no such method is readily available for solving the combined irregular 3D loading and the pickup and delivery problem.

For interested readers, we refer to Leao et al. (2020) for a recent review on the irregular packing problems including those with irregular containers, to Nascimento et al. (2021) for a review of the CLP with practical constraints and to Pollaris et al. (2015) for a review of combined CLP and vehicle routing problems.

B.4 Load plan method

This appendix introduces the load plan method (LPM) which uses a placement heuristic and determines the placement (position) of items inside trucks. This method should be capable of being efficiently embedded within the company's routing algorithm and will be called frequently when creating routes. Therefore, it is important for the LPM to produce high quality load plans in low computational runtimes.

An adaptation of the best-fit placement heuristic proposed by Burke et al. (2004) is used in this appendix which follows the efficient implementation by Imahori and Yagiura (2010). This approach was chosen given its simplicity to implement, computational performance and the high-quality solutions generated. The best-fit heuristic is a constructive method. Given a list of items and a container, the method finds the position furthest from the rear of the container and places the item that fits best in it. Items are iterated over in a predetermined order until one item can be successfully placed. Once an item is inserted, the furthest position is updated and the placement continues until all items have been loaded.

The implementation in this appendix sorts machines based on their width, length and area. Ties between machines with the same dimension are broken by either length or width. When placing a machine inside the truck, one of two insertion policies can be followed: insert the machine next to the longest neighbor machine or insert the machine next to the neighbor with the most similar length. The LPM combines each machine ordering with an insertion policy and returns the first feasible load plan generated.

To further adapt our load plan method so that it can accommodate the specific characteristics of the problem presented in Section B.2, we introduce the following parameters:

- *Extra width* (exW): for safety reasons, a certain amount of extra space must be preserved between machines when loaded side by side. In order to ensure this safety distance is respected, we extend machines' width by exW if their original width is below a threshold max_w. This extension is not needed for wider machines as it will never be possible to load multiple wide machines side by side.

- *Minimum machine length required to partially occupy the gooseneck* (*min_gNeck*): given the difference in height, only large machines are able to occupy both the loading surface and the gooseneck. Parameter *min_gNeck* indicates how big a machine must be to be positioned partially on the gooseneck and partially on the loading surface.
- *Maximum weight* (*max_weight*): the considered machines are often large and heavy, which may lead to infeasible load plans due to their total weight instead of the physical space they occupy. We set *max_weight* based on the maximum weight regulations in effect in the country where the company provides services.
- *Reduced length percentage* (*reduced_len%*): in addition to the total length, the wheelbase length is provided. While the wheelbase indicates a portion of the machine which can never overlap with another, the machine's arms or forks can often overlap depending on the combination of machines. Using *reduced_len%* we define a new machine length to be considered which is outlined in Equation (B.1). This new length is equal to the total length if *reduced_len%* = 0 and equals the wheel base if *reduced_len%* = 1.

$$RL = TL - ((TL - WBL) * reduced\_len\%) \tag{B.1}$$

Note that the company currently uses the PPM to solve a simplified version of the problem which does not consider the newly introduced LPM parameters. Given a set of machines returned by the PPM, truck drivers rely on their experience to generate a load plan and do not explicitly consider constraints associated with the introduced parameters. These parameters are thus specific to the LPM and have not previously been exploited by the company. Therefore, a careful calibration of these parameters is essential to generate a model well tuned to the problem-specific scenario and capable of efficiently classifying load plans.

In a scenario where LPM is called repeatedly, the load plan, whether it is feasible or not, can be stored in a memory structure referred to as Trie (Leung et al., 2010). Trie avoids evaluating routes that have already had their load plan checked, significantly accelerating the runtime of the overall algorithm (vehicle routing plus LPM).

B.5 Computational study

This section presents computational experiments to validate the performance of the developed approach and compare the LPM with the current practice. The company provided us with the dimensions and characteristics of their machines: length, width, height, weight, wheelbase length and predicted percentage. A set of 196 candidate load plans, including both feasible and infeasible configurations, was also provided with the classification of experienced human operators.

Table B.1 provides the necessary parameters for the LPM. The value of max_weight should be selected so as to remain compliant with national regulations. We set this value to 22 tons in our experiments which corresponds to the value stipulated in Belgian legislation. We should note here that the values of these parameters will vary from country to country and have a significant impact on possible combinations. Using a random subset of the available instances, all the remaining parameters were calibrated using *irace* (López-Ibáñez et al., 2016).

Table B.1: LPM parameters and their values

Parameter	Value
exW	25 cm
max_w	132 cm
min_gNeck	744 cm
$reduced\_len\%$	0.47
max_weight	22 tons

The goal of the LPM is to classify load plans as (in)feasible with more precision than the PPM, which simplifies the real dimensions of machines. The LPM is thus a decision method which divides machine combinations into two classes: feasible and infeasible load plans. We compare the accuracy of three different methods: (i) the predicted percentage method currently employed by the company, (ii) the LPM-Base method which corresponds to the best-fit heuristic proposed by Burke et al. (2004) for solving 2D packing problems and (iii) the LPM method which we adapted from LPM-Base by integrating all of the extra parameters described in Section B.4. For the purpose of this comparison, Table B.3 provides the *confusion matrix* of each method where columns TP, FP, TN and FN correspond to each classification method's number of true positives, false positives, true negatives and false negatives, respectively. We provide the precise meaning of these categories in Table B.2.

To assess the performance of each method, well-known metrics to compare binary classifiers are calculated. The F1-score measures the method's accuracy and is calculated as per Equation (B.2).

$$F1 - Score = \frac{TP}{TP + 0.5(FP + FN)} \tag{B.2}$$

Table B.2: Confusion categories.

Confusion parameter	Abbreviation	Method's decision	Actuality
True positive	TP	Feasible	Feasible
False positive	FP	Feasible	Infeasible
True negative	TN	Infeasible	Infeasible
False negative	FN	Infeasible	Feasible

Since F1-score do not take into account the number of true negatives, a second metric, Matthews Correlation Coefficient (MCC), is also employed and is calculated as per Equation (B.3).

$$MCC = \frac{(TP \times TN) - (FP \times FN)}{\sqrt{(TP + FP)(TP + FN)(TN + FP)(TN + FN)}} \tag{B.3}$$

Table B.3: Confusion matrix of the loading methods.

Loading method	TP	FP	TN	FN	F1-score	MCC
PPM	74	16	83	23	0.791	0.603
LPM-Base	82	19	80	15	0.828	0.654
LPM	87	10	89	10	0.897	0.796

The PPM obtains the worst results, misclassifying up to 20% of the considered instances. Since operators often assign a higher predicted percentage to machines, this method results in many feasible routes having a total predicted percentage greater than 100%. This leads to many false negatives.

LPM-base considers machines as 2D rectangles, but does not take into account constraints related to the gooseneck or safety concerns in regards to width. Consequently, this method also results in a high number of false positives. In other words, many combinations that LPM-base considers feasible are not feasible in reality. Moreover, when ignoring the possibility of machine overlapping many feasible solutions are misclassified as infeasible, which helps explain most of LPM-Base's false negatives.

Although LPM does not correctly classify all instances, it exhibits better accuracy across all metrics and classifies approximately 90% of the instances correctly. By employing this method, the company is able to significantly improve their routing plans. Moreover, LPM was able to discover several feasible load plans which were unknown to the company's operators prior to this study.

When we take a closer look at the misclassifications by LPM, we observe that these mostly occur due to special cases where operators push the limits of safety restrictions by permitting extra width or increasing maximum weight. For false negative cases, some machines can make better use of the gooseneck in reality because the LPM method is overly restrictive when calculating the amount of space they can occupy over the gooseneck. Similarly, overlapping machines may occupy a larger area in reality than the one allowed by the LPM method.

B.6 Conclusion

This appendix introduced the feasibility verification in machine packing problem (FVMP): a real-world container loading problem where one must decide whether or not a given set of non-identical irregular machines can be loaded onto a non-convex truck loading surface without violating several practical restrictions. The set of machines include a variety of forklifts, arms and cages as well as lifts that can be folded. The practical constraints include maximum weight restrictions, fixed orientation for machines and a minimum amount of empty space between the machines. Specific characteristics of the FVMP allow a partial overlapping of some machines which increases the possibility of feasibly loading the machines.

The FVMP must be solved very frequently within a computationally demanding routing algorithm. Therefore, the method for solving the FVMP should not only be highly accurate but also very efficient. In order to solve the FVMP, we developed an algorithm based on the *best-fit placement heuristic* which was originally introduced for solving the two-dimensional rectangular packing problem. The results of computational experiments conducted using real-world data indicated that the new algorithm outperforms the approach currently employed by the company.

Regarding future work, some of the defined parameters can be applied with different values depending on the machine type. For instance, for a forklift, everything which is not wheelbase (the fork) may be completely overlapped. By contrast, the articulated arm and cage of certain machines can only be partially overlapped, which means these machines needs to have a different percentage of overlapping.

The analysis and methods in this appendix focused only on the feasibility verification of loading a set of machines associated with a given route onto a truck. Future research should also focus on problem extensions which combine loading and routing decisions and address them with integrated methods. This will necessarily result in the loading problem becoming even more complex since when routing trucks internationally each country may have different loading regulations which need to be adhered to.

Bibliography

Alegre, J., Laguna, M., and Pacheco, J. (2007). Optimizing the periodic pick-up of raw materials for a manufacturer of auto parts. In: *European Journal of Operational Research* 179.3, pp. 736–746.

Baldacci, R., Bartolini, E., Mingozzi, A., and Valletta, A. (2011). An exact algorithm for the period routing problem. In: *Operations research* 59.1, pp. 228–241.

Barrera, D., Velasco, N., and Amaya, C.-A. (2012). A network-based approach to the multi-activity combined timetabling and crew scheduling problem: Workforce scheduling for public health policy implementation. In: *Computers & Industrial Engineering* 63.4, pp. 802–812.

Battarra, M., Cordeau, J.-F., and Iori, M. (2014). "Chapter 6: pickup-and-delivery problems for goods transportation". In: *Vehicle Routing: Problems, Methods, and Applications, Second Edition.* SIAM, pp. 161–191.

Beltrami, E. J. and Bodin, L. D. (1974). Networks and vehicle routing for municipal waste collection. In: *Networks* 4.1, pp. 65–94.

Boccia, M., Crainic, T. G., Sforza, A., and Sterle, C. (2010). A metaheuristic for a two echelon location-routing problem. In: *International Symposium on Experimental Algorithms.* Springer, pp. 288–301.

Boccia, M., Crainic, T. G., Sforza, A., and Sterle, C. (2011). Location-routing models for designing a two-echelon freight distribution system. In: *Rapport technique, CIRRELT, Université de Montréal*, p. 91.

Bortfeldt, A. and Wäscher, G. (2013). Constraints in container loading–A state-of-the-art review. In: *European Journal of Operational Research* 229.1, pp. 1–20.

Bostel, N., Dejax, P., Guez, P., and Tricoire, F. (2008). “Multiperiod planning and routing on a rolling horizon for field force optimization logistics”. In: *The vehicle routing problem: latest advances and new challenges.* Springer, pp. 503–525.

Breunig, U., Schmid, V., Hartl, R. F., and Vidal, T. (2016). A large neighbourhood based heuristic for two-echelon routing problems. In: *Computers & Operations Research* 76, pp. 208–225.

Burke, E. K. and Bykov, Y. (2017). The late acceptance hill-climbing heuristic. In: *European Journal of Operational Research* 258.1, pp. 70–78.

Burke, E. K., Kendall, G., and Whitwell, G. (2004). A new placement heuristic for the orthogonal stock-cutting problem. In: *Operations Research* 52.4, pp. 655–671.

Campbell, A. M. and Wilson, J. H. (2014). Forty years of periodic vehicle routing. In: *Networks* 63.1, pp. 2–15.

Cantu-Funes, R., Salazar-Aguilar, M. A., and Boyer, V. (2017). Multi-depot periodic vehicle routing problem with due dates and time windows. In: *Journal of the operational research society*, pp. 1–12.

Caramia, M. and Guerriero, F. (2010). A heuristic approach for the truck and trailer routing problem. In: *Journal of the Operational Research Society* 61.7, pp. 1168–1180.

Castillo-Salazar, J. A., Landa-Silva, D., and Qu, R. (2016). Workforce scheduling and routing problems: literature survey and computational study. In: *Annals of Operations Research* 239.1, pp. 39–67.

Chao, I.-M. (2002). A tabu search method for the truck and trailer routing problem. In: *Computers & Operations Research* 29.1, pp. 33–51.

Christiaens, J., Çalik, H., Wauters, T., Chandrasekharan, R. C., and Berghe, G. V. (2020). The prisoner transportation problem. In: *European Journal of Operational Research* 284.3, pp. 1058–1073.

Christiaens, J. and Vanden Berghe, G. (2020). Slack induction by string removals for vehicle routing problems. In: *Transportation Science* 54.2, pp. 417–433.

Clarke, G. and Wright, J. W. (1964). Scheduling of vehicles from a central depot to a number of delivery points. In: *Operations Research* 12.4, pp. 568–581.

Contardo, C., Hemmelmayr, V., and Crainic, T. G. (2012). Lower and upper bounds for the two-echelon capacitated location-routing problem. In: *Computers & operations research* 39.12, pp. 3185–3199.

Cordeau, J.-F., Laporte, G., and Ropke, S. (2008). “Recent models and algorithms for one-to-one pickup and delivery problems”. In: *The vehicle routing problem: latest advances and new challenges.* Springer, pp. 327–357.

Cordeau, J.-F., Laporte, G., and Mercier, A. (2001). A unified tabu search heuristic for vehicle routing problems with time windows. In: *Journal of the Operational research society* 52.8, pp. 928–936.

Cordeau, J.-F., Laporte, G., Pasin, F., and Ropke, S. (2010). Scheduling technicians and tasks in a telecommunications company. In: *Journal of Scheduling* 13.4, pp. 393–409.

Cordeau, J.-F. and Maischberger, M. (2012). A parallel iterated tabu search heuristic for vehicle routing problems. In: *Computers & Operations Research* 39.9, pp. 2033–2050.

Cortés, C. E., Matamala, M., and Contardo, C. (2010). The pickup and delivery problem with transfers: Formulation and a branch-and-cut solution method. In: *European Journal of Operational Research* 200.3, pp. 711–724.

Côté, J.-F., Gendreau, M., and Potvin, J.-Y. (2014). An exact algorithm for the two-dimensional orthogonal packing problem with unloading constraints. In: *Operations Research* 62.5, pp. 1126–1141.

Cuda, R., Guastaroba, G., and Speranza, M. G. (2015). A survey on two-echelon routing problems. In: *Computers & Operations Research* 55, pp. 185–199.

Danloup, N., Allaoui, H., and Goncalves, G. (2018). A comparison of two meta-heuristics for the pickup and delivery problem with transshipment. In: *Computers & Operations Research* 100, pp. 155–171.

Dantzig, G. B. and Ramser, J. H. (1959). The truck dispatching problem. In: *Management science* 6.1, pp. 80–91.

De Bruecker, P., Bergh, J. Van den, Beliën, J., and Demeulemeester, E. (2015). Workforce planning incorporating skills: State of the art. In: *European Journal of Operational Research* 243.1, pp. 1–16.

De Jong, S. P., Smit, J., and Van Drooge, L. (2015). Scientists’ response to societal impact policies: A policy paradox. In: *Science and public policy* 43.1, pp. 102–114.

Derigs, U., Pullmann, M., and Vogel, U. (2013). Truck and trailer routing—problems, heuristics and computational experience. In: *Computers & Operations Research* 40.2, pp. 536–546.

Drexl, M. (2012). Synchronization in vehicle routing—a survey of VRPs with multiple synchronization constraints. In: *Transportation Science* 46.3, pp. 297–316.

Drexl, M. (2013). Applications of the vehicle routing problem with trailers and transshipments. In: *European Journal of Operational Research* 227.2, pp. 275–283.

Drexl, M. (2020). On the one-to-one pickup-and-delivery problem with time windows and trailers. In: *Central European Journal of Operations Research*, pp. 1–48.

Drexl, M. and Schneider, M. (2015). A survey of variants and extensions of the location-routing problem. In: *European Journal of Operational Research* 241.2, pp. 283–308.

Fasano, G. (2014). "Tetris-like items". In: *Solving Non-standard Packing Problems by Global Optimization and Heuristics*. Springer, pp. 7–26.

Fowler, R. J., Paterson, M. S., and Tanimoto, S. L. (1981). Optimal packing and covering in the plane are NP-complete. In: *Information Processing Letters* 12.3, pp. 133–137.

Francis, P. M., Smilowitz, K. R., and Tzur, M. (2008). "The period vehicle routing problem and its extensions". In: *The vehicle routing problem: latest advances and new challenges*. Springer, pp. 73–102.

Franses, P. and Post, G. (2003). Personnel scheduling in laboratories. In: *Practice and Theory of Automated Timetabling IV: 4th International Conference, PATAT 2002, Gent, Belgium, August 21-23, 2002. Selected Revised Papers 4*. Springer, pp. 113–119.

Fuellerer, G., Doerner, K. F., Hartl, R. F., and Iori, M. (2009). Ant colony optimization for the two-dimensional loading vehicle routing problem. In: *Computers & Operations Research* 36.3, pp. 655–673.

Gandra, V., Çalik, H., Wauters, T., and Berghe, G. V. (2021a). A heuristic approach to feasibility verification for truck loading. In: *Proceedings of the Genetic and Evolutionary Computation Conference Companion*, pp. 1564–1569.

Gandra, V., Sartori, C. S., Çalık, H., and Smet, P. (2022a). Multi-depot periodic vehicle routing with variable visit patterns. In: *Proceedings of the Genetic and Evolutionary Computation Conference Companion*, pp. 2019–2026.

Gandra, V. M. S., Çalık, H., Wauters, T., Toffolo, T. A., Carvalho, M. A. M., and Berghe, G. V. (2021b). The impact of loading restrictions on the two-echelon location routing problem. In: *Computers & Industrial Engineering* 160, p. 107609.

Gandra, V. S., Çalık, H., Toffolo, T. A., Carvalho, M. A. M., and Berghe, G. V. (2022b). The vessel swap-body routing problem. In: *European Journal of Operational Research* 303.1, pp. 354–369.

Gaudioso, M. and Paletta, G. (1992). A heuristic for the periodic vehicle routing problem. In: *Transportation Science* 26.2, pp. 86–92.

Gendreau, M., Iori, M., Laporte, G., and Martello, S. (2006). A tabu search algorithm for a routing and container loading problem. In: *Transportation Science* 40.3, pp. 342–350.

Gendreau, M., Iori, M., Laporte, G., and Martello, S. (2008). A Tabu search heuristic for the vehicle routing problem with two-dimensional loading constraints. In: *Networks: An International Journal* 51.1, pp. 4–18.

Hadjiconstantinou, E. and Baldacci, R. (1998). A multi-depot period vehicle routing problem arising in the utilities sector. In: *Journal of the Operational Research Society* 49.12, pp. 1239–1248.

Hashimoto, H., Boussier, S., Vasquez, M., and Wilbaut, C. (2011). A GRASP-based approach for technicians and interventions scheduling for telecommunications. In: *Annals of Operations Research* 183.1, pp. 143–161.

Imahori, S. and Yagiura, M. (2010). The best-fit heuristic for the rectangular strip packing problem: An efficient implementation and the worst-case approximation ratio. In: *Computers & Operations Research* 37.2, pp. 325–333.

Iori, M. and Martello, S. (2010). Routing problems with loading constraints. In: *Top* 18.1, pp. 4–27.

Iori, M., Salazar-González, J.-J., and Vigo, D. (2007). An exact approach for the vehicle routing problem with two-dimensional loading constraints. In: *Transportation Science* 41.2, pp. 253–264.

Jacobsen, S. K. and Madsen, O. B. (1980). A comparative study of heuristics for a two-level routing-location problem. In: *European Journal of Operational Research* 5.6, pp. 378–387.

Júnior, R. R., Yanasse, H. H., Morabito, R., and Junqueira, L. (2019). A hybrid approach for a multi-compartment container loading problem. In: *Expert Systems with Applications* 137, pp. 471–492.

Kovacs, A. A., Parragh, S. N., Doerner, K. F., and Hartl, R. F. (2012). Adaptive large neighborhood search for service technician routing and scheduling problems. In: *Journal of scheduling* 15.5, pp. 579–600.

Kuhn, H. W. (1955). The Hungarian method for the assignment problem. In: *Naval research logistics quarterly* 2.1-2, pp. 83–97.

Kulkarni, R. and Bhave, P. R. (1985). Integer programming formulations of vehicle routing problems. In: *European Journal of Operational Research* 20.1, pp. 58–67.

Kumar, S. N. and Panneerselvam, R. (2012). A survey on the vehicle routing problem and its variants. In: *Intelligent Information Management* 4.03, p. 66.

Laporte, G. (2009). Fifty years of vehicle routing. In: *Transportation Science* 43.4, pp. 408–416.

Leao, A. A., Toledo, F. M., Oliveira, J. F., Carravilla, M. A., and Alvarez-Valdés, R. (2020). Irregular packing problems: a review of mathematical models. In: *European Journal of Operational Research* 282.3, pp. 803–822.

Leung, S. C., Zheng, J., Zhang, D., and Zhou, X. (2010). Simulated annealing for the vehicle routing problem with two-dimensional loading constraints. In: *Flexible Services and Manufacturing Journal* 22.1-2, pp. 61–82.

Leung, S. C., Zhou, X., Zhang, D., and Zheng, J. (2011). Extended guided tabu search and a new packing algorithm for the two-dimensional loading vehicle routing problem. In: *Computers & Operations Research* 38.1, pp. 205–215.

Li, Y., Lim, A., and Rodrigues, B. (2005). Manpower allocation with time windows and job-teaming constraints. In: *Naval Research Logistics (NRL)* 52.4, pp. 302–311.

Lin, S.-W., Vincent, F. Y., and Chou, S.-Y. (2009). Solving the truck and trailer routing problem based on a simulated annealing heuristic. In: *Computers & Operations Research* 36.5, pp. 1683–1692.

Lin, S.-W., Vincent, F. Y., and Lu, C.-C. (2011). A simulated annealing heuristic for the truck and trailer routing problem with time windows. In: *Expert Systems with Applications* 38.12, pp. 15244–15252.

Litvinchev, I, Pankratov, A, and Romanova, T (2019). 3D irregular packing in an optimized cuboid container. In: *IFAC-PapersOnLine* 52.13, pp. 2014–2019.

López-Ibáñez, M., Dubois-Lacoste, J., Cáceres, L. P., Birattari, M., and Stützle, T. (2016). The irace package: Iterated racing for automatic algorithm configuration. In: *Operations Research Perspectives* 3, pp. 43–58.

Lourenço, H. R., Martin, O. C., and Stützle, T. (2003). "Iterated local search". In: *Handbook of Metaheuristics*. Springer, pp. 320–353.

Lourenço, H. R., Martin, O. C., and Stützle, T. (2019). "Iterated local search: Framework and applications". In: *Handbook of metaheuristics*. Springer, pp. 129–168.

Luxen, D. and Vetter, C. (2011). Real-time routing with OpenStreetMap data. In: *Proceedings of the 19th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems*. GIS '11. Chicago, Illinois: ACM, pp. 513–516. DOI: 10.1145/2093973.2094062.

Masson, R., Lehuédé, F., and Péton, O. (2013a). An adaptive large neighborhood search for the pickup and delivery problem with transfers. In: *Transportation Science* 47.3, pp. 344–355.

Masson, R., Lehuédé, F., and Péton, O. (2013b). Efficient feasibility testing for request insertion in the pickup and delivery problem with transfers. In: *Operations Research Letters* 41.3, pp. 211–215.

Masson, R., Lehuédé, F., and Péton, O. (2014). The dial-a-ride problem with transfers. In: *Computers & Operations Research* 41, pp. 12–23.

Mendoza, J., Guéret, C., Hoskins, M., Pillac, V., Vidal, T., and Vigo, D. (2014). VRP-REP: the vehicle routing community repository. Third meeting of the EURO Working Group on Vehicle Routing and Logistics Optimization (VeRoLog). Oslo, Norway. Available online at http://www.vrp-rep.org/datasets/item/2017-0019.html. Last access on: 2022-02-28.

Mitrović-Minić, S. and Laporte, G. (2006). The pickup and delivery problem with time windows and transshipment. In: *INFOR: Information Systems and Operational Research* 44.3, pp. 217–227.

Nagy, G. and Salhi, S. (2007). Location-routing: Issues, models and methods. In: *European Journal of Operational Research* 177.2, pp. 649–672.

Nascimento, O. X., Queiroz, T. A. de, and Junqueira, L. (2021). Practical constraints in the container loading problem: Comprehensive formulations and exact algorithm. In: *Computers & Operations Research* 128, p. 105186.

Nguyen, P. K., Crainic, T. G., and Toulouse, M. (2014). A hybrid generational genetic algorithm for the periodic vehicle routing problem with time windows. In: *Journal of Heuristics* 20.4, pp. 383–416.

Nguyen, V.-P., Prins, C., and Prodhon, C. (2012a). A multi-start iterated local search with tabu list and path relinking for the two-echelon location-routing problem. In: *Engineering Applications of Artificial Intelligence* 25.1, pp. 56–71.

Nguyen, V.-P., Prins, C., and Prodhon, C. (2012b). Solving the two-echelon location routing problem by a GRASP reinforced by a learning process and path relinking. In: *European Journal of Operational Research* 216.1, pp. 113–126.

Öncan, T., Altınel, I. K., and Laporte, G. (2009). A comparative analysis of several asymmetric traveling salesman problem formulations. In: *Computers & Operations Research* 36.3, pp. 637–654.

OpenStreetMap (2022). OSM contributors. Planet dump retrieved from https://planet.osm.org. Available online at `https://www.openstreetmap.org`. Last access on: 2022-02-22.

Paraskevopoulos, D. C., Laporte, G., Repoussis, P. P., and Tarantilis, C. D. (2017). Resource constrained routing and scheduling: Review and research prospects. In: *European Journal of Operational Research* 263.3, pp. 737–754.

Parragh, S. N. and Cordeau, J.-F. (2017). Branch-and-price and adaptive large neighborhood search for the truck and trailer routing problem with time windows. In: *Computers & Operations Research* 83, pp. 28–44.

Parragh, S. N., Doerner, K. F., and Hartl, R. F. (2008). A survey on pickup and delivery problems (Part II: Transportation between pickup and delivery locations). In: *Journal für Betriebswirtschaft* 58.2, pp. 81–117.

Passmark (accessed April 26, 2022). *CPU benchmarks*. `http://www.cpubenchmark.net/cpu_list.php`.

Perboli, G., Tadei, R., and Vigo, D. (2011). The two-echelon capacitated vehicle routing problem: Models and math-based heuristics. In: *Transportation Science* 45.3, pp. 364–380.

Pichka, K., Bajgiran, A. H., Petering, M. E., Jang, J., and Yue, X. (2018). The two echelon open location routing problem: Mathematical model and hybrid heuristic. In: *Computers & Industrial Engineering* 121, pp. 97–112.

Pillac, V., Gueret, C., and Medaglia, A. L. (2013). A parallel matheuristic for the technician routing and scheduling problem. In: *Optimization Letters* 7.7, pp. 1525–1535.

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

Pollaris, H., Braekers, K., Caris, A., Janssens, G. K., and Limbourg, S. (2015). Vehicle routing problems with loading constraints: state-of-the-art and future directions. In: *OR Spectrum* 37.2, pp. 297–330.

Potvin, J.-Y. and Rousseau, J.-M. (1995). An exchange heuristic for routeing problems with time windows. In: *Journal of the Operational Research Society* 46.12, pp. 1433–1446.

Prodhon, C. and Prins, C. (2014). A survey of recent research on location-routing problems. In: *European Journal of Operational Research* 238.1, pp. 1–17.

Qu, Y. and Bard, J. F. (2012). A GRASP with adaptive large neighborhood search for pickup and delivery problems with transshipment. In: *Computers & Operations Research* 39.10, pp. 2439–2456.

Rahimi-Vahed, A., Crainic, T. G., Gendreau, M., and Rei, W. (2013). A path relinking algorithm for a multi-depot periodic vehicle routing problem. In: *Journal of heuristics* 19.3, pp. 497–524.

Rais, A., Alvelos, F., and Carvalho, M. S. (2014). New mixed integer-programming model for the pickup-and-delivery problem with transshipment. In: *European Journal of Operational Research* 235.3, pp. 530–539.

Ram, B. (1992). The pallet loading problem: A survey. In: *International Journal of Production Economics* 28.2, pp. 217–225.

Ropke, S. and Pisinger, D. (2006). An adaptive large neighborhood search heuristic for the pickup and delivery problem with time windows. In: *Transportation Science* 40.4, pp. 455–472.

Rothenbächer, A.-K., Drexl, M., and Irnich, S. (2018). Branch-and-price-and-cut for the truck-and-trailer routing problem with time windows. In: *Transportation Science* 52.5, pp. 1174–1190.

Salhi, S. and Nagy, G. (1999). Consistency and robustness in location-routing. In: *Studies in Locational Analysis* 13, pp. 3–19.

Salhi, S. and Rand, G. K. (1989). The effect of ignoring routes when locating depots. In: *European Journal of Operational Research* 39.2, pp. 150–156.

Savelsbergh, M. W. (1992). The vehicle routing problem with time windows: Minimizing route duration. In: *ORSA journal on computing* 4.2, pp. 146–154.

Scheuerer, S. (2006). A tabu search heuristic for the truck and trailer routing problem. In: *Computers & Operations Research* 33.4, pp. 894–909.

Schneider, M. and Drexl, M. (2017). A survey of the standard location-routing problem. In: *Annals of Operations Research* 259.1-2, pp. 389–414.

Schwengerer, M., Pirkwieser, S., and Raidl, G. R. (2012). A variable neighborhood search approach for the two-echelon location-routing problem. In: *European Conference on Evolutionary Computation in Combinatorial Optimization*. Springer, pp. 13–24.

Shiri, H, Rahmani, M, and Bafruei, M (2020). Examining the impact of transfers in pickup and delivery systems. In: *Uncertain Supply Chain Management* 8.1, pp. 207–224.

Takoudjou, R. T., Deschamps, J.-C., and Dupas, R. (2012). A MIP Formulation for the Pickup and Delivery Problem with Time Window and Transshipment. In: *IFAC Proceedings Volumes* 45.6, pp. 333–338.

Toffolo, T. A., Christiaens, J., Van Malderen, S., Wauters, T., and Vanden Berghe, G. (2018). Stochastic local search with learning automaton for the swap-body vehicle routing problem. In: *Computers & Operations Research* 89, pp. 68–81.

Tricoire, F., Bostel, N., Dejax, P., and Guez, P. (2013). Exact and hybrid methods for the multiperiod field service routing problem. In: *Central European Journal of Operations Research* 21.2, pp. 359–377.

Vidal, T., Crainic, T. G., Gendreau, M., Lahrichi, N., and Rei, W. (2012). A hybrid genetic algorithm for multidepot and periodic vehicle routing problems. In: *Operations Research* 60.3, pp. 611–624.

Vidal, T., Crainic, T. G., Gendreau, M., and Prins, C. (2014). A unified solution framework for multi-attribute vehicle routing problems. In: *European Journal of Operational Research* 234.3, pp. 658–673.

Vidal, T., Laporte, G., and Matl, P. (2019). A concise guide to existing and emerging vehicle routing problem variants. In: *European Journal of Operational Research*.

Villegas, J. G., Prins, C., Prodhon, C., Medaglia, A. L., and Velasco, N. (2013). A matheuristic for the truck and trailer routing problem. In: *European Journal of Operational Research* 230.2, pp. 231–244.

Wäscher, G., Haußner, H., and Schumann, H. (2007). An improved typology of cutting and packing problems. In: *European journal of operational research* 183.3, pp. 1109–1130.

Wei, L., Zhang, Z., Zhang, D., and Leung, S. C. (2018). A simulated annealing algorithm for the capacitated vehicle routing problem with two-dimensional loading constraints. In: *European Journal of Operational Research* 265.3, pp. 843–859.

Wei, L., Zhang, Z., Zhang, D., and Lim, A. (2015). A variable neighborhood search for the capacitated vehicle routing problem with two-dimensional loading constraints. In: *European Journal of Operational Research* 243.3, pp. 798–814.

Wolfinger, D. (2021). A Large Neighborhood Search for the Pickup and Delivery Problem with Time Windows, Split Loads and Transshipments. In: *Computers & Operations Research* 126, p. 105110.

Zachariadis, E. E., Tarantilis, C. D., and Kiranoudis, C. T. (2013). Integrated distribution and loading planning via a compact metaheuristic algorithm. In: *European Journal of Operational Research* 228.1, pp. 56–71.

Zachariadis, E. E., Tarantilis, C. D., and Kiranoudis, C. T. (2009). A guided tabu search for the vehicle routing problem with two-dimensional loading constraints. In: *European Journal of Operational Research* 195.3, pp. 729–743.

Zamorano, E. and Stolletz, R. (2017). Branch-and-price approaches for the multiperiod technician routing and scheduling problem. In: *European Journal of Operational Research* 257.1, pp. 55–68.

Zäpfel, G. and Bögl, M. (2008). Multi-period vehicle routing and crew scheduling with outsourcing options. In: *International Journal of Production Economics* 113.2, pp. 980–996.

Zhang, Y., Atasoy, B., Souravlias, D., and Negenborn, R. R. (2020). Pickup and Delivery Problem with Transshipment for Inland Waterway Transport. In: *International Conference on Computational Logistics*. Springer, pp. 18–35.

Zhao, X., Bennell, J. A., Bektaş, T., and Dowsland, K. (2016). A comparative review of 3D container loading algorithms. In: *International Transactions in Operational Research* 23.1-2, pp. 287–320.

List of publications

Articles in internationally reviewed academic journals

2022. **Gandra, V. S.**, Çalık, H., Toffolo, T. A. M., Carvalho, M. A. M., & Vanden Berghe, G. (2022). The vessel swap-body routing problem. *European Journal of Operational Research*, Vol 303, Issue 1, 354-369. `doi.org/10.1016/j.ejor.2022.02.015`.

2021. **Gandra, V. M. S.**, Çalık, H., Wauters, T., Toffolo, T. A. M., Carvalho, M. A. M., & Vanden Berghe, G. (2021). The impact of loading restrictions on the two-echelon location routing problem. *Computers & Industrial Engineering*, Vol 160, p.107609. `doi.org/10.1016/j.cie.2021.107609`.

2019. **Santos, V. G. M.**, & Carvalho, M. A. M. (2021). Tailored heuristics in adaptive large neighborhood search applied to the cutwidth minimization problem. *European Journal of Operational Research*, Vol 289, Issue 3, 1056-1066. `doi.org/10.1016/j.ejor.2019.07.013`.

2018. **Santos, V. G. M.**, & de Carvalho, M. A. M. (2018). Adaptive large neighborhood search applied to the design of electronic circuits. *Applied Soft Computing*, Vol 73, 14-23. `doi.org/10.1016/j.asoc.2018.08.017`.

Papers at international scientific conferences published in full in proceedings

2022. **Gandra, V.**, Sartori, C. S., Çalık, H., & Smet, P. (2022). Multi-depot periodic vehicle routing with variable visit patterns. *In Proceedings of the Genetic and Evolutionary Computation Conference Companion* (pp. 2019-2026). `doi.org/10.1145/3520304.3533953`.

2021. **Gandra, V.**, Çalık, H., Wauters, T., & Vanden Berghe, G. (2021). A heuristic approach to feasibility verification for truck loading. *In*

Proceedings of the Genetic and Evolutionary Computation Conference Companion (pp. 1564-1569). doi.org/10.1145/3449726.3463184.

2021. Sartori, C. S., **Gandra, V.**, Çalık, H., & Smet, P. (2021). Production Scheduling with Stock-and Staff-Related Restrictions. *In International Conference on Computational Logistics* (pp. 142-162). Springer, Cham. doi.org/10.1007/978-3-030-87672-2_10.

Conferences and scientific talks

2022. **Gandra, V.** (2022). Multi-period vehicle routing problems in logistic companies. *KU Leuven internal seminars.* Ghent, Belgium, 5 September, 2022.

2022. **Gandra, V.**, Sartori, C. S., Çalık, H., & Smet, P. (2022). Multi-depot periodic vehicle routing with variable visit patterns. *GECCO.* Boston, USA, 9-13 July, 2022.

2022. **Gandra, V.** (2022). Multi-depot periodic vehicle routing with variable visit patterns. *UFOP internal seminars.* Ouro Preto, Brazil, 9 June, 2022.

2021. **Gandra, V.** (2021). Operations Research & Society. *CODeS internal seminars.* Ghent, Belgium, 27 October, 2021.

2021. **Gandra, V.**, Çalık, H., Wauters, T., & Vanden Berghe, G. (2021). A heuristic approach to feasibility verification for truck loading. *GECCO.* Lille, France, 10-14 July, 2021.

2020. **Gandra, V.** (2020). A study on the two-echelon location routing problem with 2D loading constraints. *KU Leuven internal seminars.* Ghent, Belgium, 19 May, 2020.

2020. **Gandra, V. M. S.**, Çalık, H., Toffolo, T. A. M., Carvalho, M. A. M., & Vanden Berghe, G. (2020). A metaheuristic approach for the two-echelon location routing problem. *Annual conference of the Belgian Operational Research Society (ORBEL).* Lille, France, 30-31 January, 2020.

Data published online

- Instances for the two-echelon location routing problem with two-dimensional loading constraints: https://data.mendeley.com/datasets/hkxchx5sxp/draft?a=5ba9957a-8d42-4099-b0e0-a176be0c1fb9.

- Supplementary material for multi-depot periodic vehicle routing with variable visit patterns: `https://data.mendeley.com/datasets/s47bj3x9gm/draft?a=341f8917-a8f5-41e5-b668-c422a6230067`.
- Supplementary material for the technician routing and scheduling problem: `https://data.mendeley.com/datasets/7fy8zsy5r8/draft?a=46e6548d-a91b-406c-989a-52c3363cf471`.

FACULTY OF ENGINEERING TECHNOLOGY
DEPARTMENT OF COMPUTER SCIENCE
NUMERICAL ANALYSIS AND APPLIED MATHEMATICS
Celestijnenlaan 200A box 2402
B-3001 Leuven
vinicius.gandramartinssantos@kuleuven.be
https://wms.cs.kuleuven.be/groups/NUMA