universität wien

DISSERTATION / DOCTORAL THESIS

Titel der Dissertation / Title of the Doctoral Thesis

"Modern logistic solutions for pickup and delivery problems with alternative and roaming locations, intermodal transportation, and inventory constraints and transfers"

verfasst von / submitted by

Alina-Gabriela Dragomir

angestrebter akademischer Grad / in partial fulfilment of the requirements for the degree of

Doctor of Philosophy (PhD)

Wien, 2020 / Vienna 2020

Studienkennzahl lt. Studienblatt / degree programme code as it appears on the student record sheet:	UA 794 370 403
Dissertationsgebiet lt. Studienblatt / field of study as it appears on the student record sheet:	Logistics and Operations Management
Betreut von / Supervisor:	Univ.-Prof. Mag. Dr. Karl Franz Dörner, Privatdoz.

Acknowledgements

First of all I want to thank my supervisor and advisor Karl F. Doerner for offering me a position as part of the project FEAT - Fair and Efficient Allocation of Transportation funded by the Austrian Science Fund (FWF), for his scientific guidance, for letting me have the freedom of researching the topics I found interesting, for his reassurance, for constructive feedback and for believing in my capabilities even when I did not.

Additionally, I want to thank all members of the project FEAT: Richard F. Hartl, Rudolf Vetschera, and Margaretha Gansterer for invaluable ideas and critical evaluation of work in progress.

To Adria Soriano and Daniel Nicola I am grateful for being the best office-mates, collaborators, co-authors, colleagues and friends one can ask for. It was a pleasure working with both and they always helped me with questions I had. I am thankful to all other fellow PhD-students and colleagues for making the last four years a truly enjoyable time, whether at work or outside of it, with Jasmin Grabenschweiger, Biljana Roljic, Emilio Alarcon, Georg Brandstätter, Frank Benda, and David Wolfinger deserving a special mention.

Additionally, I would like to thank Caroline Jagtenberg, Joaquim Gromicho, Tom van Woensel, Jelke van Hoorn, and Ahmed Kheiri with whom it was a pleasure to collaborate.

I would like to gratefully acknowledge the anonymous reviewers for their constructive criticism and helpful comments.

My gratitude also goes to my parents, Daniela and Marian, for showing pride in my achievements, believing in me, and making my academic education possible. I also want to thank my sister Stefanie whose positive attitude for life helped me to remember whats important and for whom I hope to be as inspirational and aspirational as she is for me.

I would like to convey my deepest gratitude and love to my partner David who showed an extraordinaire amount of patience and understanding. Without his unrelenting words of encouragement and reassurance this work would not have been possible and without whom I wouldn't have grown professionally and personally as I have. Additionally, having David as a co-author and collaborator was the most fun I had with a research topic.

Preface

This thesis consists of research I have conducted over the past four years at the University of Vienna and is comprised of various topics of logistics solutions for pickup and delivery problems. Each chapter corresponds to a separate research topic and is either a published paper or a submitted manuscript with the exceptions of the introduction as Chapter 1, a single-authored working paper as Chapter 5, and the conclusion as Chapter 8.

More precisely, Chapter 2 with the title 'Multi-depot pickup and delivery problems in multiple regions: A typology and integrated model' is published in Dragomir et al. [43] while Chapter 3 with the title 'Solution techniques for the inter-modal pickup and delivery problem in two regions' is published in Dragomir and Doerner [42]. Chapter 6 with the title 'Tackling a VRP challenge to redistribute scarce equipment within time windows using metaheuristic algorithms' is published in Kheiri et al. [86] and Chapter 7 with the title 'Total distance approximations for routing solutions' is published in Nicola et al. [116].

Chapter 4 with the title 'The pickup and delivery problem with alternative locations and overlapping time windows' has been submitted to *Transportation Research Part B: Methodological* [44] and is under revision, while Chapter 5 with the working title 'A comparison of heuristics for the pickup and delivery problem with alternative locations' is still in preparation.

The results of these works were presented at several conferences: i.a. at the VeRoLog, the annual workshop of the EURO working group on Vehicle Routing and Logistics optimization, in 2016, 2017, and 2019; the MIC, the Metaheuristics International Conference, in 2017; the EURO, the Association of European Operational Research Societies, in 2019; the TSL, the INFORMS Transportation Science and Logistics Society Workshop, in 2019; the EULOG, Entscheidungsunterstützung in der Logistik, in 2016 and 2018; and the YAMS, the Workshop on Young Academics' Management Science, in 2015.

Contents

List of Figures

List of Tables

List of Algorithms

List of abbreviations

2E-VRP	Two-Echelon Vehicle Routing Problem
AL	Alternative Locations
ALNS	Adaptive Large Neighborhood Search
C2C	Customer to Customer
CVRP	Capacitated Vehicle Routing Problem
CVRP-TW	Capacitated Vehicle Routing Problem with Time Windows
EPI	Extended Pilot Insertion Method
FI	Farthest Insertion Algorithm
FIf	Farthest Insertion First Algorithm
GA	Genetic Algorithm
GA-ALNS	Genetic Algorithm and Adaptive Large Neighborhood Search
GRASP	Greedy Randomized Adaptive Search Procedure
LB	Locker Boxes
LH	Long-haul
LNS	Large Neighborhood Search
LRP	Location Routing Problem
MDPDP	Multi-Depot Pickup and Delivery Problem
MDVRP	Multi-Depot Vehicle Routing Problem
MDVRPMB	Multi-Depot Vehicle Routing Problem with Mixed Backhauls
MIP	Mixed Integer Programming
MR-MDPDP	Multi-region Multi-depot Pickup and Delivery Problem
MS-ALNS	Multi-start Adaptive Large Neighborhood Search
$\mathbf{NN_w}$	Modified Nearest Neighbor Algorithm
NN	Nearest Neighbor Algorithm

OV; OFV	Objective Value; Objective Function Value
PDP	Pickup and Delivery Problem
PDPAL	Pickup and Delivery Problem with Alternative Locations
PDPLH	Pickup and Delivery Problem with Long-hauls
PDPTW-SL	Pickup and Delivery Problem with Time Windows and Scheduled Lines
PI	Pilot Insertion Method
RD	Roaming Deliveries
RMP	Restricted Master Problem
RP	Roaming Pickups
SH	Short-haul
SI	Smallest Insertion Algorithm
TSP	Traveling Salesman Problem
VLNS	Very Large Neighborhood Search
VND	Variable Neighborhood Decent
VRP	Vehicle Routing Problem
VRP	Vehicle Routing Problem
VRPRDL	Vehicle Routing Problem with Roaming Delivery Locations
VRPRDL-S	Vehicle Routing Problem with Roaming Delivery Locations and Stochastic Travel Times
VRPTW	Vehicle Routing Problem with Time Windows

Chapter 1

Introduction

This thesis deals with of variations of the pickup and delivery problem, a subfield of commercial transportation problems. The increase of e-commerce and the rapid development experienced by the transportation industry in the past decades has led to the study of many configurations of transportation networks and to an explosion of variants in transportation problems, motivating researchers to look at broader logistic problems, beyond the basic vehicle routing problems. My wide research focus results in very diverse topics and chapters including characteristics like time windows, multiple regions, multiple depots, inter-modal transportation, long-haul and short-haul transportation, homogeneous and heterogeneous vehicles, alternative locations, roaming locations, locker boxes, capacitated and uncapacitated vehicles, paired and unpaired pickups and deliveries, travel time restrictions, and combination of scheduling and routing aspects.

This thesis attempts to find answers for a multitude of questions: How to extend the known mathematical models to be able to handle multiple regions and multiple depots for a pickup and delivery problem? How important is synchronization in a multi-region multi-modal transportation problem using transshipment? How much can the long-haul scheduling really impact the short-haul routing costs? This thesis also explores new concepts that transportation providers might consider for their product portfolio: A holistic approach that attempts to unify the convenience of the customer with cost savings for the carrier by incorporating alternative recipients and roaming locations. How does this concept compare to traditional home delivery? Is additional convenience only advantageous for the customer? Is it affordable, or even profitable, for the carrier as well?

In addition to these questions, some problems are of a more methodological nature: Different methods were developed and compared to find the most efficient and appropriate method and design of that method for a particular problem. The methods used in this thesis are heuristics, metaheuristics, matheuristics, and decomposition approaches. In particular, I applied adaptive large neighborhood search, variable neighborhood decent, pilot method and an extension thereof, genetic algorithms, and variations of the savings algorithm and the insertion algorithm to different problems. For all heuristics I used both standard operators from the literature and self-developed problem specific operators. To ensure the quality of the algorithms, I tested them on instances from the literature and compared them in terms of both solution quality and computation time. For all topics, I conducted distinct computational experiments, as well as solved realistic and real-life scenarios to gain a deeper understanding of relevant factors and gain managerial insights.

The remainder of this thesis is organized as follows:

Chapter 2 is a survey and typology that provides definitions, a literature review and a step-by-step construction of the mathematical models from a simple and well-known

scenario to the multi-region multi-depot pickup and delivery problem (MR-MDPDP). It suggests extensions for prospective research, and computational experiments show that considering a multi-region formulation is much more efficient than using the formulations for a single region. The chapter was published in Dragomir et al. [43] as a collaborative effort by my co-authors D. Nicola, A. Soriano and myself, with M. Gansterer taking a supervising and guiding role. The work was motivated by the project FEAT, and the models and literature review were developed in a joint effort using cooperative team-work.

Chapter 3 is an inter-modal pickup and delivery problem in two regions where direct shipments between regions are not possible and a long-haul connection has to be used. Additionally, it incorporates time window and capacity constraints. The scope is to identify the correlations and synchronization between different modes and to examine the influences of the long-haul scheduling on short-haul routing costs. This work was published in Dragomir and Doerner [42]. This problem was motivated by the project FEAT and suggested by my supervisor Karl Doerner. I developed the model, implemented the solution algorithm, conducted the computational experiments and wrote the manuscript with advice and guidance from my co-author regarding every part of the process.

Chapter 4 examines an alternative concept for transportation providers in order to attain more convenience for private customers that wish to send and receive parcels: the pickup and delivery problem with alternative locations and overlapping time windows. Additional convenience is achieved with customers having multiple roaming pickup locations throughout the day with non-overlapping time windows (since the product cannot be in two places at once), multiple roaming delivery locations, an additional alternative recipient with its own set of roaming locations, and 24-hour locker boxes. This work has been submitted to *Transportation Research Part B: Methodological* [44] and is currently under revision. The initial motivation was provided by my co-authors with whom I specified the problem. I developed the model, implemented the solution algorithm, conducted the computational experiments and wrote the manuscript. My co-authors gave advice, guidance and feedback for every part of the process.

Chapter 5 is a result of the extensive computational study of Chapter 4. Multiple solution algorithms were implemented which provides an opportunity to compare different methods while solving the same problem. Therefore, the motivation and problem specification is identical to the pickup and delivery problem with alternative locations. This chapter is a working paper for which I am the sole author. I implemented all algorithms and conducted all computational experiments, and wrote the manuscript.

Chapter 6 is published in Kheiri et al. [86] and emerged out of the participation and achievement of third place in the VeRoLog Solver Challenge 2016 - 2017: the third solver challenge facilitated by VeRoLog, the EURO Working Group on Vehicle Routing and Logistics Optimization. The work is a collaboration between the participants that achieved second place (Ahmed Kheiri), third place (my co-author David Mueller and myself), and the organizers of the challenge. The parts of the published work can be clearly distinguished as each participant describes their own method and the organizers

present an overview of the challenge, the posed problem, and the competition results. The solution algorithm of the third place was initiated by me, however the implementation, tuning and optimization of the algorithm, and writing of the manuscript emerged in a cooperative manner by my co-author and myself.

Chapter 7 presents regression-based estimation models that provide fast predictions for the travel distance in the Traveling Salesman Problem, the Capacitated Vehicle Routing Problem with Time Windows, and the Multi-Region Multi-Depot Pickup and Delivery Problem. The use of different characteristics allows us to adjust the models to different problems and to different solution methods. Regression based models are particularly advantageous when a large number of routing problems have to be solved so that even heuristic methods do not provide the required speed. This work is published in Nicola et al. [116]. My contribution is limited to the implementation of the heuristic solution methods for the different problems and participation in conducting the literature review.

For all chapters that correspond to already published work, I have obtained the rights to include the submitted version in my thesis. The only changes made, and the only differences to the published versions, concern formatting, structural changes (renumbering of sections), and updates to cited references.

Chapter 2

Multi-depot pickup and delivery problems in multiple regions: A typology and integrated model

Published in: *International Transactions in Operational Research*
Multi-depot pickup and delivery problems in multiple regions: A typology and integrated model.
A.G. Dragomir, D. Nicola, A. Soriano, & M. Gansterer.
https://doi.org/10.1111/itor.12473

Abstract The rapid development experienced by the transportation industry in the last decades has led to many configurations of networks and therefore to an explosion of variants in transportation problems, motivating researchers to look at broader logistic problems, beyond the basic vehicle routing problems. This work introduces a new type of problem scenario combining various attributes: a pickup and delivery problem with multiple regions, multiple depots and multiple transportation modes. We provide definitions, a literature review and a step-by-step construction of the mathematical models from a simple and well-known scenario to the multi-region multi-depot pickup and delivery problem (MR-MDPDP). For each step the relevant literature is examined. Furthermore we suggest possible extensions for prospective research.

2.1 Motivation

When thinking about realistic transportation networks four main features come to mind: multiple depots, multiple regions, paired pickup and deliveries, and multi-modality. The first three are the core pillars of the problems we are examining. The literature so far has mostly looked at transportation problems within one single region, although in reality transportation between different regions is common practice since the beginning of modern globalization. Also, the so often made assumption that the routing problem can be simplified to a vehicle routing problem might be justified from an academic point of view but is rather oversimplified for practical applications. Moreover, almost all companies operating on a somewhat larger scale will have a network structure that contains at least two depots, especially if transportation takes place between two geographically separated regions. The multi-modality aspect usually occurs by default by operating in different regions and by the need for a switch of transportation modes for the last-mile delivery.

The great advances in infrastructures, computation power and solution algorithms have increased the possibilities for all carriers, specially when facing long distance services. Therefore, we believe that research in the transportation field should start to look at more complex problem structures, like the ones described in this work. This paper is aimed to serve as an introduction

of this new kind of problems that have been hardly studied in their entirety in the literature. Furthermore, we want to ascertain which works consider at least part of the main building blocks. We are therefore specifically looking at multi-depot pickup and delivery problems in multiple regions to provide a foundation for further research in this field.

This research will not only benefit big transportation companies, but also by small carriers who are willing to form a collaborative network system to be able to compete with those big companies in the long distance transportation market. The paper presents the transition from single-region multi-depot problems to multi-region pickup and delivery problems, going through all the basic mathematical models and existing literature in each field. This aims to provide a better understanding of the underlying multi-region transportation problem. Our goal is to gradually introduce the building blocks of our problem to allow the reader a deeper understanding of it. Our basis are multi-depot pickup and delivery problems.

The paper is structured as follows: in Section 2.2 we will give a short problem introduction including all necessary definitions. Section 2.3 gives an overview of the relevant literature. Section 2.4 and 2.5 introduces in detail different variations of multi-depot pickup and delivery problems in one or several regions. We will present mathematical models for each problem variation and give a literature review over work already done in these areas including information about already used instances. In Section 2.6 we conclude our work and in Section 2.7 we examine possible extensions to the basic problem presented before.

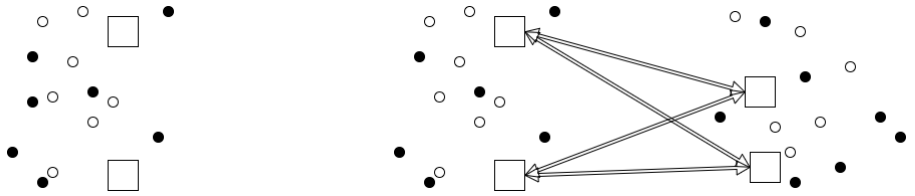
2.2 Problem introduction

Before going into detail we need to provide some definitions:

We assume a two-dimensional plane. All nodes within this plane are represented by a complete graph $G(V, E)$, where V represents a set of nodes and A represents a set of arcs connecting them. A single arc connects a node i to a node j, with a transportation cost of d_{ij}. This cost is usually represented by the distance between both end points of the arc.

Our definitions are based on the nomenclature provided by Parragh et al. [122]. We define a **region** as a group of nodes. The nodes in different regions are geographically separated. A **pickup node** is a customer, located at some specific location, where goods have to be collected by a vehicle. A **delivery node** is a customer where goods have to be dropped off. In unpaired problems, the goods are homogeneous and some quantity is collected and dropped off. There is no need do deliver specific goods. We are either looking at unpaired or at paired pickup and delivery nodes. For **unpaired nodes** the goods collected or dropped off are either transported between customers or have the depot as their origin or destination. **Paired nodes** are called a **request**. A request can have its pickup and delivery node within the same region or in different regions. A request therefore consists of exactly one pickup and one delivery node with a specific quantity. The pickup node has to be served before the delivery node. Since the goods are considered heterogeneous, the delivery node expects exactly these goods and not any goods just like it. Two modes of transportation are considered. **Short-haul vehicles** are used within a region. They are of small capacity and are mainly used for the last-mile delivery. **Long-haul vehicles** are used for transporting goods between regions. They allow the consolidation of goods and have a much bigger capacity than the short-haul vehicles. Long-haul vehicles can be slow, fast, cheap or expensive in any combination. A **depot** is the place where a vehicle starts and ends their tours. A tour starts and ends at the same depot. Also, vehicles are assumed to stay at the depot when they are not being used.

Figure 2.1 shows two cases of regions. All figures in this work use the following representation: the squares are depots, the circles are nodes. An empty (white) circle is a pickup node, a full (black) circle is a delivery node. Routes fulfilled by a short-haul vehicle are depicted as dashed lines, long-haul routes are double lines.



(a) One region with 2 Depots (b) Two regions with 4 depots and long-haul

Figure 2.1: Figure 7.1a shows one region with two depots where all nodes can be served by either depot. Figure 7.1b shows two regions with two depots each. The depots belonging to different regions are connected by long-haul lanes. The delivery node of a corresponding pickup can be in the same or in another region. Black nodes are pickup points, while white nodes are the paired delivery points.

In general, the multi-depot multi-region version of the vehicle routing problems (VRP) is closely related to network design problems (including location routing problems), two- or multi-echelon VRPs, multi-modal transportation and VRP with intermediate facilities. All of these topics are closely related to real-world problems and have been studied in the literature. However, no clear problem definition or consistent typology exists. Our goal is to provide structure and formulations for this generalization of existing problems. For a literature review on VRPs with multiple depots we refer the interested reader to the publication by Montoya-Torres et al. [109]. The works mentioned there build the basis for the problem we are examining but they do not contain the core characteristics of examining solely pickup and delivery problems or dealing with multiple regions. For a literature review on multimodal freight transportation planning we refer the reader to the work by Steadie Seifi [156]. Multi-modal problems often incorporate multiple depots, pickup and delivery problems, or multiple regions but almost never all three. The literature review by Montoya-Torres et al. [109] gives a very good overview but does not integrate all building blocks.

2.3 Literature

Our goal was to find literature that fits exactly to our problem: multiple depot pickup and delivery problem with inter-depot routes in multiple regions. However, to the best of our knowledge, the exact problem we are studying here, has not been yet addressed in the existing literature. Therefore we set the boundary at literature dealing with multi-depot problems with an underlying pickup and delivery routing problem. We loosened this boundary if a paper dealt with other important characteristics (inter-depot routes or multiple regions) even if it was ”only” a vehicle routing problem. Therefore we also included in our literature multi-depot VRPs if they seemed fitting and had an overall network structure that contained some features of the problem classes. Because of this, literature about the two-echelon routing problem and work dealing with multiple modes became relevant for this work. We want to give structure to a new problem class because there has already been some work done but it seems uncoordinated and looked at from very different point of views. We collected the relevant work to show the potential of this problem. Most of the work mentioned here used real-life instances provided by a company which had a need for a solution. Therefore the importance becomes evident.

Table 2.1 gives an overview on the included literature. The individual papers are presented in detail in their corresponding sections. The papers are sorted first by the section they appear in this work and second by year of publication.

The first block of columns shows the underlying routing problem. This refers to the way the routing is dealt with within a region. We divide the papers according to the following classification: VRPs (**VRP**), paired pickup and delivery problems (**paired P&D**), pickup and

delivery problems with backhauls (**backhauls**), and pickup and delivery problems with mixed backhauls (**mixed backhauls**). In the standard VRP each customer node has a certain quantity. It does not matter if this quantity has to be delivered or picked up since it is the same type for all nodes. Please note, that we did include literature that was similar enough to our problem, even if it only dealt with the standard VRP. The paired pickup and delivery problem (see the definitions by Parragh et al. [122]) contains for each transportation request a pickup and delivery node that belong together. The delivery can be done directly (within the same tour as the pickup) or be forced to go over a depot. The VRP with backhauls or mixed backhauls belongs to a different class of problems. Here, all goods that are delivered to a customer are loaded at the depot and all goods that are picked up at a customer are unloaded at the depot. There is no link between the pickup and delivery nodes (see the definitions by Parragh et al. [121]). In the mixed backhaul problem the pickup and delivery at the customers can be fulfilled in any sequence. However the standard backhaul problem (also called the VRP with clustered backhauls) specifies that all deliveries have to occur before the first pickup.

The second block of columns shows constraints and other problem specifications. Only constraints that were mentioned in the relevant literature are included in the table, apart from the vehicle capacity constraint, which was excluded because it appeared in every paper. These are however not all possible constraints since many problem variations have not been studied yet. For further suggestions we refer the reader to Section 2.7. The third block of columns shows whether the solution method applied was an exact method or a heuristic. Because of the difficult nature of the problems it is not surprising that heuristics were used more often. Also, it shows the number of locations (nodes) for the biggest instances solved. If only the number of requests where given, the assumption is made that each request consists of two locations and this number was added to the table.

2.4 Single region problems

Single region problems are the basic building block for our new problem class. According to the previous definition of region, single-region problems encompass the vast majority of families of VRPs studied so far. The main elements of this setting are a set of customers or nodes that are to be visited and a set of depots from and to which routes start and arrive. Many configurations and restrictions of this setting can be found in the literature, leading to a broad spectrum of problems. The most representative problems of this setting are the classical travelling salesman problem and VRPs. Both problems have one unique depot. In Laporte [93], the VRP is described in detail and solution algorithms for the VRP are presented. For the purpose of this paper we are interested in problems where the number of depots is greater than one. Routing problems with one depot can be extended to several depots, leading to the family of multi-depot problems. One of the classic problems among them is the multi-depot VRP, which is an extension of the aforementioned VRP. A formal description of this type of problem can be found in the works by Polacek et al. [129] and Renaud et al. [134]. The main elements of these settings remain the same as for the single-depot problems, only the graph $G(V, E)$ is extended to include all depots. The many existent typologies of problems in this family arise from the diversification of factors like the nature of customers, their relation, time constraints and many others. In this work we focus on pickup and delivery problems. We present some of the more relevant variants of this problem within the single-region setting, along with their mathematical model, before extending them to the multi-region setting.

Multi-depot vehicle routing problems

Dondo and Cerdá [41], consider a heterogeneous fleet of vehicles in a multi-depot scenario with time windows. The vehicles differ in capacity and speed. The underlying routing problem is

	Year	VRP	paired P&D	backhauls	mixed backhauls	capacitated depots	limited number of vehicles	time windows	heterogeneous vehicles	multimodality	compatibility constraints	exact	heuristic	instance size	section
Currie and Salhi [36]	2003	✓				✓	✓	✓	✓			✓	✓	200	2.4
Polacek et al. [129]	2004	✓					✓	✓					✓	288	2.4
Dondo and Cerdá [41]	2007	✓						✓	✓				✓	100	2.4
Min et al. [105]	1992			✓		✓							✓	161	2.4.1
Irnich [81]	2000			✓			✓	✓	✓				✓	484	2.4.1
Nagy and Salhi [114]	2005				✓		✓						✓	249	2.4.1
Bettinelli et al. [16]	2014		✓				✓	✓	✓			✓		144	2.4.2
Detti et al. [40]	2016		✓			✓		✓	✓		✓		✓	246	2.4.2
Gendron and Semet [63]	2009	✓				✓			✓			✓		701	2.4.3
Perboli and Tadei [125]	2010	✓				✓	✓			✓		✓		50	2.4.3
Crainic et al. [32]	2011	✓				✓	✓			✓			✓	50	2.4.3
Perboli et al. [126]	2011	✓				✓	✓			✓			✓	50	2.4.3
Hemmelmayr et al. [77]	2012	✓				✓	✓			✓			✓	50	2.4.3
Nguyen et al. [115]	2012	✓				✓				✓			✓	200	2.4.3
Ghilas et al. [65]	2016		✓			✓	✓	✓	✓	✓			✓	100	2.4.3
Breunig et al. [18]	2016	✓				✓	✓			✓			✓	200	2.4.3
Ghilas et al. [64]	2018		✓				✓	✓				✓		50	2.4.3
Bard et al. [9]	1998	✓					✓					✓	✓	20	2.4.3
Angelelli and Speranza [4]	2002	✓				✓	✓						✓	150	2.4.3
Crevier et al. [34]	2007	✓					✓						✓	288	2.4.3
Goel and Gruhn [68]	2008	✓					✓	✓	✓		✓		✓	2500	2.4.3
Muter et al. [112]	2014	✓					✓					✓		50	2.4.3
Salhi and Sari [143]	1997	✓							✓				✓	360	2.7.3
Ceselli et al. [22]	2009	✓				✓	✓	✓	✓		✓	✓		100	2.7.4

Table 2.1: General classifications of the literature sorted by section and year

a capacitated vehicle routing problem. Although they mention both pickups and deliveries, only one option is chosen at a time.

The proposed method is a region-based optimization including three phases. In the first phase, a set of cost effective regions is identified, while in a second phase, regions are assigned to vehicles and tours are sequenced using a region-based MILP formulation. The third phase orders nodes within regions and schedules arrival times for the vehicles at customer locations by solving a MILP model. They test the computational performance of their algorithm on instances from Solomon's work found in [150]. In addition, new test instances have been generated. They have up to 100 nodes and can solve many problems up to optimality or near-optimality. Even though the work deals only with a simple vehicle routing problem it is mentioned because of the region based solution approach which makes it interesting for our topic.

Another application presented by Currie and Salhi [36] is a full-load pickup and delivery problem with time windows. They have heterogeneous vehicles and products. In their problem they consider the delivery requirements of a large construction company. Goods have to be transported from multiple depots to a large number of customers. Since the problem consists only of deliveries it can be classified as a capacitated vehicle routing problem. The problem ignores vehicle load splits as they very rarely occur. Trips are made between locations, where the product is collected, and the depots. Each location is visited exactly once, as several requests for the same location are considered as a single requests for a location. Loading and unloading times are assumed to be constant, irrespective of the product to be transported. A constant cleaning time for vehicles, as well as waiting times might be added to the routes. 'Dayworks', which are situations in which the vehicles remain at a location for the rest of the day after their arrival, are also included. External vehicles might be hired by the company if the fleet is not sufficient to perform all trips.

To solve the problem described in Currie and Salhi's work [36], the authors present a first formulation as a 0-1 LP followed by a hybrid algorithm that chooses between a greedy heuristic and one based on regret costs. The greedy heuristic uses an index of stringency to select the next request for insertion, where requests with the lowest index values are assigned to vehicles first. Stringency includes different attributes such as quantity and length of time windows. The regret costs method calculates a penalty cost for each request, which is obtained from the cost of forfeiting the opportunity of inserting the request in its best slot. They randomly generate their instances and solve instance sizes up to 200 location sites and 500 request. These instances can be found on: *http://www.mat.bham.ac.uk/S.Salhi/others.*

2.4.1 Multi-depot vehicle routing problem with mixed backhauls

The multi-depot vehicle routing problem with mixed backhauls consists of a set of unpaired pickup and delivery customers. Unpaired nodes are independent of each other, there is no relationship between the pickup and the delivery locations. By definiton this means that all pickup goods are transported to a depot, and all delivery goods have to be loaded in the vehicle at a depot.

The underlying routing problem has been defined in different ways in the literature. In [121], Parragh et al. refer to the problem as VRP with Mixed Linehauls and Backhauls, while Salhi et al. name it Multiple Depot VRPs with Backhauling in [142]. Traditionally, the VRP with backhauls has been associated with pickup and delivery problems where pickup and delivery customers are not paired and a node has positive or negative quantities. In the simplified version (with clustered backhauls) the delivery points must always be visited before the pickup of goods to avoid infeasibility concerning the vehicle capacity restriction. However, in this section we present the more general case where no precedence relations have to be satisfied, that is, pickup and delivery points can be visited in any order. Therefore, we want to differentiate from the pure backhauling problems by referring to the problem as multi-depot VRP with mixed backhauls

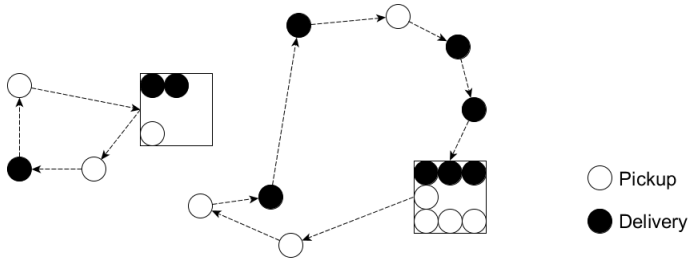


Figure 2.2: A single-region multi-depot problem with mixed backhauls. For every pickup in a tour, a delivery is done at the depot, and for every delivery in a tour, the goods are picked up at the depot as well. Note, that the pickup and delivery sequence within the tour can be mixed.

(MDVRPMB). Figure 2.2 gives a visual representation of the VRP with mixed backhauls can be found .

As described above, the main characteristics of this problem are that the nodes in graph $G(V, E)$ could be either pickup or delivery customers, and that they can be visited in any order. The objective is to visit all customers at minimum cost while satisfying the capacity restrictions of the vehicles. A set of vehicle is defined in advance, but it is assumed that this number is sufficient to fulfill all transportation requests.

The mathematical model is based on the work by Montoya et al. [109].

Sets and Parameters

N = set of customers,
L = set of depots,
V = set of all nodes, $N \cup L$,
K_l = set of vehicles in depot l,
K = $\cup_{l \in L} K_l$ = set of all vehicles,
q_i = quantity load for the node $\begin{cases} +q_i, & \text{if it is a pickup customer,} \\ -q_i, & \text{if it is a delivery customer.} \end{cases}$
Q^{SH} = capacity of each vehicle,
c_{ij} = travel cost between nodes i and j.

Decision Variables

x_{ijk} = $\begin{cases} 1, & \text{if vehicle } k \text{ travels directly from node } i \text{ to node } j, \\ 0, & \text{otherwise.} \end{cases}$
S_{ik} = loading amount of vehicle k after visiting customer i.

For simplicity in the notation, copies of depot l representing start and end points of the routes are denoted as l^0 and l^1.

$$\text{minimize} \sum_{i \in V} \sum_{j \in V} \sum_{k \in K} c_{ij} x_{ijk} \qquad \text{Subject to} \tag{2.1}$$

$$\sum_{i \in N} \sum_{k \in K} x_{ijk} = 1 \qquad \forall j \in V \tag{2.2}$$

$$\sum_{i \in V} x_{ihk} - \sum_{j \in V} x_{hjk} = 0 \qquad \forall k \in K, h \in N \tag{2.3}$$

$$\sum_{i \in \mathrm{V}} x_{ilk} = \sum_{j \in V} x_{ljk} \leq 1 \qquad \forall l \in L, k \in K_l \tag{2.4}$$

$$S_{jk} = \sum_{i \in V} x_{ijk} \cdot (S_{ik} + q_i) \qquad \forall j \in V, k \in K \tag{2.5}$$

$$0 \leq S_{ik} \leq Q^{SH} \qquad \forall i \in V,\ k \in K \tag{2.6}$$

$$\sum_{i \in S} \sum_{j \in S} x_{ijk} \leq |S| - 1 \qquad \forall k \in K, S \subseteq N, |S| \geq 2 \tag{2.7}$$

$$x_{ijk} \in \{0, 1\} \qquad \forall i, j \in V, k \in K \tag{2.8}$$

$$S_{ik} \in \mathbb{N} \qquad \forall i \in V, k \in K$$

Equation (2.1) represents the objective function, which is to minimize costs. Constraint (2.2) ensures that each node is visited once, constraint (2.3) that the vehicle entering and leaving a node should be the same (route continuity). Each short-haul tour originates and terminates at the corresponding depot (2.4). Loading restrictions for backhauls (2.5). Constraint (2.5) makes the model non-linear. For simplification purposes, linearization of the constraint is not included in the model but on the appendix. Since both pickup and delivery nodes are visited in any order, it is not sufficient to sum up over all quantities within a route. After every visited node, the quantity of goods in the vehicle has to be within our bounds (2.6). Constraint (2.7) is used for sub tour elimination.

Related problems

Different types of this model (with certain variants and assumptions) have been worked on. For example, the MDVRP with backhauling, was early proposed by Min et al. in [105]. They do not consider mixed backhauls because they assume rear-loaded trucks for their problem where reshuffling of loads is tedious and not easily possible. Therefore all deliveries are fulfilled before the first pickup is allowed. Both their vehicles and depots have a limit on capacity. For the depots this means that they are able to only serve a limited amount of loadings and unloadings per day. Their main decision problems are to determine the fleet size (the number of available vehicles is not limited), to allocate the vehicles to the depots, to allocate the customers to vehicles and the routing of the vehicles.

Irnich [81] looks at a kind of multi-depot pickup and delivery problem. The author here considers multiple depots and heterogeneous vehicles. The goods are delivered to and from a hub, instead of the depots of the particular vehicles. This is because of the particularity that all pickup or delivery locations serve as the depots for the vehicles. We classified it as a problem with backhauls under the assumption, that the hub is actually the depot and the nodes are just simple customers. However it is clear that it does not quite fit since the hub does not correspond to the definition of a depot. Their main decision is the assignment of a request to a specific vehicle. The routing is far less important.

Nagy and Salhi [114], look at a multi-depot VRP with mixed backhauls and simultaneous pickup and deliveries where a customer can both receive and send goods at the same time. Although their work focuses mainly on single-depot problems, they manage to extend this approach to a multi-depot scheme. Their vehicles are capacitated and they have a maximum route length constraint.

Methods and instances

Min et al. in [105] decompose their problem and use a three phase heuristic method. Its first phase aggregates customers (delivery nodes) and vendors (pickup nodes) to capacitated regions; in the second phase, customers and vendors are assigned to a depot and a route; the third phase designs the individual vehicle routes. Each phase takes the output of the previous phase as an input. The data and instances used were provided by the transportation division of a large U.S. company and altered to ensure confidentiality. Their data contained 161 potential nodes to visit. The input data can be found in the appendix of their published paper.

Irnich [81] formulated their problem as a network model. The goal is to find a set of trips which fulfills all requests while minimizing costs. They use a set partitioning/covering model which implies a two phase algorithm. The solution approach first enumerates relevant route-vehicle combinations; then a set T of relevant trips for each route-vehicle combination are enumerated, checked for dominance, and, if dominated, eliminated from T. Finally they solve a set covering problem with the corresponding remaining set T. The algorithm was implemented in a decision support system used by the Deutsche Post AG. They have up to 22 locations with 242 requests in total and up to 6 different vehicle types. They do not provide a comparison to exact solutions, only to weak lower bounds. However, compared to previously manually planned solutions, they where able to reduce cost by 15 per cent on average. As far as we know, the exact data has not been made public.

The multi-depot problems from Nagy and Salhi [114] are solved by dividing the customer set into borderline and non-borderline customers. For the non-borderline customers, the assignment is straightforward, whereas the borderline customers need an explicit assignment to their nearest depot. A customer is borderline if it is situated roughly midway between two depots. For each depot they find a weakly feasible solution and finally insert the borderline customers into the vehicle routes one at a time. Their instances have up to 249 customers and up to 5 depots. They also reimplemented other methods from the literature to compare against and show that their methods outperforms the others in relation to solution quality, but not necessarily in relation to computational times.

2.4.2 Multi-depot paired pickup and delivery problem

This problem represents the basic extension to a multi-depot scheme of the classic pickup and delivery problem (PDP). In PDP, customers are not independent any more, but they are paired to form requests. Each request consists thus of two customers or locations, one where the goods have to be picked up and a second one where goods are to be delivered. This problem differs from the one presented in Section 2.4.1 because the order in which customers are visited is now constrained. No delivery customer can be visited before its matching pickup customer. For further information about PDP problems we refer the interested reader to the work by Parragh et al. [122] and by Berbeglia et al. [14]. Despite the extensive research conducted on single depot PDP and its variants, very little literature exists for the multi-depot case (MDPDP). A visual representation of the VRP with paired P&D can be found in Figure 2.3.

New Sets and Parameters

R = set of requests, ={1,..n},
N = set of pickup and delivery points, ={1,..n, n+1,...,2n}.

New decision variables

t_{ik} = fulfillment time at node i on tour (vehicle) k.

The precedence relation between pickup and delivery customers is modeled by explicitly considering the visiting time as a decision variable. The pickup and delivery nodes have to be

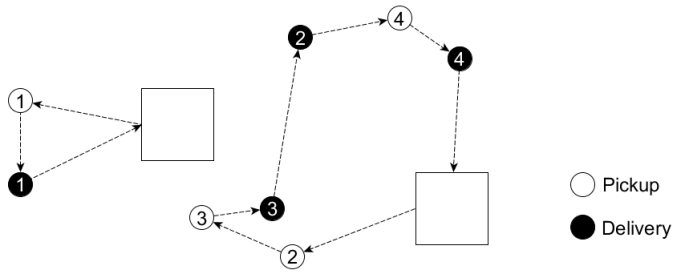


Figure 2.3: A single-region multi-depot VRP with paired P&D. Here the delivery has to be after the pickup within the same route and vehicle. The nodes belonging together share the same number.

always within the same route, it is not possible to store a package at the depot.

$$\sum_{j \in N} x_{ijk} - \sum_{j \in N} x_{i+n,j,k} = 0 \qquad \forall i \in R, k \in K \tag{2.9}$$

$$t_{ik} \leq t_{i+n,k} \qquad \forall i \in \{1, .., n\},\ \forall k \in K \tag{2.10}$$

$$t_{jk} = \sum_{i \in V} x_{ijk} \cdot (t_{ik} + c_{ij}) \qquad \forall j \in V, k \in K \tag{2.11}$$

$$t_{ik} \in \mathbb{Z} \qquad \forall i \in V, k \in K \tag{2.12}$$

Constraint (2.9) ensures that both pickup and delivery locations are served by the same vehicle. The precedence constraints for P&D are included by (2.10). (2.11) makes sure that the time continuity is given in a tour.

Related problems

Bettinelli et al. in [16] present a multi-depot heterogeneous pickup and delivery problem with soft time windows. They find a set of routes of minimum length for a fleet of vehicles with limited capacity to serve a set of customers. Routes start and end at a depot from a given set of depots. Customers have a given demand that has to be served by a single vehicle from a heterogeneous fleet. The violations of the time windows are penalized. The pickup and delivery of a package belonging to one request has to be done by the same vehicle.

Detti et al. [40] present a multi-depot dial-a-ride problem with heterogenous vehicles. They extend the problem by introducing compatibility constraints. The authors study a problem in healthcare, where patients have to be transported in vehicles depending on their needs. A fleet of heterogeneous vehicles are located in geographically distributed depots. Patients may ask to be transported by a specific vehicle, which is called patient's preferences. The problem consists on assigning transportation services to vehicles and finding a route for them. Among the constraints included in this problem we can find vehicle's capacity, patient-vehicle compatibility, pick up and delivery time windows, patients' preferences, precedence constraints, quality and timing of the service provided.

Methods and instances

As for the problem described by Bettinelli et al. in [16], the authors propose a branch-and-cut-and-price algorithm. First, a set of columns is generated for every customer, representing optimal paths when customers are served one at a time. A dummy column with a very high cost

is included in order to ensure feasibility at each node of the search tree. They solve the linear relaxation of the restricted master problem (RMP), and then search for columns which are not in the RMP, but have negative reduced cost. If this column does not exist, the solution is optimal for the linear relaxation of the master problem and provides a lower bound to the problem. Their next step is a dynamic programming heuristic to provide an upper bound on the amount of dual prizes that could be collected when completing the path. To strengthen the lower bound they add violated 2-path inequalities. The branching policies used are branching on the number of vehicles and branching on arcs. This approach is tested on instances derived from Solomon's work presented in [150], solving instances up to 144 customers. They manage to prove optimality for instances up to 75 customers.

In Detti et al. [40], the proposed solution methods include variable neighbourhood search and tabu search algorithms. A mixed integer linear programming formulation is also presented. These algorithms are compared based on their computational results for large real-world and random instances. For the tabu search, in a first phase, an initial solution is generated through a fast insertion heuristic. Then, a tabu search scheme iteratively tries to improve this solution. At each iteration, a neighbourhood is generated by perturbing the current solution. The whole process is repeated until a stopping criterion is satisfied. As for the variable neighbourhood search algorithm, two steps are proposed. First, an initial solution is generated through a fast insertion heuristic. Then, a local search step is performed on this initial solution. The second step of the algorithm is an iterative procedure. At each iteration of the algorithm, a new randomly generated solution is created in the current neighbourhood. A local search step is applied to this solution. If it is feasible and the best so far, it replaces the initial solution. If only one violation on any single type of constraint occurs, the solution is changed by an adjusting procedure to make it feasible. These methods are tested on random instances based on real-life data. The instances have up to 246 transportation services (requests) and 313 vehicles with 4 different types, distributed over 17 depots and belonging to 29 non-profit organizations.

2.4.3 Multi-depot pickup and delivery problem with inter-depot routes

Here we present one variant of the multi-depot problem where depots are connected by inter-depot lanes. Figure 2.4 shows a simple version of an inter-depot route connecting two depots in a single regions with paired pickup and delivery nodes. This setting makes sense when the vehicle type used for inter-depot transportation is different from the one used for servicing the customer nodes, exploiting the advantages of a faster or cheaper connection. Therefore, this problem lies also in the family of multi-modal transportation. The inter-depot lane can be a permanent connection or opened when needed. The connection allows the consolidation of goods and a more efficient and maybe even faster transportation per unit of distance. It can be operated on a fixed schedule and would therefore mimic a third party providing the service, or on a flexible schedule which would mimic long-haul vehicles at the carriers' own disposition. The costs can be fixed (full-truckload transportation) or variable depending on the goods transported. To represent different modes of transportation, multiple parallel lanes can be added to a model, for example a cheap and slow lane for transportation by ship and an expensive and fast lane for transportation by plane. To the best of our knowledge, this problem has not been previously addressed in this setting in literature. However it is included in this work as a basis for the development of multi region transportation problems presented in Section 2.5.

This mathematical model inherits the main characteristics of the previous models without inter-depot routes, and adds inter-depot related sets, variables and constraints. Also, decision variables regarding the beginning and ending times of the routes have to be defined to model the time consistency between the short- and long-haul vehicles.

New sets and parameters

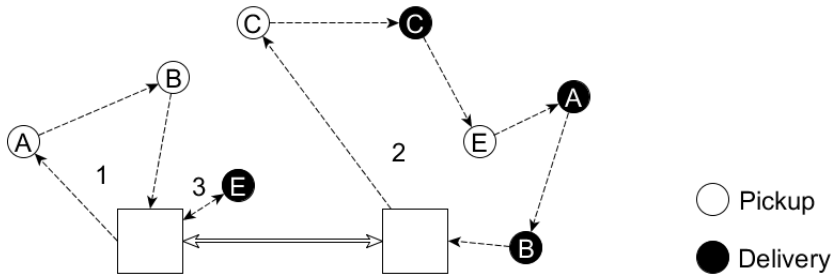


Figure 2.4: Inter-depot routes by adding a separate (faster, cheaper) vehicle between the depots, while still operating in a single region where each node can be served by any depot. Each request (pickup and delivery pair) has a letter assigned. The short-haul vehicles visit the pickup nodes first and then deliver to the corresponding delivery nodes. The optimal routes are shown and numbered according to their sequence of execution. All transportation requests, except one (C), are transported over the inter-depot route. Request C is picked up and directly delivered by the same vehicle.

H_l = set of inter-depot vehicles in depot $l \in L$,
H = set of inter-depot vehicles, $\cup_{l \in L} H_l$,
Q^{LH} = capacity of each inter-depot vehicle,
c_{lp} = cost of traveling between depots l and $p \in L$.

New decision variables

$$y_{ilph} = \begin{cases} 1, \text{if request } i \text{ is transported from depot } l \text{ to depot } p \text{ with vehicle } h, \\ 0, \text{otherwise.} \end{cases}$$

$$z_{lph} = \begin{cases} 1, \text{if inter-depot vehicle } h \text{ is used for a transport between depots } l \text{ and } p, \\ 0, \text{otherwise.} \end{cases}$$

m_h = starting time of vehicle h, $h \in H$,
b_k = beginning time of tour $k \in K$,
e_k = ending time of tour $k \in K$.

The adjusted objective function includes the costs of transporting between depots, which may be different from the costs of the vehicles used for the routes:
Constraints to add:

$$\text{minimize} \sum_{i \in V} \sum_{j \in V} \sum_{k \in K} c_{ij} x_{ijk} + \sum_{l \in L} \sum_{p \in L} \sum_{h \in H_l} c_{lp} z_{lph} \tag{2.13}$$

Constraints to add:

$$\sum_{j \in N \cup \{l\}} \sum_{k \in K_l} x_{ijk} e_k \leq \sum_{h \in H_l} y_{ilph} m_h + M(1 - \sum_{j \in N \cup \{p\}} \sum_{k' \in K_p} x_{(i+n)jk'}) \quad \forall i \in R, l, p \in L \tag{2.14}$$

$$\sum_{j \in N \cup \{l\}} \sum_{k' \in K_p} x_{(i+n)jk'} b_{k'} \geq \sum_{l \in L \setminus \{p\}} \sum_{h \in H_l} y_{ilph}(m_h + d_{lp}) \quad \forall i \in R, p \in L \tag{2.15}$$

$$\sum_{i \in R} \sum_{h \in H_l} y_{ilph} q_i \leq Q^{LH} \quad \forall l, p \in L \tag{2.16}$$

$$\sum_{i \in R} y_{ilph} \leq M \cdot z_{lph} \qquad \forall l, p \in L, h \in H_l \tag{2.17}$$

$$y_{ilph}, z_{lph} \in \{0, 1\} \qquad \forall i \in V,\ l, p \in L \tag{2.18}$$

$$m_h \in \mathbb{Z} \qquad \forall h \in H$$

Starting time of the inter-depot vehicle transporting goods of request i should be greater than the arriving time of the short-haul route serving the pickup customer corresponding to the request (2.14). The big M component is necessary to allow for requests being transported directly from pickup to delivery, without going through the inter-depot route. Starting time of the short-haul route delivering goods of request i should be greater than the arriving time of the long-haul route transporting the goods of the request (2.15). Constraint (2.16) for the quantity on the inter-depot vehicle. Constraint (2.17) ensures that requests are only assigned to long-haul transports that are performed.
Constraints (2.14) and (2.15) make the model non-linear and therefore intractable for a commercial solver. However they can be linearized, as shown in the appendix. In spite of this the model remains NP-hard and can only be solved to optimality for small instances.

Related problems

As far as we know, this type of multi-depot problem has not been yet addressed in the literature in this form. Its main feature is the existence of two different ways of transportation in two stages, which requires the combination of a scheduling and a routing problem. Extensive literature exists for similar problems, where two dependent decisions are to be made. For example, two-echelon vehicle routing problems (2E-VRP) and location routing problems (LRP) are frequently studied in the literature. Also another concept of MDPDP with inter-depot routes has been addressed in the literature, where inter-depot routes are understood as a vehicle starting and ending the route in a different depot, thus staying away from multi-modal transportation. A detailed explanation is given in Section 2.4.3.

Gendron and Semet [63] consider a multi-echelon LRP for a fast delivery service, with three levels of intershipment facilities (hubs, depots, satellites). From satellites, the products are sorted and delivered to the customers. Transportation between levels is done in different transportation modes.

In Crainic et al. [32], Perboli and Tadei [125], Perboli et al. [126], and Breunig et al. [18] a two-echelon vehicle routing problem (2E-VRP) is presented. They define one central hub, a set of intermediate depots called satellites and a set of customers with demands. The freight stored at the central hub must transit through the satellites and then be delivered to the customers. Two different types of vehicles are considered according to the level they serve. Neither route size nor number of visited customers are limited.

In Hemmelmayr et al. [77], the 2E-VRP is also considered. Satellites locations are assumed to be known and a limit on the number of vehicles for both levels is considered. They are also able to solve an LRP with their approach, by transforming the 2E-VRP to an LRP. Capacity limit and opening costs are considered for the depots.

Ghilas et al. [64, 65] use a very similar type of inter-depot routes in their work. Their inter-depot route is a scheduled active public transportation line travelling in the city of Amsterdam transporting both passengers and packages. All vehicles from a depot can serve all customer nodes without restrictions, but they have the possibility to shorten their routes by consigning the packages to a public scheduled line. At the end of the line, the packages have to be picked up by another vehicle belonging to another depot and delivered to their destination. They prove that

introducing scheduled lines leads to a decrease in transportation cost compared to the solutions obtained without them.

Methods and instances

Gendron and Semet [63] present both an arc-based and a path-based formulation for tackling the problem. Furthermore, they develop a binary relaxation as an alternative, for which both formulations are equivalent. The computational results are obtained from 32 instances generated based on real data, for 93 depots, 320 satellites and 700 customers. The linear programming relaxations are solved using CPLEX with a 2 hours CPU time limit.

For the 2E-VRP solution, Perboli and Tadei [125] present new inequalities in an exact model. Inequalities are then inserted in a Branch & Cut framework with a 10000 seconds computing time limit for each instance.

Crainic et al. [32] present heuristics to solve the 2E-VRP. An initial solution is obtained by a first-clustering second-routing strategy. Then a local search is performed over the clustering of customers around satellites. A final perturbation move is performed until a feasible solution is reached.

Perboli et al. [126] present two interesting math-based heuristics to solve the 2E-VRP. The first one forces assignment variables to be fixed to a value and, if the model is infeasible, they unfix some variables and restart the process. The second heuristic method, solves a semi-continuous 2E-VRP using a MIP solver with a 60 seconds time limit and obtains a list of best integer solutions found. For every solution in the list, they consider the customer-satellite assignments and solve the VRP instances with a time limit of 5 seconds.

Crainic et al. [32], Perboli and Tadei [125] and Perboli et al. [126] use instances from some previous work of theirs, which covers up to 50 customers and 5 satellites.

An adaptive large neighborhood search is proposed in Hammelmayr et al. [77]. Its main feature is that it allows for infeasibilities in the initial solution construction phase. Destroy and repair algorithms are used for improving the solution and making it feasible. The authors use the same instances as Crainic et al. [32] and Perboli et al. [126].

Breunig et al. [18] make use of a large neighborhood search heuristic. The method uses several destroy operators and only one repair operator. They also work on collecting the instances for the problem, obtaining different sets from several previous works and modifying them to a common structure. Instances with up to 200 customers are solved. Results show a very good performance in comparison to the results from Hemmelmayr et al. [77].

Ghilas et al. [64, 65] use both exact and heuristic methods for their problem. They decided for a branch-and-price approach as an exact method and for an adaptive large neighborhood search heuristic method. They managed to obtain exact results for small and medium size instances with up to 50 requests. In addition they studied the effects of the number of scheduled lines, the scheduled lines frequency, and of the time windows on the operating costs. Their instances contain up to 100 requests and 3 scheduled lines.

Interpretations of inter-depot routes

So far, existing literature has referred to the inter-depot routes as routes in which vehicles start and ends in different depots or when another depot than the starting depot is used as a replenishment or unloading point along the route. This interpretation leaves us with a quite different problem than the previously proposed one. Multi-modal transportation stays out of scope and it is no longer a two-level decision problem.

The first option mentioned consist of vehicles starting in a depot and ending at a different location (depot or customer) within the same region. A practical application would be to allow the vehicles to return to their start depot on another day. Figure 2.5 shows a simple route with two pickup and two delivery nodes which starts and ends at different depots. From this example

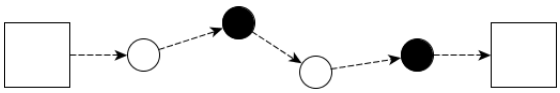


Figure 2.5: Inter-depot routes by allowing vehicles have different start and end depots.

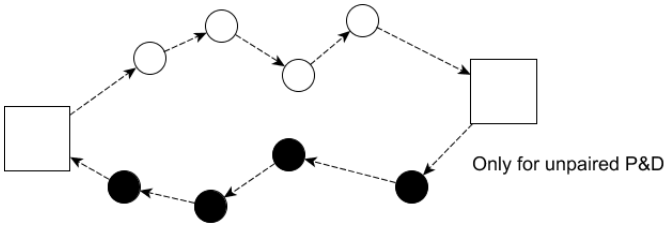


Figure 2.6: Inter-depot routes by allowing vehicles to visit depots within their route. In the first part of the route it picks up all the white nodes and delivers all or part of the goods to the nearest depot (different from the starting depot). From there, it picks up the goods that have to be delivered to the black nodes. At the end of the day it returns to the depot it started from.

it is clear that by forcing a vehicle to return to the start depot, a considerable increase in distance has to be condoned.

The paper by Goel et al. [68] deals with a rich VRP that incorporates several constraints found in real-life applications. These locations can require a pickup, a delivery, or simply a service, where no quantity of goods change hands. They do not assume that vehicles are stationed at a depot, but simply become available at a specific location, to a specific time with some predefined load. Also, vehicles do not return to a depot, but an end location for the tour is given. Therefore we classified the problem as one with inter-depot routes, since every location can be a depot. During the computation, some requests get canceled while others became known only then. In addition to the inter-depot routes, they consider several constraints like time window restrictions or drivers' working hours, among others.

They make use of a reduced large neighborhood search strategy, with several and specific neighborhoods that allow them to go from one solution to another. They solved self generated instances with up to 500 vehicles and 2500 orders.

Another approach that can be considered as a problem with inter-depot routes consists in allowing a vehicle to visit another depot during its tour while still returning to its own depot at the end of the day. This approach makes sense for unpaired pickup and delivery problems or capacitated VRP with backhauls. Especially vehicles with a tight capacity might profit from the possibility to unload or replenish the goods to be transported during their route. This scenario also applies, for example, to electric vehicles that need to use recharging stations during their route, as stated in the work by Hiermann et al. [78]. Figure 2.6 shows an example of a route in which a vehicle visits a depot during its route while transporting backhauls.

This problem has been introduced by Crevier et al. [34] where depots act as replenishment facilities, in a real-life grocery distribution network. Vehicles can have a single-depot route, which starts and ends at the same depot, or inter-depot routes which connects two different depots. Nevertheless, returning to its original depot is mandatory. They tackle the problem with a three-phase heuristic: route generation, determination of least cost feasible rotations and post-optimization phase. They adapted instances from the literature and generated benchmark instances with 48 to 288 customers and 3 to 6 depots.

Muter et al. [112] tackle the same problem and allow vehicles to stop at intermediate depots for replenishment.They solve their problem by a branch-and-price algorithm. Moreover, two

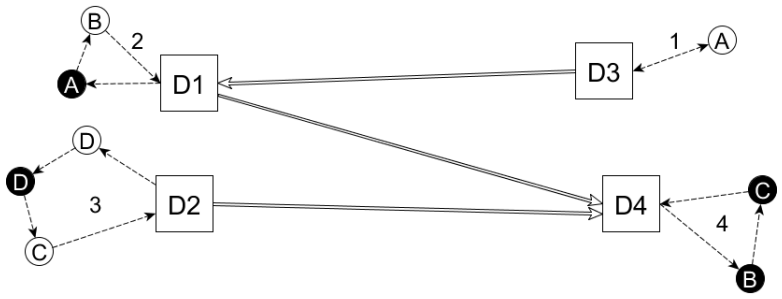


Figure 2.7: MR-MDPDP with two regions and two depots per region. Requests A, B and C are transported over long-haul lanes. Request D is an intra-region request served only by depot 2. The SH routes are numbered in their order of execution. The order is the following: Route 1 picks up request A, which then travels from D3 to D1 over the LH. Route 2 delivers A and picks up B. B travels over the LH from D1 to D4. Route 3 picks up D and C and immediately delivers D within the same route. Request C travels over the LH to D4. After both B and C have arrived in D4 they are both delivered by route 4.

different pricing sub problems are implemented and compared. The traditional one-level column generation scheme is inferior to their developed two-level decomposition scheme. They work on instances from the literature containing between 3 and 7 depots and up to 50 customers, solving some of them to optimality. They show that allowing inter-depot routes leads to an improvement of cost, although making the problem harder to solve.

Bard et al. [9], and Angelelli and Speranza [4] allow the replenishment at intermediate facilities, but differing from the previous works in that depots cannot act as intermediate facilities, hence being different kind of locations.

Angelelli and Speranza [4] propose a tabu search algorithm to iteratively move from one solution to another within the same neighborhood. A new solution is chosen minimizing a penalty that is related to the objective function. Instances from previous literature are solved, as well as random generated ones with up to 150 customers, 4 satellites and 5 vehicles.

The solution method proposed in Bard et al. [9] starts by relaxing the vehicle routing problem with satellite facilities linear program. Subsequently, a branch and cut approach is implemented. They solve self-generated instances with up to 20 customers and 2 satellites.

2.5 Multi-region pickup and delivery problems

In this section we introduce a new type of problems, the multi-region pickup and delivery problems. The main characteristic of these problems is that regions are independent of each other, meaning that customers in different regions cannot be visited by a vehicle from a depot situated in another region. Especially, we are interested in describing the multi-depot version of these problems (MR-MDPDP), where more than one depot is located in each region. This problem constitutes a special case of the MDPDP with inter-depot routes explained in Section 2.4.3. From the application point of view, we think it seems to be more realistic than the MDPDP with inter-depot routes since there exist many cases where transportation between two separated regions is unpractical for the vehicles performing the short distance transportation, and no other option than multi-modal transportation is left.

In MR-MDPDP, all transportation requests have a pickup and a delivery node. These nodes can be within the same region (intra-region requests) or they can be in different regions (inter-region requests). Customer locations (either pickup or delivery) are visited by short-haul vehicles. The goods from inter-region requests have to be transported from the region where the pickup customer is located, to the delivery customer region. This inter-region transportation is done

by long-haul vehicles. Long-haul or inter-region vehicles connect the regions and transport the consolidated goods. They have to be scheduled so that all requests are fulfilled and cost minimized. In the case where there is only one depot in each region the inter-region vehicles represent a standing connection between the regions which has to be used due to a lack of alternatives. However, the departure times can still be either fixed (scheduled) or flexible (up to optimization). If multiple depots are present in each region, the additional decision occurs, which of the possible long-haul connections to use. Since each depot can be connected to every other depot of any other region, several possibilities arise. Figure 2.7 shows a simple representation of a solution of a MR-MDPDP with two regions and two depots in each region.

To the best of our knowledge, this problem has not been considered in the literature yet. However, all related problems and literature detailed for the MDPDP with inter-depot routes in Section 2.4.3 apply to the MR-MDPDP, and even in a greater degree, since in this case multi-modal transportation is required and we face a pure two-level decision problem like the problems from 2E-VRP and LRP families.

The characteristics of this problem induce the need of introducing an index for the region in most of the sets and decision variables in the mathematical model. We also have to make a distinction between requests going over inter-region transportation and requests for which pick up and delivery are in the same region. Since the introduction of this problem is one of the main goals of this paper and due to the several changes that need to be made on the mathematical formulation, we present the complete model for the multi-region problems.

Sets

U = set of regions,
R^I = set of inter-region requests, with customers located in different regions,
R^u = set of intra-region requests, with both customers in region u,
N^u = customer points in region u,
L^u = set of depots in region u,
$V^u = N^u \cup L^u$,
K_l^u = set of short haul vehicles in depot l of region u,
$K^u = \cup_{j \in L^u} K_j^u$,
H_l^u = set of long haul vehicles in depot l of region u,
$H^u = \cup_{j \in L^u} H_j^u$.

Parameters

c_{ij}^u = distance between points i and j in region u,
c_{lp}^{uw} = distance between depots $l \in L^u$ and $p \in L^w$,

$$d_i^u = \begin{cases} 1, \text{ if request } i \text{ starts in region } u, \\ 0, \text{ otherwise.} \end{cases}$$

$$q_i^u = \text{quantity to serve at customer } i \text{ in region } u \begin{cases} > 0, \text{if it is a pick up point,} \\ < 0, \text{ if it is a delivery point,} \\ 0, \text{if customer } i \text{ is not in region } u. \end{cases}$$

Q^{SH} = capacity limitation for short-haul vehicles,
Q^{LH} = capacity limitation for long-haul vehicles.

Decision Variables

$$x^u_{ijk} = \begin{cases} 1, \text{arc } i-j \text{ in region } u \text{ is covered by vehicle } k \in K^u, \\ 0, \text{otherwise.} \end{cases}$$

$$y^{uw}_{ilph} = \begin{cases} 1, \text{request } i \in R^I \text{ transported from depot } l \in L^u \text{ to depot } p \in L^w \text{ in long-haul vehicle } h \in H^u_l, \\ 0, \text{otherwise.} \end{cases}$$

$$z^{uw}_{lph} = \begin{cases} 1, \text{vehicle } h \in H^u_l \text{ is used for transporting between depots } l \in L^u \text{ and } p \in L^w, \\ 0, \text{otherwise.} \end{cases}$$

S^u_{ik} = quantity loaded in tour $k \in K^u$ after visiting customer i,
t^u_{ik} = fulfillment time at customer i from region u with vehicle $k \in K^u$,
b^u_k = beginning time of route performed by vehicle $k \in K^u$,
e^u_k = ending time of route performed by vehicle $k \in K^u$,
m^u_h = starting time of inter-region trip with vehicle $h \in H^u$.

For simplicity in the notation, copies of depot $l \in L^u$ representing starting and ending depot respectively in each region will be denoted as l^u_0 and l^u_1. Indexes in N^u are constituted first with the customers belonging to inter-region requests and then with pickup and delivery customers belonging to intra-region requests in u (R^u).

$$\text{minimize} \quad \sum_{u\in U}\sum_{i,j\in V^u}\sum_{k\in K^u} c^u_{ij}\cdot x^u_{ijk} + \sum_{u\in U}\sum_{w\in\bar{u}}\sum_{l\in L^u}\sum_{p\in L^{\bar{u}}}\sum_{h\in H^u_l} c^{uw}_{lp}\cdot z^{uw}_{lph} \tag{2.19}$$

Subject to

$$\sum_{j\in V^u}\sum_{k\in K^u} x^u_{ijk} = 1 \qquad \forall u\in U, i\in N^u \tag{2.20}$$

$$\sum_{i\in V^u} x^u_{ihk} - \sum_{i\in V^u} x^u_{hik} = 0 \qquad \forall u\in U, h\in N^u, k\in K^u \tag{2.21}$$

$$\sum_{i\in V^u} x^u_{l^u_0,ik} = \sum_{i\in V^u} x^u_{i,l^u_1,k} \le 1 \qquad \forall u\in U, l\in L^u, k\in K^u_l \tag{2.22}$$

$$t^u_{jk} = \sum_{i\in V} x^u_{ijk}\cdot\left(t^u_{ik} + c^u_{ij}\right) \qquad \forall u\in U, j\in V, k\in K \tag{2.23}$$

$$b^u_k = t^u_{l^u_0,k} \qquad \forall u\in U, k\in K^u, l\in L^u$$

$$e^u_k = t^u_{l^u_1,k} \qquad \forall u\in U, k\in K^u, l\in L^u \tag{2.24}$$

$$t^u_{ik} < t^u_{i+|R^u|,k} \qquad \forall u\in U, i\in R^u, k\in K^u \tag{2.25}$$

$$S^u_{jk} = \sum_{i\in V^u} x^{ut}_{ijk}\cdot(S^u_{ik} + q^u_i) \qquad \forall u\in U, j\in N^u, k\in K^u \tag{2.26}$$

$$0 \le S^u_{ik} \le Q^{SH} \qquad \forall u\in U, i\in V^u, k\in K^u \tag{2.27}$$

$$\sum_{i\in R^I} y^{uw}_{ilph} \le M\cdot z^{uw}_{lph} \qquad \begin{array}{l}\forall u\in U, w\in U\setminus u, l\in L^u \\ \forall p\in L^w, h\in H^u_l\end{array} \tag{2.28}$$

$$d_i^u \cdot \sum_{j \in V^u} \sum_{k \in K_l^u} x_{ijk}^u e_k^u \leq \sum_{h \in H_l^u} \sum_{p \in L^w} y_{ilph}^{uw} m_h^u \qquad \forall u \in U, w \in U \setminus u, i \in R^I, l \in L^u \tag{2.29}$$

$$\sum_{j \in V^w} \sum_{k \in K_p^w} x_{ijk}^w b_k^w \geq d_i^u \cdot \sum_{l \in L^u} \sum_{h \in H_l^u} y_{ilph}^{uw} (m_h^u + c_{lp}^{uw}) \forall u \in U, w \in U \setminus u, p \in L^w, i \in R^I \tag{2.30}$$

$$\sum_{i \in N^u} y_{ilph}^{uw} q_i^u \leq Q^{LH} \qquad \begin{array}{l} \forall u \in U, w \in U \setminus u, l \in L^u \\ \forall p \in L^w, h \in H^u \end{array} \tag{2.31}$$

$$x_{ijk}^u \in \{0,1\} \qquad \forall i, j \in V, k \in K \tag{2.32}$$

$$y_{ilph}^{uw}, z_{lph}^{uw} \in \{0,1\} \qquad \forall i \in V,\ l, p \in L, h \in H, u, w \in U$$

$$t_{ik}^u, S_{ik}^u, b_k^u, e_k^u \in \mathbb{Z} \qquad \forall i \in V, k \in K$$

$$m_h \in \mathbb{Z} \qquad \forall h \in H$$

Constraint (2.20) ensure that all customers in region u are visited once and only once by a vehicle from region u. A vehicle arriving at a customer must leave it in the same tour (2.21). A vehicle belonging to depot l in region u has to arrive at depot l, if it leaves it (2.22). Constraint (2.23) refers to time continuity constraints: the arriving time to customer j with vehicle k equals the time the previous customer of vehicle k was serviced plus the travel time between i and j. Constraints modeling the beginning and end time of vehicle k: Start time equals the time when vehicle k leaves the depot. End time is when vehicle k arrives at the depot (2.24). Time precedence constraint for intra-region pickup and delivery requests (2.25). Constraints (2.26) and (2.27) model the load of a short-haul vehicle when visiting customer j and ensure the load stays below the short-haul capacity limitation. Constraint (2.28) binds decision variable y and z. Constraints (2.29) and (2.30) bind decision variables x and y. If inter-region request i starts in region u, the end time of the vehicle k visiting the pickup customer must be smaller than the departing time of the long-haul vehicle h transporting request i. Equivalently, the beginning time of vehicle k' visiting the delivery customer must be greater than the arriving time of the long-haul vehicle h to depot p. The load on a long-haul vehicle cannot exceed the long-haul vehicle capacity limitation (2.31).
The linearization of constraints (2.26), (2.29) and (2.30) is shown in the appendix.

Methods

As previously stated, the MR-MDPDP is a two-level decision problem in which interrelated scheduling and routing decisions have to be made. Any change in the scheduling of requests traveling in long-haul vehicles affects the routes in both regions and vice versa. Therefore we face a problem with two simpler underlying problems. The routing problem corresponds to a MDVRPMB (see Section 2.4.1). It is based on the classical capacitated VRP, which is itself a NP-hard problem (see Cordeau et al. [28]). Therefore the MR-MDPDP also belongs to the category of NP-hard problems. This means that an exact approach to the solution is only possible for instances of very small size, and therefore an heuristic approach is needed for solving problems of a realistic size.

In order to assess how far we can get with the use of commercial solvers, the previous model has been implemented and applied to instances of different sizes. A time limit of two hours was set for every run. All computational results and data description can be found as online supplements to this publication. In general, within the specified computational time we are able to solve instances with up to 4 inter-region requests and 4 intra-region requests.

Consequently, heuristic algorithms are the only feasible option for dealing with bigger instances. Many types of algorithms have been used for problems with similar two-level decision structure, like the families of 2E-VRP and LRP. Looking at the most relevant 2E-VRP literature (presented in Section (2.4.3)), however, it is possible to observe that most of heuristics used consist of two-stage heuristics due to the fact that they provide convenient solution representation. The interrelated variables make it difficult to store a solution in a format required to use them as chromosomes in a population for genetic based algorithms or as labels in a column generation scheme. Hence, large neighborhood heuristics arise as common practice to solve these problems, since the work is done mainly on one single solution. Also, most of the papers make use of two phase algorithms for obtaining initial solutions or to explore neighborhood spaces. That is, they separate the scheduling and the routing problems and consider one of them as an independent problem. This first problem is solved, and afterwards the second one is approached according to the solution obtained from the first phase. However we found an interesting case in the work from Ghilas et al. [64], where a branch-and-price algorithm is successfully applied, getting solutions that outperform in quality those from the ALNS approach, and in time those from CPLEX. However, as earlier mentioned, they need to design a complex labeling system, far from the standard labeling systems applied in general VRP problems.

Everything stated above suggests that probably the best way of approaching a solution method for the MR-MDPDP would be by making use of large neighborhood heuristics, as well as two-phase heuristics with a more particular design focused on each type of problem.

2.6 Conclusion

Multi-region problem scenarios are becoming more common everyday. Different real-world applications include intra-region as well as inter-region transportation, in which, regions are connected by vehicles of larger capacity, different speed and cost.

In-town logistics are constantly evolving in order to reduce costs and increase productivity. While big trucks and trains are an advantage when traveling from one city to another, smaller trucks are used for pickup and delivery operations inside a city. Other multi-modal models have to be used in certain cities, as the only transportation means that are allowed in some neighborhoods are small electric vehicles.

This work introduces this new type of logistic problems which main focus lies on pickup and deliveries in multiple geographically separated regions. The characteristics of these problems lead to the necessity of dealing with other issues like multi-modal transportation and multi-depot networks. An exhaustive examination of already existing literature in related topics is also presented, justifying the multi-region scenarios as a logical evolution step towards more realistic problems in transportation logistics and setting a basis for further research in this direction. This work also contains a step-by-step construction of mathematical models, which highlights the relation between classical transportation problems and the new introduced logistic networks.

2.7 Outlook and possible extensions

As logistics operations are centralized to save resources, n-regions multi-modal problems are an attractive scenario for further research. Companies can use big vehicles like airplanes, trains and trucks to connect main depots to a central location, smaller trucks can be used to reach smaller regional depots and from these, smaller depots, stores, dealers and/or customers can be visited with small vehicles routes. It is very rare to find models that are capable of solving this kind of problems.

Multi-depot, multi-region problems are of particular importance in collaborative settings. Although all cited studies assume to have one single decision maker who aims for a centralized solutions, this assumption does not necessarily hold for collaborative vehicle routing problems.

Probably many companies are involved in such collaborative networks, and therefore decentralized decision making has to be investigated.

Collaborative Networks have been studied and developed over the last years. In works like Berger and Bierwirth [15], Gansterer and Hartl [62], Wang and Kopfer [164]. These studies were, however, focusing on single depot per carrier settings. In a multiple depot approach, collaborative networks could include collaborative depots that could be used by all participants. When thinking about a multiple region structure, collaboration might be even more beneficial, by allowing carriers to share depots and short-haul vehicles on every region, and by splitting costs in combined long-haul trips.

Real world applications of collaborative networks can be found in Montoya et al. [110] and Buijs et al. [19], where a quantification of the obtained improvement suggests that collaboration is a plausible and recommendable practice. Once collaboration is performed, a final aspect to take into account is profit sharing. Guajardo and Rönnqvist in [72], present a recent overview on cost and profit sharing allocation methods.

Besides solution methods, collaborative networks could be improved with managerial implications suggesting best practices in this field. The decision on how much information does a company need to share in order to get the maximum advantage is one of the main questions.

Furthermore we introduce some possible extensions for the problems. All extensions are presented with respect to the model in Section 2.5. These show the extensive gap between already existing literature and a rough approximation to reality, making these problems an attractive field for future research since many real world extensions are not considered so far.

2.7.1 Time windows

Time windows have already been used a lot in the literature and can be added in many different ways. Usually the nodes or customers to visit have soft or hard time windows for the visit. A soft time window that is not met, results in penalties that impair the objective value. A hard time window that is not met, results in an infeasible solution. Time windows can also be considered on the depot opening hours. A clear example of a routing problem with time windows can be found in Dondo and Cerda [41].

Some changes should be done to the presented models when including time windows. Let $[a_i, f_i]$ be the time window for customer i.

- Soft time windows. The objective function must be modified to account for penalty costs. Let ϕ represent the relative penalty cost. The new objective cost would be:

$$\begin{aligned}\text{minimize} \quad & \sum_{u\in U}\sum_{i,j\in V^u}\sum_{k\in K^u} c^u_{ij}\cdot x^u_{ijk} + \sum_{u\in U}\sum_{w\in\bar{u}}\sum_{l\in L^u}\sum_{p\in L^{u*}}\sum_{h\in H^u_l} c^{uw}_{lp}\cdot z^{uw}_{lph} \\ & + \sum_{u\in U}\sum_{i\in V^u}\sum_{k\in K^u} \phi\cdot max(0, t^u_{ik} - f_i)\end{aligned} \tag{2.33}$$

- Hard time windows. New constraints have to be added.

$$a_i \leq \sum_{k\in K^u} t^u_{ik} \leq f_i \qquad \forall u\in U, i\in V^u \tag{2.34}$$

2.7.2 Multiple periods

Another common extension in routing problems is to consider multiple periods. Generally this problems introduce the possibility for requests to be performed on different days. That induces another degree of freedom to the combinatorial problem, widening the solution space.

Servicing a customer on different days can lead to significant improvements in the objective function. Storage of goods in depots is a realistic option for these problems. The work of Vidal et al. [160] contains a clear explanation of multiple period models. A hybrid genetic metaheuristic is developed to solve a variety of problems within the multi-depot and periodic VRP families.

Multiple periods induce a change on the variables of the model. A temporal index should be added to all variables that could be dependent on it. If we name o the new index representing the days in the planning horizon, the variables of our model would be the following:

$$x^u_{ijko}, y^{uw}_{ilpho}, z^{uw}_{lpho}, S^u_{iko}, t^u_{iko}, b^u_{ko}, e^u_{ko}, m^u_{ho}.$$

2.7.3 Heterogeneous vehicles

A problem with an heterogeneous fleet is a problem where vehicles with different problem-relevant characteristics exist. It refers to vehicles within the same mode of transportation and performing the same kind of service. Vehicles can be different if they have different capacities, fixed or variable cost, range (especially for electric vehicles), driving time restrictions or restrictions concerning the arcs they can travel, different speed or different requirements for the drivers (not all drivers have a license for all vehicle types). In a realistic scenario, a company will generally have an heterogeneous fleet.

Regarding heterogeneous fleets, the following works have to be mentioned: The work by Dondo and Cerdá [41] is a multi-depot VRP with heterogeneous fleet and time windows. Their vehicles differ in capacity, cost and speed. Irnich [81] works on a single region multi-depot VRP with an heterogeneous fleet. The vehicles in the fleet differ by cost and speed (regarding driving and loading), and capacity. Salhi and Sari [143] consider a single region multi-depot capacitated VRP with the objective to construct vehicle routes and determine the composition of the vehicle fleet. The vehicles in the fleet differ in capacity and cost. The aim of the authors is to construct a set of routes and determine the vehicle fleet composition minimizing costs.

The changes on our model derived from the use of heterogeneous fleet would be reflected on the parameters and sets. Also we would need an extra index for the variables. Assume we have n vehicle types for the short-haul routes and r vehicle types for long-haul routes. The new elements would be:

$K^u_{ln}, K^u_n, H^u_{lr}, H^u_r$, sets of short-haul and long-haul vehicles of each type in depot l,
$c^u_{ijn}, c^{uw}_{lpr}, Q^{SH}_n, Q^{LH}_r$, costs of transportation and capacities for each vehicle type,
$x^u_{ijkn}, y^{uw}_{ilphr}, z^{uw}_{lphr}, S^u_{ikn}, t^u_{ikn}, b^u_{kn}, e^u_{kn}, m^u_{hr}$, variables containing information about the vehicle type.

2.7.4 Multi-attribute vehicle routing problems

Multi-attribute or rich VRPs are labels for VRPs with additional constraints that aspire to express a more realistic problem structure and decision choices (multiple depots, fleets and commodities that take place in multiple periods and have to consider driver work rules, traffic congestion and much more).

A rich VRP can be found in the work by Ceselli et al. [22] who use a heterogeneous fleet of vehicles, multiple depots, multiple capacities, time windows associated with depots and customers, incompatibility constraints between goods, depots, vehicles, and customers; maximum route length and duration; upper limits on the number of consecutive driving hours and compulsory drivers' rest periods; the possibility of skipping some customers and using express courier services instead of the given fleet to fulfill some orders; the option of splitting up the orders; and the possibility of open routes that do not terminate at depots.

Vidal et al. [161] present a survey on heuristics for multi-attribute VRPs. However, in spite of their extensive pool of constraints, both papers do not consider multi-modality, the possibility of consolidation, or more complex network structures.

These considerations would lead to many different changes in our model from Section 2.5, depending on which attributes are added to the problem.

2.8 Appendix: Linearization of non-linear constraints

Some constraints in the previously presented models are not linear. Here we present the linearization of these constraints, in order to provide full information about the way the models can be implemented.

All non-linear constraints can be grouped in two main equation classes.

1. The first one applies to time continuity and capacity constraints. They appear in constraints 11, 23, 5 and 26. An additional variable A_{ijk} is defined, and big M notation is used.
   Original constraint:

$$S_{jk} = \sum_{i \in V} x_{ijk} \cdot (S_{ik} + q_i)\,, \left(t_{jk} = \sum_{i \in V} x_{ijk} \cdot (t_{ik} + c_{ij}) \right) \qquad \forall j \in V, k \in K,$$

   New constraints:

$$x_{ijk} + A_{ijk} = 1 \qquad \forall i, j \in V, k \in K,$$
$$S_{ik} + q_i - S_{jk} \leq M A_{ijk} \qquad \forall i, j \in V, k \in K,$$
$$S_{ik} + q_i - S_{jk} \geq -M A_{ijk} \qquad \forall i, j \in V, k \in K,$$

2. The second linearization applies to the flow consistency constraints for the models with inter depot routes 14, 15, 29 and 30. Two new decision variables are defined: α_{ik}, β_{ilh}.
   Original basic constraint:

$$\sum_{j \in N \cup \{l\}} \sum_{k \in K_l} x_{ijk} e_k \leq \sum_{h \in H_l} y_{ilph} m_h \qquad \forall i \in R^I, l, p \in L,$$

   New constraints:

$$\sum_{j \in N} x_{ijk} e_k = \alpha_{ik} \qquad \forall i \in R^I, k \in K, (Non - linear)$$

$$\sum_{l' \in L \setminus l'} y_{ill'h} m_h = \beta_{ilh} \qquad \forall i \in R^I, l \in L, h \in H^l, (Non - linear)$$

$$\sum_{k \in K^l} \alpha_{ik} \leq \sum_{h \in H^l} \beta_{ilh} \qquad \forall i \in R^I, l \in L,$$

   The first two of the new equations have to be linearized too. We show the linearization of the first one, and the same method applies to the second one. Big M notation is again used.

$$\alpha_{ik} \leq M \sum_{j \in N} x_{ijk} \qquad \forall i \in R^I, k \in K,$$

$$\alpha_{ik} \leq e_k \qquad \forall i \in R^I, k \in K,$$
$$\alpha_{ik} \geq e_k - M(1 - \sum_{j \in N} x_{ijk}) \qquad \forall i \in R^I, k \in K,$$

Chapter 3

Solution techniques for the inter-modal pickup and delivery problem in two regions

Published in: *Computers & Operations Research*
Solution techniques for the inter-modal pickup and delivery problem in two regions.
A.G. Dragomir & K.F. Doerner. © *2020, Elsevier*
https://doi.org/10.1016/j.cor.2019.104808

Abstract This work addresses the routing problem faced by transportation carriers and postal services that transport small parcels in large quantities. By splitting the territory into regions, these service providers can adapt a three-part network structure and solve a pickup and delivery problem with long-hauls without direct shipments between regions. That is, bilateral cross-city and cross-country requests must be met while also fulfilling capacity and time window constraints. To address this challenge, this work limits the problem to two regions and thereby can identify the correlations and synchronization between different modes. The proposed solution approach decomposes the problem into two subproblems: a long-haul assignment that can be solved exactly, and a short-haul routing problem that must be solved heuristically. The result is an efficient matheuristic, whose quality is confirmed through a comparison with findings from previous literature; it is viable in terms of solution quality and computation time. Long-haul flexibility also influences short-haul routing cost, such that improvements of up to 22% are possible merely by increasing long-haul flexibility but not long-haul cost. Finally, realistic instances are solved based on the inter-library loan system comparing the influence of selecting train or truck on the SH routing costs.

3.1 Introduction and motivation

Transportation carriers and postal services transport small parcels in large quantities and operate in vast territories, divided into regions for better planning and execution. Parcels are delivered on the same day from origin to destination, from one customer to another, though they often are located in different regions. To operate more efficiently and environmentally, many carriers have adopted inter- or multi-modal network structures that split transportation between short-haul (SH) and long-haul (LH) operations. That is, the first- and last-mile delivery operations are performed exclusively by SH vehicles, whereas LH transportation is dedicated to the connections across regions or cities, used to primarily consolidate shipments. Goods transported using the different parts of the network include at least one LH connection, whereas cross-city orders can be served solely by SH vehicles. A real life application of such a structure

is the inter-library loan system. The cities of Austria offer a vast variety of specialized libraries, archives, institutions of estate administration, offices, and museums, each with their own unique and borrowable collection. These collections are relevant for students, scholars, office holders, and the interested public alike. Specialized research would require travel to libraries located in different cities. The inter-library loan system makes sharing possible by providing a network of shipping copies or original documents to the reader.

To study such network structures, a two-region setting is the smallest non-trivial option, in that it includes all relevant properties of the different modes, reflects the influence of different LH timetables, and acknowledges the importance of synchronization among transportation modes and different regions. To serve such regions, a three-part network is typical of postal services (e.g., messenger services, express mail, parcel services) [5]. With this research setting, we consider an extension of the pickup and delivery problem (PDP), as has been studied extensively and with many variations [122], such that we adapt the PDP to this specific network structure. Dragomir et al. [43] introduce problems with a similar structure and confirm the relevance of multiple regions, inter-modality, and paired pickups and deliveries. A strongly related problem has been used for our study of the interrelation between LH flexibility and SH cost.

Other authors also address inter- and multi-modal problems (for detailed reviews see [17, 156, 166]), including contributions that offer a taxonomy of intermodal freight transportation [33]. A key trait of inter- or multi-modality is the task division between the SH (often trucks) and LH (rail, ships, or full truckload trucks) parts of the transportation chain, as well as the need to establish synchronization between modes [17]. The SH also has been described as pre-haul (first mile) or end-haul (last-mile) [156]. Previous works have studied this problem in varying degrees and are hereafter sorted by similarity to our own problem formulation.

Previous work often assumes a relaxed network structure such that the customers, even if far apart, are formally located in a single region. A direct delivery by SH vehicles is almost always an option suggesting that LH transportation is optional, which is not a realistic scenario when deliveries span multiple cities. In this sense, our research diverges notably from otherwise similar contributions from Ghilas et al. [65, 64], Aldaihani and Dessouky [3], Liaw et al. [94], and Moccia et al. [107].

Ghilas et al. [64, 65] investigate the PDP with time windows and scheduled lines, along with the novel idea of using public transportation for part of the transit. Using this public transport vehicle (or 'scheduled line'), analogous to LH, is optional. In their study the pickup and delivery nodes are all located in one region, so all SH vehicles can visit all nodes. To solve the problem, Ghilas et al. use adaptive large neighbourhood search (ALNS) and branch-and-bound in an effort to minimize the total costs (i.e., travel cost for the pickup and delivery vehicles and cost of the scheduled lines, which depends on the quantity shipped but is independent of travelled distance). Due to the similarity of their problem to ours their instances were chosen for comparison.

Aldaihani and Dessouky [3] and Liaw et al. [94] both investigate the dial-a-ride problem with up to two modes of transportation: last mile (curb-to-curb service, analogous to SH) and fixed route transit (analogous to LH) with buses. To investigate how to transport elderly passengers to and from hospitals, Aldaihani and Dessouky rely on multiple bus routes, similar to Ghilas et al. [65, 64]. Their problem is bound to a single region and does not require obligatory transit. Their solution minimizes both distance and travel time for passengers; to achieve these two separate, sometimes conflicting objectives, they adopt a lexicographic approach. Using heuristics, they show that switching from direct delivery to an option with transfers reduces overall distance and trip time, due to the clustered nature of residences and hospitals. The clusters themselves are far apart and connected by a bus line. Because the nodes are part of a single connected region, instead of two separate regions, transshipments (i.e., switching between transportation modes) are optional.

Liaw et al. [94] focus on people with handicaps who require transportation between their home and other locations of interest, like hospitals or work. To limit customer inconvenience,

they allow only direct transportation or the inclusion of a single bus trip; switching between buses or between SH vehicles is not permitted. Time windows are established for pickup, delivery, or both. With a lexicographic objective function, these authors seek to maximize the serviced number of requests. Then, for a fixed number of requests, they seek to minimize the total travel distance of the vehicles. Both the time windows and the maximum allowable excess riding time of the passengers are hard constraints that must be met; excess riding time however is relative to the direct riding time of each passenger. In an online setting, requests come in one at a time, and a simple greedy heuristic was designed to include buses whenever possible, as well as all additional locations for routing the SH vehicles. In offline cases, all requests are known in advance. An iterative improvement procedure, based on simulated annealing, including local search techniques, is applied to find a good approximate solution in an acceptable amount of computation time. Including fixed bus routes with fixed schedules, they find a solution that can accommodate more requests while decreasing the amount of direct transportation, compared with the manual solutions previously obtained by the schedule operators.

Moccia et al. [107] review a PDP in a multi-modal network, in which shipments must be routed through the network in accordance with time windows and multiple capacity constraints of consolidations (volume, weight). The individual modes operate on either fixed or flexible schedules and evoke different cost functions. The planning horizon over multiple days includes multiple non-overlapping time windows for pickups and deliveries with the goal of finding the lowest cost for routing within the network. In this case, direct delivery, or dedicated origin-destination shipment, is always an option without requiring transshipment or multi-modality. The authors use a column generation approach to compute lower bounds that they embed in a heuristic algorithm, consisting of a limited branch-and-cut search, to obtain feasible solutions. However, they were unable to solve a real-life instance of 122 requests optimally, so they created smaller instances of 10, 30 and 60 requests. Still, the quality of their solution heuristic is good compared with the best known lower bounds.

Postal services and express shipping routing problems constitute a specific aspect of the inter- or multi-modal literature with a three-part network structure. Parcels need to be transported within and between regions. The distribution of customer nodes reflects urban agglomeration and therefore is clustered. The different regions are far enough apart to warrant a LH connection, and direct delivery by SH vehicles between regions is not an option, due to the large distances and low cost demand on this industry. This network structure thus requires inter-modal transportation, which always involves the consolidation of goods in the LH vehicles [31]. Grünert and Sebastian [71] describing the Deutsche Post postal service, introduce a similar network structure and consider multiple clusters, but they only address the tactical problem (LH planning) of achieving next day delivery of all mail, without citing operational issues like explicit SH routing. In their work, SH planning consists of vehicle and driver assignments, and air transportation (direct, over hub, combination) and truck transportation are analysed in detail. They introduce different models of LH transportation for different specific parts of the network; a system-wide optimization would not be solvable within reasonable computation times for a transportation problem of relevant size. They separately model tactical planning of air transportation (aircraft scheduling and request-to-flight allocations), ground transportation (requests not assigned to the air network), and scheduling of vehicles, aircraft and drivers, without discussing explicit solution techniques.

Barnhart and Schneur [10] investigate an express shipment design problem, provide and solve a model using column generation. The elaborate LH connection allows for multiple stops and directs all LH paths over a central hub. Packages are first collected at shipment centres and then brought together in a LH station. They include various costs (i.e., fuel, drivers, crew cost) in all parts of the system (ground transport, handling, air transport) within their objective.

When considering cross-city or cross-country pickup and delivery problems one has to be aware of the vast scope that is possible to include. Although the newest topics from multi-modal city-logistics prove to be insightful and promising, they exceed the scope of this work which is focusing on the inter-modality in terms of LH transportation. Therefore we refer the interested

reader to the most recent publications by Savelsbergh and Van Woensel [145], Perboli and Rosano [124], Perboli et al. [123] and Oliveira et al. [117].

In general, prior inter-modal literature, despite some obvious similarities, does not consider all components of the problem at hand. Instead of an operational planning problem with an inter-modal structure, researchers often consider only strategic (e.g., infrastructure investment decisions like hub allocation problems) or tactical (e.g., network flow planning, service network design) planning problems, whereas operational planning problems are only rarely addressed [43]. Therefore, and reflecting the relevance for postal services and transportation carriers, we focus on studying the pickup and delivery problem with long-hauls (PDPLH). To specify the effect of the interaction between LH and SH transportation, we limit our focus to two regions, which provides the smallest non-trivial problem setting.

In turn, the contributions of this study are threefold:

1. We introduce a decomposition approach and matheuristic for the PDPLH in two regions solving the LH assignment subproblem exactly and the SH routing subproblem heuristically using a variation of the pilot method proposed by Voß et al. [162]. This method offers a fast algorithm that also can perform bundle generation in auctions in a collaborative setting [62] and a variation of this method has been proven suitable for distance approximations for routing solutions [116]. It has been successfully applied in combination with other metaheuristics for network design [80] and routing problems [103, 131].
2. We establish the quality and efficiency of the solution method applying it to state-of-the-art instances from Ghilas et al. [64] and comparing the results of their branch-and-bound algorithm. We also analyse different parameter settings for the solution method.
3. By conducting a managerial analysis of the effects of a more flexible LH schedule and its influence on SH routing and overall costs, we determine the importance of synchronization between transportation modes, as well as whether an increase in LH vehicles and departure times leads to better overall solutions in an operational planning setting.

Section 3.2 formally introduces the problem, and then in Section 4.3, we detail the proposed solution method. Along with the computational results, we offer a comparison with Ghilas et al. [64] in Section 7.4 and an evaluation of the effects of a more flexible LH on overall costs in Section 3.4.2. Finally, we conclude in Section 7.5.

3.2 The pickup and delivery problem in two regions with long-hauls

A three-part network in two regions/cities is a suitable starting point for studying the PDP for transportation carriers that operate in a network. Each city consists of geographically separated nodes and has a depot that represents the start and end point of all SH vehicles assigned to this city. Each city also contains a station for available LH connections, used as a consolidation and transshipment point. The station is usually a separate location from the depot, but the two locations may be identical. All nodes (customer nodes, station) have individual service times, reflecting their (un)loading and transfer processes. The LH vehicles depart on a schedule, given as a departure frequency (e.g., departure every 30 minutes). The greater the schedule frequency, the more flexible the LHs are, in that more departure and arrival points are available. Both SH and LH vehicles are limited in capacity and number (the available number of LH vehicles depends on the schedule). Our scope is to fulfil a set of requests as cost effectively as possible and optimize a planning horizon of a single day (10 hours). That is, a feasible solution requires all pickups, deliveries, and LH transportations to be performed within this time span. A request consists of a pickup and a delivery node, as well as a quantity. The nodes can be both within

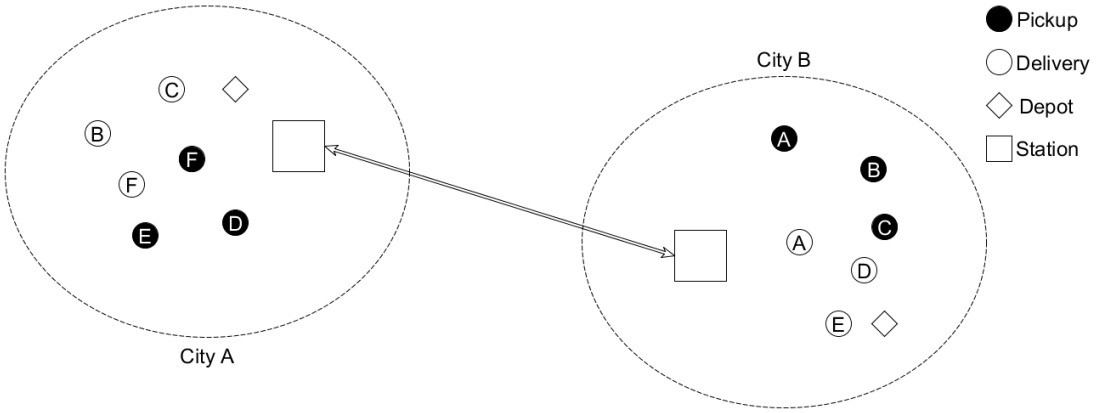


Figure 3.1: A schematic representation with two cities

the same city (cross-city) or in two different cities (cross-country). Cross-country requests have to be transported over the available LH connection and therefore need to be delivered in time to the station, so that they can depart with the scheduled LH; moreover, they are available to be picked up at a station only after the arrival of the LH. The customer nodes also have time windows. Pickup time windows have to be before delivery time windows, but they also require sufficient time for a direct delivery path, reflecting the distances of the nodes to and from the stations, the LH travelling time and the LH schedule.

The goal is to transport all requests within the planning horizon and minimize the total LH and SH costs. In particular, the objective value of a solution is the sum of the SH costs in both clusters plus the LH costs for both directions. The SH costs are the sum of all distances (or arcs) travelled by SH vehicles in each city. The LH costs are composed of the cost of the LH transportation per direction (which corresponds to the distance) multiplied by the number of LH vehicles used. The LH costs are independent of the load. The necessary decisions include the determination of a LH departure time for each request and the sequence in which the nodes should be visited by the SH vehicles. 3.6.1 provides a mathematical model of the complete problem.

Figure 3.1 offers an overview of the problem with two cities. The LH connects the stations and is depicted by a double arrow. There are six requests, indexed by the letters A to F. Requests A and F are cross-city requests; B, C, D and E are cross-country requests, and the directions of B and C are opposite those of D and E.

3.3 Solution algorithm

The complexity of the mathematical model (see 3.6.1) makes it computationally intractable to solve bigger instances within reasonable computation times. Instances with more than 9 requests cannot be solved by CPLEX. Our 8GB of RAM was not sufficient for the memory requirement of CPLEX for more than 9 requests and since using the hard drive results in a considerable increase in computation time and the gap to the optimal solution was still too large, the computations were aborted. Additionally, the problem emerged from an auction based collaboration with the necessity of creating and evaluating bundles of transportation requests within the three-part network [116]. Therefore a solution method that reaches good solutions in short computational times is chosen by focusing on a good construction heuristic, namely a variation of the pilot method by Voß et al. [162], and an improvement metaheuristic to improve the routing and minimize used vehicles.

The proposed solution algorithm splits the problem into two sub-problems which are easier to handle: the LH assignment and the SH routing. The solution algorithm consists of the following steps:

1. Optimal LH assignment for all cross-country requests to minimize LH costs and consolidate the goods according to the model presented in Section 3.3.1.
2. Establish a construction heuristic for the SH routing using variations of the pilot method [162], including a variant of cheapest insertion [83] that accounts for waiting time.
3. Establish an improvement heuristic for the SH routing using a variable neighbourhood descent [106] for minimizing route length and the number of used vehicles.

The decomposition approach ensures that each sub-problem can be solved independently, making the problem easier to solve. The LH assignment is solved optimally due to its simple nature; more sophisticated construction and improvement heuristics are required for the SH routing, which constitutes a more complex PDP with time windows.

3.3.1 Long-haul assignment

The goal of the LH assignment is to select exactly one LH departure time for each request while meeting LH capacity constraints and allowing for potentially feasible SH routing. A feasible SH routing requires consideration of the temporal aspect, such that the LH departure and arrival times are compatible with the time windows set at the customer nodes. The goal is to fulfil all requests and minimize the number of expensive LH vehicles used, while also meeting the time windows for SH vehicles visiting customers by direct delivery from and to the station. The LH assignment model requires identical constraints for both directions. For readability, we present half of the model here, with the caution that the constraints have to be implemented twice, once for each direction. Accordingly, we use the following mathematical model to find an optimal solution to the LH assignment problem:

Sets and Parameters

$\mathcal{N} = \{1, 2, \ldots, |\mathcal{N}|\}$ ... set of node indices
$R = \{1, 2, \ldots, |R|\}$ ... set of request indices
$F = \{1, 2, \ldots, |F|\}$ ... set of LH departure indices
$s^d \in \mathcal{N}$ ... index of departure station node
$s^a \in \mathcal{N}$ ... index of arrival station node

$n_r^p \in \mathcal{N}$ ... pickup node of request $r \in R$
$n_r^d \in \mathcal{N}$ ... delivery node of request $r \in R$
$d_{ij} \geq 0$ ... distance between node $i \in \mathcal{N}$ and $j \in \mathcal{N}$
q_r ... quantity of request $r \in \mathcal{R}$
τ_i^B ... begin of time window of node $i \in \mathcal{N}$
τ_i^E ... end of time window of node $i \in \mathcal{N}$
σ_i ... service time of node $i \in \mathcal{N}$
T_f ... departure time for each LH departure $f \in F$
H ... LH travel time
C ... LH capacity
D ... depot closing time
P ... penalty for not fulfilling a request

Decision variables

$$z_{rf} = \begin{cases} 1, \text{request } r \text{ departs with LH number } f, \\ 0, \text{otherwise.} \end{cases}$$

$$y_r = \begin{cases} 1, \text{request } r \text{ is outsourced (not fulfilled)}, \\ 0, \text{otherwise.} \end{cases}$$

$$g_f = \begin{cases} 1, \text{LH } f \text{ is used (at minimum one request is shipped)}, \\ 0, \text{otherwise.} \end{cases}$$

t_r^d ... time of LH departure for each request r
t_r^a ... time of LH arrival for each request r

$$\text{Minimize } P \sum_{r \in R} y_r + \sum_{f \in F} g_f \tag{3.1}$$

s.t.

$$g_f \leqslant \sum_{r \in R} z_{rf} \qquad \forall f \in F \tag{3.2}$$

$$|R| g_f \geqslant \sum_{r \in R} z_{rf} \qquad \forall f \in F \tag{3.3}$$

$$y_r = -\sum_{f \in F} z_{rf} + 1 \qquad \forall r \in R \tag{3.4}$$

$$\sum_{f \in F} z_{rf} \leqslant 1 \qquad \forall r \in R \tag{3.5}$$

$$\sum_{f \in F} z_{rf} T_f = t_r^d \qquad \forall r \in R \tag{3.6}$$

$$t_r^d + H = t_r^a \qquad \forall r \in R \tag{3.7}$$

$$\sum_{r \in R} q_r z_{rf} \leqslant C \qquad \forall f \in F \tag{3.8}$$

$$max(d_{0n_r^p}, \tau_{n_r^p}^B) + \sigma_{n_r^p} + d_{n_r^p s^d} + \sigma_{s^d} \leqslant t_r^d + My_r \qquad \forall r \in R \tag{3.9}$$

$$t_r^a + \sigma_{s^a} + d_{s^a n_r^d} \leqslant \tau_{n_r^d}^E + My_r \qquad \forall r \in R \tag{3.10}$$

$$t_r^a + \sigma_{s^a} + d_{s^a n_r^d} + \sigma_{n_r^d} + d_{n_r^d 0} \leqslant D + My_r \qquad \forall r \in R \tag{3.11}$$

Constraints (3.2) and (3.3) require that if at least one request is scheduled on a LH, it generates cost. Constraints (3.4) state that each request has to depart with a LH or be 'outsourced', then bear the penalty for not fulfilling a request. Constraints (3.5) ensures that each request can depart with at most one LH vehicle. Constraints (3.6) limit the LH departure times to the given schedule. Constraints (3.7) set the arrival times in relation to the departure times of the LH for each request. Constraints (3.8) ensures that the LH capacity is not exceeded. Constraints (3.9) ensures that the requests can arrive in time for the LH departure. Constraints (3.10) and (3.11) ensure that the requests arrive early enough to enable a delivery in the other region.

A pool of solutions is created, each with different LH departure times for the requests, which are all equally good in terms of absolute objective value. We use CPLEX to populate the pool of solutions. No objective value gap between solutions is allowed, and we apply CPLEX parameters to search specifically for diverse solutions (i.e., solution pool intensity with value 4 and solution pool replacement strategy with value 2). Note that 'outsourcing' or not fulfilling a request is in general possible by our model but avoided by setting the penalty p high enough. Therefore, the model always uses an additional LH vehicle at a different time. The SH routing is performed for all complete solutions, where no request is outsourced. Due to the differences in LH departure times, the SH routing results in different routes (and different costs) for each solution in the pool. The solution with the smallest cost is considered the best solution.

3.3.2 Short-haul routing

All cross-country requests have to travel over the LH and therefore change transportation modes at stations. The LH assignment fixes the departure and arrival times at the stations for each request. The SH routing thus can be performed for each city independently, by duplicating requests that travel over the LH and splitting them into two independent requests in each city. Instead of a station, we use a 'dummy' pickup/delivery location. The departure times of the LH result in an available drop-off time span at the station, within which requests have to arrive to be able to leave with the scheduled LH. Therefore, for each request departing by LH, we also create a 'dummy' delivery node with a drop-off time span that is equal to the time window begin of the paired pickup node and the LH departure time. The 'dummy' pickup nodes are created in the same way: For requests arriving by LH a 'dummy' pickup node is created with a release time span of the LH arrival time and the time window end of the paired delivery node. Within this release time span, requests are available for pickup by the SH vehicles. Figure 3.2 depicts examples of each 'dummy' node. The transshipment time for loading/unloading SH and LH vehicles is a service time at the station node and incorporated in the SH routing.

The solution algorithm is based on the cheapest insertion algorithm inspired by Jaw et al. [83]. The procedure calculates the cost of inserting each request into each vehicle at the best possible position without changing the sequence of already inserted nodes. From this set of possible insertions (with different allocated costs), one insertion is selected and performed. In general, only insertions with the smallest cost are considered. After every successful insertion, the cost for each request-vehicle-combination has to be recalculated. This procedure is repeated until all requests are inserted in the solution.

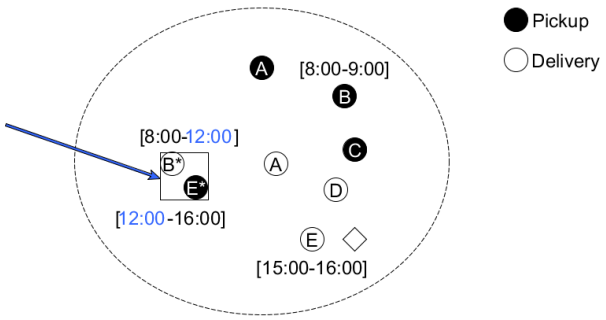


Figure 3.2: Dummy delivery node B* and dummy pickup node E* for cross-country requests. Request E arrives and request B departs with the LH at 12:00. These values have been set by the LH assignment model. The release time span for request E at the station is therefore 12:00-16:00; the drop-off time span for request B is therefore 8:00-12:00.

The insertion cost is the minimum additional distance that a vehicle has to travel to include a request into its route. For an empty vehicle, it is always the direct distance between the nodes to be inserted (see Equation 3.12 for the insertion cost c_r for request r with pickup node i and delivery node j). In addition to the cost of travelled distance, we include the waiting time, with the assumption that waiting at a customer node until the beginning of the time window is unproductive time that prevents the vehicle from servicing other nodes. The weights φ_d for the distance and φ_w for the waiting times w_i and w_j are therefore included in the calculation of the insertion cost. Figure 3.3 shows a graphical representation of the waiting time calculation for a single request, inserted into a previously empty vehicle route. In this example, waiting time only occurs before visiting node i, not before visiting node j.

$$c_r = \varphi_d(d_{0i} + d_{ij} + d_{j0}) + \varphi_w(w_i + w_j) \tag{3.12}$$

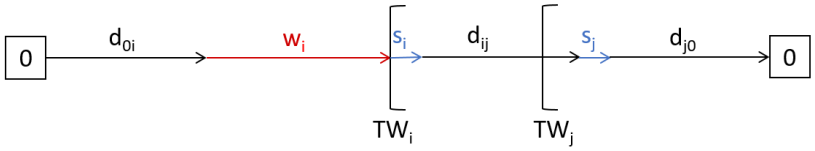


Figure 3.3: Determination of waiting time w_i for insertion into an empty vehicle. 0 represents the depot. No waiting time occurs before visiting node j.

If a vehicle is not empty, all possible positions are considered for inserting an additional request without changing the node sequence. If the pickup node i and delivery node j of the request to be inserted are next to each other, as in Figure 3.4, the insertion calculation is equivalent to Equation 3.13, and the set $V = \{1, ..., v\}$ contains all nodes in the new route. The waiting time w is calculated for each node v depending on the difference between the arrival time at that node and the earliest possible start of service. If the pickup node i and delivery node j of a request to be inserted are separated by at least one other node, the calculation is slightly different, as shown in Figure 3.5 and Equation 3.14. The nodes $i+1$ and $j-1$ can be the same if i and j are separated by only one node.

$$c_r = \varphi_d(-d_{i-1,j+1} + d_{i-1,i} + d_{ij} + d_{j,j+1}) + \varphi_w \sum_{v \in V} w_v \tag{3.13}$$

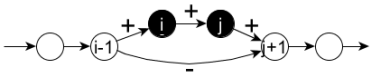


Figure 3.4: Inserting nodes next to each other

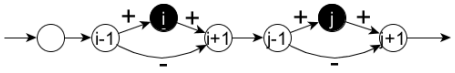


Figure 3.5: Inserting nodes separate from each other

$$c_r = \varphi_d(-d_{i-1,i+1} - d_{j-1,j+1} + d_{i-1,i} + d_{i,i+1} + d_{j-1,j} + d_{j,j+1}) + \varphi_w \sum_{v \in V} w_v \tag{3.14}$$

Information about the position of the nodes gets saved together with the cost of inserting a particular request into a particular vehicle. The cost depend on the position and sequence of the nodes and contain the sum of necessary detours, as well as the weighted waiting time throughout the whole route.

Combining cheapest insertion with the pilot method

In the pilot method proposed by Voß et al. [162], the concept is to 'look ahead' to a possible result when building a solution. Decisions are based solely on the quality of that insertion, defined by the immediate increase in the objective value (greedy approach). The pilot insertion method (PI) evaluates multiple decisions at each step to achieve a better solution at the end of the construction process. Figure 3.6 shows the underlying principle. At each step, the m-best possible choices are considered for insertion, and the solution evaluation only takes place after looking ahead n steps. The decision about which insertion to select depends on the best partial solution obtained. A complete solution, where all requests are fulfilled, is evaluated by distance only, whereas a partial solution needs a cost estimation accounting for not yet fulfilled requests. If distance alone were to be considered for the quality of a partial solution, an empty solution would always (and wrongly) be considered 'better' than one in which all nodes are visited. We want to eliminate the possibility of selecting decisions leading to a partial solution that is 'better' because it has unfulfilled requests, in contrast to another partial solution which is 'worse' because it fulfils all requests and travels a longer distance. Therefore we estimate the cost of a partial solution by considering a penalty for not (yet) included requests, in addition to distance. For n look-ahead steps, the estimated cost differs greatly if one solution path fulfils n requests but the other only fulfils, for example, $n-1$ requests (because the n^{th} request could not be feasibly inserted). The penalty is always greater than any reduction in distance costs. If the same number of requests were inserted, the values of the estimated costs would differ only by distance. For the PI, with a greedy approach for all n look-ahead steps, the best insertion is always selected. By looking ahead, we can determine the solution quality according to the cost estimation and use this as a basis for the current insertion decision. This procedure is repeated at every step of the solution construction, i.e. every time an insertion is permanently chosen for the solution.

The PI was augmented to create an extended pilot method (EPI). The idea is, to not only assume greedy decision for all look-ahead steps but rather always consider all m possible choices.

Figure 3.7 shows the principle of the EPI method. The m possible choices are only drawn for the first decision each, but are considered for every decision in that decision tree. The EPI evaluates more of the solution space than PI. Although it is more computationally expensive, the EPI can find better solutions than the PI.

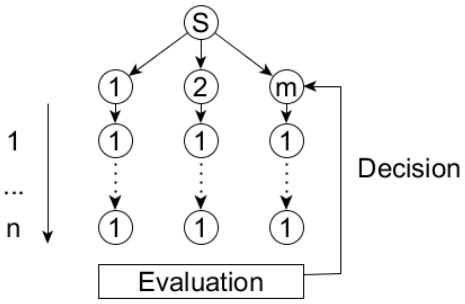


Figure 3.6: Pilot insertion method (PI) with m decisions and n look-ahead steps

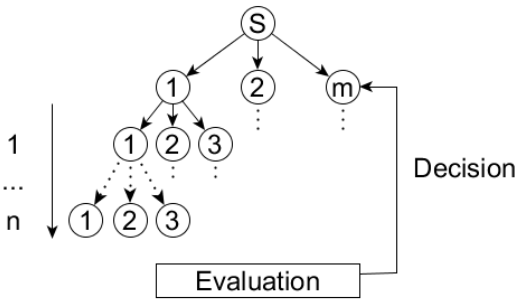


Figure 3.7: Extended pilot insertion method (EPI) with m decisions and n look-ahead steps

VND for the improvement of the short-haul routing

The constructed solution by PI or EPI is further improved by a variable neighbourhood descent (VND); a local search and a variation of the variable neighbourhood search by Mladenović and Hansen [106]. The purpose of the VND is to further improve the routing and reach feasibility (if not previously achieved, due to unfulfilled requests). If unfulfilled requests remain, additional temporary vehicles can be created and the requests are individually inserted into the vehicles. The VND uses 3 different neighbourhoods ($MoveRequest$, $SwapRequest$ and $2Opt*$, in that order) to improve the solution and remove previously added vehicles whenever possible. It starts with the solution provided by the pilot method as an incumbent solution, then moves to the next neighbourhood if no more improvement can be found in the current neighbourhood as a means to escape from a local minimum. Each improvement results in a new incumbent solution and a restart of the VND, such that the number of unsuccessful tries for each neighbourhood is reset to zero. A solution that is equal or worse to the incumbent solution is discarded. The VND ends after no more improvement can be found for a certain number of iterations in each neighbourhood, thus resulting in a local optima for all three neighbourhoods [159].

The neighbourhood $MoveRequest$ picks a random vehicle and a random request served by this vehicle. This request is removed from the vehicle, and the corresponding pickup and delivery nodes are simultaneously inserted at their best positions into the best other vehicle. For this, all possible insertion location combinations are tried out without changing the already present node sequence. The best vehicle is determined by trying all vehicles, apart from the vehicle from which the request was previously removed. If the cost of the newfound solution is better than the cost of the incumbent solution, the move is accepted; otherwise, it is discarded.

The neighbourhood $SwapRequest$ picks two random vehicles and one random request each within those vehicles. The requests are swapped, and the corresponding pickup and delivery nodes again inserted at the most favourable possible position without changing the order of already present nodes.

The neighbourhood $2Opt*$ picks two random vehicles and a random position within the route (a random arc). The routes are 'cut' at this position. The first parts of the cut routes remain with the original vehicle, but the second parts are switched among the vehicles. This cut procedure is not trivial, because the pickup and delivery nodes of a request are not allowed to end up in different routes. Routes can only be cut at places where whole requests would be transferred to another vehicle.

Both the $MoveRequest$ and the $2Opt*$ neighbourhood can result in empty vehicles. A vehicle can be empty if the only remaining request is moved to another vehicle or if the 'cut' of a route occurs before the very first visit of a vehicle, such that the entirety of the route is transferred to another vehicle while simultaneously cutting at the very end of the other route, such that nothing

gets transferred back. Figure 3.8 shows a $2Opt*$ that results in an empty vehicle, because it cuts both routes in such a way that after the swap, the first vehicle is empty and can be eliminated.

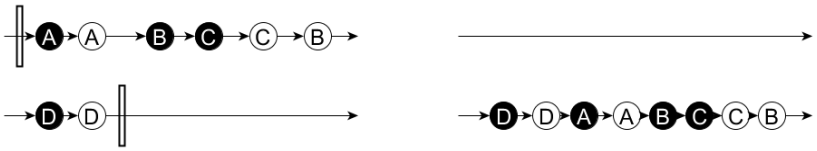


Figure 3.8: $2Opt*$ operator: cuts and swap result in an empty vehicle.

3.4 Computational Experiments and Results

In Section 3.4.1, we provide the instance descriptions. Section 3.4.2 provides managerial insights related to the impact of more LH flexibility on SH costs, while in Section 3.4.3 we validate the quality our algorithm by comparing with existing literature. Then in Section 3.4.4 we apply our solution algorithm to instances from the inter-library loan network. Finally, Section 3.4.5 explores technical differences among the solution methods.

The computational experiments were performed on a desktop computer with an Intel(R) Core(TM) i7-4790 CPU @ 3.60GHz processor (4 cores) and 8 GB RAM running on Windows 10.

3.4.1 Instance description

Our algorithm is tested on three different types of instances.
First, we report computational results from 40 randomly generated instances. The number of requests (R) range from 10 to 100, with a random quantity between 1 and 4. Each instance contains between 0% and 50% of cross-city requests (R^{cc}). There is a limited number of SH vehicles (V^c) at each depot. The instances are generated either without time windows (TW) (i.e., time windows spanning the whole day of 600 minutes) or with TW of 180 minutes each. The coordinates for the customer nodes are randomly distributed in two areas on an Euclidean plane corresponding to two cities. The cities are so far apart that the LH transportation time takes approximately 2.5 hours. The planning horizon is one day (600 minutes). The LH departure frequency ranges from 300 to 30 minutes. Each city contains one depot and one station. The depot can have the same location as the station (S=D), but if it is separate from the station, the locations are generated randomly. In addition to locations for pickup and delivery, each request has a predefined city assignment that establishes if it is a cross-city or cross-country request, according to the assigned city of origin and destination.
Second, we compare our algorithm to the results and instances provided by Ghilas et al. [64]. They provide three sets of instances with a range of requests from 10 to 50. The instances are clustered around the stations (C), uniformly distributed (R), or randomly clustered (RC). The planning horizon is 10 hours. The requests have randomly generated time windows between 26 and 91 minutes, a random demand between 1 and 3 units, and a random loading time up to 3 minutes. The SH vehicles are limited in number and heterogeneous in capacity (up to 20 units). The LH vehicles depart every 30 minutes throughout the day.
Third, we apply our algorithm to realistic instances based on the inter-library loan system. The instances simulate 25 to 40 orders to be delivered within a day. Since the parcels are comprised of single books or documents, the SH and LH vehicles are assumed to have no relevant capacity constraint. The network includes 115 real locations in Vienna and 40 real locations in Graz, each with their (sometimes very restricted) real opening hours and service times between 5 and 10 minutes. The bigger and more popular libraries and archives have opening hours corresponding to a standard work day (e.g. 9 a.m. to 5 p.m.) whereas the smaller and less frequented archives

might close around lunch time or are open only for one to two hours in the afternoon. For all locations that are only accessible by appointment we assumed a general availability from 9 a.m. to 6 p.m. All distances and travel times are the shortest driving route provided by Google Maps. The orders are mostly comprised of inter-city requests, however some intra-city requests are also present. We compare a LH connection by train to one by truck. The train connection assumes the use of a passenger train, on a weekday, between Vienna and Graz, by the time table set by the Austrian Federal Railway. Trains from Vienna to Graz leave hourly at minute 58 (e.g., 06:58, 07:58), trains from Graz to Vienna leave hourly at minute 26 (e.g., 06:26, 07:26) and take 155 minutes. The truck reduces the LH travel time to 120 minutes and allows a more flexible schedule (departures are assumed to be possible every 15 minutes).

3.4.2 Evaluation of the influence of more flexible LH on costs

Since we study a pickup and delivery problem with LH in two regions one of the main questions is not only which LH to choose from the ones offered but rather how many connections to offer. Because costs only accrue for utilized connections, not for offered ones, a possible solution could be to offer as many LHs as possible, so a possible connection exists at every moment of the day. However, this solution greatly increases the complexity of the problem. We therefore study the costs associated with limiting or else extending the choice of possible connections. The possible departure frequencies of LH vehicles, in minutes, range from an extremely limited choice of a single possible LH at an interval of 300 minutes to a more flexible choice of 19 possible LH connections at an interval of 30 minutes. The LH is assumed to depart simultaneously from both stations at the scheduled times. We are particularly interested in the influence of a more flexible LH on SH costs; the LH cost are secondary. Therefore we relax the capacity constraint on the LH vehicles to be able to fulfil all requests even with a single departure.
Table 3.1 shows the absolute EPI values for a LH departure frequency of 75 minutes with and without VND. The first column depicts the instances from 1 to 40. The next five columns give more detailed information about each instance: the number of requests R, the share of cross-city requests R^{cc}, whether the location of the station is identical to the location of the depot 'S=D', the size of the time windows 'TW' and the number of available SH vehicles in each city V^c.

We also list the total costs and objective value 'OV', LH costs 'LH', SH costs 'SH' and computation time in seconds 'T'. The relation of the LH cost to total OV depends mostly on the instance size. For larger instances, more goods can be consolidated, and the LH gets cheaper in relation to SH. The column 'Δ OV' specifies the improvement obtained by including the VND ($\frac{EPI\&VND}{onlyEPI} - 1$). The VND does not result in much improvement for smaller instances but produces improvements up to 21.8% for larger instances. Overall, the VND leads to an average improvement of 11.8%.

Tables 3.2 and 3.3 detail the improvements in SH costs 'SH' and difference in computational time 'T' when changing the LH frequency from 300 to 150 minutes, from 150 to 75 minutes, and from 75 to 30 minutes. The last block of columns shows the overall differences obtained by increasing the LH frequency from 300 to 30 minutes. Negative numbers indicate decreased costs. Table 3.2 shows the values obtained using PI, and Table 3.3 contains the values using EPI. For all instances, we used five look-ahead steps n, three decisions m, a waiting time weight φ_w of 0.3, and 10^4 iterations without improvement until the VND termination. The LH costs are not shown, because they mostly remain identical, with only the timing changed.

Overall, the SH costs improve significantly; only sporadically is a worse solution obtained. On average, the SH costs are 12% lower for the PI and 11% lower for the EPI method, with up to 22% cost improvement for both methods. However, we also observe a large increase in the required computational time.

By increasing the LH departure frequency, we obtain a change in SH routing and in SH costs. The SH total costs can be reduced while keeping LH costs at the same level, because no additional LHs are actually used (though they are offered), and because empty LHs do not

						75LH, only EPI				75LH, EPI & VND				Δ OV
	R	R^{cc}	S=D	TW	V^c	OV	LH	SH	T	OV	LH	SH	T	
1	10	0%	no	600	4	1054.4	354.2	700.2	2.9	1054.4	354.2	700.2	5.2	0.0%
2	10	0%	no	600	4	1217.2	316.8	900.5	2.0	1217.2	316.8	900.5	3.4	0.0%
3	10	0%	no	600	4	894.5	272.1	622.4	3.4	894.5	272.1	622.4	5.7	0.0%
4	10	50%	no	600	4	902.7	202.5	700.2	2.2	902.7	202.5	700.2	4.9	0.0%
5	10	50%	no	600	4	880.2	238.0	642.1	2.2	880.2	238.0	642.1	5.5	0.0%
6	10	0%	no	180	4	1013.3	272.1	741.2	2.2	991.5	272.1	719.4	4.0	-2.2%
7	10	0%	no	180	4	1200.2	215.7	984.4	2.5	1194.4	215.7	978.7	5.5	-0.5%
8	10	30%	no	180	4	964.2	275.5	688.7	0.9	955.1	275.5	679.5	1.7	-1.0%
9	10	50%	no	180	4	1066.6	202.5	864.1	0.7	1032.1	202.5	829.6	1.7	-3.2%
10	10	50%	no	180	4	1084.1	238.0	846.0	0.9	1084.1	238.0	846.0	2.2	0.0%
11	20	30%	no	600	7	1577.4	274.9	1302.4	16.1	1394.3	274.9	1119.4	21.7	-11.6%
12	20	30%	no	600	7	1665.4	278.1	1387.3	16.7	1531.6	278.1	1253.4	24.1	-8.0%
13	20	50%	no	600	7	1463.0	325.4	1137.6	17.0	1380.9	325.4	1055.5	23.6	-5.6%
14	20	50%	no	600	7	1272.3	247.3	1025.0	16.5	1184.2	247.3	936.9	23.8	-6.9%
15	20	50%	no	600	7	1406.7	292.9	1113.8	11.6	1384.1	292.9	1091.2	16.4	-1.6%
16	20	0%	yes	180	7	1509.6	247.4	1262.2	17.3	1429.9	247.4	1182.5	24.2	-5.3%
17	20	10%	yes	180	7	1488.0	247.4	1240.6	20.6	1441.8	247.4	1194.4	28.7	-3.1%
18	20	20%	yes	180	7	1675.3	247.4	1427.9	8.3	1513.6	247.4	1266.2	11.8	-9.6%
19	20	30%	no	180	7	1834.4	278.1	1556.3	10.1	1701.2	278.1	1423.1	15.6	-7.3%
20	20	50%	no	180	7	1609.7	247.3	1362.4	9.0	1450.3	247.3	1203.0	12.8	-9.9%
21	50	30%	no	600	18	2500.2	272.1	2228.2	407.9	2135.9	272.1	1863.9	464.3	-14.6%
22	50	30%	no	600	18	2306.1	262.1	2044.0	480.8	2105.6	262.1	1843.6	543.0	-8.7%
23	50	50%	no	600	18	2683.2	276.0	2407.2	281.1	2237.6	276.0	1961.6	317.3	-16.6%
24	50	50%	no	600	18	2246.7	221.8	2024.9	167.9	1944.8	221.8	1723.0	207.1	-13.4%
25	50	50%	no	600	18	2360.4	346.0	2014.4	292.2	2183.4	346.0	1837.4	339.6	-7.5%
26	50	0%	no	180	18	2955.9	284.1	2671.8	143.2	2569.0	284.1	2284.9	172.8	-13.1%
27	50	0%	no	180	18	2842.0	258.3	2583.7	190.8	2304.7	258.3	2046.5	225.9	-18.9%
28	50	10%	no	180	18	2837.6	218.4	2619.2	210.1	2596.3	218.4	2377.8	237.5	-8.5%
29	50	10%	no	180	18	2785.1	226.9	2558.2	194.3	2391.5	226.9	2164.6	225.7	-14.1%
30	50	30%	yes	180	18	3197.4	247.4	2950.0	109.1	2584.8	247.4	2337.4	127.3	-19.2%
31	100	0%	yes	600	35	3958.5	247.4	3711.1	2477.1	3440.6	247.4	3193.2	3041.8	-13.1%
32	100	0%	yes	600	35	4254.0	247.4	4006.6	2473.4	3587.6	247.4	3340.2	2986.6	-15.7%
33	100	30%	no	600	35	3591.8	225.8	3366.0	1741.3	3112.8	225.8	2887.0	2160.7	-13.3%
34	100	50%	no	600	35	3971.4	274.5	3696.9	689.7	3105.7	274.5	2831.3	888.0	-21.8%
35	100	50%	no	600	35	3800.0	240.4	3559.6	1693.3	3144.2	240.4	2903.8	2169.1	-17.3%
36	100	50%	yes	180	35	4947.0	371.1	4575.9	491.1	4274.1	371.1	3903.1	555.7	-13.6%
37	100	50%	yes	180	35	5293.3	247.4	5046.0	190.0	4508.8	247.4	4261.5	212.5	-14.8%
38	100	50%	yes	180	35	4908.2	247.4	4660.8	500.7	4142.1	247.4	3894.7	564.6	-15.6%
39	100	50%	yes	180	35	4982.9	247.4	4735.5	389.0	4251.5	247.4	4004.2	457.2	-14.7%
40	100	50%	yes	180	35	5214.6	247.4	4967.2	252.5	4650.2	247.4	4402.8	278.8	-10.8%
Avg.						2435.4	262.1	2173.3	338.5	2147.2	262.1	1885.2	410.4	-11.8%

Table 3.1: Absolute EPI values for a LH departure frequency of 75 minutes (LH75), with and without VND and improvement in percentage (Δ OV). The table includes the number of requests (R), the amount of cross-city requests (R^{cc}), whether the station and depot have identical locations (S=D), the time window size (TW), the number of SH vehicles (V^c), the total objective value (OV), LH cost (LH), SH cost (SH), and CPU time in seconds (T).

generate cost. In addition to a reduction in SH costs, we observe a reduction in required SH vehicles. These effects emerge for each increase in LH departure frequency, but they are greatest for the increase from 300 to 30 minutes. The effects also occur independent of time windows, the share of cross-city requests, or the size of the instances.

However, the universal claim that more LH connections are better is insupportable, because the goal must be to offer the right connections, not the most. Ascertaining the right departure times creates the same difficulties as the choice between departures at every minute of the day.

To illustrate the change in SH routing when increasing the LH departure frequency, we depict two explicit examples in Figures 3.10 and 3.11 in 3.6.2. In both cases, we compare a LH departure frequency of 300 minutes with a departure frequency of 30 minutes.

In the case of LH300, a single LH departs in the middle of the planning horizon for each direction, so LHs depart simultaneously from each station. Each city requires at least two routes, though not necessarily two vehicles, namely one route performing all pickups and one route performing all deliveries. First, all pickups are performed in both cities (more or less simultaneously) and brought to the station. Then, all parcels are simultaneously transported via LH to their destination city with both LH vehicles leaving at minute 300. Finally, upon arriving, all parcels are delivered by a separate route (possibly using the same SH vehicles).

In the case of LH30, a potential LH can depart every 30 minutes, throughout the day, starting from the beginning of the day (time 0), including the departure time of LH300. First, the parcels are being picked up in one city and transported to the station on a route that contains only pickups. Then, the first LH leaves. After arrival in the other city, the parcels are picked up from the station, and the SH vehicles perform combined pickup and delivery routes. At the end of this combined route, the station is visited again to drop off all parcels destined for the other city. Then, the LH returns to the first city and finally, upon arriving, the parcels are brought to their destinations by a SH vehicle servicing only delivery nodes.

By allowing multiple LH departure times and a higher departure frequency, we make combined pickup and delivery routes possible, and therefore make room for improving the SH routes. In both instances shown in 3.6.2, the SH and total costs can be improved while having the same LH cost (the same number of utilized LH departures).

To evaluate the robustness of the claim that a more flexible LH leads to improvements in SH costs, we conduct a sensitivity analysis where different LH capacity restrictions are tested. In particular we test the SH improvement of our instances when limiting the LH capacity to 25%, 50%, 75% or 100% of total demand. The box plot in Figure 3.9 shows the average results of those tests. The mean values for all capacity restrictions are stable and below zero (improvements ranging from 4% to 7%) indicating a strong claim that more flexible LHs lead to improvement of SH cost. Additionally, the boxes of the boxplot are stable below zero as well (improvement ranging from 2% to 10%).

Note that, not all instances can be solved with certain LH departure frequencies in combination with a restricted LH capacity. Out of our 40 instances we can solve 25 instances with a LH departure frequency of 150 minutes and a LH capacity restriction at 25%. For a LH departure frequency of 300 minutes (which is the equivalent of a single departure per day) we can solve 38 instances at a capacity restriction set to 75%, 28 instances at a capacity restriction set to 50%, and only 2 instances at a capacity restriction set to 25%. This is due to the fact that we need to fulfil all requests in order to obtain a feasible solution.

The box plots for the more detailed tests comparing different LH departure schedules can be found in 3.6.3. They show, like in Tables 3.2 and 3.3, the results of the SH cost reduction when the LH departure frequency increases from 300 to 150 minutes in Figure 3.13a, 150 to 75 minutes in Figure 3.13b, 75 to 30 minutes in Figure 3.13c, and 300 to 30 minutes in Figure 3.13d. It can be seen that the mean values of SH improvement range from zero to 15% with outliers just over zero (slight worsening) and improvements up to 22%.

						LH150 rel. to LH300		LH75 rel. to LH150		LH30 rel. to LH75		**LH30 rel. to LH300**	
	R	R^{cc}	S = D	TW	V^c	SH	T	SH	T	SH	T	SH	T
1	10	0%	no	600	4	-9%	228%	-10%	134%	0%	324%	-19%	3158%
2	10	0%	no	600	4	-12%	39%	-10%	189%	0%	509%	-21%	2349%
3	10	0%	no	600	4	-11%	185%	0%	92%	0%	555%	-11%	3481%
4	10	50%	no	600	4	-2%	272%	-13%	197%	0%	418%	-14%	5634%
5	10	50%	no	600	4	-5%	210%	-4%	202%	-2%	411%	-11%	4678%
6	10	0%	no	180	4	0%	72%	-9%	140%	0%	294%	-9%	1527%
7	10	0%	no	180	4	0%	85%	-12%	211%	-1%	603%	-13%	3953%
8	10	30%	no	180	4	0%	-30%	-14%	335%	-4%	491%	-17%	1699%
9	10	50%	no	180	4	0%	-30%	-1%	204%	-5%	842%	-6%	1910%
10	10	50%	no	180	4	0%	8%	-1%	422%	-4%	611%	-5%	3912%
11	20	30%	no	600	7	-1%	87%	-6%	187%	-4%	627%	-11%	3796%
12	20	30%	no	600	7	-5%	120%	-1%	135%	-11%	566%	-16%	3333%
13	20	50%	no	600	7	-14%	207%	-1%	118%	-8%	560%	-21%	4320%
14	20	50%	no	600	7	-14%	349%	-1%	72%	-6%	479%	-19%	4375%
15	20	50%	no	600	7	-3%	62%	-5%	176%	1%	713%	-6%	3538%
16	20	0%	yes	180	7	0%	44%	-4%	188%	-3%	384%	-7%	1903%
17	20	10%	yes	180	7	0%	62%	-7%	206%	-1%	318%	-8%	1970%
18	20	20%	yes	180	7	0%	-1%	-5%	346%	-5%	774%	-9%	3766%
19	20	30%	no	180	7	0%	96%	-8%	180%	-1%	514%	-9%	3281%
20	20	50%	no	180	7	-1%	45%	-19%	167%	0%	502%	-20%	2233%
21	50	30%	no	600	18	-2%	353%	-12%	100%	-5%	341%	-18%	3883%
22	50	30%	no	600	18	-2%	471%	-5%	107%	-3%	617%	-10%	8359%
23	50	50%	no	600	18	-8%	250%	-1%	123%	-7%	358%	-15%	3471%
24	50	50%	no	600	18	0%	107%	-1%	202%	-5%	813%	-6%	5605%
25	50	50%	no	600	18	-11%	312%	-6%	48%	-4%	377%	-20%	2810%
26	50	0%	no	180	18	0%	0%	-20%	296%	-2%	481%	-22%	2212%
27	50	0%	no	180	18	0%	-3%	-19%	419%	-3%	280%	-21%	1814%
28	50	10%	no	180	18	0%	-9%	-10%	425%	-2%	345%	-12%	2019%
29	50	10%	no	180	18	0%	21%	-2%	140%	-8%	339%	-10%	1172%
30	50	30%	yes	180	18	0%	-3%	0%	184%	-13%	158%	-13%	611%
31	100	0%	yes	600	35	0%	73%	2%	145%	-3%	415%	-1%	2080%
32	100	0%	yes	600	35	-4%	122%	-4%	243%	-2%	414%	-10%	3815%
33	100	30%	no	600	35	-5%	107%	-5%	124%	-1%	706%	-11%	3626%
34	100	50%	no	600	35	0%	-8%	-9%	179%	-1%	886%	-10%	2440%
35	100	50%	no	600	35	0%	339%	1%	90%	-3%	503%	-3%	4931%
36	100	50%	yes	180	35	0%	2%	-10%	349%	0%	165%	-10%	1115%
37	100	50%	yes	180	35	0%	-11%	-4%	101%	-6%	841%	-9%	1577%
38	100	50%	yes	180	35	0%	-6%	-9%	695%	0%	139%	-9%	1685%
39	100	50%	yes	180	35	0%	-7%	-7%	940%	-4%	284%	-11%	3623%
40	100	50%	yes	180	35	0%	-2%	-4%	28%	-5%	537%	-9%	704%
Min						-14%	-30%	-20%	28%	-11%	139%	-22%	704%
Avg.						-3%	105%	-7%	223%	-3%	493%	-12%	3081%
Max						0%	471%	2%	940%	1%	886%	-1%	8359%

Table 3.2: Comparison for different LH frequencies (between 30 and 300 minutes) with the PI method. The table includes the number of requests (R), the amount of cross-city requests (R^{cc}), whether the station and depot have identical locations (S=D), the time window size (TW), the number of SH vehicles (V^c), the SH cost (SH), and the CPU time difference (T). The table shows the increases in SH cost reduction and CPU time as a percentage when the LH departure frequency increases from 300 to 150 minutes, 150 to 75 minutes, 75 to 30 minutes, and 300 to 30 minutes.

						LH150 rel. to LH300		LH75 rel. to LH150		LH30 rel. to LH75		**LH30 rel. to LH300**	
	R	R^{cc}	S = D	TW	V^c	SH	T	SH	T	SH	T	SH	T
1	10	0%	no	600	4	-10%	173%	-10%	152%	0%	335%	-19%	2890%
2	10	0%	no	600	4	-12%	62%	-9%	246%	-1%	546%	-21%	3514%
3	10	0%	no	600	4	-11%	198%	0%	141%	0%	593%	-11%	4883%
4	10	50%	no	600	4	-1%	156%	-12%	224%	0%	418%	-13%	4187%
5	10	50%	no	600	4	-4%	152%	-5%	232%	0%	390%	-8%	4007%
6	10	0%	no	180	4	0%	47%	-9%	272%	0%	260%	-9%	1868%
7	10	0%	no	180	4	0%	33%	0%	336%	-1%	556%	-1%	3712%
8	10	30%	no	180	4	0%	-4%	-14%	336%	-4%	549%	-17%	2624%
9	10	50%	no	180	4	0%	-26%	0%	297%	-5%	778%	-5%	2462%
10	10	50%	no	180	4	0%	-30%	-1%	376%	-4%	610%	-5%	2259%
11	20	30%	no	600	7	-3%	49%	-10%	222%	0%	504%	-13%	2805%
12	20	30%	no	600	7	0%	89%	0%	233%	-12%	489%	-12%	3601%
13	20	50%	no	600	7	-9%	263%	-1%	162%	-4%	519%	-14%	5789%
14	20	50%	no	600	7	-7%	207%	-5%	139%	-1%	432%	-13%	3807%
15	20	50%	no	600	7	0%	74%	0%	207%	-2%	706%	-2%	4200%
16	20	0%	yes	180	7	-1%	63%	-8%	193%	0%	382%	-9%	2203%
17	20	10%	yes	180	7	0%	76%	-3%	230%	-1%	306%	-4%	2255%
18	20	20%	yes	180	7	0%	-18%	-5%	382%	-3%	655%	-9%	2888%
19	20	30%	no	180	7	0%	52%	-6%	277%	-5%	411%	-10%	2826%
20	20	50%	no	180	7	-4%	66%	-18%	252%	-1%	507%	-22%	3436%
21	50	30%	no	600	18	-1%	355%	-16%	152%	-5%	275%	-20%	4206%
22	50	30%	no	600	18	-8%	324%	0%	153%	-7%	519%	-14%	6539%
23	50	50%	no	600	18	-16%	277%	1%	154%	-3%	334%	-18%	4057%
24	50	50%	no	600	18	0%	95%	-9%	220%	-1%	579%	-10%	4134%
25	50	50%	no	600	18	-7%	270%	-5%	154%	-2%	332%	-13%	3955%
26	50	0%	no	180	18	0%	-8%	-15%	391%	-6%	407%	-20%	2179%
27	50	0%	no	180	18	0%	-9%	-19%	312%	-3%	292%	-21%	1364%
28	50	10%	no	180	18	0%	-14%	-9%	390%	-4%	284%	-12%	1507%
29	50	10%	no	180	18	0%	-10%	0%	354%	-6%	252%	-7%	1342%
30	50	30%	yes	180	18	0%	-11%	-13%	464%	1%	176%	-12%	1280%
31	100	0%	yes	600	35	0%	111%	0%	208%	-1%	376%	-1%	2989%
32	100	0%	yes	600	35	2%	86%	-6%	256%	-3%	403%	-7%	3225%
33	100	30%	no	600	35	-3%	116%	-3%	196%	-4%	601%	-10%	4391%
34	100	50%	no	600	35	0%	-5%	-9%	359%	-3%	700%	-12%	3395%
35	100	50%	no	600	35	-4%	349%	-4%	155%	-2%	465%	-9%	6359%
36	100	50%	yes	180	35	0%	-12%	-11%	373%	-3%	202%	-14%	1161%
37	100	50%	yes	180	35	0%	-11%	-6%	157%	-8%	573%	-14%	1437%
38	100	50%	yes	180	35	0%	-11%	-5%	393%	-5%	125%	-9%	884%
39	100	50%	yes	180	35	0%	-12%	-3%	426%	-2%	309%	-5%	1792%
40	100	50%	yes	180	35	0%	-12%	1%	118%	-10%	508%	-8%	1062%
Min						-16%	-30%	-19%	118%	-12%	125%	-22%	884%
Avg.						-2%	89%	-6%	255%	-3%	447%	-11%	3102%
Max						2%	355%	1%	426%	0%	778%	-1%	6539%

Table 3.3: Comparison for different LH frequencies (between 30 and 300 minutes) with the EPI method. The table includes the number of requests (R), the amount of cross-city requests (R^{cc}), whether the station and depot have identical locations (S=D), the time window size (TW), the number of SH vehicles (V^c), the SH cost (SH), and the CPU time difference (T). The table shows the increases in SH cost reduction and CPU time as a percentage when the LH departure frequency increases from 300 to 150 minutes, 150 to 75 minutes, 75 to 30 minutes, and 300 to 30 minutes.

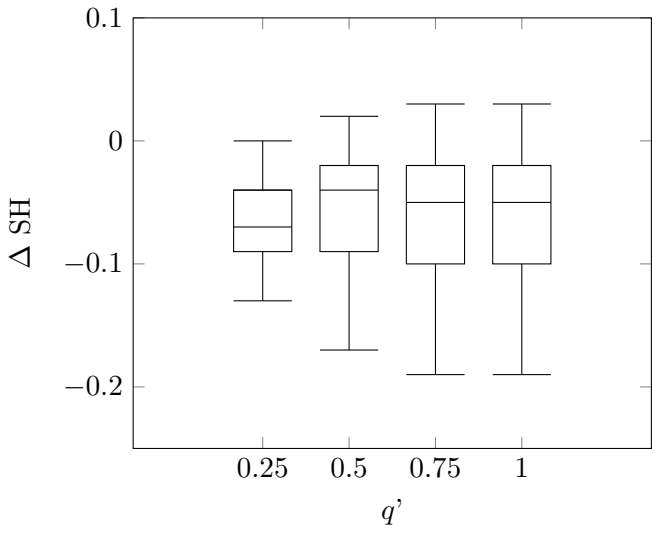


Figure 3.9: SH improvement with different LH capacity constraints for various LH frequency comparisons. The y-axis shows SH improvements up to 19 percent (Δ SH), the x-axis indicates the available LH capacity as a percentage of the total demand (q'), ranging from 25% to 100%.

3.4.3 Evaluation of the algorithm in comparison with Ghilas et al. [64]

We compare our algorithm with the branch-and-price algorithm of Ghilas et al. [64], developed for a PDP with time windows and scheduled lines (PDPTW-SL). Not all instances are suitable for comparison though. We eliminate instances from the computational experiments if the same solution could not be reached by our algorithm, because our constraints do not allow it. The PDPTW-SL operates in one big region, with two depots, multiple stations, and multiple scheduled lines or long-hauls. In our problem, a SH vehicle may not leave the region, whereas in some solutions by Ghilas et al. [64], a SH vehicle visits both the station of its own region and the station of the opposite region. For the comparison, we include only those instances whose solutions are valid by our definition. Also, to enable the comparisons, we split the nodes and assigned them to two regions, each with one depot and one station, by assigning the same depot as in the solution by Ghilas et al. We also adapted the objective function to be identical with that from Ghilas et al., where SH vehicle driving time (equal to distance) is multiplied by 0.5, and the LH cost is 1 per unit shipped, regardless of LH distance. Table 3.4 compares the objective value obtained from our algorithms and the results provided by Ghilas et al. [64]. For all instances, we include both PI and EPI and report the better solution. Identical parameters are used for all instances, with an initial LH assignment solution pool of 30, six look-ahead steps, four considered decisions, a waiting time weight of 0.8, and 10^5 iterations without improvement until VND termination. Among the 28 instances, 20 are optimal, as indicated in the table by asterisks. On average, our algorithm deviates by 1% from the optimal solution, with a maximum deviation of 5.9%. Table 3.5 shows the best solution over all runs with different parameters. Here we find 22 optimal solutions. On average, our algorithm is only 0.3% worse than the optimal solution with a maximum deviation of 5.0%. By including different parameter settings, we can improve the mean objective value and we obtain an optimal solution for two additional instances (C12 and RC50). There seem to be no differences in performance according to instance type or customer node distribution. The exact parameters used for each instance can be found in Table 3.10 in 3.6.4. Table 3.6 shows the run times for all instances in seconds for the results obtained by Table 3.5. On average, our algorithm is about four times faster, mainly due to the large

instances that take exponentially longer with an exact solution approach. Both algorithms are implemented in Java. Ghilas et al. use four threads on a 2.6 GHz Intel Core i5 processor with 4 GB of RAM; whereas our computational experiments were performed with a 3.60 GHz Intel Core i7 processor with 8 GB of RAM. Comparing the single thread ratings of both processors, we find ours to be about 30 percent more powerful and have therefore provided both the actual achieved computational times as well as projected times that account for the different CPU benchmarks.

3.4.4 Application of the solution algorithm on the inter-library loan network

Our algorithm is applied to realistic instances for the inter-library loan network. The computational results of 20 instances are summarized in Table 3.7 and two different LH options (train and truck) and their influence on the SH costs are compared. The table provides information about the number of requests R, the share of cross-city loans R^{cc}, the SH costs 'SH' and number of SH vehicles '#SH' of both the train and truck LH option, and the change in SH costs 'Δ SH'. The train departs hourly according to the timetable of the Austrian Federal Railway while the truck offers more flexibility by being able to depart every 15 minutes. Identical parameters are used for all instances, namely the same as the ones used to obtain Table 3.2: PI method with an initial LH assignment solution pool of 30, five look-ahead steps n, three decisions m, a waiting time weight φ_w of 0.3, and 10^4 iterations without improvement until VND termination.
On average, the SH costs reduce by 2.1% when choosing truck over train, while the required number of SH vehicles increases slightly. However, the changes in costs are very volatile. They range from a 13.06% reduction to a 10.70% increase. This distribution of costs is independent of the instance size or the share of cross-city requests, however they to seem to correlate with the increase in SH vehicles. Figure 3.12 shows two exemplary SH solutions in Graz of instance 1 of the inter-library loan system using a LH via train or truck. The different schedule results in a different SH routing when switching from a LH via train to truck.
With both LH options, the absolute number of used LH vehicles was minimized in the first step of the solution algorithm with the aim to use the least amount of connections possible. We made the assumption that minimizing LH vehicles would sufficiently minimize LH costs since all LH connections are provided by a third party and LH costs consists only of a factor of the quantity shipped (only variable costs, no fixed costs). This assumption is suitable for modelling transport via train, but is probably inapt for transport via truck. The additional fixed costs of using trucks as LH vehicles and the slight increase in required SH vehicles, makes an average of 2.1% SH costs decrease seemingly irrelevant. As previous results on the artificial instances indicated, it seems that also on the real-world based instances a careful decision when to schedule LH departures is of more importance than simply offering more LH connections.

3.4.5 Analysis of the algorithm and parameter settings

Influence of the waiting time weight, number of decisions, and look-ahead steps

The weight of the waiting time influences evaluations of an insertion. The weights range from 0 to 0.9 in 0.1 increments and are set in relation to the distance weight, which is always 1. A weight of 0 implies no penalty for waiting times, whereas higher weights result in higher penalties. The waiting time penalty is added to the distance cost, and the insertion list is sorted by this combined cost, so the weights influence the order of the sorted list and, by extension, determine if an insertion will be considered at a specific point in the construction process.

The number of considered insertion decisions m and the number of look-ahead steps n also could have a considerable influence on the quality of the solutions. To find the combination of parameters that leads to the best solutions, we solved an instance several times with different

Instance	OV-G	Ghilas	Δ
C10*	510.01	510.01	0.0%
C11*	522.85	522.85	0.0%
C12	549.80	541.60	1.5%
C20	778.82	751.23	3.7%
C25	922.48	879.94	4.8%
R6*	416.16	416.16	0.0%
R7*	473.06	473.06	0.0%
R8*	558.17	558.17	0.0%
R9*	632.42	632.42	0.0%
R10*	636.05	636.05	0.0%
R11	780.69	748.29	4.3%
R12*	913.68	913.68	0.0%
R15*	1083.50	1083.50	0.0%
R20*	1542.93	1542.93	0.0%
R25*	1636.12	1636.12	0.0%
R40	2085.31	1969.77	5.9%
RC6*	572.76	572.76	0.0%
RC7*	575.95	575.95	0.0%
RC8*	585.32	585.32	0.0%
RC9*	593.70	593.70	0.0%
RC10*	599.95	599.95	0.0%
RC11*	624.48	624.48	0.0%
RC12*	662.03	662.03	0.0%
RC15*	1068.16	1068.16	0.0%
RC20*	1200.36	1200.36	0.0%
RC25	1533.89	1530.43	0.2%
RC40	2020.98	1969.45	2.6%
RC50	2327.10	2200.88	5.7%
		Mean	1.0%

Table 3.4: Comparison of objective value (OV-G) to Ghilas et al. [64] (Ghilas) and the difference (Δ), best run. Identical parameters are used on all instances. 20 of 28 instances are optimal, marked with '*'.

Instance	OV-G	Ghilas	Δ
C10*	510.01	510.01	0.0%
C11*	522.85	522.85	0.0%
C12*	541.60	541.60	0.0%
C20	751.30	751.23	0.0%
C25	888.34	879.94	1.0%
R6*	416.16	416.16	0.0%
R7*	473.06	473.06	0.0%
R8*	558.17	558.17	0.0%
R9*	632.42	632.42	0.0%
R10*	636.05	636.05	0.0%
R11	754.70	748.29	0.9%
R12*	913.68	913.68	0.0%
R15*	1083.50	1083.50	0.0%
R20*	1542.93	1542.93	0.0%
R25*	1636.12	1636.12	0.0%
R40	2067.75	1969.77	5.0%
RC6*	572.76	572.76	0.0%
RC7*	575.95	575.95	0.0%
RC8*	585.32	585.32	0.0%
RC9*	593.70	593.70	0.0%
RC10*	599.95	599.95	0.0%
RC11*	624.48	624.48	0.0%
RC12*	662.03	662.03	0.0%
RC15*	1068.16	1068.16	0.0%
RC20*	1200.36	1200.36	0.0%
RC25	1531.95	1530.43	0.1%
RC40	1978.65	1969.45	0.5%
RC50*	2200.88	2200.88	0.0%
		Mean	0.3%

Table 3.5: Comparison of objective value (OV-G) to Ghilas et al. [64] (Ghilas) and the difference (Δ), best solutions found over all runs with different parameters. 22 of 28 instances are optimal, marked with '*'.

Instance	CPU	i5-CPU	Ghilas
C10	51	66	67
C11	67	87	9
C12	70	91	4
C20	7	10	83
C25	144	187	2114
R6	2	2	0
R7	2	2	1
R8	2	2	0
R9	43	55	2
R10	46	59	1
R11	35	45	1
R12	72	94	2
R15	2	2	15
R20	1	2	12
R25	52	68	50
R40	3	4	895
RC6	3	4	0
RC7	4	5	0
RC8	4	5	0
RC9	22	28	0
RC10	24	32	0
RC11	29	38	1
RC12	22	28	1
RC15	2	2	1
RC20	65	84	6
RC25	69	90	28
RC40	20	26	738
RC50	145	189	1235
Mean	36.0	46.7	188.1

Table 3.6: Run time comparison with Ghilas et al. [64], in seconds, showing the actual achieved times (CPU) and projected times (i5-CPU) accounting for the different CPU benchmarks.

			Train		Truck		
	R	R^{cc}	SH	#SH	SH	#SH	Δ SH
1	25	20%	209.40	5	194.59	8	-7.07%
2	25	20%	220.07	5	219.10	8	-0.44%
3	25	20%	197.99	7	199.85	7	0.94%
4	25	20%	234.39	4	237.32	6	1.25%
5	25	20%	210.63	6	223.88	4	6.29%
6	30	33%	240.38	6	225.22	6	-6.31%
7	30	33%	245.32	6	234.40	4	-4.45%
8	30	33%	263.15	7	251.87	6	-4.29%
9	30	33%	235.54	6	231.90	6	-1.55%
10	30	33%	237.02	6	242.45	6	2.29%
11	35	14%	241.08	8	209.59	8	-13.06%
12	35	14%	290.86	9	263.59	5	-9.38%
13	35	14%	287.82	8	268.82	8	-6.60%
14	35	14%	264.18	6	265.59	6	0.53%
15	35	14%	246.23	5	272.58	9	10.70%
16	40	25%	274.75	10	243.91	7	-11.22%
17	40	25%	288.96	8	275.10	9	-4.80%
18	40	25%	252.93	8	244.18	8	-3.46%
19	40	25%	272.55	6	274.02	7	0.54%
20	40	25%	279.58	4	299.60	7	7.16%
Mean				6.5		6.8	-2.1%

Table 3.7: Comparison of different LH options for the inter-library loan network between Vienna and Graz. The table includes the number of requests (R), the share of cross-city requests (R^{cc}), the SH cost when using the train or truck (SH), the total number of SH vehicles used in both cities (#SH), and the difference in objective value (Δ SH) when comparing the SH costs.

parameter combinations. In general, having very few decisions or looking-ahead steps leads to inferior solutions. However, when these values grow larger, the computation time lengthens, depending on the number of partial solutions that need to be evaluated. The PI method always evaluates m solutions, no matter how many n steps are looked-ahead (though each additional look-ahead step takes time), so the number of solutions to evaluate depends solely on the size of the instance (number of requests r), rm. The EPI evaluates more solutions, depending on both the number of decisions and look-ahead steps, rm^n. The values are therefore restricted for our computational experiments to $3 \leqslant m \leqslant 6$ decisions and $1 \leqslant n \leqslant 8$ look-ahead steps.

A good tuning of the parameters is crucial to finding good or optimal solutions. The settings depend on the instance. We evaluate two instances from Ghilas et al. [64] in detail, namely, R15 and R20. The plots are in 3.6.5 and represent the objective value obtained with different parameter combinations of insertion decisions n, look-ahead steps m, and waiting time weight φ_w. In general, overly small values of $n < 3$ lead to worse or even infeasible solutions. Values of $n = 5$ and $n = 6$ seem promising and do not demand too much computational time. However, values of $n = 7$ or $n = 8$ can deteriorate solution quality or stay promising, depending on the instance. Additionally, different values of insertion decisions are tested. At least $m = 3$ insertion

decisions are always included. Any increase in decisions included in the selection led almost as often to a decline in solution quality as it led to an improvement.

A comparison of the pilot method (PI) and the extended pilot method (EPI)

The EPI method performs slightly better than the PI method on average, but it also is much more computationally expensive. The PI evaluates rm solutions with r requests and m insertion decisions; the EPI evaluates rm^n with n look-ahead steps. Regarding the trade-off between computation time and solution quality, the difference in both also depends on the application of the improvement heuristic, which can increase both time and solution quality.
Table 3.8 details the performance of both PI and EPI for our own instances with LH75 (LH departure every 75 minutes), with and without VND. The relation is calculated using $\frac{PI}{EPI} - 1$. Without VND, the PI method is on average 2.5% worse (i.e., has 2.5% higher costs) than the EPI method. The results range from being 6.1% better to 14.3% worse. When including VND, the difference between the two algorithms diminishes, and PI is only 0.4% worse on average, with a range of 8.8% better to 12.1% worse. We do not include LH costs in the table, because they are identical for both methods. We find no significant differences in PI and EPI performance with regard to instance size, the share of cross-city requests, the location of the station, or the size of the time windows.

Why the PI method can be better than the EPI method

Although it may seem counterintuitive, the EPI does not always result in a better solution than the PI for the same number of considered decisions and look-ahead steps.

The EPI goes beyond the m decisions in the first step to consider all m possibilities for each of the n look-ahead steps. Based on this much more extended decision tree, the decision for the first step is made. The solutions evaluated by the PI thus are by definition a subset of the solutions evaluated by the EPI.

To understand how the EPI might fail though, we offer an example: Assume that both algorithms perform identically until a specific point at which the evaluations of the insertions diverge, due to a potential solution that could be superior in objective value. On the path considered by the EPI, the expected partial solution has a lower estimated cost (i.e., smaller distance while fulfilling the same number of requests). However, if not all requests have been included, such that one request is still missing in each partial solution, looking ahead just one more step would reveal that the decision path of the EPI was misguided, and the last request cannot be feasibly inserted. When the algorithm finishes, it becomes evident that the solution achieved with the PI is superior to that from the EPI, because the EPI was misled down a seemingly good solution path that would not have been feasible if followed.

Both methods might yield infeasible solutions that do not fulfil all requests. A change in the look-ahead steps also exerts a great impact on the solutions obtained (see Figure 3.14 in 3.6.5). Adding a single look-ahead step to our example would have prevented the EPI from selecting that inappropriate path. Still, from our computational results, it is clear that both methods should be used if time allows.

Some remarks on computational time

Table 3.9 gives an overview of the average running time for an instance in seconds. In general:

- Computation time increases for larger instances and larger time windows.
- A tighter LH departure frequency increases running time significantly, because more short-haul sub-problems must be solved (also see Tables 3.2 and 3.3).
- EPI is much more computationally expensive then PI (see also Table 3.8)

						LH75, only (E)PI			LH75, (E)PI & VND		
	R	R^{cc}	S=D	TW	V^c	OV	SH	T	OV	SH	T
1	10	0%	no	600	4	-0.5%	-0.8%	-95.9%	-0.5%	-0.8%	-57.5%
2	10	0%	no	600	4	0.6%	0.7%	-97.4%	-0.9%	-1.2%	-52.7%
3	10	0%	no	600	4	0.9%	1.3%	-97.3%	0.9%	1.3%	-52.0%
4	10	50%	no	600	4	-0.6%	-0.7%	-94.9%	-0.6%	-0.7%	-42.5%
5	10	50%	no	600	4	1.7%	2.3%	-97.6%	1.7%	2.3%	-45.6%
6	10	0%	no	180	4	0.5%	0.6%	-96.2%	0.5%	0.6%	-52.4%
7	10	0%	no	180	4	1.2%	1.5%	-97.6%	0.0%	0.0%	-55.7%
8	10	30%	no	180	4	4.5%	6.3%	-95.5%	0.7%	0.9%	-39.9%
9	10	50%	no	180	4	1.2%	1.4%	-95.3%	0.7%	0.9%	-44.9%
10	10	50%	no	180	4	0.0%	0.0%	-94.3%	0.0%	0.0%	-41.4%
11	20	30%	no	600	7	-0.7%	-0.8%	-98.5%	2.3%	2.9%	-79.8%
12	20	30%	no	600	7	-1.2%	-1.5%	-98.5%	3.0%	3.7%	-75.5%
13	20	50%	no	600	7	5.6%	7.2%	-98.3%	2.3%	2.9%	-75.1%
14	20	50%	no	600	7	3.3%	4.1%	-98.5%	5.6%	7.1%	-76.3%
15	20	50%	no	600	7	0.2%	0.3%	-98.2%	-0.7%	-0.9%	-74.8%
16	20	0%	yes	180	7	2.8%	3.4%	-98.4%	0.4%	0.5%	-72.2%
17	20	10%	yes	180	7	1.5%	1.8%	-98.3%	-3.5%	-4.2%	-71.5%
18	20	20%	yes	180	7	4.5%	5.3%	-97.6%	5.3%	6.3%	-70.5%
19	20	30%	no	180	7	2.3%	2.7%	-98.0%	-1.6%	-2.0%	-74.9%
20	20	50%	no	180	7	6.9%	8.1%	-98.4%	0.1%	0.1%	-71.1%
21	50	30%	no	600	18	1.4%	1.5%	-98.5%	0.4%	0.5%	-88.7%
22	50	30%	no	600	18	2.9%	3.3%	-98.5%	-3.7%	-4.3%	-89.9%
23	50	50%	no	600	18	1.2%	1.3%	-98.7%	0.3%	0.3%	-88.2%
24	50	50%	no	600	18	2.8%	3.1%	-98.4%	5.5%	6.2%	-88.3%
25	50	50%	no	600	18	2.0%	2.3%	-98.6%	0.5%	0.6%	-89.2%
26	50	0%	no	180	18	-6.1%	-6.8%	-98.6%	-8.8%	-9.9%	-84.6%
27	50	0%	no	180	18	2.7%	2.9%	-98.6%	-0.6%	-0.6%	-83.6%
28	50	10%	no	180	18	14.3%	15.4%	-98.7%	-4.2%	-4.6%	-90.4%
29	50	10%	no	180	18	-0.2%	-0.2%	-98.7%	1.3%	1.5%	-90.7%
30	50	30%	yes	180	18	1.7%	1.8%	-98.2%	12.1%	13.4%	-83.6%
31	100	0%	yes	600	35	5.4%	5.7%	-98.7%	2.3%	2.5%	-84.2%
32	100	0%	yes	600	35	8.8%	9.3%	-98.8%	0.1%	0.1%	-82.4%
33	100	30%	no	600	35	4.8%	5.1%	-98.6%	-1.2%	-1.3%	-92.3%
34	100	50%	no	600	35	-2.7%	-2.9%	-98.6%	-1.1%	-1.2%	-92.7%
35	100	50%	no	600	35	-3.7%	-3.9%	-98.4%	0.7%	0.8%	-91.5%
36	100	50%	yes	180	35	7.3%	7.9%	-98.8%	-2.4%	-2.6%	-90.6%
37	100	50%	yes	180	35	4.1%	4.4%	-98.6%	-1.2%	-1.3%	-91.3%
38	100	50%	yes	180	35	4.5%	4.7%	-98.5%	-1.4%	-1.5%	-90.1%
39	100	50%	yes	180	35	4.8%	5.1%	-98.6%	1.9%	2.1%	-86.7%
40	100	50%	yes	180	35	9.3%	9.8%	-98.7%	-1.9%	-2.0%	-90.8%
Min						-6.1%	-6.8%	-98.8%	-8.8%	-9.9%	-92.7%
Avg						2.5%	2.8%	-97.9%	0.4%	0.5%	-74.9%
Max						14.3%	15.4%	-94.3%	12.1%	13.4%	-39.9%

Table 3.8: PI in relation to EPI performance for a LH departure frequency of 75 minutes (LH75) with and without VND. The table includes the number of requests (R), the amount of cross-city requests (R^{cc}), whether the station and depot have identical locations (S=D), the time window size (TW), the number of SH vehicles (V^c), the total objective value (OV), SH cost (SH), and computational time (T). The PI method is 2.5% worse on average (i.e. has 2.5% higher cost) than the EPI method without VND and 0.4% worse than the EPI method with VND.

	PI				EPI			
	LH300	LH150	LH75	LH30	LH300	LH150	LH75	LH30
R10	0.37	0.72	2.00	11.59	0.68	1.22	3.98	22.89
R10 No TW	0.35	0.97	2.47	13.41	0.68	1.72	4.94	27.30
R10 TW	0.38	0.47	1.52	9.78	0.67	0.73	3.03	18.47
R20	1.02	2.01	5.19	31.95	3.50	6.54	20.25	115.63
R20 No TW	0.90	2.30	5.18	35.24	3.43	7.65	21.91	135.49
R20 TW	1.11	1.68	5.02	29.51	3.49	5.42	18.23	101.87
R50	5.88	13.68	33.40	173.47	42.98	93.33	286.05	1317.42
R50 No TW	5.03	20.50	41.15	238.42	38.73	144.50	374.26	1871.03
R50 TW	6.61	9.84	27.48	119.58	45.39	57.41	221.47	880.84
R100	32.12	59.68	163.30	909.99	225.03	413.10	1331.50	7193.12
R100 No TW	53.07	108.61	283.97	1651.47	330.05	720.34	2249.22	12797.38
R100 TW	12.98	25.09	66.21	325.31	131.61	229.98	706.33	3365.52

Table 3.9: Average computational times in seconds by instance size, type, and LH departure frequency

- The smallest instances need an average of 0.37 seconds to be solved. The largest computational times for instances with 100 requests, no time windows, EPI, and a high LH departure frequency (LH30) need approximately 3.5 hours.

3.5 Conclusions

We have proposed a matheuristic algorithm to solve the PDPLH in two regions by decomposing the problem into LH assignment, SH routing, and SH improvement. The LH assignment is solved exactly and then provides input for the SH routing. To demonstrate the quality of the solution algorithm, we compared it with the branch-and-price algorithm described by Ghilas et al. [64] for a related problem. We managed to obtain optimal solutions in 22 of 28 instances with an average difference of 0.3% in reasonable computation times. We also provide the results for self-generated instances. Both the achieved solution quality and computational times make the algorithm suitable as a method for evaluating bundles of transportation requests in an auction mechanism for a collaborative setting. Such a setting requires solving problems of the PDPLH type to evaluate potential additional costs or consolidation possibilities when transportation carriers exchange requests among themselves.

The numerical experiments involving both instances from prior literature and our own analyses the need to fine-tune the parameters and provide interesting insights into the mechanics of the pilot insertion method (PI) and the extended pilot insertion method (EPI). The comparison indicates that the EPI can be superior but is not always. Single instances can be improved up to 14%, yet the overall benefits, after taking the high computation times into account, remain uncertain.

To study the effect of a more flexible LH connection, we limit the number of available LH and gradually expand availability, then check the effects on SH costs. Improvements in SH costs of up to 22% can be observed, merely due to additional LH departure times without additional LH costs. On average, more LH connections result in a 11% (for the PI) or 12% (for the EPI) cost decrease. This insight is especially relevant for decision makers and illustrates the importance of synchronization between transportation modes, as well as the need to remain flexible with

respect to predefined schedules. When simple measures, such as offering additional departure times, results in potential cost savings of that magnitude, decision makers must balance potential SH savings against the cost of flexibility, which is intangible and harder to compute.

Future possible extensions of the problem might be the increase the number of regions or cities, as well as additional modes of (LH) transportation. Small steps in this direction have already been made with a basic comparison of train or trucks as LH vehicles for realistic instances of the inter-library loan network. However, more sophisticated cost functions, that depict the real world more accurately, are necessary. Additional constraints limiting the storage space at the transshipment point, and potential holding costs also might be of interest. Finally, an extension to a multiple-period problem and the inclusion of dynamic or stochastic components would be of interest.

3.6 Appendix

3.6.1 Mathematical problem formulation

Since our problem is based on two cities, they have been named city 'A' and city 'B'. c will be used as a place-holder for either 'A' or 'B'. The same principle is applied for indicating if a set, variable or parameter belongs to a pickup 'P' or a delivery 'D'. In that case π will be used as a place-holder. Also note, that the notation for this mathematical problem formulation is independent of the initial LH assignment in Section 3.3.1.

Sets

V^c ... set of SH vehicles in each city c
V^l ... set of LH vehicles
R ... set of all requests
R^{lc} ... set of requests travelling over the LH from city c
R^{cc} ... set of requests travelling from/to/within city c
N^c ... set of nodes in each city c
$N_0^c = N^c \cup \{0\}$, set of nodes including the depot
$N_r^{\pi c}$ ... P/D node of each request r in c, a set of size 1
B^c ... set of LH single trips departing from c

Parameters

Q^v ... capacity of SH vehicle v
Q^l ... capacity of LH vehicles
H ... LH travel time per direction
C ... cost of LH transportation per direction
d_{ij}^c ... distance from node i to j per city c
σ_i ... service time of node i
σ_s ... loading time at station s
E ... end of day

$q_i^c \ldots \begin{cases} q_i, \text{positive quantity for pickup nodes } i \text{ in } c \\ -q_i, \text{negative quantity for delivery nodes } i \text{ in } c \end{cases}$

$h_b \ldots$ departure time of LH vehicle b in both cities

Fixed time windows of request

$e_r^{PA}, e_r^{PB}, e_r^{DA}, e_r^{DB} \ldots$ earliest P/D time of request r travelling within city A/B

$u_r^{PA}, u_r^{PB}, u_r^{DA}, u_r^{DB} \ldots$ latest P/D time of request r travelling within city A/B

$e_r^{PAA}, e_r^{DAA}, e_r^{DAB} \ldots$ earliest P/D time of request r travelling from A in A/B

$u_r^{PAA}, u_r^{PAB}, u_r^{DAB} \ldots$ latest P/D time of request r travelling from A in A/B

Requests are present in all sets that apply and are therefore in multiple sets at once. For requests travelling over the LH, the time windows are fixed at the customer nodes and treated as decision variables at the stations. For example, for a request travelling from city A to city B, the pickup TW in city A and delivery TW in city B (at customer nodes) are fixed. Delivery TW in city A and pickup TW in city B (at the stations) are decision variables. For cross-city requests (not travelling over the LH), all of the TWs are fixed.

Decision variables

$$x_{ijv}^c = \begin{cases} 1, \text{if vehicle } v \text{ travels from } i \text{ to } j \text{ in } c, \\ 0, \text{otherwise.} \end{cases}$$

$$z_{rb} = \begin{cases} 1, \text{if request } r \text{ is departing with LH } b, \\ 0, \text{otherwise.} \end{cases}$$

$$g_b^c = \begin{cases} 1, \text{if any request is departing with LH } b \text{ from } c, \\ 0, \text{otherwise.} \end{cases}$$

$t_i^c \ldots$ visit at node i in c

$t_r^{Pc} \ldots$ pickup time of request r in c

$t_r^{Dc} \ldots$ delivery time of request r in c

$S_{iv}^c \ldots$ load after visiting node i in c

$t_{\delta v}^c \ldots$ departure time from the depot in c

$t_{\alpha v}^c \ldots$ arrival time at the depot in c

$t_r^{l\delta} \ldots$ departure time of r with LH vehicle

$t_r^{l\alpha} \ldots$ arrival time of r with LH vehicle

Time window decision variables at the stations

e_r^{PAB}, e_r^{PBA} ... Pickup begin TW of r travelling from A in B (or from B in A)
u_r^{DAA}, u_r^{DBB} ... Delivery end TW of r travelling from A in A (or from B in B)

Objective

The objective is to minimize the total cost (SH A + LH + SH B):

$$\text{Minimize} \sum_{i \in N_0^A} \sum_{j \in N_0^A} \sum_{v \in V^A} d_{ij}^A x_{ijv}^A + \sum_{b \in B^A} C g_b^A + \sum_{b \in B^B} C g_b^B + \sum_{i \in N_0^B} \sum_{j \in N_0^B} \sum_{v \in V^B} d_{ij}^B x_{ijv}^B \quad (3.15)$$

Constraints for the SH

$$\sum_{i \in N_0^c} x_{ijv}^c - \sum_{i \in N_0^c} x_{jiv}^c = 0 \qquad \forall j \neq i, j \in N^c, v \in V^c \quad (3.16)$$

$$x_{iiv}^c = 0 \qquad \forall i \in N^c, v \in V^c \quad (3.17)$$

$$\sum_{i \in N_0^c} x_{i0v}^c = \sum_{j \in N_0^c} x_{0jv}^c = 1 \qquad \forall v \in V^c \quad (3.18)$$

$$\sum_{i \in N_0^c, i \neq j} \sum_{v \in V^C} x_{ijv}^c = 1 \qquad \forall j \in N^c \quad (3.19)$$

$$t_i^c + \sigma_i + d_{ij}^c \leqslant t_j^c + M(1 - x_{ijv}^c) \qquad \forall i, j \in N^c, v \in V^c \quad (3.20)$$

$$0 \leqslant t_i^c \leqslant E \qquad \forall i \in N^c \quad (3.21)$$

$$t_r^{\pi c} = t_i^c \qquad \forall r \in R, i \in N_r^{\pi c} \quad (3.22)$$

$$t_i^c \leqslant t_j^c \qquad \forall i \in N_r^{Pc}, j \in N_r^{Dc}, r \in R \quad (3.23)$$

$$e_r^{\pi c} \leqslant t_r^{\pi c} \leqslant u_r^{\pi c} \qquad \forall r \in R^{cc} \quad (3.24)$$

$$e_r^{\pi Ac} \leqslant t_r^{\pi c} \leqslant u_r^{\pi Ac} \qquad \forall r \in R^{AB} \quad (3.25)$$

$$e_r^{\pi Bc} \leqslant t_r^{\pi c} \leqslant u_r^{\pi Bc} \qquad \forall r \in R^{BA} \quad (3.26)$$

$$S_{jv}^c \leqslant S_{iv}^c + q_i^c + M(1 - x_{ijv}^c) \qquad \forall i, j \in N_0^c, v \in V^c \quad (3.27)$$

$$S_{jv}^c \geqslant S_{iv}^c + q_i^c - M(1 - x_{ijv}^c) \qquad \forall i, j \in N_0^c, v \in V^c \quad (3.28)$$

$$0 \leqslant S_{iv}^c \leqslant Q^v \qquad \forall i \in N^c, v \in V^c \quad (3.29)$$

$$S_{0v}^c = 0 \qquad \forall v \in V^c \quad (3.30)$$

$$t_{\delta v}^c + d_{0i}^c \leqslant t_i^c + M(1 - x_{0iv}^c) \qquad \forall i \in N^c, v \in V^c \quad (3.31)$$

$$t_i^c + \sigma_i + d_{i0}^c \leqslant t_{\alpha v}^c + M(1 - x_{i0v}^c) \qquad \forall i \in N^c, v \in V^c \quad (3.32)$$

$$0 \leqslant t_{\delta v}^c \leqslant t_{\alpha v}^c \leqslant E \qquad \forall v \in V^c \quad (3.33)$$

$$t_{\delta v}^c \leqslant M(1 - x_{00v}^c) \qquad \forall v \in V^c \quad (3.34)$$

$$t_{\delta v}^c \geqslant -M(1 - x_{00v}^c) \qquad \forall v \in V^c \quad (3.35)$$

$$t_{\alpha v}^c \leqslant M(1 - x_{00v}^c) \qquad \forall v \in V^c \quad (3.36)$$

$$t_{\alpha v}^c \geqslant -M(1 - x_{00v}^c) \qquad \forall v \in V^c \quad (3.37)$$

Constraints (4.3) state that if vehicle v visits a node in a tour, it has to leave the node again. Constraints (4.4) state that travelling between the same node i is forbidden (except the depot),

because that would indicate an empty tour. Constraints (4.5) state that each SH tour originates and terminates at the depot, in both cities. Constraints (4.6) makes sure that each node is visited exactly once. Constraints (4.7) state that if we travel from i to j in vehicle v in city c, the fulfilment times at the nodes must be consecutive. Therefore, the fulfilment time t_j^c at node j in city c has to be greater than or equal to the sum of the fulfilment time t_i^c at node i in city c, service time σ_i, and distance d_{ij}^c between the nodes. These constraints also eliminate subtours, because all tours begin and end at the depot. Constraints (3.21) state that the fulfilment time of requests have to be positive and smaller than the end of the time span E. Constraints (4.12) and (4.14) state that the P&D times $t_r^{\pi c}$ of request r in city c have to be set to the fulfilment times t_i^c at node i, where i is part of the set $N_r^{\pi c}$ containing exactly one P/D node of request r in c, and the pickup must be completed before the delivery is possible. Constraints (4.8) to (3.26) restrict the fulfilment times at the nodes to the time windows. Constraints (4.15) to (4.18) are the loading restrictions for the SH vehicles. If we travel from i to j with vehicle v, the load S_{jv}^c before serving node j equals the load before servicing node i plus the quantity q_i^c of node i. This quantity can be negative if a delivery occurs. Constraints (4.19) and (4.20) state that all arrival times $t_{\alpha v}^c$ and departure times $t_{\delta v}^c$ at the depot have to be consistent with travelling times and fulfilment times of the SH vehicles. Constraints (4.21) state that the departure and arrival times at the depot have to be during the time span as well as ensuring that each vehicles first departs and then arrives at the depot. Constraints (4.23) to (4.26) states that if a SH tour is empty, the departure time and arrival time at the depot are 0.

Constraints for the LH

$$\sum_{b\in B} z_{rb} = 1, \qquad \forall r \in R^{lc} \tag{3.38}$$

$$g_b^c = \sum_{r\in R^{lc}} z_{rb} \qquad \forall b \in B^c \tag{3.39}$$

$$\sum_{r\in R^{Lc}} z_{rb} q_i^c \leqslant Q^L \qquad \forall i \in N_r^{Pc}, b \in B^c \tag{3.40}$$

$$t_r^{L\delta} \leqslant h_b + M(1 - z_{rb}) \qquad \forall r \in R^{Lc}, b \in B^c \tag{3.41}$$

$$t_r^{L\delta} \geqslant h_b - M(1 - z_{rb}) \qquad \forall r \in R^{Lc}, b \in B^c \tag{3.42}$$

$$t_r^{L\alpha} \leqslant h_b + H + M(1 - z_{rb}) \qquad \forall r \in R^{Lc}, b \in B^c, \tag{3.43}$$

$$t_r^{L\alpha} \geqslant h_b + H - M(1 - z_{rb}) \qquad \forall r \in R^{Lc}, b \in B^c, \tag{3.44}$$

$$u_r^{DAA} \leqslant t_r^{l\delta} - \sigma_s + M(1 - z_{rb}) \qquad \forall r \in R^{AB}, b \in B^c \tag{3.45}$$

$$u_r^{DAA} \geqslant t_r^{l\delta} - \sigma_s - M(1 - z_{rb}) \qquad \forall r \in R^{AB}, b \in B^c \tag{3.46}$$

$$e_r^{PAB} \leqslant t_r^{l\alpha} + \sigma_s + M(1 - z_{rb}) \qquad \forall r \in R^{AB}, b \in B^c \tag{3.47}$$

$$e_r^{PAB} \geqslant t_r^{l\alpha} + \sigma_s - M(1 - z_{rb}) \qquad \forall r \in R^{AB}, b \in B^c \tag{3.48}$$

$$u_r^{DBB} \leqslant t_r^{l\delta} - \sigma_s + M(1 - z_{rb}) \qquad \forall r \in R^{BA}, b \in B^c \tag{3.49}$$

$$u_r^{DBB} \geqslant t_r^{l\delta} - \sigma_s - M(1 - z_{rb}) \qquad \forall r \in R^{BA}, b \in B^c \tag{3.50}$$

$$e_r^{PBA} \leqslant t_r^{l\alpha} + \sigma_s + M(1 - z_{rb}) \qquad \forall r \in R^{BA}, b \in B^c \tag{3.51}$$

$$e_r^{PBA} \geqslant t_r^{l\alpha} + \sigma_s - M(1 - z_{rb}) \qquad \forall r \in R^{BA}, b \in B^c \tag{3.52}$$

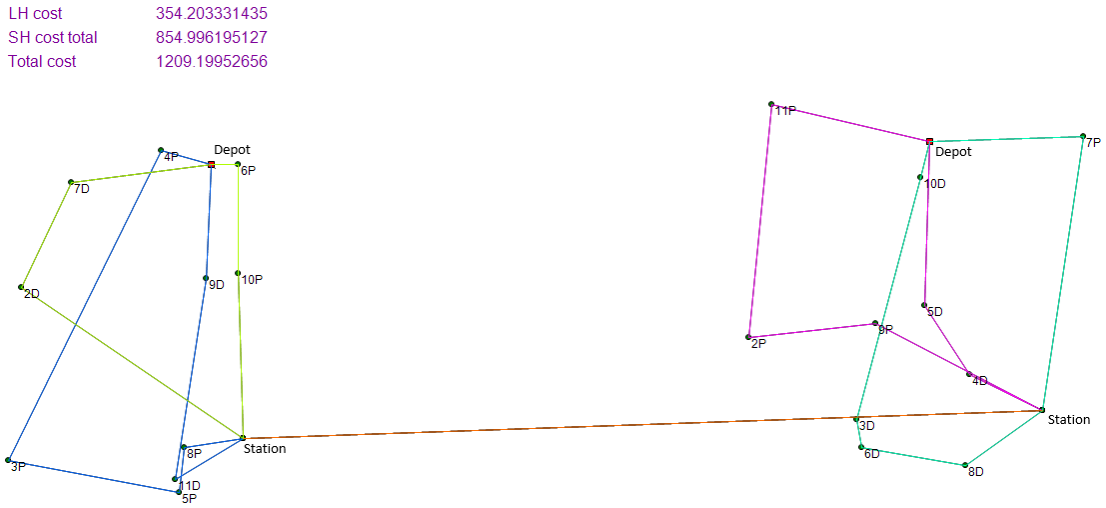
$$z_{r0} = 1 \qquad \forall r \in R \setminus R^{lc} \tag{3.53}$$

$$t_r^{l\delta} = t_r^{l\alpha} = 0 \qquad \forall r \in R \setminus R^{lc} \tag{3.54}$$

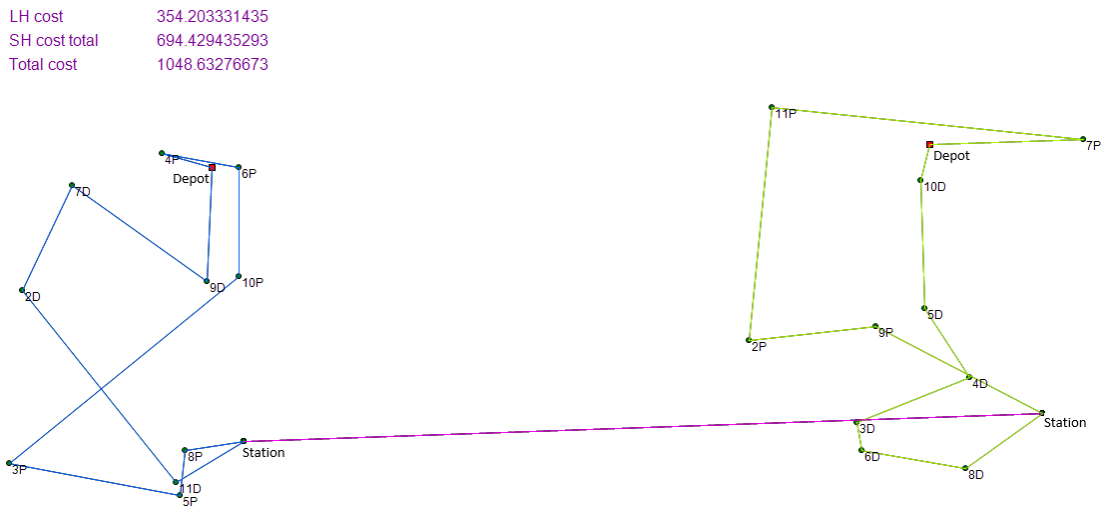
Constraints (3.38) state that all requests that are supposed to travel over the LH have to be scheduled on a LH. Constraints (4.27) determine if a LH is in use and therefore if it generates costs or not. Constraints (3.40) is the capacity constraint for the LH vehicles. Constraints (4.29) to (3.52) state that for each request that departs with the LH vehicle, the departure and arrival times are set. The LH departure and arrival times also correspond to the time windows for pickup and drop-off of requests at the stations (after considering service time σ_s). Constraints (4.31) and (4.32) leave no decision variable undefined for cross-city requests, which do not need to travel over the LH connection.

The constraints for the LH are not the same as the long-haul assignment model in Section 3.3.1; they serve different purposes. The long-haul assignment as part of the solution algorithm is specifically designed to generate solutions in which all requests are transported by the LH, without taking the SH routing into account, whereas above constrains for the LH are part of the complete model.

3.6.2 Visualization of exemplary instances

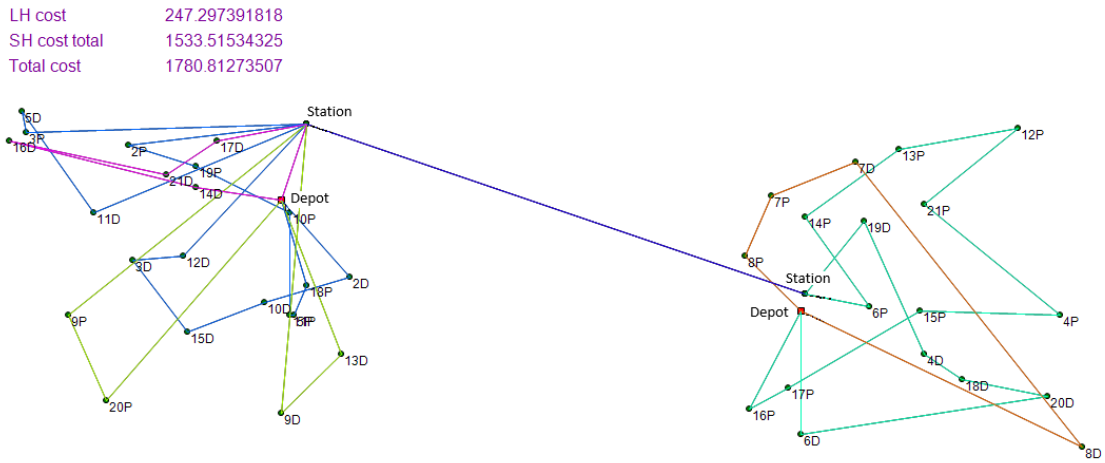


(a)

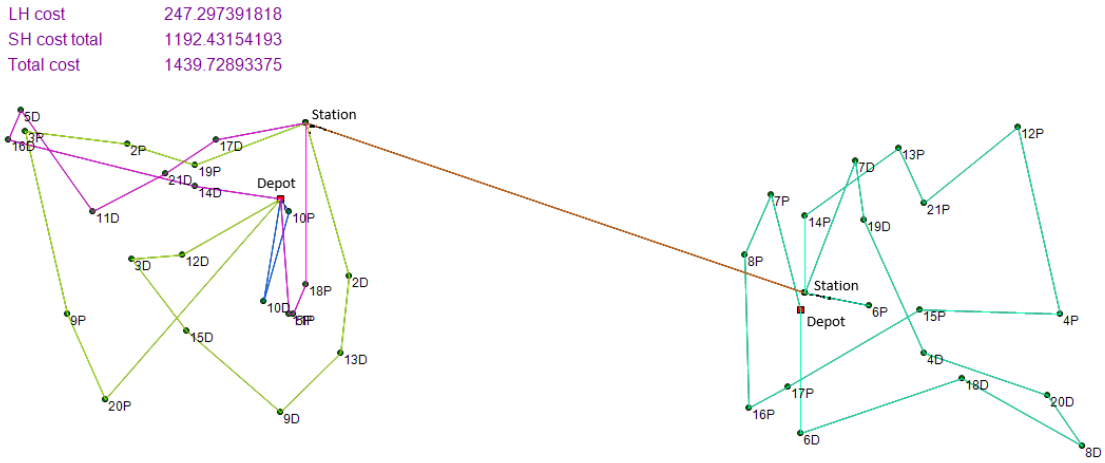


(b)

Figure 3.10: Solution of instance 1 with PI, with (3.10a) LH300 or (3.10b) LH30. Instance 1 has 10 requests, no time windows, and no cross-city requests. Requests are numbered from 1 to 10, each with a pickup node 'P' and a delivery node 'D'. Pickups are always serviced before deliveries of the same request.

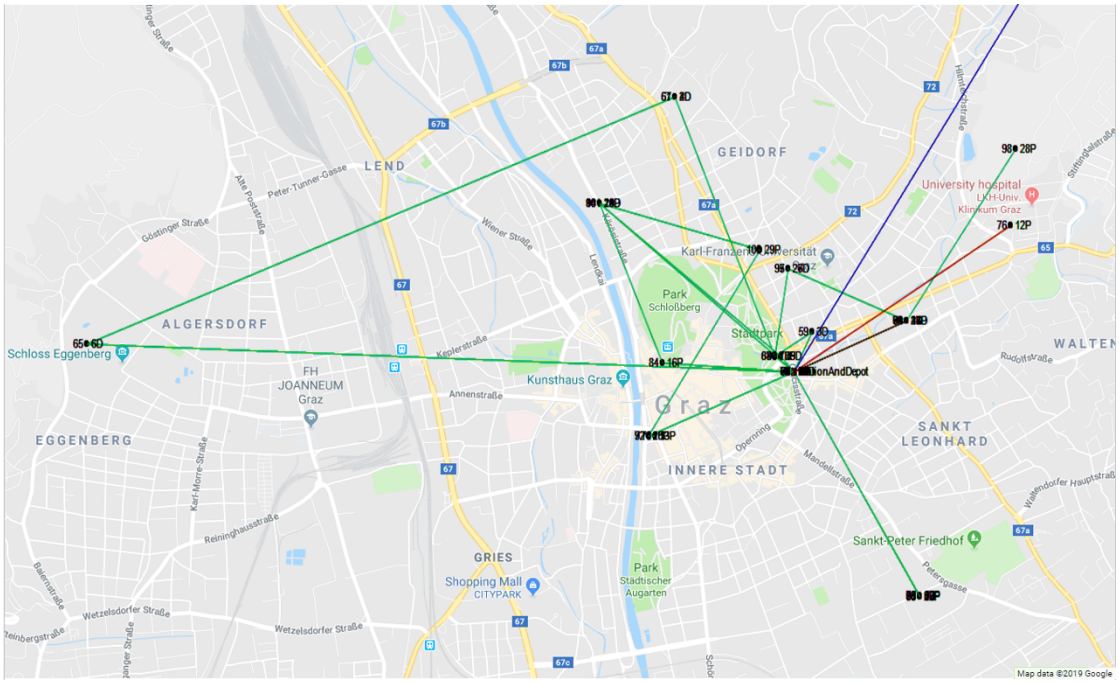


(a)

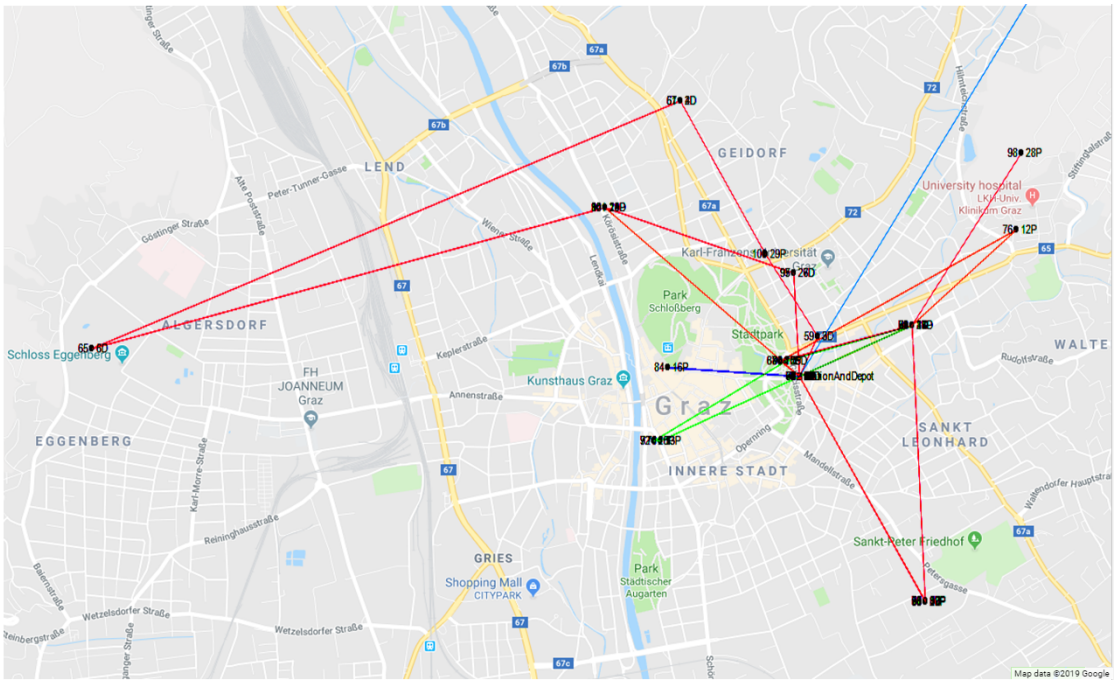


(b)

Figure 3.11: Solution of instance 20 with EPI, with (3.11a) LH300 or (3.11b) LH30. Instance 20 has 20 requests, time windows, and a cross-city share of 50%. Requests are numbered from 1 to 20, each with a pickup node 'P' and a delivery node 'D'. Pickups are always serviced before deliveries of the same request.



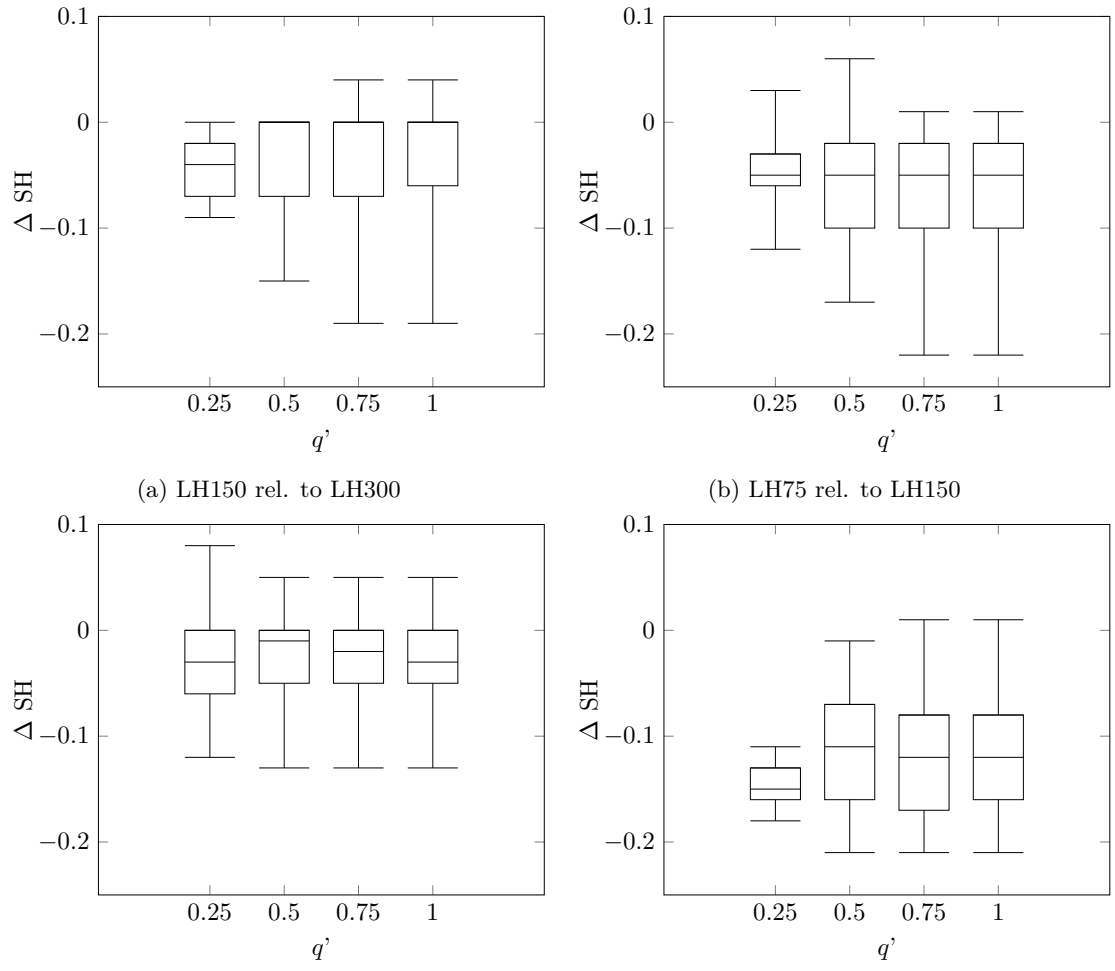
(a)



(b)

Figure 3.12: SH solution in Graz of instance 1 of the inter-library loan system with PI, using a LH via (3.12a) train or (3.12b) truck. The different schedule results in a different SH routing when switching from a LH via train to truck. Although the instances are composed of real travel times, this figure is a simplified depiction and does not include real travel paths (only direct connections).

3.6.3 Sensitivity analysis of SH improvements with more LH flexibility



(a) LH150 rel. to LH300

(b) LH75 rel. to LH150

(c) LH30 rel. to LH75

(d) LH30 rel. to LH300

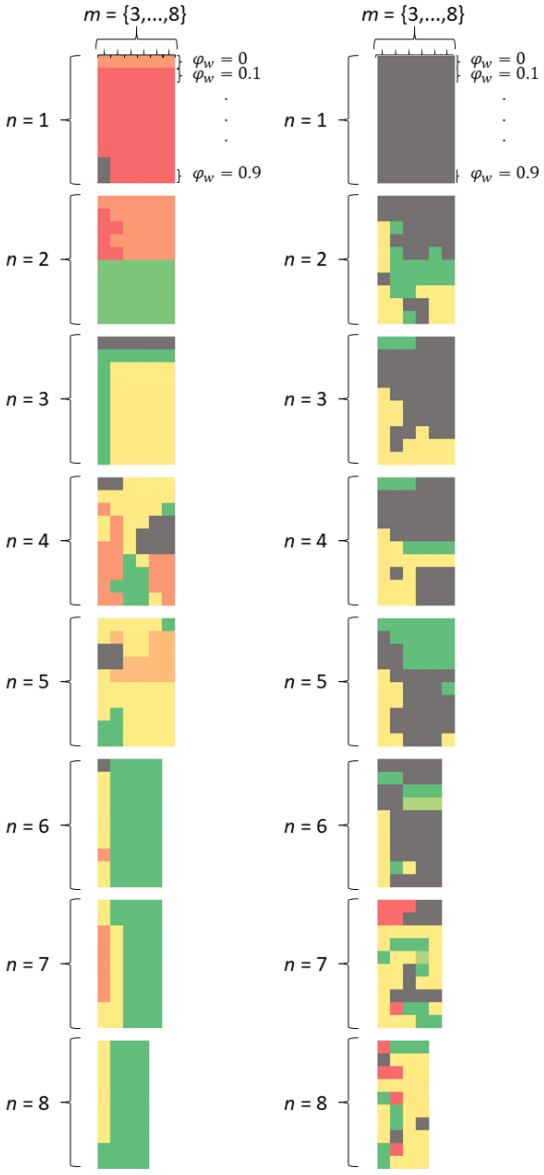
Figure 3.13: SH improvement with different LH capacity constraints for various LH frequency comparisons. The y-axis shows SH improvements up to 22 percent (Δ SH), the x-axis indicates the available LH capacity as a percentage of the total demand (q'), ranging from 25% to 100%.

3.6.4 Parameter settings

Instance	LH sol. Pool	VND tries	n	m	φ_w
C10	30	10^5	6	4	0.8
C11	30	10^5	6	4	0.8
C12	30	10^5	5	3	0.7
C20	30	10^4	5	3	0.4
C25	30	10^4	6	4	1
R6	30	10^5	6	4	0.8
R7	30	10^5	6	4	0.8
R8	30	10^5	6	4	0.8
R9	30	10^5	6	4	0.8
R10	30	10^5	6	4	0.8
R11	30	10^4	5	3	1
R12	30	10^5	6	4	0.8
R15	30	10^5	6	4	0.8
R20	30	10^5	6	4	0.8
R25	30	10^5	6	4	0.8
R40	30	10^6	7	4	0.2
RC6	30	10^5	6	4	0.8
RC7	30	10^5	6	4	0.8
RC8	30	10^5	6	4	0.8
RC9	30	10^5	6	4	0.8
RC10	30	10^5	6	4	0.8
RC11	30	10^5	6	4	0.8
RC12	30	10^5	6	4	0.8
RC15	30	10^5	6	4	0.8
RC20	30	10^5	6	4	0.8
RC25	30	10^4	5	3	0.2
RC40	30	-	7	4	0.8
RC50	30	10^5	5	3	0.2

Table 3.10: Individual parameters for the results provided in Table 3.5. The table includes the number of solutions in the pool after the LH assignment (LH sol. pool), the number of unsuccessful iterations after which the VND terminates (VND tries), the number of look-ahead steps n, the number of insertion decisions considered m, and the waiting time weight φ_w.

3.6.5 Solution quality plots



(a) (b)

Figure 3.14: Solution quality plots for instance R15 (3.14a) and R20 (3.14b) for different numbers of insertion decisions m, look-ahead steps n, and waiting time weight φ_w. Each block corresponds to the objective value obtained by explicit combinations of m, n and φ_w. The blocks are color coded and range from green (optimal solution found) over yellow and red (solution gets worse) to grey (no feasible solution found). For $n = 6$ and $n = 7$, the number of decisions changed to $m = \{3, ..., 7\}$, and for $n = 8$, it changed to $m = \{3, ..., 6\}$. For instance R15, it was relatively easy to obtain feasible solutions, and many parameter combinations lead to the optimal or very good objective values. For instance R20, feasibility was much harder to reach, and most parameter combinations lead to infeasible solutions.

Chapter 4

The pickup and delivery problem with alternative locations and overlapping time windows

Submitted to: *Transportation Research Part B: Methodological*
under revision, February 25th 2020.
A.G. Dragomir, T. Van Woensel & K.F. Doerner.

Abstract Sending and receiving parcels can be an inconvenience in both B2C and C2C settings. To facilitate transportation providers to expand their product portfolio with alternative concepts, we consider the pickup and delivery problem (PDP) with alternative locations and overlapping time windows. The transportation requests have to be served by a fleet of homogeneous capacitated vehicles. Each request may have multiple roaming pickup locations throughout the day with non-overlapping time windows (since the product cannot be in two places at once). Request may also have multiple roaming delivery locations and additionally an alternative recipient with its own set of roaming locations. As such, multiple persons in different locations can be available simultaneously to accept a delivery. Additionally, recipients can use 24-hour locker boxes if they are located near their home.

We propose a solution approach based on a multi-start adaptive large neighborhood search with problem specific operators to solve the pickup and delivery problem with alternative locations. We compare our algorithm with similar problems from the literature and examine in detail different scenarios based on real data provided by an Austrian logistics provider. In particular, we explore the benefits of locker boxes, roaming locations, alternative recipients, and mixed customer profiles with different preferences concerning data sharing and convenience. We found that an increase in flexibility and convenience for the customers translates into cost savings up to almost 30% for the carriers.

4.1 Introduction and motivation

In a C2C (Customer to Customer) market consumers interact directly with each other without a commercial player in the middle. Well-known online platforms such as eBay, Chraigslist or Willhaben enable persons to sell no longer needed or self-made items to private buyers. eBay alone almost doubled their gross merchandise volume in the last years to 95 billion U.S. dollars (full year 2018) [49]. Willhaben is one of the biggest online platforms in Austria for (mainly) C2C second-hand goods with a broad range of categories. The increasing popularity of smartphone applications for C2C transactions, i.e. OfferUp, Vinted and Tradesy, greatly simplified the process and enabled a further utilization increase. Additionally, the C2C market gains relevance due to

the second-hand consumption movement [157] where environmentally conscious consumers try to give goods a second life through repairs, repurposing, recycling or reusing instead of buying new goods.

With the growth of the online C2C market comes an increase in transportation requirements. Since a major motivation of online shopping is time savings and convenience [2, 137], we can assume that at least a considerable part of C2C participants will opt for shipping instead of a personal meeting for exchanging goods and money, even if both live in the same city. However, opting for a transportation carrier brings its own inconveniences: Public post services or commercial shipping companies are not very accommodating for private mail by enforcing personal visits at pickup points or simply by maintaining rigid schedules and opening hours. Sender of parcels have to make an appearance at their local post office within its opening hours. On the other hand, missed deliveries for parcel receivers are a prevalent problem in the industry. The cost of a missed delivery is more than just the last mile cost of a second delivery attempt. One has to include the environmental impact and the frustration of the carrier and receiver alike. Complaints take up customer service resources of the sender and the carrier, and a repetition of missed deliveries result in the loss of customer loyalty, bad reviews and long-term losses of potential new customers. Most current delivery options are not sufficiently accommodating for a C2C setting.

However, some private companies already noticed the need for more convenience for pickup and delivery problems (PDP) and implemented solutions accordingly: most big C2C platforms have included shipping options. Some, like Willhaben (which collaborates with Veloce) or Tradesy (which collaborates with USPS), took big steps towards convenience by offering a pickup option, same-day delivery, customer chosen delivery time slots, or even packaging material. Others, like OfferUp or Vinted, just act as an intermediary for standard shipping options.

Motivated by these real life challenges, we study a variant of the PDP in which both the seller and the buyer can specify a daily itinerary of their whereabouts. An itinerary is a detailed plan of places scheduled to visit during the day and the estimated time frame. The transportation carrier can then choose between multiple locations and their corresponding time windows for pickup and delivery of the parcel to minimize its transportation cost. It can be very expensive for the transportation carrier, even infeasible, to allow a single customer-selected pickup and delivery time window. However, by having multiple options spread throughout the day the problem becomes both harder to solve and cheaper in terms of transportation cost. The option of multiple locations counterbalances the restricted consolidation potential. However a direct one-to-one PDP ensures that the transportation requests are delivered faster and immediately without detours over a distribution center. To the best of our knowledge this problem, the pickup and delivery problem with alternative locations (PDPAL), was not previously studied.

This work focuses on the PDPAL where we know the itineraries of both parties in advance and optimize accordingly. While there is little doubt about the revenue potential of perceived convenience in time-based parcel delivery [66], we assume that consumers are willing to share their itineraries in exchange for the added benefits [52]. This is also supported by the fact that companies like Fetchr (fetchr.us) use live GPS tracking to find the recipient of a parcel, while Roadie (roadie.com) tries to estimate the whereabouts of recipients based on the GPS data collected by their phone.

The PDP is a generalization of the vehicle routing problem (VRP) [48] where a set of vehicles have to fulfill transportation requests that consist of a pickup location and a delivery location with a precedence constraint. In the literature [14, 122], our problem is classified as a *one-to-one* problem where the origin and destination of each request are coupled and, if not allowing transshipments, have to be served by the same vehicle. Savelsbergh and Sol [146] introduce the general PDP where each transportation request can be picked up and delivered to a set of possible locations. The general variant seems to have been studied much less than its special cases namely the PDP, dial-a-ride (on-demand transportation of people), or the VRP.

The PDPAL includes characteristics of the VRP with roaming delivery locations (VRPRDL) introduced by Reyes et al. [135] and Ozbaygin et al. [119] where a fleet of vehicles deliver parcels to the trunk of cars, wherever they may be parked during the day. The pickups all take place at the depot. Since the cars move around during the day and are never in two places at once, their problem has non-overlapping time windows for all roaming locations. The VRPRDL is similar to our problem, therefore we chose the work of Reyes et al. [135] and Ozbaygin et al. [119] for comparison and evaluation of the efficiency of our algorithm. Both, Lombard et al. [96] and Sampaio et al. [144] extended the VRPRDL by including stochastic travel times (VRPRDL-S). Lombard et al. [96] propose a combined Monte-Carlo sampling method and enhanced greedy randomized adaptive search procedure (GRASP) to solve this new problem variant while Sampaio et al. [144] use a scenario-based sample average approximation to get a heuristic solution. Ozbaygin and Savelsbergh [120] extended their original problem to include the possibility that the formerly fixed customer itineraries can change on short notice.

The VRP with floating targets was studied by Gambella et al. [61] where the targets (customers) move away from their home location during the day. Their problem models new applications in drone routing and ridesharing where the customers move towards the vehicles and all customers share a common destination. In this paper a dynamic variant is studied. It is relevant to note that customers have no choice in the matter and their movement towards a vehicle is part of the optimization. Both the customer and vehicle movement speed are randomly generated. For the literature concerning moving-target traveling salesman problems, see [24, 76].

Minimizing the chance of missed deliveries is studied in different variations: Florio et al. [57] used realistic availability profiles to optimize hit rates for finding customers at home. Local pickup/delivery points like gas stations, supermarkets or parcel lockers are considered as well [82, 102, 111, 151] either as a primary delivery location or only after delivery at home was unsuccessful. Other attempts for more convenience have been made by including anticipatory shipping [153], where customer behaviour is predicted and products shipped before they are ordered, or drone delivery [128].

Our main contribution is extending the VRPRDL to include pickup and deliveries and both non-overlapping and overlapping time windows. We provide an efficient algorithm to solve this new problem: a multi-start adaptive large neighborhood search (MS-ALNS). The ALNS has been developed foremost for a PDP [139] as well as successfully used for a wide variety of VRPs. In particular [65, 67, 100, 113] applied it to various PDPs. In our computational study we solve realistic instances based on data from an Austrian logistics provider in Vienna and are competitive when comparing with previous work from Reyes et al. [135] and Ozbaygin et al. [119]. Furthermore, we provide detailed comparisons of different scenarios where various degrees of flexibility are compared by including alternative recipients, roaming pickup and delivery locations, or limited participation of customers in the system. Additionally, we study the relevance of locker boxes and their cost savings potential. Locker boxes are unattended parcel delivery boxes, placed in strategic locations by the service provider, where recipients can autonomously pick up their parcel at their convenience.

The remainder of this paper is organized as follows: Section 4.2 provides a detailed problem formulation and a mathematical model. Section 4.3 describes the solution algorithm. Section 4.4 shows the computational results of our algorithm. We provide numerical results of detailed problem scenarios in Sections 4.4.1 and 4.4.2 and compare with the VRPRDL from the literature in Sections 4.4.3 and 4.4.4. Section 7.5 concludes and adds final remarks.

4.2 Problem definition and formulations

For the PDPAL, we have a complete directed graph $G = (N_0, A)$ with $N_0 = 0, 1, ..., n$ where node 0 corresponds to the depot and all other nodes correspond to either a potential pickup or a potential delivery location. N is a set of nodes excluding the depot. Each node i has a service

time σ_i and a quantity q_i associated with it, which has a positive value for a pickup location and a negative value for a delivery location. Furthermore, each node i has a time window associated with it during which a visit is allowed. The time window for each node i opens at a_i and closes at b_i. If a vehicle arrives before the opening of the time window, it has to wait. For each arc $(i, j) \in A$ a cost and a symmetrical travel time c_{ij} is given.

The PDPAL consists of a collection of transportation request R where an item or parcel has to be transported from the seller (where it is picked up) to the buyer (where it is delivered). Since the seller is a private person and has usually no commercial store or designated pickup location, it is assumed that on the day of the pickup the parcel travels with the seller on his/her itinerary. We also assume that the whereabouts of the seller are known in advance. Each place the seller visits throughout the day corresponds to a pickup location. Time windows are given for each pickup location indicating the time the package is available for pickup at each location. Since the seller moves between locations throughout the day, we define those as roaming pickup locations with non-overlapping time windows. The set of unique nodes N_r^P includes all possible pickup nodes for each request r.

The buyer moves around as well and has his/her own set of roaming delivery locations given by N_r^D. Additionally, each buyer has an alternative person available for receiving the parcel. This alternative person has their own set of roaming delivery locations as well. Combining both persons and adding 24-hour locker boxes, we obtain alternative delivery locations with overlapping time windows. The set N_r^D represents the union of all those locations. However, the locker boxes are only a viable option and included if they are located less then five minutes from the home location of the primary delivery recipient. Furthermore we assume that the locker boxes have enough remaining capacity to serve as a delivery location for requests and they do not generate additional cost since, in the case of Vienna, the locker box network is already build and operating.

For each request, we therefore have roaming pickup locations, alternative delivery locations and a fixed quantity. Figure 4.1 shows an example where request A has three (possible) pickup locations and request B has three (possible) delivery locations. A choice has to be made which of the alternative locations to include in the tour. The arrows depict the tour of the vehicle. For request B, the nodes nearest to the depot are selected, however for request A a more distant pickup node is selected due to time window restrictions (time windows are not depicted). Figure 4.2 shows an example for the time windows of a request with roaming pickups and alternative deliveries. There are three possible delivery locations with overlapping time windows.

Figure 4.1: VRPTW with multiple pickup and delivery locations

All transportation requests have to be served by a fleet $V = 1, 2, ..., v$ of homogeneous vehicles that start and end their tour at the single depot. All vehicles have a maximum tour duration L and have to return to the depot before the end of day E. For our instances, we assume a time frame (depot opening hours) of a working day of 16.5 hours, starting at 5 a.m. and ending at 9:30 p.m., the smallest unit of time is one minute. The goal is to select exactly one of the pickup and one of the delivery location possibilities for each request and to determine the node sequence for all vehicles while minimizing travel time and while fulfilling time window and capacity constraints.

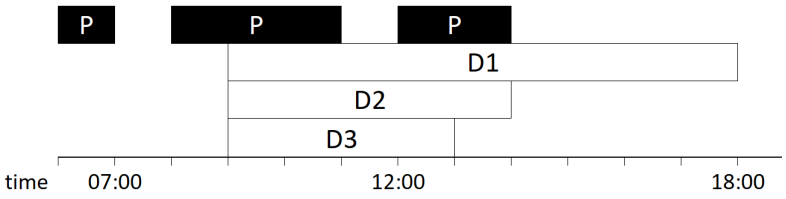


Figure 4.2: Possible time windows for a request with roaming pickups and alternative deliveries

4.2.1 Mathematical problem formulation

In Table 4.1 the mathematical notation can be found. It is understood that each node $i \in N$ is uniquely assigned to a single request $r \in R$. Furthermore each node i is either a pickup or delivery node:

$$N_r^P \cap N_{r'}^D = \emptyset \qquad \forall r, r' \in R \tag{4.1}$$

Parameter	Description
V	Set of vehicles
R	Set of all requests
N	Set of nodes
N_0	$N \cup \{0\}$, set of nodes including the depot
N_r^P	Set of possible pickup nodes for each request r
N_r^D	Set of possible delivery nodes for each request r
Q	Capacity of vehicle
c_{ij}	Travel time from node i to j
σ_i	Service time of node i
L	Maximum tour duration
E	End of day and depot closing time
q_i	Quantity at node i, positive for pickup, negative for delivery
a_i	Opening of time window for node i
b_i	Closing of time window for node i
x_{ijv}	$\begin{cases} 1, \text{if vehicle } v \text{ travels from node } i \text{ to } j, \\ 0, \text{otherwise.} \end{cases}$
t_i	Time of visit at node i
t_r^P	Pickup time of request r
t_r^D	Delivery time of request r
s_{iv}	Load after visiting node i with vehicle v
t_v^δ	Departure time of vehicle v from the depot
t_v^α	Arrival time of vehicle v at the depot

Table 4.1: Sets, parameters, and decision variables for the mathematical problem formulation

The objective is to minimize the travel time:

$$\text{Minimize} \sum_{i \in N_0} \sum_{j \in N_0} \sum_{v \in V} c_{ij} x_{ijv} \tag{4.2}$$

The constraints are the following:

$$\sum_{i\in N_0} x_{ijv} - \sum_{i\in N_0} x_{jiv} = 0 \qquad \forall j \neq i, j \in N, v \in V \tag{4.3}$$

$$x_{iiv} = 0 \qquad \forall i \in N, v \in V \tag{4.4}$$

$$\sum_{i\in N_0} x_{i0v} = \sum_{j\in N_0} x_{0jv} = 1 \qquad \forall v \in V \tag{4.5}$$

$$\sum_{i\in N_0, i\neq j} \sum_{v\in V} x_{ijv} \leqslant 1 \qquad \forall j \in N \tag{4.6}$$

$$t_i + \sigma_i + c_{ij} \leqslant t_j + M(1 - x_{ijv}) \qquad \forall i \in N_0, j \in N, v \in V \tag{4.7}$$

$$a_i \sum_{v\in V} \sum_{j\in N_0\setminus i} x_{ijv} \leqslant t_i \leqslant b_i \sum_{v\in V} \sum_{j\in N_0\setminus i} x_{ijv} \qquad \forall i \in N \tag{4.8}$$

$$0 \leqslant t_i \leqslant M \sum_{j\in N_0} x_{ijv} \qquad \forall i \in N, v \in V \tag{4.9}$$

$$\sum_{i\in N_r^P} \sum_{j\in N_0\setminus i} x_{ijv} = \sum_{i\in N_r^D} \sum_{j\in N_0\setminus i} x_{ijv} \qquad \forall r \in R, v \in V \tag{4.10}$$

$$\sum_{i\in N_r^P} \sum_{j\in N_0\setminus i} \sum_{v\in V} x_{ijv} = \sum_{i\in N_r^D} \sum_{j\in N_0\setminus i} \sum_{v\in V} x_{ijv} = 1 \qquad \forall r \in R \tag{4.11}$$

$$t_r^P = \sum_{i\in N_r^P} t_i \qquad \forall r \in R \tag{4.12}$$

$$t_r^D = \sum_{i\in N_r^D} t_i \qquad \forall r \in R \tag{4.13}$$

$$t_r^P \leqslant t_r^D \qquad \forall r \in R \tag{4.14}$$

$$s_{jv} \leqslant s_{iv} + q_j + M(1 - x_{ijv}) \qquad \forall i, j \in N_0, v \in V \tag{4.15}$$

$$s_{jv} \geqslant s_{iv} + q_j - M(1 - x_{ijv}) \qquad \forall i, j \in N_0, v \in V \tag{4.16}$$

$$0 \leqslant s_{iv} \leqslant Q \qquad \forall i \in N, v \in V \tag{4.17}$$

$$s_{0v} = 0 \qquad \forall v \in V \tag{4.18}$$

$$t_v^\delta + c_{0i} \leqslant t_i + M(1 - x_{0iv}) \qquad \forall i \in N, v \in V \tag{4.19}$$

$$t_i + \sigma_i + c_{i0} \leqslant t_v^\alpha + M(1 - x_{i0v}) \qquad \forall i \in N, v \in V \tag{4.20}$$

$$0 \leqslant t_v^\delta \leqslant t_v^\alpha \leqslant E \qquad \forall v \in V \tag{4.21}$$

$$t_v^\alpha - t_v^\delta \leqslant L \qquad \forall v \in V \tag{4.22}$$

$$t_v^\delta \leqslant M(1 - x_{00v}) \qquad \forall v \in V \tag{4.23}$$

$$t_v^\delta \geqslant -M(1 - x_{00v}) \qquad \forall v \in V \tag{4.24}$$

$$t_v^\alpha \leqslant M(1 - x_{00v}) \qquad \forall v \in V \tag{4.25}$$

$$t_v^\alpha \geqslant -M(1 - x_{00v}) \qquad \forall v \in V \tag{4.26}$$

$$x_{ijv} \in \{0, 1\} \qquad \forall i, j \in N_0, v \in V \tag{4.27}$$

$$0 \leqslant t_i \leqslant E \qquad \forall i \in N_0 \tag{4.28}$$

$$0 \leqslant t_r^P \leqslant E \qquad \forall r \in R \tag{4.29}$$

$$0 \leqslant t_r^D \leqslant E \qquad \forall r \in R \tag{4.30}$$

$$0 \leqslant t_v^\delta \leqslant E \qquad \forall v \in V \tag{4.31}$$

$$0 \leqslant t_v^\alpha \leqslant E \qquad \forall v \in V \tag{4.32}$$

Constraints (4.3) state that if vehicle v visits a node in a tour, it has to leave the node again. Constraints (4.4) state that traveling between the same node i is forbidden (except the depot because that would indicate an empty tour). Constraints (4.5) state that each vehicle tour originates and terminates at the depot. Constraints (4.6) make sure that each node is visited at most once. Constraints (4.7) state that if a vehicle v travels from i to j, the fulfillment times at the nodes must be consecutive. Therefore, the fulfillment time t_j at node j has to be greater than or equal to the sum of the fulfillment time t_i at node i, service time σ_i, and travel time c_{ij} between the nodes. These constraints also eliminate subtours, because all tours begin and end at the depot. Constraints (4.8) restrict the fulfillment times at the nodes to the time windows. Constraints (4.9) restrict the fulfillment times at the nodes to be at least zero and at most zero or a large number depending on whether the node i is visited. Constraints (4.10) and (4.11) ensure that from all nodes i, from a set of pickup or delivery nodes belonging to a single request r, exactly one is visited. They also ensure that the pickup and delivery is fulfilled by the same vehicle by forcing, for each vehicle, the same number of arcs to leave the pickup nodes N_r^P of a request as well as arrive at the delivery nodes N_r^D of the same request r. Constraints (4.12) and (4.13) state that the pickup and delivery times t_r^P and t_r^D of request r have to be set to the sum of the fulfillment times t_i at all nodes i from the set N_r^P or N_r^D respectively. Since it was ensured by Constraints (4.10) and (4.11) that exactly one pickup node and one delivery node is visited for each request, and ensured by Constraints (4.9) that the times from not visited nodes are set to zero, the sum of the fulfillment times of all nodes is the fulfillment time at the visited request. Constraints (4.14) state that the pickup must be completed before the delivery is possible. Constraints (4.15) to (4.18) are the loading restrictions for the vehicles. If we travel from i to j with vehicle v, the load s_{jv} before serving node j equals the load before servicing node i plus the quantity q_i of node i. This quantity can be negative if a delivery occurs. Constraints (4.19) and (4.20) state that all arrival times t_v^α and departure times t_v^δ at the depot have to be consistent with traveling times and fulfillment times of the vehicles. Constraints (4.21) state that the departure and arrival times at the depot have to be during the time span of the day, as well as ensuring that each vehicles first departs and then arrives at the depot. Constraints (4.22) limit the maximum vehicle tour duration. Constraints (4.23) to (4.26) state that if a tour is empty, the departure time and arrival time at the depot are 0. Constraints (4.27) to (4.32) limit the range of the decision variables.

4.3 Solution algorithm

For our solution algorithm we decompose the problem in two sub-problems: a request to vehicle assignment and a single vehicle routing which define the specific pickup and delivery locations as well as the visiting order. Specifically, to solve the PDPAL, three decisions have to be made:

- Select exactly one pickup and one delivery location for each request
- Assign requests to vehicles
- Determine the sequence in which the vehicles visit the nodes

The request-to-vehicle assignment is determined by our metaheuristic algorithm: a multi-start adaptive large neighborhood search (MS-ALNS). The pickup and delivery locations and the node sequence are determined by a separate algorithm that is responsible for the routing. It is based on a farthest insertion and a local search procedure. All parameters used for the algorithm can be found in Table 4.9 in the Appendix. The algorithm terminates after a runtime limit is reached. The runtime depends on the instance size so that larger instances have a longer runtime.

4.3.1 Construction heuristic

For a multi-start approach, we create a pool of size $|P|$ of different initial solutions. A larger solution pool means a higher diversification in the starting points for the metaheuristic which can be advantageous. On the other hand, this advantage is balanced out when considering a limited runtime where not every solution might be selected for improvement. Therefore, we select a value as big as possible to allow for as much diversification as possible, and as small as necessary to ensure that each solution can be selected for improvement. We use a simple construction heuristic where each request has a probability between ρ_{min} and ρ_{max} to be assigned to a random vehicle. Otherwise they are assigned to a separate vehicle. This leads to a randomized grouping of requests which proved quite successful in combination with a multi-start approach. It is also possible, that requests are grouped which can not feasibly served together due to time window or route length constraints. In this case, the solution is discarded and the construction heuristic is applied again. When all requests are assigned to vehicles, the routing heuristic (see Section 4.3.4) is used to determine the pickup and delivery locations, the node sequence, and the travel time and consequent costs.

4.3.2 Multi-start Adaptive Large Neighborhood Search (MS-ALNS)

Algorithm 1 Multi-start Adaptive Large Neighborhood Search (MS-ALNS)

```
1: s''                                                    ▷ Best found solution
2: procedure MS-ALNS
3:     while solution pool is not filled do
4:         create feasible initial solutions
5: while time limit not reached do
6:     apply a roulette wheel selection for choosing the next incumbent solution s
7:     improve the solution s with ALNS
```

Algorithm 1 gives an overview of the MS-ALNS. First, a solution s is chosen from the solution pool using a roulette wheel selection [7] where better solutions have a higher probability in proportion to their fitness of being chosen. For a minimization problem, a better fitness means a smaller objective function value (OFV). This solution s is then improved with an ALNS that is described in detail in Algorithm 2.

At the start of the algorithm the variable c is initialized to 0, a counter to keep track of how many iterations have passed where no improvement was found. If c reaches c_L the ALNS is terminated, the modified solution s' returned to the pool, replacing the solution s, and a new solution is selected for improvement. Alternatively, the algorithm also stops, if the runtime limit is reached.

Also, the variable w is initialized to 1, the threshold for accepting a worse solution in relation to the starting solution s. This helps to avoid getting stuck in a local optima by allowing us to escape it when we can no longer find improvements. The standard values for the parameters c_w, $c_{\gamma_{max}}$ and w_{max} can be found in Table 4.9 in the Appendix, where c_w is the number of iterations without improvement after which the threshold w is increased by w' until at most w_{max}. A new solution is accepted as the new incumbent solution if its OFV is smaller than the OFV of s times w. So, for $w = 1$ no worse solution is accepted and for $w = 1.2$ a solution is accepted if it is less than 20 percent worse than s. $c_{\gamma_{max}}$ is the number of iterations without improvement after which the solution destroy percentage γ_{max} is doubled, to the end that a local optima might be escaped.

A copy s' of s is made as an incumbent solution to be modified. A destroy and repair operator is applied. If the modified solution s' is feasible and better than the previously found best solution

Algorithm 2 Adaptive Large Neighborhood Search (ALNS)

$c \leftarrow 0$ ▷ counter for iterations without improvement
$c_w \leftarrow 0.5 * c_L$ ▷ after how many iterations without improvement a worse solution is accepted
$w_{max} \leftarrow 1.5$ ▷ how much worse the incumbent solution is allowed to be in relation to s
$w \leftarrow 1$ ▷ Initialize w, the threshold for allowing worse solutions
$w' \leftarrow 0.05$ ▷ by how much w changes if no solution could be found
$\gamma_{max} \leftarrow 0.4$ ▷ Initialize γ_{max}, the upper destroy limit for a solution
$c_{\gamma_{max}} \leftarrow 0.75 * c_L$ ▷ after how many iterations without improvement the destroy limit is changed
while $c < c_L$ **do**
    $c \leftarrow c + 1$
    create new incumbent solution $s' \leftarrow s$
    destroy s' by selecting destroy operator
    repair s' by selecting repair operator
    if s' is feasible and OFV $s' < s''$ **then**
        $s'' \leftarrow s'$
        $s \leftarrow s'$
        $w \leftarrow 1$
        $c \leftarrow 0$
        reset γ_{max} to original value
        update probabilities for destroy and repair operators
    else if s' is feasible and OFV $s' < s * w$ **then**
        $s \leftarrow s'$
    else if $s' \geqslant s * w$ or infeasible **then**
        $s' \leftarrow s$ ▷ s' is discarded
    if $c > c_w$ **and** $w < w_{max}$ **then**
        $w \leftarrow w + w'$
    if $c > c_{\gamma_{max}}$ **then**
        $\gamma_{max} \leftarrow 2 * \gamma_{max}$

s'', the best solution s'' is replaced by s', s is replaced by s', the parameters w and c are reset, and the probabilities for choosing destroy and repair operators are updated. In all other cases, c is increased by one to keep track of the number of iterations without improvement.

If the modified solution s' is feasible, worse than the best solution s'' but better than s, s is replaced by the modified solution s'. If s' is feasible but worse than s, it depends on the value of w if we take it as the new incumbent solution. If the modified solution s' is infeasible or worse than the threshold set by w, s' is discarded and replaced by s.

At the end of each iteration we check if $c > c_w$ and $w < w_{max}$, that is we check if the number of iterations without improvement has exceeded a certain threshold and if the multiplier, that determines the maximum OFV of an accepted solution, has not yet reached its maximum. If this is the case we increase the multiplier to accept even worse solutions. Additionally, we check if $c > c_{\gamma_{max}}$, that is if the number of iterations without improvement has exceeded an even higher threshold. If this is the case we double the allowed solution destroy percentage. Both, accepting a worse solution and increasing the amount of the solution to be destroyed, helps us to escape from a local optima.

4.3.3 Destroy and repair operators

The ALNS uses three destroy and six repair operators. Each destroy operator removes between γ_{min} and γ_{max} percent of the solution, i.e. requests-to-vehicles assignments. Simultaneously, the pickup and delivery nodes are removed from the vehicle tours while leaving the rest of the node sequence unchanged. This is necessary because we need to be able to evaluate additional request removals from the same vehicle tours. The method of removal is identical for all destroy operators, they are only distinguished by their selection of which requests to remove.

The *destroy greedy requests* operator calculates the reduction in tour length when removing a request from their respective tour. Since we always remove two nodes, a pickup node and a delivery node, we adapted the standard savings calculation by Clarke and Wright [26] and the reduction in tour length will henceforth be termed 'savings' value as well. If the delivery node is visited right after the pickup node, the savings $s_{p,d}$ for removing the pickup node p and delivery node d is calculated as follows:

$$s_{p,d} = c_{p-1,p} + c_{p,d} + c_{d,d+1} - c_{p-1,d+1} \tag{4.33}$$

with c_{ij} denoting the cost and travel time between two nodes. If there are other nodes between the pickup node p and the delivery node d, the savings is calculated as follows:

$$s_{p,d} = c_{p-1,p} + c_{p,p+1} + c_{d-1,d} + c_{d,d+1} - c_{p-1,p+1} - c_{d-1,d+1} \tag{4.34}$$

We use a roulette wheel selection for choosing which request to remove from the solution with bigger savings having a proportionally bigger chance for selection.

The *destroy random requests* operator chooses the requests to remove randomly, as does the *destroy random vehicles* operator. The difference is, that the former picks single requests from all vehicles, and the latter picks random vehicles and removes all requests within those vehicles. We did not use a roulette wheel selection for selecting the vehicles to destroy, since vehicles with a high cost are not necessarily bad. On the contrary, high cost vehicles usually serve many requests and this is not something that needs to be discouraged. Neither can we make universal statements about low cost vehicles, therefore we opted for randomness.

The *repair cheapest insertion* operator calculates the insertion cost of each request for each vehicle. The insertion cost are the minimum additional cost that arise when serving an additional request. For determining the insertion cost of a request into a specific vehicle, we try all possible pickup and delivery positions. The best position with the smallest cost is selected. We use the same calculation as Dragomir and Doerner [42] with the weights φ_c of one for the travel time and φ_w of 0.2 to account for waiting time. To allow for some variability we randomly choose between the best three vehicles.

The *repair request compatibility pairs* and *repair request compatibility triple* operators assign a request to a vehicle based on a compatibility measure. The compatibility measure tries to indicate how well a pair/triple of requests fit together when being served by the same vehicle. The calculation is based on the travel times and time windows and is calculated as follows: For each pair of requests, r_1 and r_2, a single compatibility indicator is calculated. This is done only once as a preprocessing step and the values are stored in a two dimensional matrix for all request pairs (three dimensions for request triples). We look how 'compatible' all pickup and delivery locations of r_1 are to all locations (pickup and delivery) of r_2. The compatibility of a pickup (delivery) location to all other pickup (delivery) locations of the same request are not relevant since both locations will never be visited simultaneously.

The compatibility κ of r_1 and r_2 is therefore

$$\begin{aligned}\kappa_{r_1 r_2} = &\sum_{p_1 \in N^P_{r_1}} \sum_{p_2 \in N^P_{r_2}} max(|b_{p_1} - a_{p_2} - c_{p_1 p_2}|, |b_{p_2} - a_{p_1} - c_{p_1 p_2}|) \\ + &\sum_{p_1 \in N^P_{r_1}} \sum_{d_2 \in N^D_{r_2}} max(|b_{p_1} - a_{d_2} - c_{p_1 d_2}|, |b_{d_2} - a_{p_1} - c_{p_1 d_2}|) \\ + &\sum_{d_1 \in N^D_{r_1}} \sum_{p_2 \in N^P_{r_2}} max(|b_{d_1} - a_{p_2} - c_{d_1 p_2}|, |b_{p_2} - a_{d_1} - c_{d_1 p_2}|) \\ + &\sum_{d_1 \in N^D_{r_1}} \sum_{d_2 \in N^D_{r_2}} max(|b_{d_1} - a_{d_2} - c_{d_1 d_2}|, |b_{d_2} - a_{d_1} - c_{d_1 d_2}|) \end{aligned} \quad (4.35)$$

with N^P_r and N^D_r as the sets of alternative pickup and delivery locations of request r, c_{ij} as the travel times between locations i and j, a_i and b_i as the begin and end time windows for location i. Request pairs have a higher compatibility if the travel time between them and their time windows make a visit easily feasible. For the triples we calculate all pairs separately and take the average.

For assigning a request to a vehicle we calculate the most promising feasible vehicle using the compatibility measure. The vehicle compatibility is the average of all compatibilities of already-assigned requests to be served by that vehicle and the request in question. It is important to take the average because vehicles that serve more requests would otherwise have a higher compatibility, no matter how high the values actually are. Of all possible feasible vehicles, the selection is made by a weighted roulette mechanism, prioritizing the vehicles with the highest compatibility.

For the *repair location compatibility pairs* and *repair location compatibility triple* the same calculation is used. However the compatibilities over the pickup and delivery locations of the requests are not summed up, but rather the maximum is used. By using the maximum we can identify if request pairs (triples) fit together well if certain locations are selected even if the other locations do not seem promising.

The *repair request randomly* operator inserts a request in a random non-empty vehicle, if it can be feasible inserted. The feasibility is verified by quickly calculating the insertion cost and ensuring that no constraints are violated (capacity, time windows, tour length). If an insertion is not feasible, any other non-empty vehicle is selected. If no insertion is feasible in any non-empty vehicle, it is inserted alone in an empty vehicle.

After destroying and repairing the request-to-vehicle assignment and calculating the routing (see description in Section 4.3.4) we can determine if the solution was improved by the changes. If this is the case, we update the probabilities of selecting those specific destroy and repair operators. At the beginning of the algorithm, all operators have equal probabilities to be selected. For each operator we count the number of successful applications of that specific operator s_o. We also count how often the solution was improved with all operators s_s. The probability p_o to select a

specific operator o is calculated as

$$p_o = \frac{s_o}{s_s} \tag{4.36}$$

4.3.4 Routing algorithm

The node sequence is determined for each vehicle separately and all vehicles start empty. For the tour construction we use a modified farthest insertion heuristic with a roulette wheel selection that considers all possible locations and has a penalty for waiting time. The constructed tour is improved by local search using *move* and *swap* operators embedded in a variable neighborhood descent (VND)[106] framework. Since the routing calculation is quite fast but stochastic, we calculate it multiple (five) times to guarantee a good solution.

For our construction heuristic, we need to calculate the insertions for each request. An *insertion* is defined by a request and a vehicle, a determined pickup location, a determined delivery location, the places of those locations within the (already existing) node sequence, and cost. The insertion calculation is identical to the savings Eq. (4.33) and (4.34) and it guarantees that all constraints are satisfied (time windows, maximum tour length, depot opening hours, vehicle capacity). A separate insertion is created for each pickup/delivery location combination and for each place in the node sequence. From this list, we first select one insertion for each request, namely the one with the smallest cost. Ties are broken randomly. All other insertions are discarded. From the remaining insertions one is picked by a roulette wheel selection prioritizing the highest cost. The request is inserted into the vehicle, the insertions recalculated and the process begins anew until all request are inserted into all vehicles. A small example with two requests can be found in Figure 4.3 and its subfigures.

For improvement, the VND utilizes neighborhoods that are applied in order of complexity. Each neighborhood is performed until no further improvement can be found for θ iterations and a local optimum is reached. To escape from this local optimum, the algorithm moves to the next neighborhood. As soon as a new solution is found, the algorithm restarts and returns to the first neighborhood since it's the least complex and 'cheapest' in terms of computation time. If no more improvements can be found for θ iterations for all neighborhoods, the algorithm is terminated. The consequent solution is a local optimum for all neighborhoods applied [159]. For our VND we use two neighborhoods.

The *move* neighborhood picks a random request, removes it from its original tour, and inserts it into the best possible position (in terms of node sequence) of any other vehicle (including its former vehicle). The goal is to reduce the distance. For determination of the distance reduction we compare the savings value when removing a request (see Eq. (4.33) and (4.34)) to the insertion cost in the new vehicle and/or the new position of the nodes. If a better vehicle and/or position can be found, the request is moved, otherwise it remains in its current vehicle and its current position.

The *swap* neighborhood picks two random requests from separate vehicles and swaps the vehicle assignment of the requests. The requests are first removed from their original tour and reinserted in the tour of the other request respectively. Only the best position within the node sequence is considered when inserting a request, no matter the position of the original request. The cost savings when removing and reinserting both requests are determined while both insertions have to be feasible. If the total cost is smaller, the swap counts as a success and is saved. Otherwise the changes are discarded.

4.4 Computational results

Our computational results section is structured as follows: in Section 4.4.1 a description of our own instances and the created scenarios can be found. Section 4.4.2 reports the results of

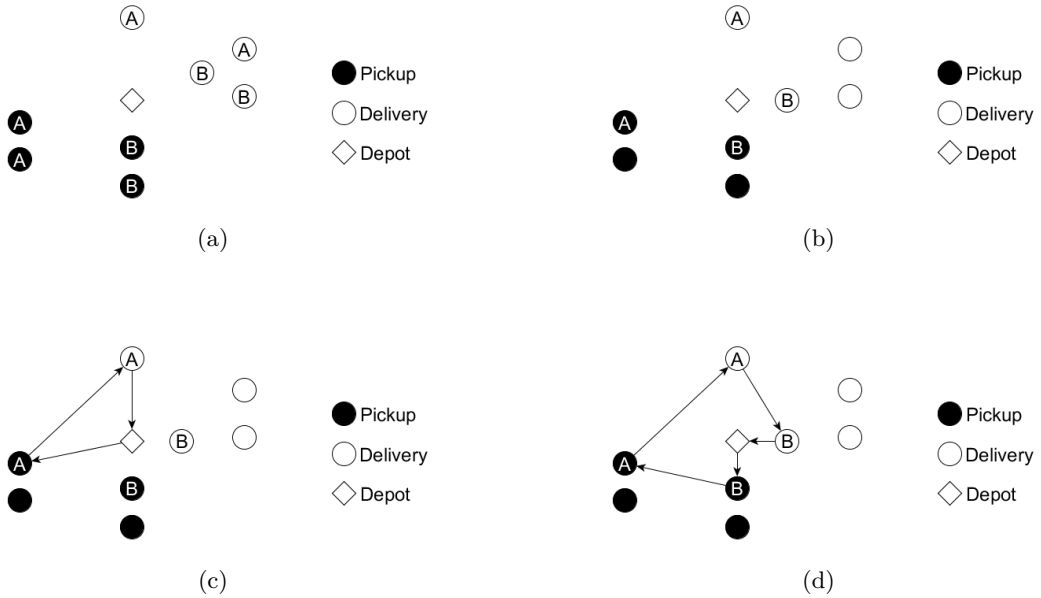


Figure 4.3: Depiction of farthest insertion. Figure 4.3a shows two requests 'A' and 'B' with two pickup and two delivery locations each. Not depicted is the insertion calculation that determines the cost for each pickup and delivery location combination for each request. Figure 4.3b shows how exactly one insertion is selected for each request, namely the one with the smallest cost. Choosing from the remaining (in this example two) insertions, Figure 4.3c shows the selection of the farthest insertion, request 'A'. Here, the insertions are recalculated. For the only remaining request 'B' the same nodes are selected, however the placement of the nodes within the sequence are different, since the tour already serves a request. Figure 4.3d show the selection of the next-farthest (and last) insertion, request 'B', and its placement in the tour.

those scenarios. To evaluate the quality of our solution algorithm, we provide numerical results and compare to Reyes et al. [135] in Section 4.4.3 and to Ozbaygin et al. [119] in Section 4.4.4.

All computational experiments are performed on a desktop computer with an Intel(R) Core(TM) i7-4790 CPU @ 3.60GHz processor (4 cores) and 8 GB RAM running on Windows 10.

4.4.1 Instance and scenario descriptions

Locations and distances

For our self generated instances, we had access to Viennese delivery data from 2017 from an Austrian logistics provider. We put a grid over the city and adjusted its coarseness until around 700 locations remained, so that each grid square contained exactly one location. Those were classified into 'home', 'work', and 'other' roaming locations, depending on their nature. 'Home' and 'work' are self-explanatory, however an 'other' roaming location was defined as a place where people typically and regularly spend a block of their time (fitness center, library or other educational institutions, pub/restaurants). A location can have multiple classifications, for example a university is both 'work' for its employees and 'other' for its students and a multistory apartment building with a fitness studio and several medical practices have all three classifications. Additionally, we have around 100 locations for the 24-hour locker box stations belonging to the Austrian Post in Vienna (with a separate classification 'postal locker box'). From these 799 locations (including the depot), the actual travel time matrix was created using Open Street Map [118]. A real-life travel time matrix is neither symmetric nor does it fulfill the triangular

inequality. Therefore, we symmetrized it and then determined the shortest path for all arcs to fulfill the triangular inequality by using the Floyd-Warshall Algorithm [58]. All travel times are given in minutes.

Customers and requests

Each request consists of a seller (sender), a buyer (receiver) of a parcel and a random generated quantity between one and five. We created 17 different customer profiles: six profiles representing full-time employment, eight profiles representing part-time employment, and three profiles representing unemployment (including pensioners) where each profile is given as a daily itinerary of the whereabouts of a customer. Therefore all scheduled visits to different locations during the day are known. The detailed profiles can be found in Table 4.7 in the Appendix. The profiles have on average 3.1 locations, ranging from one to five. There are always 30 minutes between locations to account for unavailability while the customer moves between places. For Vienna, the statistical probability to be employed is 65 percent. If one is employed, there is a 71.4 percent chance for full-time employment [155]. Additionally, each person has a probability of living and working in a certain sector in Vienna. The probabilities can be found in Table 4.8 in the Appendix. Based on this data each customer was created by selecting the employment type based on the probabilities mentioned above, then selecting a random profile of this employment type. Additionally, the home location, work location and up to two 'other' roaming locations were selected based on the probabilities in Table 4.8 ('other' roaming locations have an equal probability for each sector). After selecting the sector for each type of node, the actual location was selected randomly out of all locations in that particular sector. For each request we need to create 3 customers: one seller (pickup), one buyer (delivery) and one alternative person (alternative delivery recipient). Additionally, the buyers have the possibility to get their parcel delivered to a 24-hour locker box if one is located within five minutes of their home location.

Instances types

We created 19 different types (scenarios) of the same instance where the customer profiles, all relevant locations and the travel time matrix are identical for all types. The home location and itineraries of all customers are exactly the same among the types even if one type includes deliveries to roaming or alternative locations and the other type doesn't. Therefore we made a comparison possible to determine the usefulness of roaming locations, alternative persons or locker boxes. Table 4.2 gives an overview for all types and a detailed description. All instances are provided online at `https://bda.univie.ac.at/research/data-and-instances/vehicle-routing-problems/`.

4.4.2 Scenario comparisons

We created 10 sets of each type in three different sizes: 30 requests, 50 requests and 100 requests (570 instances in total). Table 4.3 shows the average objective function values (OFV) for each scenario type over all 10 sets. The scenarios in the table are sorted by complexity (less flexibility on top, more flexibility on the bottom) and the color gradient already gives a good impression about the usefulness of more complexity. To examine the scenarios in more detail we provide explicit comparisons in Table 4.4. Explicit results of all instances are provided as complementary material and can be found at `https://bda.univie.ac.at/research/data-and-instances/vehicle-routing-problems/`

Comparison of 'traditional' systems to 'full flexibility'.

First, we compare the possible cost benefits when switching from a more 'traditional' system to one with full flexibility. A 'traditional' system is one where only home delivery is considered,

Type	Description
H	Customers are assumed to be available at home all day. Both pickup and delivery locations are the home locations. No time windows.
H+TW	Only home pickup/delivery within the customer specified time windows. Option with highest convenience.
H+TW+LB	Home pickup/delivery only within the customer specified time windows. Additionally, delivery to a nearby 24-hour locker box is possible.
RP+RD(%)	Roaming pickup locations and roaming delivery locations for (%) of requests. Pickup/delivery locations and time windows are based on the customer itineraries. All other requests can only be served at home within the TW specified (like 'H+TW').
RP+RD+LB(%)	Identical to RP+RD(%). Additionally, 24-hour locker boxes are available for all requests.
RP+RD+AL(%)	Identical to RP+RD(%). Additionally, each buyer (delivery customer) has an additional alternative person to whom can be delivered instead. This person has its own itinerary.
RP+RD+AL+LB(%)	Identical to RP+RD+AL(%). Additionally, 24-hour locker boxes are available for all requests (including the ones that can only be served at home).

Table 4.2: Description of all instance types. Types containing a '%' are created in four variants: 25%, 50%, 75% and 100%.

	30 req.	50 req.	100 req.	Avg.
H+TW	628	954	1723	1102
H+TW+LB	592	911	1620	1041
H	543	843	1484	957
RP+RD(25)	609	944	1697	1083
RP+RD(50)	582	923	1672	1059
RP+RD(75)	558	908	1640	1035
RP+RD(100)	545	865	1598	1003
RP+RD+LB(25)	583	888	1620	1031
RP+RD+LB(50)	565	872	1598	1011
RP+RD+LB(75)	539	872	1565	992
RP+RD+LB(100)	541	859	1540	980
RP+RD+AL(25)	514	819	1453	929
RP+RD+AL(50)	502	815	1437	918
RP+RD+AL(75)	486	797	1438	907
RP+RD+AL(100)	498	753	1428	893
RP+RD+AL+LB(25)	496	809	1416	907
RP+RD+AL+LB(50)	490	795	1412	899
RP+RD+AL+LB(75)	487	791	1401	893
RP+RD+AL+LB(100)	485	776	1410	890

Table 4.3: Average OFV over 10 sets of 19 different scenarios rounded commercially to the nearest integer (rounded down until including .4, rounded up from including .5). The column 'Avg.' is the average value over all instance sizes for a scenario. For a detailed description of the scenarios see Table 4.2. The scenarios are sorted from lowest to highest flexibility. Low flexibility types include only the home locations (with and without time windows or locker boxes). Middle flexibility types include a roaming option and high flexibility types include an alternative persons. The color gradient depicts the relation of a OFV in relation to the other OFV within the same column and ranges from 'red' (high OFV, worse solution) to 'green' (low OFV, good solution). The results were obtained with a runtime multiplier $mu = 1$ where the runtime is the number of requests in seconds.

either the whole day without time windows (H) or only within the time windows specified (H+TW). We also include the possibility to deliver to a nearby locker box (H+TW+LB) in our definition of 'traditional' systems. All those options are similar to what the Austrian Post and other transportation carriers already have implemented. 'Full flexibility' is our version of including a roaming pickup possibility, roaming deliveries, and locker boxes for all requests. H+TW is of course the option with the highest restrictions for the transportation carrier. When compared to 'full flexibility' we obtain savings of up to almost 30% for the scenarios with 30 requests. For bigger scenarios the numbers are slightly smaller, however an average over all scenarios of 23.7% cost savings remains. When including the locker boxes (H+TW+LB) the potential savings diminish slightly by 6-7%. When comparing the all-day home delivery scenario (H) with 'full flexibility' the savings are slightly less with 7.4% on average over all scenarios. In general we can be confident with the results and the potential real life benefits of a similar system.

The relevance of locker boxes

Locker boxes are postal boxes used by the Austrian postal service. A delivery to a locker box can simply be managed by having it as the delivery address on the package. Recipients get an electronic notification when the parcel arrived and can pick it up around the clock (with a personalized code submitted simultaneously with the notification). Although in practice recipients can select any locker box of their choice, we included them only as an option if one was located within 5 minutes of their home location. Looking at the computational results, we observe that including locker boxes do provide cost savings in comparison to the option without locker boxes. However, the savings diminish the more flexible the basic problem is. For the option with home delivery within customer specified time windows (H+TW), adding locker boxes provide up to 6.4% cost savings (with an average of 5.8%), while adding the option to roaming pickup and roaming deliveries (RP+RD) the benefits amount up to 4.5% (with an average of 4.2%). When considering alternative persons (RP+RD+AL), adding locker boxes improves the solution only slightly with benefits up to 2.1% and 1.6% on average. Note, that for RP+RD(+LB) and RP+RD+AL(+LB) the percentages in the table are calculated by comparing the averages over all instance types (25/57/75/100%).

The relevance of an alternative recipient

An alternative recipient is defined as a separate person with his or her own home location, itinerary and roaming delivery locations. Alternative persons can only be included for the delivery of a parcel, never for a pickup, since the goods cannot be simultaneously in two places at once. If an alternative person is included, all his or her roaming locations are included as potential delivery options. However, alternative persons never have a locker box assigned, because we fear that the motivation to pick up a parcel that does not concern them directly is low and prone to be forgotten. Therefore, for instances including both alternative persons and locker boxes, only the primary recipient might have a locker box. When including alternative persons as an option for delivery, the solutions are on average 14.7% better. If both locker boxes and alternative persons are included, the benefit of having an additional person diminishes somewhat and the results are on average only 11.8% better than when not including their additional locations. Also, the improvements prove quite stable across different instance sizes.

The relevance of roaming pickup and delivery locations

Reyes et al. [135] report cost savings up to 40% for their realistic instances with a depot in the southern part of the city. They also report, that the most benefits can be achieved in cities with a small number of work locations clustered together and dispersed residential locations. In Vienna,

where our instances are based, the population density is highest in the more central districts (see the Figure 4.4 in the Appendix), while the workplaces are scattered throughout the city. It is therefore not surprising that our cost savings are not nearly as high, even though the depot of the Austrian logistics provider is located in the southern outskirts of the city. Another possibility is, that cost savings are diminished when addressing a PDP instead of a VRP. Comparing the scenario of roaming locations (RP+RD) with the scenario of only home delivery within the time windows (H+TW), including roaming locations leads to improvements up to 15.2% and 9.8% on average over all instance sizes. When making the same comparison including locker boxes, the advantage is smaller with 6.3% on average and 9.4% at most. Nonetheless, even in a mixed city like Vienna, the potential cost improvements are obviously present and can be highly relevant when considering real world scenarios.

The relevance of partial inclusion of flexible requests

The assumption that all sender and receiver of parcels are willing to share their itinerary is not prudent. No matter how high the convenience, it is necessary to take mixed customer profiles into account where both data-sharing customers and data-withholding customers have to be served. Therefore, we also test if only a certain percentage of all requests allow data sharing, roaming locations, or alternative persons. The percentages used are 25, 50, 75 or 100% and are given in parentheses for each instance type. Our results show, that even when only 25% of customers participate in the more convenient system, the potential cost savings for the transportation carrier are up to 14.1% with an average of 11.6% over all instance sizes. When including more and more customers, both the average and the maximum cost savings are continuously growing. The maximum savings range from 14.1% to 21.5% and the average savings range from 11.6% to 17%. The smallest gap can be found for values between 75% and 100% which indicates that it might not be necessary for all customers to participate in the system for the carrier to reap the cost benefits.

4.4.3 Comparison with Reyes et al. [135]

We compare our work to Reyes et al. [135] since their paper on vehicle routing with roaming delivery locations (VRPRDL) is a special case of our problem. For the VRPRDL, the pickups all take place at the depot and the time windows of the delivery locations are not overlapping. They examine the potential of trunk deliveries where delivery personnel have a one-time permission to open a customers car trunk to deliver the parcel. This concept is built on the premise that cars spend most of their day parked somewhere while the owner is at work, shopping, or at home. They generated 40 'general' instances with the number of customers ranging from 15 to 120. Each customer has up to six roaming delivery locations, with the first always being the home location. In addition to the location coordinates, they provide a travel time matrix that are not the Euclidean distances between the coordinates. They provide multiple sets of instances, however for our comparison we choose the ones where the triangle inequality is fulfilled. The OFVs of Reyes et al. [135] were obtained from their paper, Table 2, c_{heur}.

The results are presented in Table 4.5. We compare the OFVs obtained by Reyes et al. [135] '$\text{OFV}_{\text{Reyes}}$' with the values obtained by our algorithm 'OFV'. The value of ΔR is the difference between our results and their heuristic and is calculated by $\frac{OFV - OFV_{Reyes}}{OFV_{Reyes}}$.

The table is divided in two parts. The first part shows the runtime needed to surpasses the results reported by Reyes et al. [135]. If we terminate our algorithm as soon as we reach the objective function value $\text{OFV}_{\text{Reyes}}$, the solutions found are on average -0.9% better (smaller) with a minimum improvement of -8.9% and a median improvement of -0.2%. The runtime is at least 0.9 seconds and up to 5160 seconds (slightly less than 90 minutes), with an average of 272 seconds (around 4.5 minutes), and a median of 28 seconds. Since Reyes et al. [135] do not report their runtimes (only an iteration limit is reported), a more precise comparison is not possible.

	30 req.	50 req.	100 req.	Avg.
Comparison to 'traditional' systems				
H+TW vs Full Flexibility	29.6%	22.9%	22.2%	23.7%
H+TW+LB vs Full Flexibility	22.1%	17.4%	14.9%	16.9%
H vs Full Flexibility	12.1%	8.6%	5.2%	7.4%
The relevance of locker boxes				
H+TW vs H+TW+LB	6.2%	4.7%	6.4%	5.8%
Avg RP+RD vs Avg RP+RD+LB	3.0%	4.3%	4.5%	4.2%
Avg RP+RD+AL vs Avg RP+RD+AL+LB	2.1%	0.4%	2.1%	1.6%
The relevance of alternative recipients				
Avg. RP+RD vs Avg. RP+RD+AL	14.7%	14.3%	14.8%	14.7%
Avg. RP+RD+LB vs Avg. RP+RD+AL+LB	13.8%	10.1%	12.1%	11.8%
The relevance of roaming locations				
H+TW vs RP+RD(100)	15.2%	10.2%	7.8%	9.8%
H+TW+LB vs RP+RD+LB(100)	9.4%	6.1%	5.3%	6.3%
The relevance of partial inclusion of convenience				
H+TW to avg. all(25)	14.1%	10.2%	11.4%	11.6%
H+TW to avg. all(50)	17.5%	12.1%	12.7%	13.4%
H+TW to avg. all(75)	21.4%	13.2%	14.1%	15.1%
H+TW to avg. all(100)	21.5%	17.2%	15.4%	17.0%

Table 4.4: Explicit comparison between different scenarios.

The second part of the table reports our results with a longer runtime of at most 3600 seconds (one hour). The runtime depends on the instance size to allow bigger instances a longer runtime and ranges between 7.5 minutes for the small instances with 15 requests to 60 minutes for the large instances. Our solutions are on average -1.8% better (smaller) with a minimum improvement of -8.9% and a median improvement of -0.8%. We also have some instances where we did not find a better solution than Reyes et al. [135], at most we are 0.7% worse.

4.4.4 Comparison with Ozbaygin et al. [119]

Furthermore we compare our work with Ozbaygin et al. [119]. They worked on the same problem as Reyes et al. [135] and extended it to include an all-day home delivery option by replacing the time windows of the home location with ones that span all day. They used an exact solution method, namely a branch-and-price algorithm on their own instances and on the instances provided by Reyes et al. [135]. Therefore they made it possible for us to compare our heuristic to the optimal solutions for most of the instances. For the biggest instances with 120 requests they reached their runtime limit (two hours for instances up to 60 customers, and six hours for instances with 120 customers). The OFV of Ozbaygin et al. [119] were obtained from their paper, Table 9, column VRPRDL. ΔO is the difference between our results and theirs and is calculated by $\frac{OFV - OFV_{Ozbaygin}}{OFV_{Ozbaygin}}$.

The computational results can be found in Table 4.6. We used two runtime multipliers and the runtimes were therefore between one and 60 minutes. When comparing to the branch-and-price algorithm, we find that, for the short runtimes, we are on average 1.33% worse with a median of 0.14% and 7.84% at most. For the longer runtimes we are on average 0.77% worse, with a median of 0.08% and deviate at most by 5.66%.

Note that although the same instances were used for both comparisons the OFVs are vastly different because Reyes et al. [135] used a different objective cost function than Ozbaygin et al. [119]. For the cost calculation Reyes et al. [135] used the provided travel time matrix where the travel time of each arc is roughly twice its Euclidean distance. Ozbaygin et al. [119] used the Euclidean distances rounded to the nearest integer for evaluating the solution cost. Furthermore, Reyes et al. [135] and Ozbaygin et al. [119] do not have the same numbering of instances reported in their tables. We chose to keep the nomenclature and sorting of Reyes et al. [135]. A conversion key can be found in the Appendix, Table 4.10.

4.5 Conclusions

Adequate convenience is especially lacking in the C2C market where customers interact directly and goods are both picked up and delivered by transportation carriers. Including the possibility to add both roaming locations for pickup and deliveries as well as an alternative recipient and even 24-hour locker boxes, makes for a competitive and comfortable system with advantages for the customers and carrier alike. There are big potential improvements with up to almost 30% in cost savings when comparing more traditional home delivery system with more flexible ones. Even though the benefits of locker boxes are small (up to 6.4%) in comparison to alternative recipients or roaming locations, it is the option most easily realized. Partly because the locker boxes are already build and in use, and partly because the only variability in success depends on their capacity. Including roaming locations or alternative persons enable the carrier to drive shorter tours but in reality they carry a risk. Even well-meaning and cooperative customers cannot guarantee their itineraries and changes on short notice means that the customers are unavailable for service at the planned time. To avoid missed pickups or deliveries other methods can be included to extend this deterministic problem. For example, Florio et al. [57] use different realistic availability profiles to represent the likelihood of a customer being available for receiving a delivery at their home location to maximize successful deliveries. This premise can be extended

Instance	#req.	# loc.	OFV_{Reyes}	time(s)	OFV	Δ R	time(s)	OFV	Δ R
1	15	51	2128	15.0	2128	0.0%	450	2128	0.0%
2	15	53	2007	6.7	1984	-1.1%	450	1984	-1.1%
3	15	53	3661	15.0	3661	0.0%	450	3661	0.0%
4	15	58	2582	1.5	2573	-0.3%	450	2572	-0.4%
5	15	63	1802	15.0	1801	-0.1%	450	1801	-0.1%
6	20	64	3374	1.1	3374	0.0%	600	3374	0.0%
7	20	67	2588	20.0	2588	0.0%	600	2588	0.0%
8	20	69	2489	2.3	2310	-7.2%	600	2310	-7.2%
9	20	77	2536	0.9	2521	-0.6%	600	2521	-0.6%
10	20	81	3196	28.6	2912	-8.9%	600	2912	-8.9%
11	30	76	3659	4.2	3646	-0.4%	900	3646	-0.4%
12	30	99	4173	3.5	4173	0.0%	900	4172	0.0%
13	30	104	3849	30.0	3849	0.0%	900	3849	0.0%
14	30	107	3668	1.7	3667	0.0%	900	3658	-0.3%
15	30	108	2548	22.1	2544	-0.2%	900	2543	-0.2%
16	30	114	4695	3.6	4690	-0.1%	900	4654	-0.9%
17	30	119	3507	25.0	3497	-0.3%	900	3491	-0.5%
18	30	120	3877	25.1	3877	0.0%	900	3875	-0.1%
19	30	125	3397	28.9	3392	-0.1%	900	3388	-0.3%
20	30	131	3939	30.0	3934	-0.1%	900	3934	-0.1%
21	60	209	6049	40.7	5848	-3.3%	1800	5750	-4.9%
22	60	214	5873	36.0	5818	-0.9%	1800	5661	-3.6%
23	60	220	8391	55.3	8374	-0.2%	1800	8365	-0.3%
24	60	226	7670	26.4	7656	-0.2%	1800	7562	-1.4%
25	60	226	9218	38.0	9121	-1.1%	1800	9090	-1.4%
26	60	227	8057	10.1	8012	-0.6%	1800	7942	-1.4%
27	60	230	7032	21.4	6937	-1.4%	1800	6778	-3.6%
28	60	235	6434	60.1	6400	-0.5%	1800	6377	-0.9%
29	60	236	8971	331.8	8956	-0.2%	1800	8909	-0.7%
30	60	239	8428	10.1	8280	-1.8%	1800	8225	-2.4%
31	120	423	13414	273.0	13366	-0.4%	3600	13210	-1.5%
32	120	423	11434	96.3	11327	-0.9%	3600	11279	-1.4%
33	120	429	12987	41.5	12822	-1.3%	3600	12396	-4.6%
34	120	442	11379	148.8	11357	-0.2%	3600	11176	-1.8%
35	120	452	11713	600.1	11586	-1.1%	3600	11136	-4.9%
36	120	456	11374	146.9	11104	-2.4%	3600	10902	-4.1%
37	120	462	10516	322.8	10475	-0.4%	3600	9949	-5.4%
38	120	463	11045	66.9	11033	-0.1%	3600	10372	-6.1%
39	120	468	10115	5160.1	10094	-0.2%	3600	10186	0.7%
40	120	472	10492	3111.3	10481	-0.1%	3600	10480	-0.1%
Min.				0.9		-8.9%			-8.9%
Avg.				272.0		-0.9%			-1.8%
Med.				27.5		-0.2%			-0.8%
Max.				5160.1		0.0%			0.7%

Table 4.5: Comparison to Reyes et al. [135]. The table is divided in three parts: the left part gives information about the instances, the middle part reports the computational time 'time(s)' needed until our algorithm surpasses the results from the heuristic of Reyes et al. [135], and the right part shows the results we are able to obtain when setting a fixed runtime limit of $\mu = 30$. The columns are labeled as follows: 'Instance' is the instance number (from 1 to 40) that corresponds to the number reported in Reyes et al. [135] '#req.' shows the number of requests and '#loc.' shows the number of locations each instance has. 'OFV_{Reyes}' reports the original objective function value from Reyes et al. [135] 'OFV' is the absolute objective function value obtained by our algorithm and the difference in percent is given by 'Δ R' with negative values signifying an improvement (a smaller) objective function value.

Instance	#req.	# loc.	$OFV_{Ozbaygin}$	time(s)	OFV	Δ O	time(s)	OFV	Δ O
1	15	51	1062	60	1062	0.00%	450	1062	0.00%
2	15	53	991	60	991	0.00%	450	991	0.00%
3	15	53	1832	60	1832	0.00%	450	1832	0.00%
4	15	58	1286	60	1286	0.00%	450	1286	0.00%
5	15	63	901	60	902	0.11%	450	902	0.11%
6	20	64	1684	80	1684	0.00%	600	1684	0.00%
7	20	67	1294	80	1294	0.00%	600	1294	0.00%
8	20	69	1155	80	1155	0.00%	600	1155	0.00%
9	20	77	1260	80	1260	0.00%	600	1260	0.00%
10	20	81	1455	80	1456	0.07%	600	1456	0.07%
11	30	76	1822	120	1823	0.05%	900	1823	0.05%
12	30	99	2083	120	2083	0.00%	900	2083	0.00%
13	30	104	1922	120	1922	0.00%	900	1922	0.00%
14	30	107	1827	120	1827	0.00%	900	1827	0.00%
15	30	108	1273	120	1274	0.08%	900	1274	0.08%
16	30	114	2324	120	2327	0.13%	900	2325	0.04%
17	30	119	1747	120	1751	0.23%	900	1748	0.06%
18	30	120	1938	120	1938	0.00%	900	1938	0.00%
19	30	125	1694	120	1697	0.18%	900	1696	0.12%
20	30	131	1965	120	1965	0.00%	900	1965	0.00%
21	60	209	2865	240	2871	0.21%	1800	2869	0.14%
22	60	214	2828	240	2832	0.14%	1800	2830	0.07%
23	60	220	4173	240	4185	0.29%	1800	4180	0.17%
24	60	226	3761	240	3785	0.64%	1800	3777	0.43%
25	60	226	4536	240	4548	0.26%	1800	4540	0.09%
26	60	227	3964	240	3969	0.13%	1800	3968	0.10%
27	60	230	3378	240	3386	0.24%	1800	3385	0.21%
28	60	235	3161	240	3194	1.04%	1800	3180	0.60%
29	60	236	4440	240	4471	0.70%	1800	4447	0.16%
30	60	239	4107	240	4112	0.12%	1800	4108	0.02%
31	120	423	6498	480	6663	2.54%	3600	6585	1.34%
32	120	423	5608	480	5656	0.86%	3600	5636	0.50%
33	120	429	5849	480	6253	6.91%	3600	6180	5.66%
34	120	442	5278	480	5662	7.28%	3600	5573	5.59%
35	120	452	5519	480	5780	4.73%	3600	5555	0.65%
36	120	456	5218	480	5520	5.79%	3600	5427	4.01%
37	120	462	4935	480	5186	5.09%	3600	4959	0.49%
38	120	463	5048	480	5195	2.91%	3600	5181	2.63%
39	120	468	4845	480	5225	7.84%	3600	5082	4.89%
40	120	472	5083	480	5328	4.82%	3600	5215	2.60%
Min.						0.00%			0.00%
Avg.						1.33%			0.77%
Med.						0.14%			0.08%
Max.						7.84%			5.66%

Table 4.6: Comparison to Ozbaygin et al. [119] that provided an exact method and optimal solutions for the instances from Reyes et al. [135]. The table is divided in three parts. The left part of the table gives an overview over the instances: 'Instance' is the instance number (from 1 to 40) that corresponds to the number reported in Reyes et al. [135] '#req.' shows the number of requests each instance has and '#loc.' shows the number of locations each instances has. '$OFV_{Ozbaygin}$' reports the original objective function values from Ozbaygin et al. [119]. The middle and right part of the table report our results: 'OFV' is the absolute objective function value obtained by our algorithm and the difference in percent is given by 'Δ O' with negative values signifying an improvement (a smaller) OFV. The runtime multiplier for the middle table is $\mu = 4$ and for the right table $\mu = 30$.

to other locations outside of the home as well and would make a suitable extension by adding stochasticity. Additionally, not all customers might be willing to share their itineraries, live GPS data or historical movement patterns. For this case, we analyzed the benefits of having mixed customer preferences where the percentage of customers opting for convenience was set to 25, 50, 75 or 100%. We showed that even with just 25% of customers opting in on the system, it is profitable to allow for roaming locations and alternative recipients.

4.6 Appendix

		Employed full-time			
EF1	H 05:30 07:30	W 08:00 16:00	H 16:30 21:00		
EF2	H 05:30 07:30	O 08:00 09:00	W 09:30 18:00	H 18:30 21:00	
EF3	H 05:30 08:00	W 08:30 17:00	O 17:30 20:00	H 20:30 21:00	
EF4	H 05:30 07:30	W 08:00 16:00	O 16:30 19:30	H 20:00 21:00	
EF5	H 05:30 10:30	W 11:00 19:30	H 20:00 21:00		
EF6	H 05:30 08:30	O 09:00 10:30	W 11:00 19:30	H 20:00 21:00	
		Employed part-time			
EP1	H 05:30 06:00	W 06:30 12:00	H 12:30 15:30	O 16:00 18:00	H 18:30 21:00
EP2	H 05:30 07:30	W 08:00 14:00	O 14:30 17:30	H 18:30 21:00	
EP3	H 05:30 11:30	W 12:00 18:00	H 18:30 21:00		
EP4	H 05:30 11:30	W 12:00 18:00	O 18:30 21:00		
EP5	H 05:30 09:00	O 09:30 11:30	W 12:00 18:00	H 18:30 21:00	
EP6	H 05:30 13:00	W 14:00 20:00	H 20:30 21:00		
EP7	H 05:30 09:30	O 10:00 13:30	W 14:00 20:00	H 20:30 21:00	
EP8	H 05:30 09:30	O 10:00 12:30	O 13:00 15:30	W 16:00 21:00	
		Unemployed			
UN1	H 05:30 21:00				
UN2	H 05:30 09:30	O 10:00 14:00	H 14:30 21:00		
UN3	H 05:30 13:00	O 14:00 18:00	H 18:30 21:00		

Table 4.7: Customer profiles for full-time employed, part-time employed and unemployed persons. Complete daily itinerary for each profile given with 'H' meaning 'Home', 'W' meaning 'Work' and 'O' meaning 'Other' roaming locations.

Sector	% living	% working
1	0.009	0.111
2	0.056	0.068
3	0.048	0.103
4	0.018	0.029
5	0.029	0.021
6	0.017	0.029
7	0.017	0.034
8	0.014	0.016
9	0.023	0.051
10	0.107	0.078
11	0.054	0.038
12	0.052	0.039
13	0.029	0.025
14	0.049	0.030
15	0.042	0.030
16	0.055	0.029
17	0.030	0.015
18	0.027	0.015
19	0.038	0.033
20	0.046	0.030
21	0.086	0.057
22	0.099	0.064
23	0.055	0.055
Sum	1.000	1.000

Table 4.8: Probabilities and percentage of population living and working in each sector in Vienna. Data provided by the magistrate MA23 [97, 98] from Statistik Austria statistik.at

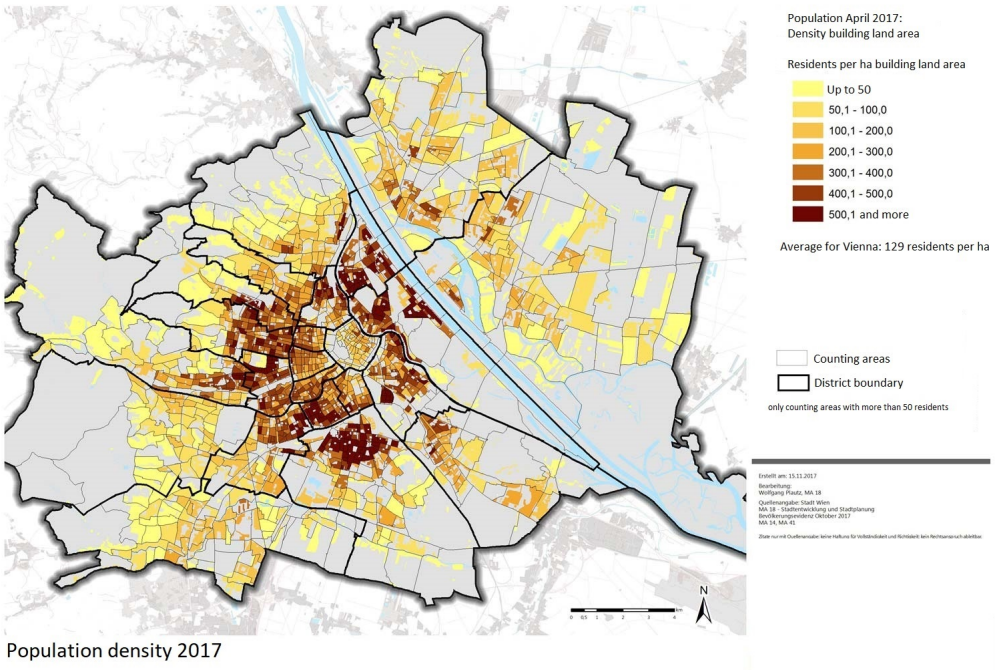


Figure 4.4: Depiction of population density from 2017 in Vienna. Original picture provided by the municipality MA 18 of Vienna [154], translated to English.

Parameter	Meaning	Value
$\|P\|$	Size of initial solution pool	10
ρ_{min}	Min. probability of assigning a random vehicle for a request during the construction heuristic	0.01
ρ_{max}	Max. probability of assigning a random vehicle for a request during the construction heuristic	0.04
$\#r$	Number of requests. The value depends on the instance	–
μ	Runtime multiplier. The value is variable and indicated for each table separately.	–
c_L	Iteration limit for the ALNS. If after c_L iterations no further improvement could be found, the ALNS is terminated.	50
c_w	Iteration limit without improvement after which the threshold w to accept a worse solution is increased	$\frac{1}{2}c_L$
$c_{\gamma_{max}}$	Iteration limit without improvement after which upper destroy limit γ_{max} for a solution is doubled	$\frac{3}{4}c_L$
w_{max}	How much worse the incumbent solution is at most allowed to be in relation to the best solution	1.5
w'	By how much the parameter for a worse solution acceptance changes if no solution could be found	0.05
γ_{min}	Minimum percentage of solution to be destroyed	0.1
γ_{max}	Maximum percentage of solution to be destroyed	0.4
φ_c	Weight of the travel time for calculating the insertion cost of a request into a tour	1
φ_w	Weight of the waiting time for calculating the insertion cost of a request into a tour	0.2

Table 4.9: Table of all parameters used for the computational results. The standard values are given. If any values deviate for specific computational results, the changes are reported in the respective tables.

Reyes et al. [135]	Ozbaybin et al. [119]	File numbering
1	4	3
2	3	2
3	5	4
4	2	1
5	1	0
6	10	9
7	6	5
8	7	6
9	9	8
10	8	7
11	20	19
12	19	18
13	11	10
14	18	17
15	14	13
16	12	11
17	13	12
18	16	15
19	15	14
20	17	16
21	27	26
22	22	21
23	28	27
24	21	20
25	26	25
26	29	28
27	24	23
28	25	24
29	23	22
30	30	29
31	36	35
32	38	37
33	39	38
34	32	31
35	35	34
36	34	33
37	31	30
38	40	39
39	37	36
40	33	32

Table 4.10: A conversion key for the numbering of the instances. For example: The same instance is numbered '1' int the paper by Reyes et al. [135], '4' in the paper by Ozbaygin et al. [119] and '3' for the file. We chose to keep the numbering of the instances from Reyes et al. [135]. Therefore our objective function value reported of instance '1' needs to be compared to instance '4' in the work of Ozbaygin et al. [119]. The file numbering is provided as well to avoid confusion for future uses and comparisons of those instances.

Chapter 5

A comparison of heuristics for the pickup and delivery problem with alternative locations

Technical Report
A comparison of heuristics for the PDP with alternative locations.
A.G. Dragomir

Abstract This work aims to provide insights into the efficiency of different solution algorithms for the pickup and delivery problem with alternative locations (PDPAL). For an identical general solution approach, namely a decomposition of the problem into a request-to-vehicle assignment and a routing algorithm, different heuristics for both parts are compared to each other. For solving the vehicle routing problem we examine five construction heuristics: a nearest neighbor algorithm (NN), a modified nearest neighbor algorithm incorporating penalties for waiting time (NN_w), a modified farthest insertion heuristic (FI), a modified smallest insertion heuristic (SI), and a combination of farthest and smallest insertion heuristic (FIf). The methods are evaluated by their ability to generate feasible solutions, and by the overall solution quality. The potential of local search is examined, as well as the importance of considering waiting time when applying the methods to a PDPAL with time windows. For the request-to-vehicle assignment we compare an Adaptive Large Neighborhood Search (ALNS), a Genetic Algorithm (GA), a multi-start ALNS (MS-ALNS), and a combination of GA and ALNS (GA-ALNS). From our results we conclude the MS-ALNS to lead to the best solutions with potential for further improvement for longer run times with the GA-ALNS as a close second.

5.1 Introduction

Transportation research problems have been widely studied in the literature, with the most common being Vehicle Routing Problems (VRP) and Pickup and Delivery Problems (PDP). Both, VRPs and PDPs, have a wide variety of applications and the research interest has lead to it being covered in several surveys: Savelsbergh and Sol [146], Berbeglia et al. [14], Cordeau et al. [30], and Parragh et al. [122]. Additionally, the broad range of applications for VRPs and PDPs have lead to many different definitions and variants, with the most recent being VRP with roaming delivery locations (VRPRDL) [96, 119, 120, 135, 144] and the PDP with alternative locations (PDPAL) [44].

The solution methods applied for both VRP and PDP range from exact methods to heuristics (e.g. constructive heuristics, local-improvement heuristics, metaheuristics, mathheuristics, decomposition approaches, population-based methods) with both, exact and heuristic methods, having their legitimacy. For an analysis of several heuristics for the traveling salesman problem

(TSP) see the work by Rosenkrantz [140]. For a pedagogical introduction of heuristics for VRPs the work by Røpke [138] can be recommended as well as Vidal et al. [161]. For a comprehensive introduction to metaheuristics the reader is referred to Zäpfel et al. [168] and to Rothlauf [141].

For newer problems, like VRPRDL or PDPAL, both exact and heuristic methods exist [44, 119, 135]. However, when approaching realistic instance sizes, exact methods often reach their limit. Also, since both problems were defined quite recently, not many solution methods have been applied to them. Especially for the PDPAL there is, to the best of our knowledge, little literature regarding the efficiency of different solution algorithms. This work attempts to close this gap insofar as to provide comparisons of different heuristics for the same general solution approach, namely a decomposition approach with a request-to-vehicle assignment and an underlying routing heuristic.

The rest of this paper is structured as follows: Section 5.2 provides a problem definition. Section 5.3 presents different solution algorithms for the routing problem which are then compared amongst each other in Section 5.4. Section 5.5 presents different metaheuristics for the request-to-vehicle assignment and Section 5.6 reports the computational experiments of the metaheuristics. The conclusion can be found in Section 5.7.

5.2 Problem definition

For the PDPAL, we have a set of transportation requests. Each request has a seller (from whom the parcel is picked up), a buyer (to whom the parcel is delivered), and a quantity.

We assume that on the day of pickup the parcel stays with the seller and can be picked up wherever they are. We also assume that we know the itinerary, that is the whereabouts, of both the seller and the buyer on the day the service occurs. The itinerary specifies the time windows for each location. Each location the seller visits on their itinerary corresponds to a (roaming) pickup location with non-overlapping time windows. Each location the buyer visits corresponds to a (roaming) delivery location with non-overlapping time windows. The buyer can specify an additional person to receive their parcel. This person has its own itinerary with their own roaming delivery locations. Additionally, the buyer has the option to have the parcel delivered to a 24-hour locker box for self-pickup. From the perspective of the buyer, these locations are alternative locations with overlapping time windows to their own locations.

For each request, we therefore have roaming pickup locations, alternative delivery locations and a fixed quantity. Figure 5.1 shows an example where request A has three (possible) pickup locations and request B has three (possible) delivery locations. A choice has to be made which of the alternative locations to include in the tour. The arrows depict the tour of the vehicle. For request B, the nodes nearest to the depot are selected, however for request A a more distant pickup node is selected due to time window restrictions (time windows are not depicted). Figure 5.2 shows an example for the time windows of a request with roaming pickups and alternative deliveries. There are three possible delivery locations with overlapping time windows.

Figure 5.1: VRPTW with multiple pickup and delivery locations

All requests have to be served by a fleet of homogeneous vehicles that start and end their tour at a single depot. All vehicles have a maximum tour duration and have to return to the

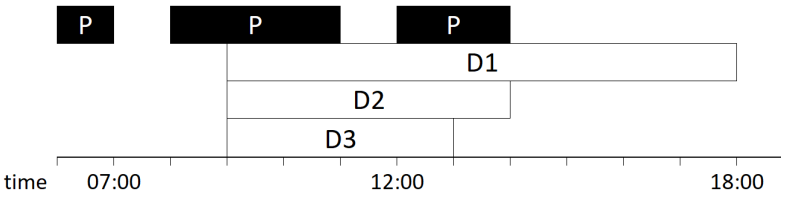


Figure 5.2: Possible time windows for a request with roaming pickups and alternative deliveries

depot before the end of day. For our instances, we assume a time frame (depot opening hours) of a working day of 16.5 hours, starting at 5:00 a.m. and ending at 9:30 p.m., the smallest unit of time is one minute. The goal is to select exactly one of the pickup and one of the delivery location possibilities for each request and to determine the node sequence for all vehicles while minimizing travel time and while fulfilling time window and capacity constraints.

For the mathematical problem formulation and the detailed instance descriptions we refer the interested reader to Dragomir et al. [44].

The problem was decomposed into two parts: First, the assignment of requests to vehicles, and second, the determination of the pickup and delivery locations as part of the routing algorithm that also determines the node sequence. The following sections will examine in detail different construction heuristics for the routing problem (with and without local search) and metaheuristics for the overall problem.

5.3 Solution algorithms for the routing problem

The routing algorithm is split into a construction heuristic and an improvement heuristic. For constructing the initial routes, we compare five different methods: a nearest neighbor algorithm (NN), a modified nearest neighbor algorithm incorporating penalties for waiting time (NN_w), a modified farthest insertion heuristic (FI), a modified smallest insertion heuristic (SI), and a combination of farthest and smallest insertion heuristic (FIf). The heuristics are adaptations from the standard traveling salesman application [26, 69] to incorporate paired pickup and delivery nodes and time windows. All insertion heuristics include a penalty for waiting time. The node sequence is determined for each vehicle separately and all vehicles start empty.

The constructed tour is improved by local search using *move* and *swap* operators embedded in a variable neighborhood descent (VND) [106] framework.

5.3.1 Construction heuristic NN and NN_w

The construction heuristics NN and NN_w have been adapted for a PDP. They distinguish between pickup and delivery nodes, and take precedence constraints into account. Both, NN and NN_w, are in principle identical, except that NN_w considers a penalty for waiting time, which is given as a weight for influencing the determined routing cost. When the waiting time weight φ_w is zero, NN and NN_w lead to the same solution. Therefore, both heuristics are described with the same pseudo-code. Algorithm 1 shows two procedures: the construction heuristic named Nearest neighbor and the function Get nearest feasible neighbor that describes in detail how the next node is selected.

The procedure Nearest neighbor works as follows: starting from an empty route, all possible nodes that can be visited are added to the set L. Since we are working with paired pickups and deliveries, the first iteration adds all pickup locations of all requests assigned to this vehicle to the set L. One of those locations is selected as the next node to visit with the function Get nearest feasible neighbor and this location is appended to the tour. Now, the set L needs to be updated. The just selected pickup node belongs to a request. Since only one pickup is

required, all other pickup locations of this particular request are no longer available as a possible candidate to be visited next and are removed from the set L. However, now all delivery nodes of this particular request are available to be visited next and are added to the set L. In the next iteration, the next node to be visited can be either a pickup node of another request, or a delivery node of the same request. The iterative approach adds and removes locations from the set L depending on the last selected node. Any time a pickup node is selected the aforementioned steps are taken. Any time a delivery node is selected, all other delivery nodes of the same request are removed from the set L and none are added. The procedure continues until the set L is empty, which means that for each request one pickup and one delivery location has been chosen and the node sequence has been established.

For determining which of the possible locations present in set L to select, the function GET NEAREST FEASIBLE NEIGHBOR is called. A cost c_j is calculated for each location j in L by

$$c_j = d_{ij} + \varphi_w k(\sigma_j - t_i) \tag{5.1}$$

with i as the previous visited node, j the current possible next location, d_{ij} as the distance and cost between i and j, φ_w as the waiting time weight, t_i as the departure time from location i, σ_j as the earliest service time at j and calculated by

$$\sigma_j = max(t_j, a_j) \tag{5.2}$$

with t_j as the arrival time at location j and a_j as the time window opening of j. The variable k is calculated by

$$k = \frac{\sigma_j - t_i}{\bar{t}_{rest}} \tag{5.3}$$

and indicates the necessary time allocation for this location.

The objective of this calculation is the following: When considering the next node to visit in a tour in a problem with time windows, no general assertion can be made whether waiting time is disadvantageous or inconsequential. In the case of a low workload, that is, few requests need to be served by a vehicle, waiting time might not be relevant enough to be considered, especially if the remaining time to serve them is long enough. In the case of a high workload, every wait increases the probability of not finishing the tour in time and, as a result, could lead to an infeasible solution. When calculating the cost c_j of visiting the next node j, we determine the remaining time this vehicle has for its tour by calculating $t_{rest} = E - t_i$, with E denoting the end of day and t_i as the departure time from location i. Then, we calculate the average remaining time per node, by dividing the remaining time by the number of remaining nodes to visit $\bar{t}_{rest} = \frac{t_{rest}}{n_{rest}}$. This is then used for calculating the time allocation k.

The difference between the two methods can be illustrated using a small example with one depot '0' and three nodes 'A','B' and 'C'. Table 5.1 provides the distance and cost matrix d_{ij} and the time windows. For our example we assume an end of day E of 100 and a waiting time weight φ_w of 0.5. Figures 5.3 to 5.6 show the step by step construction of a route and the calculations in a table below it. All nodes have their time windows a_j and b_j depicted above them (e.g. from zero to 30 for node 'A') and the arrival time t_j and start of service σ_j depicted below them in square brackets.

In Figure 5.3 both calculations select node 'A' to be inserted first. In Figure 5.4 the algorithms diverge. The NN selects node 'C' because the distance is only five. The arrival time t_j at 'C' is 15, however the start of service σ_j is 85 due to the time windows. NN_w selects node 'B' because it considers the waiting times although with a distance of 30 it is much farther away than node 'C'. Here, the arrival time t_j at 'B' is 40, and the start of service σ_j is 50. Figure 5.5 shows the selection and calculation of the next node. For both examples only one node remains to be inserted. NN attempts to insert node 'B'. However, after the long wait at the previous node, it arrives past the time window closing b_j and a service is no longer possible. NN_w successfully

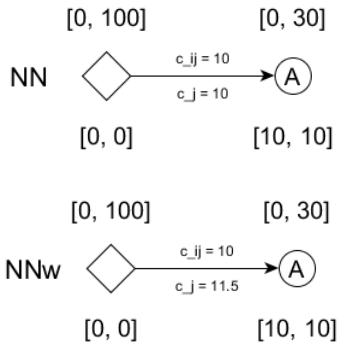
Algorithm 1 (Modified) Nearest Neighbor (NN and $\mathrm{NN_w}$)

procedure NEAREST NEIGHBOR(waiting time weight φ_w, set of requests assigned to vehicle R_v)
  $L \leftarrow \emptyset$ ▷ initialize the set of possible nodes to visit
  $tour \leftarrow \{0\}$ ▷ Vehicle tour gets initialized with node 0, the depot
  $t_i \leftarrow 0$ ▷ departure time from the depot is set to 0
  $s_i \leftarrow 0$ ▷ load when leaving depot is set to 0
  $n_{rest} \leftarrow |R_v|$ ▷ remaining nodes to be served
  for all $r \in R_v$ **do** ▷ for all requests assigned to this vehicle
    $L \leftarrow L \cup N_r^P$ ▷ all pickup nodes are added to the set L
  while $|L| > 0$ **do**
    $N \leftarrow$ GET NEAREST FEASIBLE NEIGHBOR$(\varphi_w, L, t_i, s_i, n_{rest})$
    if $N = -1$ **then** ▷ if no feasible location could be found
      break
    $tour \leftarrow tour \cup N$ ▷ append node to tour
    $n_{rest} \leftarrow n_{rest} - 1$
    if $N \in \bigcup_{r \in R_v} N_r^P$ **then** ▷ if node was a pickup node
      $L \leftarrow L \setminus N_r^P$ ▷ remove all other pickup nodes of r from L
      $L \leftarrow L \cup N_r^D$ ▷ add all delivery nodes of r to L
    else if $N \in \bigcup_{r \in R_v} N_r^D$ **then** ▷ if node was a delivery node
      $L \leftarrow L \setminus N_r^D$ ▷ remove all delivery nodes of r to L
function GET NEAREST FEASIBLE NEIGHBOR(waiting time weight φ_w, set of locations L, departure time from previous location t_i, quantity at departure from previous location s_i, remaining nodes to be served n_{rest})
  $c_{min \leftarrow M}$ ▷ initialize smallest cost to a big number M
  $l_{min \leftarrow -1}$ ▷ initialize best neighbor
  $t_{rest} \leftarrow E - t_i$ ▷ remaining time for this vehicle
  $\bar{t}_{rest} \leftarrow \frac{t_{rest}}{n_{rest}}$ ▷ average remaining time per node
  for all $j \in L$ **do**
    $t_j \leftarrow t_i + c_{ij}$ ▷ arrival time at location j
    if $t_j > b_j$ **then** ▷ If arrival time exceeds time window at location j
      continue
    $s_j \leftarrow q_i + q_j$ ▷ quantity at location j
    if $s_j > Q$ **then** ▷ If quantity exceeds vehicle capacity
      continue
    $\sigma_j \leftarrow max(t_j, a_j)$ ▷ earliest service time at j
    $k \leftarrow \frac{\sigma_j - t_i}{\bar{t}_{rest}}$ ▷ time allocation k
    $c_j \leftarrow d_{ij} + \varphi_w k(\sigma_j - t_i)$ ▷ calculate cost
    if $c_j < c_{min}$ **then**
      $c_{min} \leftarrow c_j$
      $l_{min} \leftarrow j$
  return l_{min}

d_{ij}	0	A	B	C
0	0	10	31.6	11.2
A	10	0	30	5
B	31.6	30	0	30.4
C	11.2	5	30.4	0

TW	0	A	B	C
a_j	0	0	50	85
b_j	100	30	100	100

Table 5.1: The distance and cost matrix d_{ij}, and the time windows for a small example



	t_i	t_{rest}	n_{rest}	$\bar{t}_{rest}$	t_j	a_j	σ_j	k	d_{ij}	c_j
A	0	100	3	33.3	10.0	0	10	0.3	**10.0**	**11.5**
B	0	100	3	33.3	31.6	50	50	1.5	31.6	69.1
C	0	100	3	33.3	11.2	85	85	2.6	11.2	119.6

Figure 5.3 & Table 5.2: Example of NN and NN_w, first node. Both algorithms insert node 'A'.

inserts node 'C'. Figure 5.6 shows the completion of the route and the return to the depot. For the NN algorithm, the vehicle arrives past the depot opening hours and the solution is therefore infeasible. The NN_w algorithm creates a feasible solution.

5.3.2 Construction heuristic FI, SI, and FIf

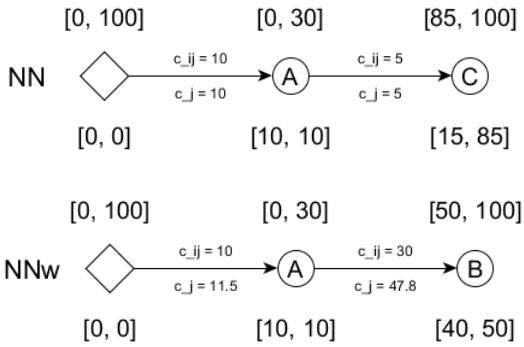
For all insertion based construction heuristics (FI, SI, and FIf) we need to calculate the insertions for a request. An *insertion* is defined by a request and a vehicle, a determined pickup location, a determined delivery location, the places of those locations within the (already existing) node sequence, and cost. The insertion cost $I_{i,j}$ for adding the pickup node i and delivery node j is calculated as follows:

$$I_{i,j} = d_{i-1,i} + d_{i,j} + d_{j,j+1} - d_{i-1,j+1} \tag{5.4}$$

with d_{ij} denoting the cost and distance between two nodes. The indices n_{i+1} and n_{j+1} denote the succeeding nodes in the tour from i and j respectively, while n_{i-1} and n_{j-1} denote the preceding nodes. If the tour is empty, the nodes n_{i-1} and n_{j+1} are both the depot. If the insertion of i and j is done in such a way that they are not sequential, the insertion cost is calculated as follows:

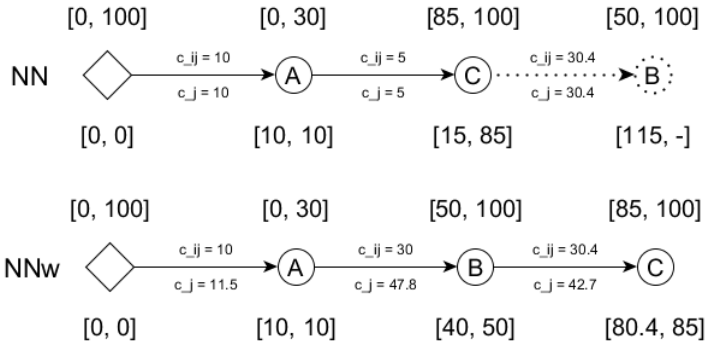
$$I_{i,j} = d_{i-1,i} + d_{i,i+1} + d_{j-1,j} + d_{j,j+1} - d_{i-1,i+1} - d_{j-1,j+1} \tag{5.5}$$

All insertion based construction heuristics use the same basic principle that is outlined in Algorithm 2. Also, they all use the same algorithm to create the list of initial insertions which are the insertion costs based on an empty tour (see Algorithm 3).



	t_i	t_{rest}	n_{rest}	$\bar{t}_{rest}$	t_j	a_j	σ_j	k	d_{ij}	c_j
B	10	90	2	45	40	50	50	0.9	30	**47.8**
C	10	90	2	45	15	85	85	1.7	**5**	67.5

Figure 5.4 & Table 5.3: Example of NN and NN_w, second node. The algorithms diverge: NN inserts node 'C' as the nearest node, NN_w inserts node 'B' due to the waiting time influencing the cost calculation.



	t_i	t_{rest}	n_{rest}	$\bar{t}_{rest}$	t_j	a_j	σ_j	k	d_{ij}	c_j
B	85	15	1	15	115.4	50	115.4	2.0	**30.4**	61.2
C	50	50	1	50	80.4	85	85.0	0.7	30.4	**42.7**

Figure 5.5 & Table 5.4: Example of NN and NN_w, third node. NN attempts to insert node 'B', however the arrival time is past the time window of 'B' and the solution is therefore infeasible. NN_w successfully inserts node 'C'.

Algorithm 2 Insertion Algorithm (FI, SI, FIf)

1: Create initial insertions
2: **while** $L \neg \emptyset$ **do**
3: Select insertion
4: Insert request into vehicle
5: Recalculate insertions

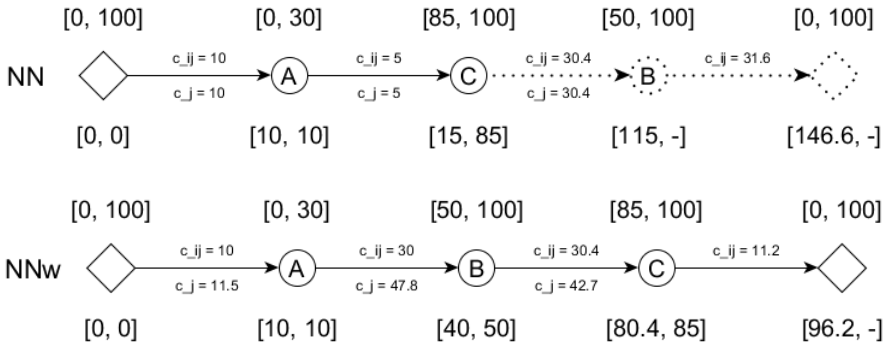


Figure 5.6: Example of NN and NN_w, arrival back at the depot. The choices made by NN do not result in a feasible solution. NN_w is successful in constructing a feasible solution.

Algorithm 3 Create initial insertions for FI, SI, and FIf

for all $r \in R$ **do**
  for all $i \in N_r^P$ **do**
    for all $j \in N_r^D$ **do**
      $t_i \leftarrow max(d_{0,i}, a_i)$ ▷ arrival time at pickup i
      if $t_i > b_i$ **then** ▷ if arrival time exceeds the time window
        continue
      $t_j \leftarrow max(t_i + \sigma_i + d_{ij}, a_j)$ ▷ arrival time at delivery j
      if $t_j > b_j$ **then** ▷ if arrival time exceeds the time window
        continue
      $t_0 \leftarrow t_j + \sigma_j + c_{j0}$ ▷ arrival time at depot
      if $t_0 > E$ **then**
        continue
      $d \leftarrow d_{0i} + d_{ij} + d_{j0}$ ▷ calculate distance
      $w \leftarrow t_0 - d - \sigma_i - \sigma_j$ ▷ calculate waiting time
      $I_{i,j} \leftarrow d + w\varphi_w$ ▷ calculate insertion cost

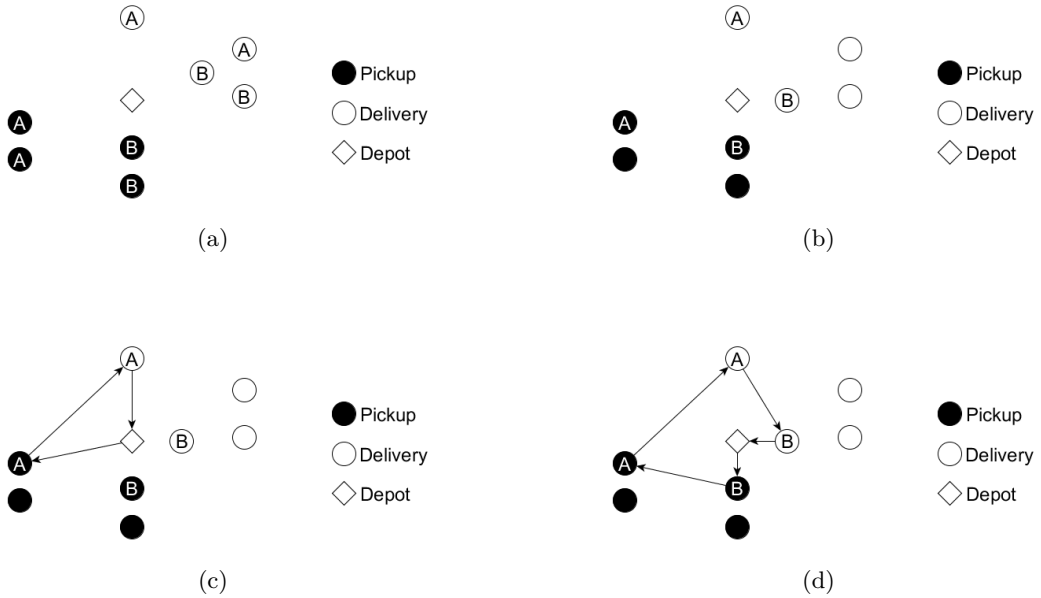


Figure 5.7: Depiction of farthest insertion. Figure 5.7a shows two requests 'A' and 'B' with two pickup and two delivery locations each. Not depicted is the insertion calculation that determines the cost for each pickup and delivery location combination for each request. Figure 5.7b shows how exactly one insertion is selected for each request, namely the one with the smallest cost. Choosing from the remaining (in this example two) insertions, Figure 5.7c shows the selection of the farthest insertion, request 'A'. Here, the insertions are recalculated. For the only remaining request 'B' the same nodes are selected, however the placement of the nodes within the sequence are different, since the tour already serves a request. Figure 5.7d show the selection of the next-farthest (and last) insertion, request 'B', and its placement in the tour.

The difference between FI, SI, and FIf lies in their manner of choosing the next insertion.

FI, which stands for farthest insertion, pre-selects one insertion for each request, namely the one with the smallest cost. Ties are broken randomly. All other insertions are discarded. From the remaining insertions one is picked by a roulette wheel selection prioritizing the highest cost. This function is named *Get biggest by roulette wheel selection* and is part of the procedure described in Algorithm 4. A small example for FI with two requests can be found in Figure 5.7.

SI, which stands for smallest insertion, does no pre-selection. Instead, it simply picks an insertion by a roulette wheel selection prioritizing the smallest cost. A small example with two requests can be found in Figure 5.8. The procedure is described in Algorithm 5. Note, that $max(c_i, 1)$ is used when adding the cost for each insertion to account for costs smaller than 1 and zero.

FIf is a combination of FI and SI and stands for farthest insertion first. The first request to be inserted in a tour is selected by a FI principle while all subsequent insertions follow the SI principle. The main idea is to combine the advantages of both methods. For the example in Figures 5.7 and 5.8, the solution is identical to the FI algorithm. However, the computational experiments performed in Section 5.4 indicate it to be the superior algorithm.

5.3.3 Improvement heuristic using local search (LS)

For improvement of the constructed vehicle routes, we use local search embedded in a VND [106] framework. The VND utilizes neighborhoods that are applied in order of complexity. Each neighborhood is performed until no further improvement can be found for θ iterations and a

Algorithm 4 Pick Farthest Insertion (FI)

$L_s \leftarrow \emptyset$ ▷ An empty set of insertions
for $r \in R$ **do** ▷ for each request
    $L_s \leftarrow L_{r,min}$ ▷ the insertion with the smallest cost is added to L_s
$L_{s,max} \leftarrow$ GET BIGGEST BY ROULETTE WHEEL SELECTION(L_s)
return $L_{s,max}$
function GET BIGGEST BY ROULETTE WHEEL SELECTION(L_s)
    $c_c \leftarrow 0$ ▷ initialize variable for cumulative cost
    for all $i \in L_s$ **do** ▷ for each insertion in the list
        $c_c \leftarrow c_c + c_i$ ▷ add cost of each insertion c_i to c_c
    $r \leftarrow$random number between 0 and c_c
    for $i \in L_s$ **do**
        $r \leftarrow r - c_i$
        if $r < 0$ **then**
            return i

(a) (b)

Figure 5.8: Depiction of smallest insertion with two requests 'A' and 'B' and each request with two pickup and two delivery locations. Figure 5.8a shows how the first request, request 'B', is selected and inserted into the route. Figure 5.8b selects and inserts the next request, request 'A', and depicts how the other locations of request 'B' are no longer an option. Although the algorithm uses a roulette wheel selection, this small example always chooses the request with the smallest cost for illustrative purposes.

Algorithm 5 Pick Smallest Insertion (SI)

$c_c \leftarrow 0$ ▷ initialize variable for cumulative cost
for all $i \in L$ **do** ▷ for each insertion in the list
    $c_c \leftarrow c_c + \frac{1}{max(c_i,1)}$ ▷ add cost of each insertion to c_c
$r \leftarrow$random number between 0 and c_c
for $i \in L$ **do**
    $r \leftarrow r - \frac{1}{max(c_i,1)}$
    if $r < 0$ **then**
        return i

local optimum is reached. To escape from this local optimum, the algorithm moves to the next neighborhood. As soon as a new solution has been found, the algorithm restarts and returns to the first neighborhood since it is the least complex and 'cheapest' in terms of computation time. If no more improvements can be found for θ iterations for all neighborhoods, the algorithm is terminated. The found solution is a local optimum for all neighborhoods applied [159].

For our VND, decribed in Algorithm 6, we use two neighborhoods: The *move request* neighborhood picks a random request, removes it from its original tour, and inserts it into the best possible position (in terms of node sequence) of any other vehicle (including its former vehicle). The goal is to reduce the net distance. For determination of the net distance reduction, we compare the savings when removing a request to the insertion cost in the new vehicle and/or the new position of the nodes (see Eq. 5.4 and 5.5). If a better vehicle and/or position can be found, the request is moved, otherwise it remains in its current vehicle and its current position.

The *swap request* neighborhood picks two random requests from separate vehicles and swaps the vehicle assignment of the requests. The requests are first removed from their original tour and reinserted in the tour of the other request respectively. Only the best position within the node sequence is considered when inserting a request, no matter the position of the original request. The cost savings when removing and reinserting both requests are determined while both insertions have to be feasible. If the total cost is smaller, the swap counts as a success and is saved. Otherwise the changes are discarded.

Additional neighborhoods were implemented but did not prove successful: *2-Opt*, *2-Opt**, *move section*, and *swap section*. All those were limited by the present time window restrictions and precedence constraints prevalent in a PDP. Also, *move section* and *swap section* between vehicles were additionally limited by the constraint that a request has to be served by a single vehicle and requests are not allowed to be split up.

Algorithm 6 Variable Neighborhood Decent (VND)

```
1:  t                                                    ▷ Iteration counter
2:  while true do
3:      t ← 0
4:      while t < θ do                          ▷ While iteration limit θ not reached
5:          t ← t + 1
6:          Move request
7:          if Move request lead to improvement then
8:              t ← 0
9:      t ← 0
10:     while t < θ do
11:         t ← t + 1
12:         Swap request
13:         if Swap request lead to improvement then
14:             break
15:     if t < θ then
16:         continue
17:     else
18:         break
```

Size	NN	NN_w	FI	SI	FIf	Avg.	Best	Δ avg.	Worst	Δ best
10	294.5	295.0	292.7	292.8	292.8	293.5	FI	-0.3%	NN_w	-0.8%
30	927.9	928.4	908.8	909.6	907.0	916.3	FIf	-1.0%	NN_w	-2.3%
50	1386.8	1387.2	1375.4	1376.1	1374.6	1380.0	FIf	-0.4%	NN_w	-0.9%
100	3209.9	3210.1	3176.4	3179.7	3168.6	3188.9	FIf	-0.6%	NN_w	-1.3%

Table 5.5: Comparison of different construction heuristics in absolute OV by instance size.

Size	NN	NN_w	FI	SI	FIf	Avg.	Best	Δ avg.	Worst	Δ best
10	252.1	252.2	250.1	250.6	250.5	251.1	FI	-0.4%	NN_w	-0.8%
30	693.2	692.7	686.2	685.3	682.2	687.9	FIf	-0.8%	NN	-1.6%
50	1019.3	1019.0	1018.8	1017.4	1017.8	1018.5	SI	-0.1%	NN	-0.2%
100	1895.7	1890.5	1883.5	1881.4	1880.8	1886.4	FIf	-0.3%	NN	-0.8%

Table 5.6: Comparison of different construction heuristics with LS in absolute OV by instance size.

5.4 Computational experiments for the routing algorithms

5.4.1 Comparison of the construction heuristics

The following construction heuristics are being compared: nearest neighbor (NN), nearest neighbor modified by a waiting time weight (NN_w), farthest insertion (FI), smallest insertion (SI), and the combination of farthest insertion and smallest insertion (FIf). The computational comparison was performed with the following method: for each request-to-vehicle assignment each construction heuristic was applied once. The request-to-vehicle assignments were generated as part of the solution algorithms described in Section 5.5. All generated vehicle routes were saved for evaluation. In total 18,450 vehicle routes were generated. Of those, 15,045 were from instances with time windows, and 3,405 were from instances without time windows.

Table 5.5 shows the comparison of different construction heuristics by instance size, the average OV for each method-instance size combination, and the average objective value (OV). The best methods are FI for the smallest instances and FIf for all others with improvements up to 1 percent in relation to the average OV. The worst method is NN_w with a larger OV of up to 2.3 percent compared to the best method. Table 5.6 includes LS in the results. The best methods are FI for the smallest instances and FIf and SI for all others with improvements up to 0.8 percent in relation to the average OV. The worst method is NN_w for the smallest instances and NN for all others with an increase of OV of up to 1.6 percent compared to the best method.

Table 5.7 and 5.8 show the results for instances without time windows. The construction heuristic FIf is the best, both for the case with LS as well as the case without LS. The difference between the best and the worst method is up to 4.7 percent larger for instances without time windows.

Concerning feasibility

Table 5.9 shows, in percent, how many times different construction heuristics did not find a feasible solution for different waiting time weights φ_w. For the instances without time windows, all algorithms seem equally successful in finding feasible solutions; there are however differences for instances with time windows. NN performs the worst with 6.5 percent of infeasible solutions on average, while both SI and FIf share the best results of 3.1 percent. It is interesting to

Size	NN	NN_w	FI	SI	FIf	Avg.	Best	Δ avg.	Worst	Δ best
10	323.4	323.4	320.2	320.6	320.0	321.5	FIf	0.00	NN	-1.1%
30	953.6	953.6	913.3	916.9	908.8	929.2	FIf	-2.2%	NN	-4.7%
50	1284.9	1284.9	1255.4	1256.8	1253.9	1267.2	FIf	-1.0%	NN	-2.4%
100	3066.5	3066.5	3009.3	3015.6	2991.2	3029.8	FIf	-0.01	NN	-2.5%

Table 5.7: Comparison of different construction heuristics in absolute OV by instance size without time windows.

Size	NN	NN_w	FI	SI	FIf	Avg.	Best	Δ avg.	Worst	Δ best
10	251.0	251.0	249.3	249.7	249.2	250.1	FIf	-0.3%	NN	-0.7%
30	631.9	631.9	621.4	622.8	606.5	622.9	FIf	-2.6%	NN	-4.0%
50	925.0	925.0	926.7	925.9	925.6	925.6	NN	-0.1%	FI	-0.2%
100	1810.0	1810.0	1801.1	1800.7	1799.3	1804.2	FIf	-0.3%	NN	-0.6%

Table 5.8: Comparison of different construction heuristics with LS in absolute OV by instance size without time windows.

note that, a waiting time weight φ_w of zero leads to the most infeasible solutions. We therefore conclude that considering waiting time in the cost calculation provides the benefit of a wider range of feasible solutions.

Table 5.10 shows the best and worst methods regarding infeasibility. In 35 percent of cases, every method found a feasible solution, except one. If only one method did not manage to find a feasible solution, while all other methods did, it was in 95 percent of cases the NN. This phenomenon, that one method was particularly unsuccessful, only occurred for instances with time windows. In 13 percent of cases, no method found a feasible solution, except one. If only one method found a feasible solution, it was NN_w (37 percent), FIf (29 percent), SI (20 percent), or FI (14 percent), but never NN. For cases without time windows the best methods are SI (71 percent) and FI (29 percent). These methods therefore seem most promising when only considering feasibility and not solution quality.

	with time windows						without time windows					
φ_w	NN	NN_w	FI	SI	FIf	Avg. by φ_w	NN	NN_w	FI	SI	FIf	Avg. by φ_w
0	6.5%	3.7%	5.5%	4.7%	5.1%	5.1%	0.1%	0.1%	0.1%	0.1%	0.1%	0.1%
0.2	6.5%	3.7%	2.7%	2.7%	2.6%	3.7%	0.1%	0.1%	0.1%	0.1%	0.1%	0.1%
0.5	6.5%	3.7%	2.8%	2.6%	2.5%	3.6%	0.1%	0.1%	0.1%	0.1%	0.1%	0.1%
0.7	6.5%	3.7%	2.7%	2.7%	2.6%	3.6%	0.1%	0.1%	0.1%	0.1%	0.1%	0.1%
1	6.5%	3.7%	2.8%	2.7%	2.5%	3.6%	0.1%	0.1%	0.1%	0.1%	0.1%	0.1%
Avg.	6.5%	3.7%	3.3%	3.1%	3.1%		0.1%	0.1%	0.1%	0.1%	0.1%	

Table 5.9: Percentage of infeasible solutions by construction heuristic and waiting time weight φ_w. Percentages for instances with time windows are based on the 15,045 vehicle routes with time windows, and percentages for instances without time windows are based on the 3,405 vehicle routes without time windows. For instances without time windows φ_w has no influence on the percentage of infeasible solutions.

Every method found a feasible solution, except ...											
with time windows						without time windows					
Total	NN	NN_w	FI	SI	FIf	Total	NN	NN_w	FI	SI	FIf
35%	95%	0%	2%	1%	1%	0%	0%	0%	0%	0%	0%
No method found a feasible solution, except ...											
with time windows						without time windows					
Total	NN	NN_w	FI	SI	FIf	Total	NN	NN_w	FI	SI	FIf
13%	0%	37%	14%	20%	29%	28%	0%	0%	29%	71%	0%

Table 5.10: Best and worst methods regarding infeasibility. The 'Total' column refers to the percentage of infeasible vehicle routes.

Size	NN+LS vs NN	NN_w+LS vs NN_w	FI+LS vs FI	SI+LS vs SI	FIf+LS vs FIf
10	-14.4%	-14.5%	-14.5%	-14.4%	-14.4%
30	-25.3%	-25.5%	-24.5%	-24.7%	-24.8%
50	-26.5%	-26.5%	-25.9%	-26.1%	-26.0%
100	-40.9%	-41.1%	-40.7%	-40.8%	-40.6%
Avg.	-26.8%	-26.9%	-26.4%	-26.5%	-26.5%

Table 5.11: Average improvement of local search by method and instance size. Averaged over all φ_w.

The potential of local search

After eliminating infeasible vehicle routes, 13,387 feasible routes remain for evaluation. Of those, there are 9,809 routes from instances with size 10, 2,021 routes from size 30, 964 routes from size 50 and 593 routes from size 100. Tables 5.11, 5.12 and 5.13 give an overview of the potential of local search. Local search seem to be more effective the larger the instance size is, with improvements of around 41 percent on average for instances of size 100 in comparison of 14 percent for instances of size 10. Waiting time weight φ_w on the other hand, does not seem to have much influence on the effectiveness of LS, and neither does the choice of construction heuristic. For a weight of zero, the improvements seem slightly bigger using LS which corresponds to the observation that $\varphi_w = 0$ generates the least amount of feasible solutions. However, both waiting time weight and construction heuristic, might affect the solution quality, even though they do not seem to influence the effectiveness of LS. In general, we see that LS is a powerful procedure with average improvements of around 27 percent (Table 5.11) or 16 percent (Table 5.13). The difference between the two reported average values is the following: Table 5.13 computes the values over all vehicle routes. Since the generated vehicle routes are not equally distributed over the different instance sizes, the values for the average and median are skewed towards the smaller instances with a multitude of routes. Table 5.11 however, calculates the average of the averages by instance size. Overall improvements of LS are up to 69 percent. The minimum improvement is zero, as there is never any decrease in solution quality by applying LS.

For instances without time windows, the computational experiments with LS lead to similar results. Tables 5.14, 5.15 and 5.16 give an overview over the results for instances without time windows. The improvements are slightly larger overall. The differences are especially visible for smaller instance sizes. The results indicate that for instances without time windows there is a

φ_w	NN+LS vs NN	NN_w+LS vs NN_w	FI+LS vs FI	SI+LS vs SI	FIf+LS vs FIf
0	-27.2%	-27.3%	-26.8%	-26.8%	-26.9%
0.2	-26.9%	-27.1%	-26.7%	-26.7%	-26.6%
0.5	-26.8%	-26.9%	-26.4%	-26.5%	-26.4%
0.7	-26.6%	-26.7%	-26.2%	-26.3%	-26.2%
1	-26.4%	-26.5%	-26.1%	-26.2%	-26.1%
Avg.	-26.8%	-26.9%	-26.4%	-26.5%	-26.5%

Table 5.12: Average improvement of local search by method and waiting time weight φ_w. Averaged over all instance sizes.

	NN+LS vs NN	NN_w+LS vs NN_w	FI+LS vs FI	SI+LS vs SI	FIf+LS vs FI
Max.	-69%	-69%	-69%	-69%	-69%
Median	-15%	-15%	-15%	-14%	-15%
Avg.	-16%	-16%	-16%	-16%	-16%
Min.	0%	0%	0%	0%	0%

Table 5.13: Improvement range of local search over all vehicle routes.

slight difference in the effectiveness of LS when comparing different construction heuristics. The differences are small (up to 2 percent) but 'simpler' construction heuristics (NN and NN_w) seem to profit slightly more from LS than more complex construction heuristics (FI, SI, FIf). The same impression is obtained from Table 5.15, where the value of φ_w seems to be irrelevant, whereas the choice of construction heuristic makes a small difference. For instances without time windows the average improvement is 23 percent with a median of 25 percent. Improvements reached by LS are up to 69 percent.

The importance of waiting time weight

Table 5.17 shows the average objective values (OV) by instance size for each waiting time weight φ_w. The table shows the standard deviation in absolute numbers as well as a percent of the average OV for each particular instance size. A different weight leads to a difference of up to 0.5 percent. When considering feasibility and the effectiveness of LS, we conclude that a positive weight is beneficial. However, when considering the absolute OV, the best results are achieved with $\varphi_w = 0$. Since the absolute improvement is quite small but the amount of infeasible

Size	NN+LS vs NN	NN_w+LS vs NN_w	FI+LS vs FI	SI+LS vs SI	FIf+LS vs FIf
10	-22.4%	-22.4%	-22.1%	-22.1%	-22.1%
30	-33.7%	-33.7%	-32.0%	-32.1%	-33.3%
50	-28.0%	-28.0%	-26.2%	-26.3%	-26.2%
100	-41.0%	-41.0%	-40.1%	-40.3%	-39.8%
Avg.	-31.3%	-31.3%	-30.1%	-30.2%	-30.4%

Table 5.14: Average improvement of local search by method and instance size for instances without time windows. Averaged over all φ_w.

φ_w	NN+LS vs NN	NN_w+LS vs NN_w	FI+LS vs FI	SI+LS vs SI	FIf+LS vs FIf
0	-31.3%	-31.3%	-30.1%	-30.2%	-30.4%
0.2	-31.3%	-31.3%	-30.1%	-30.2%	-30.4%
0.5	-31.3%	-31.3%	-30.1%	-30.2%	-30.4%
0.7	-31.3%	-31.3%	-30.1%	-30.2%	-30.4%
1	-31.3%	-31.3%	-30.1%	-30.2%	-30.4%
Avg.	-31.3%	-31.3%	-30.1%	-30.2%	-30.4%

Table 5.15: Average improvement of local search by method and waiting time weight φ_w for instances without time windows. Averaged over all instance sizes.

NN+LS vs NN	NN_w+LS vs NN_w	FI+LS vs FI	SI+LS vs SI	FIf+LS vs FIf	
Max.	-69%	-69%	-69%	-69%	-69%
Median	-25%	-25%	-25%	-25%	-26%
Avg.	-23%	-23%	-23%	-23%	-23%
Min.	0%	0%	0%	0%	0%

Table 5.16: Improvement range of local search over all vehicle routes without time windows.

solutions is decreased by half, we propose a positive, albeit small weight of around 0.2 as a means to profit from both effects. Table 5.18 shows that for instances without time windows, the weight of waiting time has no influence on the absolute OV.

φ_w/Instance size	10	30	50	100
0	250.10	683.56	1011.42	1872.50
0.2	250.51	686.00	1013.91	1881.26
0.5	251.14	688.81	1017.81	1885.48
0.7	251.69	690.15	1023.14	1892.77
1	252.02	691.02	1026.03	1899.87
Std. dev.	0.71	2.76	5.48	9.41
in % of avg.	0.3%	0.4%	0.5%	0.5%

Table 5.17: Average OV by waiting time weight and instance size.

φ_w/Instance size	10	30	50	100
0	250.06	622.88	925.64	1804.22
0.2	250.06	622.88	925.64	1804.22
0.5	250.06	622.88	925.64	1804.22
0.7	250.06	622.88	925.64	1804.22
1	250.06	622.88	925.64	1804.22
Std. dev.	0.00	0.00	0.00	0.00
in % of avg.	0.0%	0.0%	0.0%	0.0%

Table 5.18: Average OV by waiting time weight and instance size for instances without time windows.

5.5 Solution algorithms for the request-to-vehicle assignment

For the request-to-vehicle assignment we implemented different metaheuristics: an Adaptive Large Neighborhood Search (ALNS), a Genetic Algorithm (GA), a multi-start ALNS (MS-ALNS), and a combination of GA and ALNS (GA-ALNS).

For a successful implementation of those methods, and an efficient application of our search operators, a well thought out solution representation needs to be used [141]. We utilized both a genotype, for applying our search operators, and a phenotype for the evaluation of our solutions. Our genotype is represented as an array. Its size depends on the number of requests. For each request an integer indicating a vehicle ID is set. All search operators, be it of the destroy/recreate type for the neighborhood based heuristics or the recombination/mutation type for the evolutionary algorithms, are based on the modification of the integers in this array. Our phenotype for evaluating our solution is our routing algorithm which determines the specific pickup and delivery locations and node sequence of the vehicles.

For all methods we use two simple construction heuristics. The first construction method assigns each request to a separate vehicle. The second construction method assigns each request to a random vehicle. The ALNS, as the only single-solution approach, uses the solution of the first construction method as a starting solution. The other methods (GA, MS-ALNS, GA-ALNS) use pools of solutions which consists of both methods (the one solution from the first construction method, and the rest from the second random construction method). When all requests are assigned to vehicles, the routing heuristic is used to determine the pickup and delivery locations, the node sequence, and the travel time and cost. Based on the computational experiments of the routing algorithms (see Section 5.4), we selected the farthest insertion first (FIf) algorithm as the standard routing algorithm for all metaheuristics, so that they remain comparable.

Gaining information in a preprocessing step

An important step in our solution algorithm is the detection of additional instance specific information or constraints. Depending on the instance, it is entirely possible that some requests can not be assigned to be served by the same vehicle. Due to distance or time windows it might be impossible to feasibly serve both requests with the same vehicle. This knowledge can be exploited to make the search space smaller. This information is therefore saved and is used by all our methods when determining the request-to-vehicle assignment.

Additionally, a request compatibility measure is calculated. This measure has already been introduced in Dragomir et al. [44] and is only shortly recapitulated here. The compatibility

measure tries to estimate how well a pair or triple of requests fit together when being served by the same vehicle. The calculation is based on the travel times and time windows and is calculated as follows: For each pair of requests, r_1 and r_2, a single compatibility indicator is calculated. This is done only once as a preprocessing step and the values are stored. The compatibility κ of request r_1 and request r_2 is

$$\begin{aligned}\kappa_{r_1 r_2} = &\sum_{p_1 \in N^P_{r_1}} \sum_{p_2 \in N^P_{r_2}} max(|b_{p_1} - a_{p_2} - c_{p_1 p_2}|, |b_{p_2} - a_{p_1} - c_{p_1 p_2}|) \\ + &\sum_{p_1 \in N^P_{r_1}} \sum_{d_2 \in N^D_{r_2}} max(|b_{p_1} - a_{d_2} - c_{p_1 d_2}|, |b_{d_2} - a_{p_1} - c_{p_1 d_2}|) \\ + &\sum_{d_1 \in N^D_{r_1}} \sum_{p_2 \in N^P_{r_2}} max(|b_{d_1} - a_{p_2} - c_{d_1 p_2}|, |b_{p_2} - a_{d_1} - c_{d_1 p_2}|) \\ &+ \sum_{d_1 \in N^D_{r_1}} \sum_{d_2 \in N^D_{r_2}} max(|b_{d_1} - a_{d_2} - c_{d_1 d_2}|, |b_{d_2} - a_{d_1} - c_{d_1 d_2}|) \quad (5.6)\end{aligned}$$

with N^P_r and N^D_r as the sets of alternative pickup and delivery locations of request r, c_{ij} as the travel times between locations i and j, a_i and b_i as the begin and end time windows for location i. Request pairs have a higher compatibility if the travel time between them and their time windows make a visit easily feasible. For triples of requests we calculate all pairs of requests separately and take the average. The compatibility measure is only used by the ALNS and MS-ALNS as part of a repair operator.

5.5.1 Adaptive large neighborhood search (ALNS)

Shaw [149] introduced a search method for a vehicle routing problem based on destroying and recreating a solution. The destroy and recreate operators are applied on large portions of a solution which gives the large neighborhood search (LNS) its name. This concept was extended in 2006 by Ropke and Pisinger [139] (and renamed to destroy and repair) by letting the algorithm steer the probabilities of which operators to use, depending on their past successes. The ALNS was developed primarily for PDPs with time windows. Since then it has gained in popularity and has been applied to many different problems.

For the detailed description of the ALNS we refer the reader to Dragomir et al. [44]. In principle, the ALNS works as follows: A solution is (1) destroyed, (2) repaired, and (3) replaces the incumbent solution according to an acceptance criterion.

The destroy operators remove between γ_{min} and γ_{max} percent of the solution. Our ALNS uses three destroy operators: *destroy greedy requests*, *destroy random requests*, and *destroy random vehicles*, which are all standard operators. All operators are described in detail in Dragomir et al. [44]. Each destroy operator removes between γ_{min} and γ_{max} percent of the solution, i.e. request-to-vehicle assignments. Simultaneously, the pickup and delivery nodes are removed from the vehicle tours while leaving the rest of the node sequence intact. This is necessary because we need to be able to evaluate additional request removals from the same vehicle tours. The method of removal is identical for all destroy operators, they are only distinguished by their selection of which requests to remove. Should no improvement be found for $c_{\gamma_{max}}$ iterations, the solution destroy percentage γ_{max} is doubled.

Our ALNS uses six repair operators: *repair cheapest insertion* and *repair request randomly* are standard operators while *repair request compatibility pairs*, *repair location compatibility pairs* and *repair location compatibility triple* are self-developed problem specific operators that use the information gained during the preprocessing step (for details see Dragomir et al. [44]). The repair operators reassign requests to vehicles and determine the vehicle routes.

The changed solution is evaluated according to our objective function and, if it is better, always accepted. If the solution is improved, we update the probabilities of selecting those specific destroy and repair operators. If it is worse, it is only accepted if the algorithm was unsuccessful for c_w iterations and the solution is only slightly worse in comparison to the starting solution. The parameter w, indicating how much increase in solution objective value is accepted, is initialized to 1 (where no depreciation in solution quality is accepted) and grows by w' with each unsuccessful iteration after c_w iterations until at most w_{max}. Infeasible solutions are never accepted. As soon as an improvement is found, all counters and thresholds are reset. Both accepting a worse solution and increasing the amount of the solution to be destroyed, is beneficial to escape from a local optima.

The stopping criterion for ALNS is either c_L iterations without improvement or a run time limit. For the computational experiments using ALNS alone, c_L was set to infinity and only a run time limit was provided. When calling the ALNS procedure as part of MS-ALNS or GA-ALNS, an iteration limit $c_L = 50$ was provided.

5.5.2 Genetic algorithm (GA)

Genetic algorithms are computational models of evolution and classified as population based search approaches [59]. The basic principle relies on creating an initial population of individuals (individual solutions), and iteratively creating new generations by recombining and mutating those individuals. Influenced by Darwin's natural selection theory and Herbert Spencer's iconic phrase 'Survival of the fittest', the 'fittest' or best individuals will procreate and pass on their superior genes to the next generation. In terms of operations research, the fittest solution is defined by our fitness function, which (in our case) is identical to the objective function.

The GA is described in Algorithm 7. For the creation of the initial solution pool, the simple construction heuristic mentioned at the beginning of Section 5.5 is used and the pool is filled with randomly generated solutions.

The probability of selecting the parents for procreation is the same as in Kheiri et al. [86] with a probability of 1/2 to select out of the top 50 percent of the population, a probability of 1/3 to select out of the less-than-average quarter and a probability of 1/6 to select out of the worst quarter. Within those parts, the probability to select a specific solution is uniform.

As in Kheiri et al. [86], children are created using *uniform crossover* where the decision which parent property to use for each request is made independently. In our case, each request needs to be assigned to a vehicle and each request has a probability of 50 percent to inherit the vehicle assignment of one of its parents. Additionally, we consider the information obtained in the preprocessing step and only assign requests to vehicles where no conflicting requests are present. At least one parent will always fulfill this requirement.

Each child has a mutation probability $m_{prob} = 0.5$ to mutate and a mutation percentage of $m_{perc} = 0.05$. That is, if a mutation occurs, 5 percent of requests are randomly assigned to a different 'allowed' vehicle where no conflicts with other requests exist.

According to Zäpfel et al. [168], the population should consist of elite and diverse solutions. We therefore keep the best k percent of each generation as an elite. After creating all children of a new generation, the elite is added to the solution pool of the new generation. The solutions are sorted by fitness and the worst solutions removed so as to keep the population size stable.

5.5.3 Multi-start adaptive large neighborhood search (MS-ALNS)

For a multi-start approach we create a pool of different initial solutions. After many computational experiments, the size of the solution pool was set to 10. A larger solution pool means a higher diversification in the starting points for the metaheuristic which can be advantageous. On the other hand, this advantage is balanced out when considering a limited run time where not every solution might be selected for improvement. Therefore we selected a value as big as

Algorithm 7 Genetic Algorithm (GA)

```
1: while solution pool is not filled do
2:     create feasible initial solutions
3: while time limit not reached do
4:     CREATE NEW GENERATION
5: function CREATE NEW GENERATION(population size P_GA)
6:     G ← ∅                          ▷ Create an empty solution pool for the new generation
7:     while |G| < P_GA do
8:         select 2 parent solutions
9:         apply uniform crossover to generate a child solution c
10:        mutate child c
11:        if child is feasible then
12:            G ← G ∪ {c}                          ▷ Add child to new generation
13:    add top k percent of parents to child generation pool                ▷ Keep elite
14:    while |G| ⩾ P_GA do
15:        remove worst solution          ▷ After adding elite, reduce population size
```

possible to allow for as much diversification as possible, and as small as necessary to ensure that each solution could be selected for improvement. The disadvantages of a small solution pool is counteracted by the random solution construction which ensures an adequate amount of diversification.

Algorithm 8 gives an overview of the MS-ALNS. First, a solution s is chosen from the solution pool using a roulette wheel selection [7] where better solutions have a higher probability of being chosen in proportion to their fitness. For a minimization problem a better fitness means a smaller OV. This solution s is then improved with an ALNS procedure (see Section 5.5.1). When the iteration limit without improvement c_L for the ALNS is reached, the algorithm is not simply terminated. Instead, the modified solution s' is returned to the pool, replacing the solution s, and a new solution is selected for improvement. Altough the iteration limit c_L for a single solution seems small, it ensures that the algorithm does not needlessly spend time on unfruitful solutions.

Algorithm 8 Overview of MS-ALNS

```
1: while solution pool is not filled do
2:     create feasible initial solutions
3: while time limit not reached do
4:     pick a solution s by roulette wheel selection
5:     improve solution s with ALNS
```

5.5.4 Genetic algorithm and adaptive large neighborhood search (GA-ALNS)

The GA-ALNS is a simple combination of both GA and ALNS, namely that they share the run time. Algorithm 9 gives an overview of the method. The GA is performed until μ_{GA} percent of the total run time is reached and additionally for a minimum number of ζ_{GA} generations. After this, only the elite part (k percent) of the population is kept and may proceed with the ALNS. The ALNS used for the GA-ALNS follows the same principle as the MS-ALNS.

Algorithm 9 Genetic algorithm and adaptive large neighborhood search (GA-ALNS)

```
1: while solution pool is not filled do
2:     create feasible initial solutions
3: while GA time limit or min. number of generations not reached do
4:     create new generation
5: keep only the best k percent of population
6: while time limit not reached do
7:     pick a solution s by roulette wheel selection
8:     improve solution s with ALNS
```

5.6 Computational experiments for the request-to-vehicle assignment

For the comparison of the different metaheuristics we solved 30 instances of various sizes (30, 50, and 100 requests) with each method. For each instance, each method has the same run time limit. We compare not only the overall result, but also the performance during the run time. The run time limit depends on the instance size and is five times the number of requests in seconds (i.e. 150 seconds for instances with 30 requests, 250 seconds for instances with 50 requests, and 500 seconds for instances with 100 requests). Since we have no optimal solution to compare our methods to, we compare to the best found solution over all methods. We deem this sufficient since the MS-ALNS has already been compared to both heuristics and exact methods from the literature in Dragomir et al. [44]. The evaluation of the methods is conducted for each instance size separately where the results are given as a percentage of how far away the method is from the best found solution (also named 'base' solution or 'base' objective value). Additionally, all results are reported as a percent of passed run time to make instances of different size, and therefore different run times, comparable to each other.

Table 5.19 provides a comparison of all methods. More detailed results can be found in Table 5.21 in the Appendix. Both tables represent the results at the end of the allocated run time with ΔOV indicating the deviation in percent to the 'base' OV (OV_B) and the rank. The methods are ranked from 1 to 4 for each instance. If two methods reached the same OV, they share a rank and the next rank is skipped. For example for instance 20, both ALNS and GA-ALNS reached the same solution and are ranked number 2 while rank number 3 is skipped. This results in the sum of the rows '1st place' and 'last place' of Table 5.19 to be larger than 100 percent, since ranks can be assigned twice.

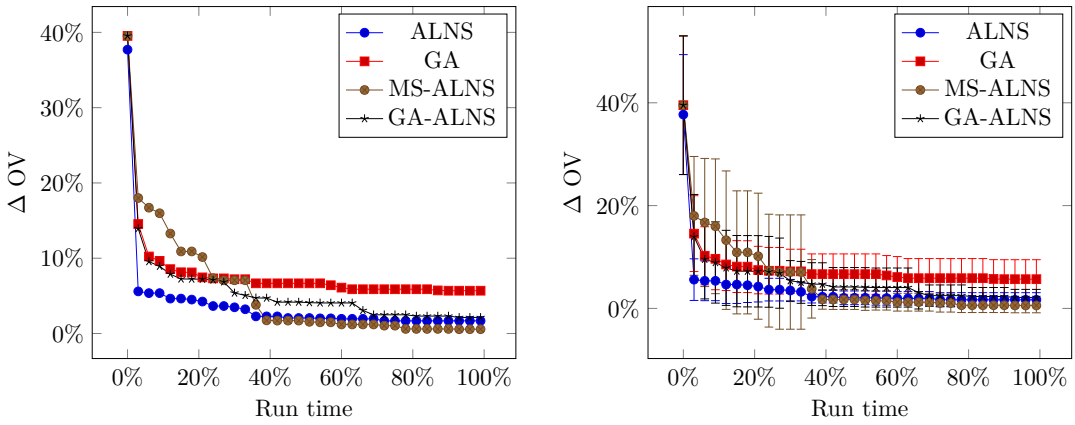
The best method according to the average rank is the MS-ALNS with a rank of 1.6. It is followed by GA-ALNS with 2.3, third place is the ALNS with an average rank of 2.5 and GA with 3.2. Table 5.19 also provides the average, median, and maximum deviation from 'base' solution. GA-ALNS is superior to all other methods with an average deviation ('Avg. ΔOV') of 0.9 percent, a median deviation ('Median ΔOV') of 0.1 percent and a maximum deviation ('Max. ΔOV') of 6.4 percent. The second best method is MS-ALNS with an average deviation of 1.4 percent while the worst method is GA with an average deviation of 4.4 percent.

When looking at the number of times a method reached first and last place in the ranking, compared to the other methods, a similar picture emerges: MS-ALNS reached first place in 50 percent of the instances and was never last place. GA-ALNS reached first place in 27 percent of the instances and was last place in 20 percent of the instances. The worst method is again the GA, with only 10 percent of the times reaching first place and 53 percent last place.

This leads to the following conclusions: the GA is by far the worst performing method. It was most often last place, least often first place. It has the highest average and median deviation from the best-known objective value and the worst maximum deviation. In our computational

	ALNS	GA	MS-ALNS	GA-ALNS
Avg. rank	2.5	3.2	**1.6**	2.3
Avg. ΔOV	2.1%	4.4%	1.4%	**0.9%**
Median ΔOV	2.0%	4.0%	0.8%	**0.1%**
Max. ΔOV	8.3%	15.3%	8.4%	**6.4%**
1st place	27%	10%	**50%**	27%
last place	23%	53%	**0%**	20%

Table 5.19: Method comparison



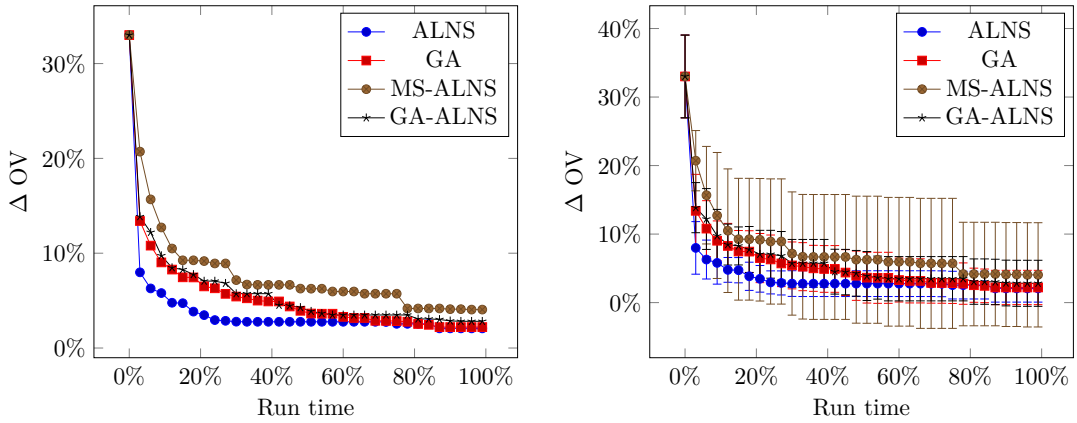
(a) Without standard deviations

(b) With standard deviation

Figure 5.9: Method comparison for instances of size 30

experiments MS-ALNS is the best method. It has the best average rank, is most often first place and never last place. However, considering the deviations from the best objective value, it is always slightly worse than GA-ALNS.

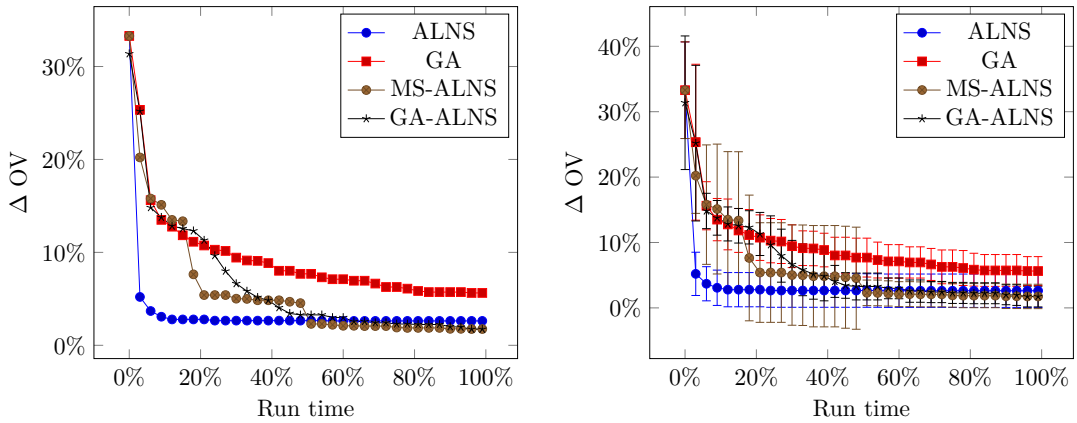
Figures 5.9 to 5.12 show the performance of the methods over the course of the run time. For all instance sizes the MS-ALNS has the slowest start but leads to the best solutions. Both GA and GA-ALNS show a slow decline even in the later parts of the run time. The ALNS is often the fastest to reach a big drop in OV, however after that it is quite stagnant and seems to get caught in some local optimum. All figures are depicted in two versions, with and without error bars of standard deviation to increase readability. Although MS-ALNS is most often ranked first place and never ranked last place, it exhibits the highest volatility as seen from the large error bars (especially for larger instances). Large error bars indicate large differences between the OVs of the different solutions at a specific run time percentage. While small error bars indicate a strong performance and an accurate predictability of the OV at a certain run time percentage. In contrast, ALNS and GA seem to have the smallest standard deviations over all methods and seem to be the most stable.



(a) Without standard deviations

(b) With standard deviation

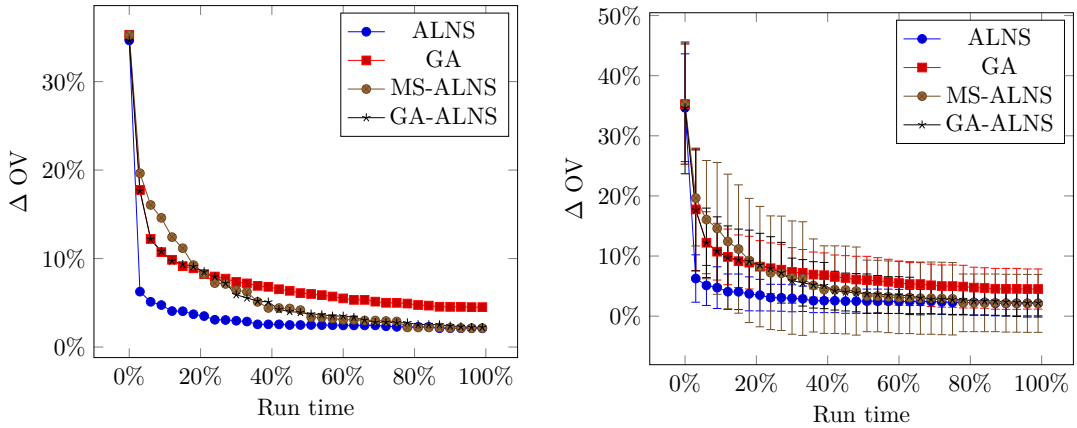
Figure 5.10: Method comparison for instances of size 50



(a) Without standard deviations

(b) With standard deviation

Figure 5.11: Method comparison for instances of size 100



(a) Without standard deviations

(b) With standard deviation

Figure 5.12: Method comparison for all instances

5.7 Conclusions

As a construction heuristic FIf is the most promising, leading to the best results for instances with and without time windows, with and without using LS, and over instances of different sizes. Both, NN and NN_w, are not recommended.

Concerning the feasibility of vehicle routes we conclude that more complex routing methods (FI, SI, and FIf) are superior to simpler methods (NN, NN_w), if the instances have time windows. For instances without time windows, all methods produce solutions of equal feasibility. Furthermore, a waiting time weight φ_w of zero is detrimental to create feasible solutions. Therefore, we conclude that it is beneficial to account for waiting time for the construction of vehicle routes.

The potential of local search is not to be underestimated. With improvements of up to almost 70 percent it is absolutely essential for routing heuristics. In general, it does not seem to matter which construction heuristic is used or the choice of φ_w, since they do not influence the effectiveness of the LS.

A waiting time weight φ_w of a positive value greater than zero is best to improve feasibility, however a value of zero is best for the best OV. Since the gains of improved feasibility are larger than the losses of OV we propose a positive, albeit small, weight of at most 0.2. For instances without time windows, φ_w has no influence over the OV.

In our computational experiments, GA is the worst method by far. It has the highest rank compared to the other methods, the highest average and median deviation from the best objective value and the worst maximum deviation. However, it seems to be a stable method, without much deviation between the instances, and provides a steady decline in OV.

ALNS performs better than GA but not as good as MS-ALNS or GA-ALNS. It has a very quick drop in OV but seems stagnant and struggles to get out of local optima (even though the worse solution acceptance criteria and the solution destroy percentage are greatly increased if no improvement can be found). Compared to the other methods, it is almost as often first place, as it is last place.

MS-ALNS and GA-ALNS perform very differently during run time. MS-ALNS has the slower start, but at some point it often surpasses the GA-ALNS. Comparing the methods to each other, MS-ALNS is clearly better, with a better average rank, most often reaching first place and never last place. However, when comparing the methods to their deviation from the best found OV, GA-ALNS is slightly better. However, since the differences in rank are greater than the differences in deviation from the best found OV, we can declare the MS-ALNS as the best method in this computational experiment.

5.8 Appendix

ID	#req.	time(s)	Method	OV	OV_B	ΔOV	Rank
20	30	151	ALNS	523	522	0.2%	2
20	30	160	GA	542	522	3.8%	4
20	30	150	GA-ALNS	523	522	0.2%	2
20	30	150	MS-ALNS	522	522	0.0%	1
21	30	150	ALNS	646	646	0.0%	1
21	30	151	GA	662	646	2.5%	4
21	30	150	GA-ALNS	646	646	0.0%	1
21	30	150	MS-ALNS	646	646	0.0%	1
22	30	150	ALNS	614	597	2.8%	3
22	30	150	GA	618	597	3.5%	4
22	30	150	GA-ALNS	598	597	0.2%	2
22	30	150	MS-ALNS	597	597	0.0%	1

Table 5.21 continued from previous page

ID	#req.	time(s)	Method	OV	OV_B	ΔOV	Rank
23	30	151	ALNS	522	522	0.0%	1
23	30	153	GA	602	522	15.3%	4
23	30	150	GA-ALNS	531	522	1.7%	3
23	30	150	MS-ALNS	527	522	1.0%	2
31	30	150	ALNS	489	473	3.4%	3
31	30	153	GA	486	473	2.7%	2
31	30	150	GA-ALNS	492	473	4.0%	4
31	30	150	MS-ALNS	473	473	0.0%	1
58	30	150	ALNS	558	546	2.2%	3
58	30	150	GA	594	546	8.8%	4
58	30	151	GA-ALNS	553	546	1.3%	2
58	30	150	MS-ALNS	546	546	0.0%	1
59	30	150	ALNS	642	623	3.0%	2
59	30	150	GA	646	623	3.7%	4
59	30	150	GA-ALNS	645	623	3.5%	3
59	30	150	MS-ALNS	623	623	0.0%	1
60	30	150	ALNS	627	609	3.0%	3
60	30	151	GA	627	609	3.0%	3
60	30	150	GA-ALNS	615	609	1.0%	2
60	30	150	MS-ALNS	609	609	0.0%	1
61	30	150	ALNS	509	498	2.2%	3
61	30	158	GA	533	498	7.0%	4
61	30	150	GA-ALNS	500	498	0.4%	2
61	30	150	MS-ALNS	498	498	0.0%	1
69	30	150	ALNS	462	462	0.0%	1
69	30	169	GA	492	462	6.5%	4
69	30	150	GA-ALNS	477	462	3.2%	3
69	30	151	MS-ALNS	473	462	2.4%	2
210	50	250	ALNS	802	788	1.8%	3
210	50	253	GA	794	788	0.8%	2
210	50	251	GA-ALNS	808	788	2.5%	4
210	50	251	MS-ALNS	788	788	0.0%	1
211	50	250	ALNS	946	904	4.6%	4
211	50	255	GA	918	904	1.5%	3
211	50	250	GA-ALNS	904	904	0.0%	1
211	50	250	MS-ALNS	916	904	1.3%	2
212	50	250	ALNS	899	851	5.6%	4
212	50	251	GA	870	851	2.2%	3
212	50	250	GA-ALNS	851	851	0.0%	1
212	50	250	MS-ALNS	861	851	1.2%	2
213	50	250	ALNS	833	833	0.0%	1
213	50	256	GA	838	833	0.6%	2
213	50	250	GA-ALNS	847	833	1.7%	4
213	50	250	MS-ALNS	838	833	0.6%	2
221	50	253	ALNS	763	763	0.0%	1
221	50	259	GA	797	763	4.5%	3
221	50	256	GA-ALNS	798	763	4.6%	4
221	50	307	MS-ALNS	790	763	3.5%	2

Table 5.21 continued from previous page

ID	#req.	time(s)	Method	OV	OV_B	ΔOV	Rank
248	50	251	ALNS	852	837	1.8%	4
248	50	270	GA	837	837	0.0%	1
248	50	250	GA-ALNS	843	837	0.7%	3
248	50	250	MS-ALNS	837	837	0.0%	1
249	50	250	ALNS	978	975	0.3%	3
249	50	252	GA	975	975	0.0%	1
249	50	250	GA-ALNS	982	975	0.7%	4
249	50	250	MS-ALNS	975	975	0.0%	1
250	50	250	ALNS	950	912	4.2%	4
250	50	252	GA	912	912	0.0%	1
250	50	250	GA-ALNS	920	912	0.9%	2
250	50	250	MS-ALNS	920	912	0.9%	2
251	50	250	ALNS	904	883	2.4%	3
251	50	256	GA	920	883	4.2%	4
251	50	250	GA-ALNS	888	883	0.6%	2
251	50	251	MS-ALNS	883	883	0.0%	1
259	50	254	ALNS	794	794	0.0%	1
259	50	257	GA	858	794	8.1%	3
259	50	309	GA-ALNS	861	794	8.4%	4
259	50	255	MS-ALNS	795	794	0.1%	2
400	100	500	ALNS	1648	1598	3.1%	3
400	100	513	GA	1676	1598	4.9%	4
400	100	500	GA-ALNS	1598	1598	0.0%	1
400	100	500	MS-ALNS	1611	1598	0.8%	2
401	100	500	ALNS	1868	1771	5.5%	4
401	100	501	GA	1861	1771	5.1%	3
401	100	500	GA-ALNS	1771	1771	0.0%	1
401	100	500	MS-ALNS	1785	1771	0.8%	2
402	100	500	ALNS	1760	1708	3.0%	4
402	100	522	GA	1749	1708	2.4%	3
402	100	500	GA-ALNS	1711	1708	0.2%	2
402	100	500	MS-ALNS	1708	1708	0.0%	1
403	100	500	ALNS	1655	1655	0.0%	1
403	100	519	GA	1763	1655	6.5%	4
403	100	500	GA-ALNS	1671	1655	1.0%	3
403	100	502	MS-ALNS	1665	1655	0.6%	2
411	100	501	ALNS	1456	1429	1.9%	3
411	100	501	GA	1526	1429	6.8%	4
411	100	501	GA-ALNS	1451	1429	1.5%	2
411	100	501	MS-ALNS	1429	1429	0.0%	1
438	100	500	ALNS	1453	1451	0.1%	2
438	100	519	GA	1473	1451	1.5%	4
438	100	502	GA-ALNS	1463	1451	0.8%	3
438	100	501	MS-ALNS	1451	1451	0.0%	1
439	100	500	ALNS	1739	1605	8.3%	4
439	100	501	GA	1679	1605	4.6%	3
439	100	500	GA-ALNS	1605	1605	0.0%	1
439	100	500	MS-ALNS	1651	1605	2.9%	2

Table 5.21 continued from previous page

ID	#req.	time(s)	Method	OV	OV_B	ΔOV	Rank
440	100	500	ALNS	1527	1513	0.9%	2
440	100	518	GA	1576	1513	4.2%	4
440	100	500	GA-ALNS	1513	1513	0.0%	1
440	100	500	MS-ALNS	1546	1513	2.2%	3
441	100	501	ALNS	1547	1547	0.0%	1
441	100	518	GA	1646	1547	6.4%	4
441	100	501	GA-ALNS	1570	1547	1.5%	2
441	100	504	MS-ALNS	1585	1547	2.5%	3
449	100	501	ALNS	1367	1325	3.2%	2
449	100	522	GA	1459	1325	10.1%	4
449	100	523	GA-ALNS	1325	1325	0.0%	1
449	100	501	MS-ALNS	1410	1325	6.4%	3

Table 5.21: Results of method comparison for all instances. Although the run time limit was identical, the algorithm always finishes already started iterations. Therefore the reported run times can deviate.

Parameter	Description	Value
$P_{MS-ALNS}$	Size of initial solution pool for MS-ALNS	10
P_{GA}	Size of initial solution pool for GA and GA-ALNS	100
m_{prob}	Mutation probability for GA	0.5
m_{perc}	Mutation percentage for GA	0.05
k	Elite population for GA	0.1
μ_{GA}	Percentage of GA for GA-ALNS	0.2
ζ_{GA}	Min. generations for GA for GA-ALNS	5
w	Initial threshold to accept a worse solution for ALNS	1
w_{max}	ALNS parameter controlling how much worse the incumbent solution is at most allowed to be in relation to the best solution	1.5
w'	ALNS Parameter controlling by how much w changes if no solution could be found	0.05
c_L	ALNS iteration limit. If after c_L iterations no further improvement could be found, the ALNS is terminated.	∞
c_L	Iteration limit for ALNS as part of MS-ALNS or GA-ALNS.	50
c_w	ALNS iteration limit without improvement after which the threshold w to accept a worse solution is increased	50
c_w	Iteration limit without improvement after which the threshold w to accept a worse solution is increased for ALNS as part of MS-ALNS or GA-ALNS	$\frac{1}{2}c_L$
$c_{\gamma_{max}}$	ALNS iteration limit without improvement after which upper destroy limit γ_{max} for a solution is doubled	75
$c_{\gamma_{max}}$	Iteration limit without improvement after which upper destroy limit γ_{max} for a solution is doubled for ALNS as part of MS-ALNS or GA-ALNS	$\frac{3}{4}c_L$
γ_{min}	Minimum percentage of solution to be destroyed for ALNS	0.1
γ_{max}	Maximum percentage of solution to be destroyed for ALNS	0.4
φ_w	Weight of the waiting time for calculating the insertion cost of a request into a tour	0.2

Table 5.20: Table of all parameters used for the computational results.

Chapter 6

Tackling a VRP challenge to redistribute scarce equipment within time windows using metaheuristic algorithms

Reprinted by permission from Springer Nature Customer Service Centre GmbH: Springer.
EURO Journal of Transportation and Logistics Tackling a VRP challenge to redistribute scarce equipment within time windows using metaheuristic algorithms. A. Kheiri, A.G. Dragomir, D. Mueller, J. Gromicho, C. Jagtenberg, & J.J. van Hoorn

https://doi.org/10.1007/s13676-019-00143-8

Abstract This paper reports on the results of the VeRoLog Solver Challenge 2016 - 2017: the third solver challenge facilitated by VeRoLog, the EURO Working Group on Vehicle Routing and Logistics Optimization. The authors are the winners of second and third places, combined with members of the challenge organizing committee. The problem central to the challenge was a rich VRP: expensive and therefore scarce equipment was to be redistributed over customer locations within time windows. The difficulty was in creating combinations of pickups and deliveries that reduce the amount of equipment needed to execute the schedule, as well as the lengths of the routes and the number of vehicles used. This paper gives a description of the solution methods of the above-mentioned participants. The second place method involves sequences of 22 low level heuristics: each of these heuristics is associated with a transition probability to move to another low level heuristic. A randomly drawn sequence of these heuristics is applied to an initial solution, after which the probabilities are updated depending on whether or not this sequence improved the objective value, hence increasing the chance of selecting the sequences that generate improved solutions. The third place method decomposes the problem into two independent parts: first, it schedules the delivery days for all requests using a genetic algorithm. Each schedule in the genetic algorithm is evaluated by estimating its cost using a deterministic routing algorithm that constructs feasible routes for each day. After spending 80 percent of time in this phase, the last 20 percent of the computation time is spent on Variable Neighborhood Descent to further improve the routes found by the deterministic routing algorithm. This article finishes with an in-depth comparison of the results of the two approaches.

6.1 Introduction

The VeRoLog Solver Challenge 2016 - 2017 was facilitated by VeRoLog, the EURO Working Group on Vehicle Routing and Logistics Optimization and organized in cooperation with ORTEC. This challenge inspired a total of 28 teams, worldwide, to participate. Table 6.1 lists the institutions to which the participants were affiliated. The first, second and third prize were awarded during the VeRoLog conference 2017 in Amsterdam[1].

ORTEC designed and ran the third solver challenge organized by the VeRoLog. The previous two editions took place in 2014[2] and 2015[3] and were designed and run by PTV. This paper concerns the third challenge[4][5].

The routing problem central to the VeRoLog Solver Challenge 2016 - 2017 was based on a real-life problem for one of ORTEC's clients. From this real-life problem, a few aspects were selected that lead to a new research topic in vehicle routing optimization.

The problem concerns a large cattle improvement company, that must regularly measure the milk quality at a number of farms (customers). This requires special measuring tools, which have to be delivered to the customers at their request. After the measurement, typically a few days later, the tools have to be picked up again. The scheduling of these deliveries to days and the routing for the planned deliveries and pickups are the issues to address in this challenge.

The problem of this challenge is deterministic and revolves over a long horizon, meaning that the scheduling of the individual delivery dates has a large impact on solution quality. The problem was first introduced by Gromicho et al. [70] and in the context of the challenge by Dullaert et al. [47]. It combines the following decisions:

- On which day each delivery request should be served. This leads to a second automatic decision: on which day it should be picked up again.
- For each day in the planning horizon, which deliveries and which pickups to combine in each route - and in what sequence.

The main objective is to serve all requests at a minimum cost (see Section 6.2.1), subject to the following constraints:

- The number of items (equipment or tools) available per type is limitative, making them scarce and forcing reuse.
- Items can be loaded at the depot or at a customer, but the items on board must always satisfy the capacity constraints.

We may recognize some of the aspects of the problem in the available literature. The routing part of our problem consists of unpaired pickups and deliveries which is central to the Pickup and Delivery VRP (PDVRP) as defined in Parragh et al. [122], where they argued that this problem had received the least attention of all problems which they have surveyed. Only one paper was known to the authors of Parragh et al. [122] addressing unpaired pickups and deliveries being repositioned by multiple vehicles, namely Dror et al. [45]. The problem being addressed there resembles in many aspects the routing of a specific day in our challenge, including the scarcity of the goods and the possibility of them being repositioned (shared electric cars). Dror et al. [45] developed an exact methodology based on a mixed integer programming model, but the applicability was limited to very small instances.

[1] https://verolog2017.sciencesconf.org/
[2] www.euro-online.org/websites/verolog/verolog-solver-challenge-2014/
[3] www.euro-online.org/websites/verolog/news/verolog-solver-challenge-2015/
[4] www.euro-online.org/websites/verolog/news/verolog-solver-challenge-2016-2017/
[5] https://verolog2017.ortec.com/

Table 6.1: The participants' institutions. If known, the names of the participants are also mentioned.

Team	Institution	Country	Participants
–	CIRRELT	Canada	
–	Erasmus University Rotterdam	The Netherlands	
–	Ho Chi Minh City International University	Vietnam	
–	Koc University	Turkey	
akhe	Lancaster University	United Kingdom	Ahmed Kheiri
–	Los Andes University	Colombia	
–	Maastricht University	The Netherlands	
Success	Shiraz University	Iran	Morteza Keshtkaran
–	Tages s.c.	Italy	
–	Tata Consultancy Services	Unknown	
–	Universidad de Buenos Aires	Argentina	
Tau17	Tel Aviv University	Israel	Yael Arbel Dafna Piotro Alona Raucher Tal Raviv
MLS	Universidad de Los Andes	Colombia	María Ángel Lucia Paris
–	Universidad Torcuato Di Tella	Argentina	
–	University College Dublin	Ireland	
–	University of Groningen	The Netherlands	
–	University of Laguna	Spain	
–	University of Pavia	Italy	
–	University of Pisa	Italy	
mjg	University of the Federal Armed Forces Hamburg	Germany	Martin Geiger
–	University of Twente	The Netherlands	
ADDM	University of Vienna and Vienna University of Technology	Austria	Alina Dragomir David Müller
–	Vrije Universiteit Amsterdam	The Netherlands	
–	Warwick University	United Kingdom	
SunBeams	Zaporizhzhya National University	Ukraine	Igor Kozin Sergey Borue Olena Kryvtsun

Close to the time of writing Parragh et al. [122], Montané and Galvão [108] designed the first metaheuristic for the routing of unpaired pickups and deliveries by multiple vehicles, but no repositioning was considered: each vehicle departs from a single depot with the loads to deliver and returns to the same depot with the picked up loads. This seems to be the setting generally followed by subsequent research.

Battarra et al. [11] consider the VRP with simultaneous pickup and delivery demands and attribute its origin to the work of Min [104] which considered simultaneous pickup and delivery in the context of a public library. The study of Min [104], which seems to be overlooked by Parragh et al. [122], also assumes that the deliveries come from a depot and the pickups return to the same depot, hence disallows relocation. It is worthy to mention that their proposed mathematical model includes a parameter to model the traffic congestion, which makes sense since the setting of their study was a large urban area. This is simply done by adjusting the travel times on the arcs. However, just as in the case of our challenge, no detailed consideration is given to the time aspect and the travel times appear only in the objective function.

Another aspect that received substantial attention is that of pickups and deliveries of individual items, which is central to so-called dial-a-ride models Cordeau and Laporte [29]. These models deal with the transportation of people and tend to focus on the journey of each item (person), which starts at pickup and finishes at delivery. Some modifications to this theme include the usage of transfer points as in Masson et al. [101] and anticipation on expected return transports as in Schilde et al. [147]. The latter adds a coordination aspect that relates to the subject of our challenge in the sense that the items being delivered need to be picked up again; however, their items (people) are not 'reusable' nor 'exchangeable'.

Simultaneous pickups and deliveries are also central in the literature on routing within reverse logistics, see Dethloff [39]. Researchers in this field tend to focus on the transportation of reusable packaging, which seemingly relates to our relocation of scarce items. However, the relocation - and hence reuse - is out of scope of these models: packaging is brought back to depots to be reused in subsequent, not yet planned, routes. Typically, no careful inventory of packages is kept since they are not perceived as scarce.

In general, routing papers assume the pickup and delivery orders to be a priori defined and they should be served on the day of planning.

One area where we do find the multiple day aspect and the choice of delivery day is in *inventory routing* problems as surveyed by Coelho et al. [27]. The main difference is that in inventory routing models the inventory is managed at the delivery location and typically *estimated* by the routing operator. Goods are not relocated, nor they are scarce. Inventory inside the vehicles is simple to manage since loading takes place at the depot only, whereas unloading takes place during the route. The intrinsic difficulty of such models comes from the combination of demand forecasting (leading to inventory estimation) with routing and scheduling decisions. These scheduling and routing aspects are present in our problem; however, classical inventory routing models lack the relocation and reuse of goods.

Finally, we mention the research on relocation planning for bike sharing systems (see Fishman [56] for a survey). In this problem, the goal is to transport bikes from locations where they tend to accumulate to those where they are sought. The decision on what level of bike inventory to maintain at every location greatly depends on estimations of flows between locations, which makes this problem highly stochastic and complex. This complexity is mentioned in, for example, Schuijbroek et al. [148], which notes that finding provably optimal solutions is practically intractable. Perhaps that is why this domain is quite rich in approaches, including Variable Neighborhood Search heuristics in Rainer-Harbach et al. [130], branch-and-cut algorithm in Raviv and Kolka [133], cluster-first route-second heuristic in Schuijbroek et al. [148], and simulation optimization in Jian et al. [84]. Furthermore, these models are sometimes extended with different aspects of the problem, such as combining staff-based vehicle redistribution and real-time price incentives for customers Pfrommer et al. [127].

It is interesting to mention that the repositioning by multiple vehicles addressed by Dror et al. [45] was also in a context of vehicle sharing, in their case electric cars, which are indeed much more scarce than the bikes being shared.

We believe that the combination of decisions found in the problem of this challenge, which is a practical problem faced by some of ORTEC's customers, is quite unique and may lead to subsequent research. Also, the richness of the objective function contributes to the versatility and difficulty of this problem: emphasizing tool minimization leads to different methodologies being effective than emphasizing total distance. In order to support future research with benchmarking, the challenge site remains alive after the challenge has ended.

The problem, including the format of the instance and solution files, and the challenge rules are described by the challenge team (Gerhard Post, Daan Mocking, Jelke van Hoorn, Caroline Jagtenberg and Joaquim Gromicho) which can be found on the competition website. Note that this paper contains a recap of the problem description and the challenge rules.

The challenge problem is a simplification of a richer version found by ORTEC's clients, which includes among other features multiple resource capabilities, heterogeneous fleet, multiple depots, route synchronization, tight time-windows and adherence to working and driving time directives. Furthermore, the real problem as solved by ORTEC for its clients includes an additional phase which is not part of this challenge: the scheduling and routing of *inspectors*, who should visit the farms while the equipment is present. Each inspector has his or her own home base, skills and periods of availability, which makes the whole problem an even greater challenge. The problem instances used during the VeRoLog challenge can be downloaded from the competition website.

As a curiosity, we mention that a group of undergraduate Business Analytics students at the Vrije Universiteit in Amsterdam ran a preliminary version of the challenge composed of smaller and less restrictive instances, as a case study during a course taught by Joaquim Gromicho. During this case study it became evident that there are at least two main ways to tackle the problem: route first, schedule second or schedule first, route second. Those that focus first on routing day by day and just schedule to meet restrictions were the first to obtain solutions to all instances, while those that develop a sophisticated scheduling of the visits prior to routing took longer to design and implement their algorithms but reached higher solution quality.

The remainder of this paper is structured as follows. Section 6.2 provides a description of the tackled problem. Section 6.3 describes the algorithms that ended up second and third in the ranking of the challenge. Section 6.4 presents the results, and Section 6.5 describes the conclusions.

6.2 Problem description

The problem discussed in this paper consists in planning of deliveries and pickups of tools to customers at their requests to achieve objectives under the presence of several constraints.

The problem consists of a set C of *customers*, a set T of *tool kinds*, and a set R of *tool requests*. A request $r(n, t, c, d, w)$ asks for $n \in \mathbb{N}$ tools of one kind $t \in T$, that need to be present at customer $c \in C$ for a given number of consecutive days d. The delivery of the tools has to fall within a certain time window w given in full days. If a customer requires several kinds of tools, this means separate requests are made. Note that all requests are known at the moment the planning is made. The tools of the request have to be picked up by one vehicle the day after the request is completed, i.e., precisely $d + 1$ days after the tools were delivered.

Each problem instance has one depot location where all tools are located at the beginning and end of the planning horizon. A vehicle can load a tool at the depot and unload it at a customer. Alternatively, after the first day, a vehicle can also pick up a tool at customer c_1 and deliver it to customer c_2 *without* visiting the depot in between. Vehicles can also visit the depot multiple times per day, leaving tools and picking them up later for redistribution. To avoid the need for synchronization between the vehicles on the same day, only the vehicle that left the tool

at the depot may pick it up again. Relaxing this constraint would force detailed arrival moments at the depot to be modeled in order to enable checking that tools already brought by a vehicle are available to be taken by another during the same day. If the tools cannot be exchanged between vehicles on the same day, the arrival and departure times of vehicles at the depot do not need to be synchronized among the vehicles. This restriction does not apply if the tool is being picked up on a later day. All vehicles must start and end their day at the depot. If a vehicle visits the depot during the day, the vehicle route consists of multiple tours.

Each tool kind has a certain size, and the available vehicles all have the same capacity with respect to the tool sizes. During any part of a route, the total size on board of a vehicle may not exceed its capacity. There is no maximum amount on the number of vehicles one can use (although vehicles are not free).

Every problem instance provides coordinates for each customer as well as a depot, allowing the participants to compute the Euclidean distance between any two locations. There is an upper bound on the distance that a vehicle can travel in one day.

6.2.1 Objectives

All requests must be satisfied, and the objective is to minimize a cost function that consists of four parts: (1) costs per distance traveled, (2) costs for using a vehicle for a day, (3) costs for using a vehicle at all, and (4) a cost per tool, depending on the tool kind. The latter was inspired by the real-life problem that this challenge originated from: the tools involved are in fact rather expensive, and hence it is worthwhile to investigate whether routes can be created which allow for fewer tools to be purchased. Each problem instance includes a definition for costs (1) - (4), which means that different problem instances emphasize different aspects of this problem. This makes it more challenging for the participants to come up with one algorithm that tackles all problem instances.

For the challenge rules, including how algorithms are evaluated, we refer the reader to 6.6.

6.3 Competitors' algorithms

Search methodologies are at the core of decision support systems, particularly while dealing with computationally difficult optimization problems. The cutting-edge methods are often tailored for a specific problem domain by the experts in the area. Such systems are custom-made and, often, costly to build. When exact methods cannot be applied, practitioners and researchers resort to heuristics, which are 'rule of thumb' methods for solving a given optimization problem. There is a growing interest towards more general, cheaper and intelligent methods. Metaheuristics [152] and hyper-heuristics [20] are such methodologies that automate the search process. This section presents the methods that won the runner-up and the second runner-up prizes in the VeRoLog Solver Challenge 2016 - 2017. The former method employs a hyper-heuristic technique and the latter applies an improved genetic algorithm metaheuristic.

6.3.1 A sequence-based selection hyper-heuristic (Team: akhe)

The main components of selection hyper-heuristics as identified in [20] are (i) *heuristic selection* which selects a low level heuristic from a pre-defined set of low level heuristics and applies it to a candidate solution at each decision point; and (ii) *move acceptance* which decides whether to continue with the newly generated solution or the previous solution. A new field of hyper-heuristic methods embedding data science techniques has recently been developed [6]. Experiments on a hyper-heuristic benchmark framework [87], urban transit route design problem [1], wind farm layout optimization problem [167], high school timetabling problem [88] and on water distribution optimization problem [89] have shown that applying a sequence of low level

heuristics can potentially improve the quality of solutions more than those that simply select and apply a single low level heuristic.

Overall model

The competing method that took the second place uses a method that applies sequences of heuristics. To achieve this, each low level heuristic is associated with two probabilities: a transition probability to move to another low level heuristic including itself, and another to determine whether to terminate the sequence of low level heuristics at this point.

Let $[llh_0, llh_1, \ldots, llh_{n-1}]$ be the set of low level heuristics. A transition matrix ($Transition$) of size $n \times n$ stores scores for each of the n low level heuristics, from which we calculate the probabilities of moving from one low level heuristic to another (by normalizing the scores given in the matrix). We also define another matrix referred to as sequence status matrix ($Status$) of size $n \times 2$ which specifies scores for each of the n low level heuristics in one of two options: *add* and *end*.

Initially, elements in both matrices ($Transition$ and $Status$) are assigned the value 1. Figure 6.1 shows the initial score values of the two matrices for $n = 4$ low level heuristics.

$Transition$

	llh_0	llh_1	llh_2	llh_3
llh_0	1	1	1	1
llh_1	1	1	1	1
llh_2	1	1	1	1
llh_3	1	1	1	1

$Status$

	add	*end*
llh_0	1	1
llh_1	1	1
llh_2	1	1
llh_3	1	1

Figure 6.1: Initial score values of the two matrices for $n = 4$ low level heuristics

At first, a randomly selected low level heuristic (assume llh_2 is selected) is added to the sequence of low level heuristics. [SEQUENCE: llh_2]

The $Status$ matrix is used to decide whether another low level heuristic will be selected and added to the sequence or the sequence will end at this point. To make one of these two choices, a roulette wheel selection method is applied. For llh_2, the probability of adding another low level heuristic is 1/2. Assume that the chosen status is *add*. [SEQUENCE: llh_2,]

The decision now is to add another low level heuristic to the sequence. This will be chosen by a selection procedure based on the roulette wheel selection strategy. In our example, the probability of selecting any low level heuristic, given that the recently added low level heuristic was llh_2, is 1/4. Assume that the chosen low level heuristic is llh_1. [SEQUENCE: llh_2, llh_1]

The $Status$ matrix is used again to decide whether another low level heuristic will be selected and added to the sequence or the sequence will end at this point. For llh_1, which is the recently added low level heuristic, the probability of adding another low level heuristic to the sequence is 1/2. Assume that the chosen status is *end*. [SEQUENCE: llh_2, llh_1].

In this case the current sequence of low level heuristics ($[llh_2, llh_1]$) will be applied to the candidate solution in this given order to generate a new solution.

If the new solution improved over the best solution, the scores in both matrices for the relevant low level heuristics are increased by 1 as a reward. This is illustrated in Figure 6.2. If the new solution does not improve the quality of the best solution in hand, then the scores in the matrices will not be updated. This way we only increase the chance of selecting the sequences that generate improved solutions.

Transition

	llh_0	llh_1	llh_2	llh_3
llh_0	1	1	1	1
llh_1	1	1	1	1
llh_2	1	2	1	1
llh_3	1	1	1	1

Status

	add	*end*
llh_0	1	1
llh_1	1	2
llh_2	2	1
llh_3	1	1

Figure 6.2: Updated score values of the two matrices

We are now at llh_1, and we continue with the same strategy to construct and apply the next sequence of heuristics using the updated scores.

The move acceptance method used in this work is Record-to-Record Travel (RRT) move acceptance criterion [46]. The idea of RRT is based on the simple notion that any new solution, which is not much worse than the best solution recorded, is accepted. A candidate solution is in the form of a three-dimensional array (days × routes × visits).

Note that the quality of a given solution is evaluated using the main objective to be minimized and an estimated (secondary) objective depending on which cost type (described in Section 6.2.1) of a given problem instance is set highest. As an example, if the main objective is to minimize the number of vehicles, then the algorithm will locate the day that has the most number of vehicles (routes) running and the secondary objective becomes the trip distance of the route that has the least total distance on that day. Similarly for the number of used vehicles per day, the algorithm attempts to minimize the number of vehicles used per day as the main objective, and the trip distance of the route that has the least total distance as the secondary objective.

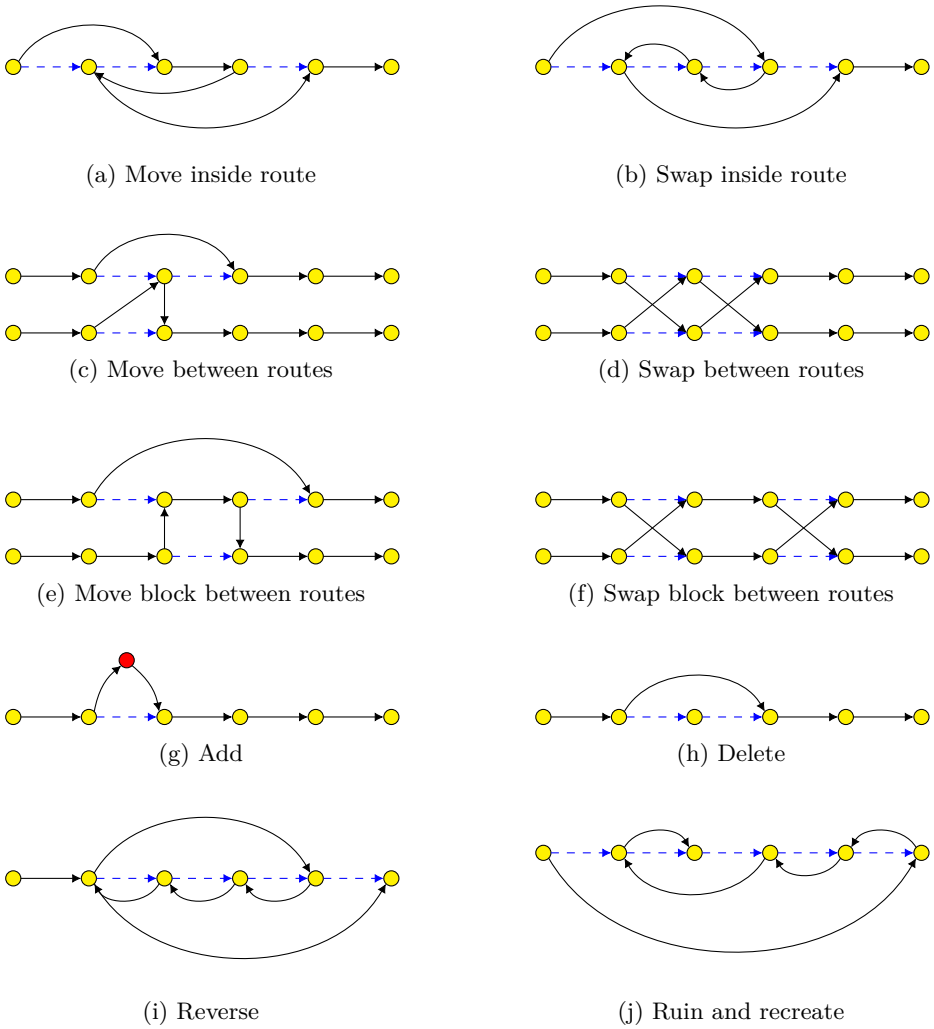
The organizers of the challenge provided a set of feasible instances, and confirmed that a feasible solution to the problem can be achieved by selecting for each visit (delivery or pickup) one vehicle to carry out only this visit. Following this, we developed a greedy algorithm to construct an initial feasible solution. However, the implemented simple greedy algorithm, which runs in milliseconds, often yields a poor quality solution requiring further enhancement.

Low level heuristics

The sequence-based selection hyper-heuristic approach in this work controls a set of 22 low level heuristics to improve the quality of an initially generated solution. The low level heuristics are grouped into the following 6 categories: *move*, *swap*, *reverse*, *add*, *delete* and *ruin and recreate*.

Move low level heuristics

- **LLH0:** Moves a visit (delivery, pickup or depot) into a new location inside a route (Figure 6.3a).
- **LLH1:** Selects two random routes, same day, and a random position on each route. The visit in the first position is moved into the second position on the second route (Figure 6.3c).
- **LLH2:** Selects two random routes from different days and a random position on each route. The visit in the first position is moved into the second position on the second route. Corresponding visits (pickup or delivery) will be moved to satisfy the time window constraint.
- **LLH3:** Moves a block of visits, that is a set of consecutive visits, into a new location inside a route.



(a) Move inside route (b) Swap inside route

(c) Move between routes (d) Swap between routes

(e) Move block between routes (f) Swap block between routes

(g) Add (h) Delete

(i) Reverse (j) Ruin and recreate

Figure 6.3: Straight arcs are visits in the route, dashed arcs are visits removed after applying the heuristic, curved arcs are visits added after applying the heuristic

- **LLH4:** Moves a block of visits into a randomly selected location from another route in the same day (Figure 6.3e).
- **LLH5:** Moves a block of visits into a randomly selected location from another route in different day. Corresponding visits will be moved to satisfy the time window constraint.
- **LLH6:** Moves a tour, that is visits between two depots, from a route into another route in the same day.
- **LLH7:** Moves a tour from a route into another route in different day. Corresponding visits will be moved to satisfy the time window constraint.

Swap low level heuristics

- **LLH8:** Selects a random route and two random positions and swaps the two visits in these positions (Figure 6.3b).

- **LLH9:** Selects two random routes from a randomly selected day, and a random position on each route and swaps the visits in these positions (Figure 6.3d).
- **LLH10:** Selects two random routes from two different days, and a random position on each route and swaps the visits in these positions. Corresponding visits will be moved to satisfy the time window constraint.
- **LLH11:** Exchanges block of visits inside a route.
- **LLH12:** Exchanges block of visits between two routes both from the same day (Figure 6.3f).
- **LLH13:** Exchanges block of visits between two routes from different days. Corresponding visits will be moved to satisfy the time window constraint.
- **LLH14:** Exchanges two tours in a randomly selected route.
- **LLH15:** Exchanges two tours in two different routes in a randomly selected day.
- **LLH16:** Exchanges two tours in two different routes from different days. Corresponding visits will be moved to satisfy the time window constraint.

Reverse low level heuristics

- **LLH17:** Consists in a chronological reversal of a block of visits in a randomly selected route (Figure 6.3i).

Add low level heuristics

- **LLH18:** Selects a random route and a random position in this route and adds a depot visit into this position (Figure 6.3g).

Delete low level heuristics

- **LLH19:** Deletes a depot visit (Figure 6.3h).

Ruin and recreate low level heuristics

- **LLH20:** Destructs a randomly selected rout generating a partial solution and then reconstructs a complete solution at random (Figure 6.3j).
- **LLH21:** Same as LLH20 but destructs/reconstructs several routes from a randomly selected day.

Additional remarks and conclusions

Although, the ultimate goal of the development of hyper-heuristic methods is to increase the level of generality, by offering methods that have the ability to work on a wide range of optimization problems, still it would be interesting to know the position of hyper-heuristics with respect to other problem-specific solution methods in a particular optimization problem while still being general. In this work, a sequence-based selection hyper-heuristic has been developed which aims to intelligently and effectively control the application of sequences of heuristics as opposed to simple selection of single heuristic. The method effectively exploits the features of the problems on the fly as indicated in [87]. This is a viable approach considering that at different points during the search, different sequences of heuristics may be performing well.

Preliminary experiments did indicate that large low level heuristics at tour (or block of visits) level that tend to move tours around could lead to better results. Of course better understanding of this effect requires further work and much more exploration. Final results of the challenge

suggest that the low level heuristics would need significant adjustment to handle the problem more effectively. Interesting future work might well try to explore features of instances that are correlated with the different objectives defined in Section 6.2.1.

6.3.2 A genetic algorithm metaheuristic (Team: ADDM)

The proposed algorithm decomposes the problem into two independent parts. The first part finds a schedule for the delivery day of each request. The second part finds the routes of the vehicles for each day according to the predetermined schedule.

Scheduling of delivery days

For the scheduling, a genetic algorithm (GA) is used. For a pedagogical introduction on GA see Wall [163]. Hart et al. [74] provide a review on evolutionary scheduling literature. The genome sequence represents the day of delivery for each request. Therefore, the sequence has a length equal to the number of requests. The day specified for each genome has to be within the delivery time window for the request. Since every request needs the tools for a specific amount of days, the corresponding pickup days can be easily determined. As far as the scheduling is concerned, all necessary information is stored in this sequence. Figure 6.4 shows an example of a genome sequence.

Request	1	2	3	4	…														
Day of delivery	1	3	5	6	2	1	1	1	1	5	5	4	3	2	6	2	1	3	4	4

Figure 6.4: An example of a genome sequence for the genetic algorithm

Building a population

For building an initial population, two different approaches are used: a greedy heuristic and a random schedule generator.

The greedy heuristic picks a request and assigns a delivery day that increases costs the least for the overall schedule. Here, a cost estimation is used based on the number of required vehicles and tools. If the resulting sequence is feasible, it is added to the initial population, otherwise the heuristic restarted. Feasibility is determined by checking if the required resources are not exceeding the available limit. To avoid a deterministic heuristic, the order of requests is chosen randomly.

The random schedule generator assigns each request a random day for delivery as long as it is within the allotted time windows. Therefore, the day for the sequence is selected between the first and last possible day for delivery. If the sequence is determined to be feasible, it is added to the initial population.

The size of the population is adapted to the size of the instance, i.e. the number of requests. Depending on the problem instance, we choose between 70 and 300 individuals. Both the greedy heuristic and the random schedule generator initially generate a pool of individuals two times the population size. The better half is kept, the worse half discarded.

Selection, genetic operators and mutation

The parents for reproduction are chosen out of the whole population with decreasing probability the higher the cost of an individual: first, we sort the population by the score of each individual. The population is then separated into three parts: the top half and the next two

quarters. A parent is then chosen with 1/2 probability out of the top half, 1/3 probability out of the less-than-average quarter and 1/6 probability out of the worst quarter. The same procedure is used for the second parent. This way of selecting parents guarantees that individuals with low cost are chosen most of the time, but also allows for less than average candidates to take part in the reproduction step. As a consequence, it takes longer for populations to converge to local optima. Clearly, constructing a selection method is not a rigorous task and there is a lot of freedom in the exact details of how to choose individuals. After some testing we settled on the above described selection method due to its simplicity and because it achieved acceptable results.

For reproduction, a uniform crossover operator is used. Each gene of the child sequence has an equal probability of being selected from one of the parents. Figure 6.5 shows an example of a uniform crossover.

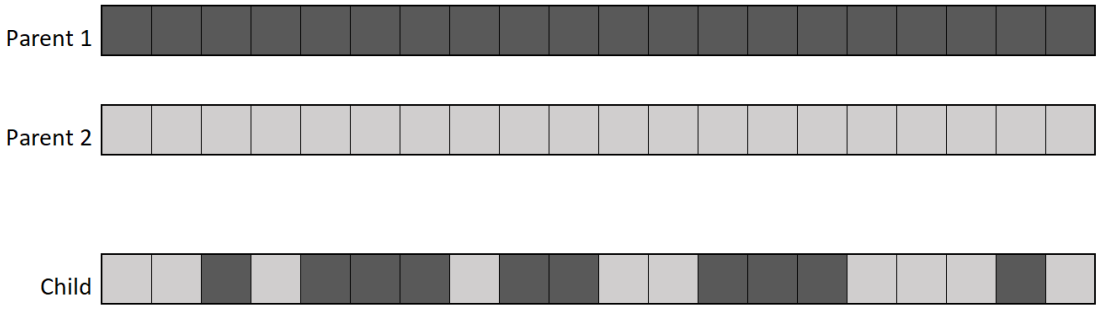


Figure 6.5: An example of a uniform crossover

After the crossover, some mutation might occur: we mutate up to 10 random requests, each having a probability of 1/2 to mutate. The mutation shifts a request to a different delivery day within the available time windows.

The genetic algorithm runs up to 400 generations or until the population 'converges'. We define a population to be converged, if the new generation contains more than 80 per cent of identical individuals compared to the last generation.

Routing and schedule evaluation

To get the exact score of a schedule, the routing for the whole planning horizon has to be computed. The routing algorithm has to run very often, because our genetic approach relies on evaluating the score of a large number of individual schedules in a short amount of time. In particular, the genetic algorithm might encounter the same individual more than once during its run. Therefore, the score evaluation of individual schedules represents the main performance bottleneck of the solver.

These problems can be solved by choosing a deterministic routing algorithm, which provides each schedule with a unique set of routes. As a result, two identical schedules have the same routing and the same score. Once the routes for a schedule are calculated, the score and a hash value of the schedule can be cached in a lookup table. The lookup table provides an easy and fast way of checking if a particular schedule has been encountered before and obtaining its score without running the routing algorithm again. Furthermore, memory use is also greatly improved because it is not necessary to store the routes of each schedule in the population. In the rare event that the exact routes are needed (for example when writing the best known solution to a file), we simply run the deterministic routing algorithm again and obtain the same result as before.

The routing algorithm takes a schedule as input and tries to solve the routing problem for each day individually. Since each vehicle can return to the depot multiple times within a day, each vehicle route can consist of multiple tours. The algorithm is comprised of four stages:

1. A modified parallel savings algorithm [26, 132] with $s_{ij} = k(s_{i0} + s_{j0} - s_{ij})$ where $k > 1$, if i is a pickup and j a delivery of the same tool. Otherwise $k = 1$. Since the problem has the quite unique characteristic of allowing tools to be passed on from one customer to the next, we favor savings where this is the case to minimize tool use. The algorithm is initialized with single requests tours. Each initial tour starts at the depot, fulfills a single pickup or delivery request and then returns to the depot. The savings algorithm is based on merging pairs of tours into increasingly longer tours. For every merge we determine if the capacity constraint is fulfilled by iterating through the arcs where each pickup reduces the remaining capacity and each delivery increases the remaining capacity of the vehicle. Additionally, maximum distance constraints are checked by summing over arc lengths. If constraints are violated, the savings pair is skipped and the next pair is considered.
2. A 2-opt heuristic [35] using best improvement where all combinations of arc pairs are selected and the sub-route between the arcs is reversed. Every time tours are modified we check for capacity and distance constraints.
3. A 3-opt heuristic [95] using best improvement where all combinations of arc triplets are selected. As with the 2-opt heuristic, we check for constraint violations.
4. A best fit decreasing bin packing heuristic [85, 99] to combine tours into vehicles routes with the goal to reduce the amount of vehicles in use. When packing tours into routes the capacity constraint is always fulfilled, as each tour starts and ends at the depot. However, the distance constraint must still be checked.

Stages 2 – 4 use simple cost estimations with different weight parameters for tool use, vehicle number and route length. Tools of the same type are assumed to always be passed on from a customer to the next to minimize the number of tools required from the depot. Depending on the parameter k and the weights, the results of the routing algorithm can vary dramatically for a given schedule. The final version of our solver uses 5 different parameter sets, which put emphasis on particular aspects of the cost function. During the initial stage of the solver, we choose the parameter set which gives the best average score of the initial population.

Post optimization for the routing

Once the genetic algorithm finishes (either due to convergence of the population or due to reaching the time limit) and a candidate schedule has been found, more computationally intensive improvement heuristics can be used to further improve the routes that were determined by the deterministic routing algorithm. Our solver sets aside 20 per cent of the available runtime for this post optimization stage. We use Variable Neighborhood Descent (VND) [73] and the following neighborhoods:

1. *Move* Moves a random node (delivery or pickup) to another position and/or tour and/or vehicle route.
2. *Swap* Swaps two random nodes from randomly selected tours and vehicles.
3. *Tour move* Moves a random tour to another vehicle.
4. *Tour swap* Swaps two random tours from randomly selected vehicles.

The VND terminates after n iterations without improvement in each neighborhood or until available runtime is reached.

Additional remarks and conclusions

The decomposition approach (i.e. 'scheduling first, routing second') was chosen to simplify the problem and accommodate the genetic algorithm. In particular, it naturally leads to an appropriate solution representation of individuals in terms of schedules. The genetic algorithm proves to be a powerful metaheuristic for a problem like this, where sub-optimal but feasible solutions can easily be found. The deterministic routing and schedule evaluation ensures that routing calculations are not needlessly repeated for identical schedules. This results in a great acceleration of the evaluation of the later generations. Such optimization is especially important for the (time) resource restricted challenge. The parameter values that account for the vastly different costs of each instance and the general parameters of the genetic algorithm were set in a trial-and-error fashion. In the end, participation in the all-time-best challenge with strong competition from other teams helped us choose the particular values used in the solver.

6.4 Competition results

6.4.1 All-time-best challenge

Table 6.2 shows the characteristics of the all-time-best instances and the cost of best obtained solutions. Instances are named using two numbers with the prefixes r for requests and d for days. The instances range from 100 to 1000 customers and from 5 to 30 days. The last number indicates which cost type is set highest, with 1: tool cost followed by vehicle cost, 2: tool cost followed by vehicle day cost, 3: vehicle cost, 4: vehicle day cost and 5: distance cost. For example, the Instance VeRoLog_r100d5_1 has 100 requests over 5 days, with tools having the highest cost, and vehicle has the second highest cost. The instances also have between 2 and 5 different tool kinds that have to be distributed.

Table 6.3: The costs of the best five solutions achieved by the competitors and the date the solutions were submitted during the all-time-best challenge

Instance		First Team	Second Team	Third Team	Fourth Team	Fifth Team
r100d5_1	Team	mjg	Success	**akhe**	**ADDM**	Sunbeams
	Cost	1,552,435,049	1,552,437,090	1,552,472,997	1,552,667,702	1,552,674,956
	Date	10/04/2017	12/06/2017	22/03/2017	24/03/2017	13/05/2017
r100d5_2	Team	mjg	**akhe**	Sunbeams	**ADDM**	Success
	Cost	996,709,544	997,125,464	997,131,853	997,536,075	997,975,026
	Date	17/04/2017	22/03/2017	19/05/2017	24/03/2017	13/03/2017
r100d5_3	Team	mjg	Success	**ADDM**	Sunbeams	EquipoMLS
	Cost	119,957,689	119,999,604	120,174,619	125,484,960	141,027,538
	Date	17/04/2017	15/06/2017	08/03/2017	13/05/2017	30/05/2017
r100d5_4	Team	Success	**ADDM**	mjg	Sunbeams	EquipoMLS
	Cost	1,359,088,350	1,359,171,004	1,388,155,986	1,506,904,422	1,599,316,230
	Date	12/03/2017	10/03/2017	13/04/2017	03/05/2017	30/05/2017
r100d5_5	Team	mjg	Success	**ADDM**	Sunbeams	EquipoMLS
	Cost	300,125,049,016	302,122,548,017	302,145,047,015	312,987,048,016	324,679,054,017
	Date	17/04/2017	15/06/2017	02/03/2017	03/05/2017	26/05/2017
r100d10_1	Team	mjg	Success	Sunbeams	**ADDM**	VeRoLog050
	Cost	1,313,786,538	1,313,800,238	1,313,843,042	1,383,818,456	1,404,313,470
	Date	05/03/2017	17/06/2017	19/05/2017	09/03/2017	03/03/2017
r100d10_2	Team	mjg	Success	Sunbeams	VeRoLog050	**ADDM**
	Cost	1,555,438,898	1,555,866,637	1,556,076,294	1,608,781,502	1,616,279,307
	Date	14/02/2017	15/04/2017	19/05/2017	02/03/2017	09/03/2017

Continued on next page

Table 6.3 – *Continued from previous page*

Instance		First Team	Second Team	Third Team	Fourth Team	Fifth Team
	Team	mjg	Success	**ADDM**	Sunbeams	TeamTau2017
r100d10_3	Cost	155,316,178	155,451,452	155,606,071	155,859,870	156,154,622
	Date	13/04/2017	30/03/2017	09/03/2017	19/04/2017	02/06/2017
	Team	Success	mjg	**ADDM**	TeamTau2017	goc-ar
r100d10_4	Cost	1,114,888,217	1,152,790,895	1,192,096,909	1,350,637,725	1,351,320,617
	Date	09/05/2017	28/04/2017	06/03/2017	02/06/2017	09/12/2016
	Team	Success	mjg	**ADDM**	Sunbeams	goc-ar
r100d10_5	Cost	43,871,262,004	43,901,664,004	45,036,866,004	52,393,174,005	52,474,576,006
	Date	23/04/2017	13/04/2017	10/03/2017	28/04/2017	05/12/2016
	Team	mjg	Sunbeams	Success	VeRoLog050	**ADDM**
r500d15_1	Cost	3,256,089,143	3,288,333,346	3,346,228,685	3,487,816,461	3,506,242,192
	Date	03/05/2017	09/06/2017	20/04/2017	03/03/2017	06/03/2017
	Team	mjg	Sunbeams	Success	**ADDM**	VeRoLog050
r500d15_2	Cost	3,800,352,751	3,811,482,643	3,888,199,515	3,951,378,786	4,052,471,132
	Date	03/05/2017	14/06/2017	13/04/2017	07/03/2017	03/03/2017
	Team	mjg	**ADDM**	goc-ar	Success	TeamTau2017
r500d15_3	Cost	402,839,699	454,318,339	511,857,243	558,690,479	607,429,601
	Date	03/05/2017	07/03/2017	29/12/2016	02/04/2017	02/06/2017
	Team	mjg	**ADDM**	goc-ar	TeamTau2017	EquipoMLS
r500d15_4	Cost	2,807,990,462	2,892,747,784	3,652,534,665	3,732,841,802	4,051,526,171
	Date	03/05/2017	02/03/2017	31/12/2016	02/06/2017	01/06/2017
	Team	mjg	**ADDM**	goc-ar	TeamTau2017	VeRoLog050
r500d15_5	Cost	251,378,880,010	259,677,305,009	306,490,210,012	312,221,625,011	330,726,465,011
	Date	03/05/2017	02/03/2017	17/12/2016	02/06/2017	03/03/2017
	Team	mjg	Sunbeams	**akhe**	**ADDM**	VeRoLog050
r1000d25_1	Cost	7,004,087,706	7,166,663,786	7,469,954,246	7,494,227,661	7,546,340,291
	Date	03/05/2017	05/06/2017	14/04/2017	12/03/2017	02/03/2017
	Team	mjg	Sunbeams	NSA	**ADDM**	VeRoLog050
r1000d25_2	Cost	6,486,405,100	6,811,349,274	7,177,503,250	7,229,613,701	7,232,820,503
	Date	03/05/2017	05/06/2017	12/05/2017	06/03/2017	02/03/2017
	Team	mjg	**ADDM**	goc-ar	EquipoMLS	TeamTau2017
r1000d25_3	Cost	207,087,083	239,642,618	246,934,382	298,214,337	308,695,312
	Date	03/05/2017	10/03/2017	09/12/2016	03/06/2017	02/06/2017
	Team	mjg	**ADDM**	goc-ar	TeamTau2017	EquipoMLS
r1000d25_4	Cost	5,598,405,178	6,205,511,061	7,553,650,631	8,204,868,671	8,857,593,182
	Date	03/05/2017	03/03/2017	09/12/2016	02/06/2017	08/06/2017
	Team	mjg	**ADDM**	goc-ar	TeamTau2017	EquipoMLS
r1000d25_5	Cost	161,446,120,006	174,254,946,006	196,592,076,007	218,038,116,009	234,970,338,008
	Date	03/05/2017	03/03/2017	09/12/2016	02/06/2017	08/06/2017
	Team	mjg	Sunbeams	Success	**ADDM**	Eva
r1000d30_1	Cost	5,220,068,560	5,545,670,163	5,572,741,083	5,919,953,818	6,029,750,153
	Date	03/05/2017	05/06/2017	14/04/2017	06/03/2017	16/05/2017
	Team	mjg	Sunbeams	VeRoLog050	Success	**ADDM**
r1000d30_2	Cost	5,181,409,255	5,363,167,525	5,696,835,520	5,703,651,327	5,736,716,754
	Date	03/05/2017	10/06/2017	02/03/2017	13/04/2017	06/03/2017
	Team	mjg	**ADDM**	EquipoMLS	TeamTau2017	goc-ar
r1000d30_3	Cost	187,843,389	219,104,500	282,173,474	284,664,999	285,062,660
	Date	03/05/2017	12/03/2017	29/04/2017	02/06/2017	05/12/2016

Continued on next page

Table 6.3 – *Continued from previous page*

Instance		First Team	Second Team	Third Team	Fourth Team	Fifth Team
r1000d30_4	Team	mjg	**ADDM**	goc-ar	TeamTau2017	Success
	Cost	4,562,837,156	5,139,734,143	6,157,957,481	6,367,255,071	6,644,215,518
	Date	03/05/2017	02/03/2017	09/12/2016	02/06/2017	12/06/2017
r1000d30_5	Team	mjg	**ADDM**	goc-ar	Success	TeamTau2017
	Cost	257,497,762,007	287,103,606,008	353,163,918,012	367,894,412,012	380,128,952,013
	Date	03/05/2017	03/03/2017	31/12/2016	12/06/2017	02/06/2017

While team *mjg* - who won the finals - is frequently at the top of the ranking, there is an example where this team is outperformed[6] by team *ADDM*, who won third prize in the finals. This happens, for the problem instance VeRoLog_r100d5_4 (see Table 6.3), consisting of 100 requests and a 5 day planning horizon, where *ADDM* and *mjg* provided the second and third best solution respectively. For the other instances where *mjg* did not provide the best solution it provided the second best solution. It is still possible to upload solutions to the all-time-best challenge.

6.4.2 Restricted resources challenge

Based on the challenge ranking system, the organizers selected potential finalists, and verified that their reported results could have realistically been produced by their submitted algorithms. This was done by running the algorithms on the same ORTEC-late instances and random seeds. Eventually, three participants were selected as finalists for the second part of the challenge.

Figures 6.6 to 6.10 show the performance variation of all competing methods on all five versions of VeRoLog_late_r1000d25 dataset. mjg achieved the best results in all instances. akhe performs better than ADDM on the first three versions, but ADDM found slightly better results compared to akhe on the last two versions (i.e. when vehicle day cost and distance cost are highly penalized, respectively). The same can be observed with the other instances.

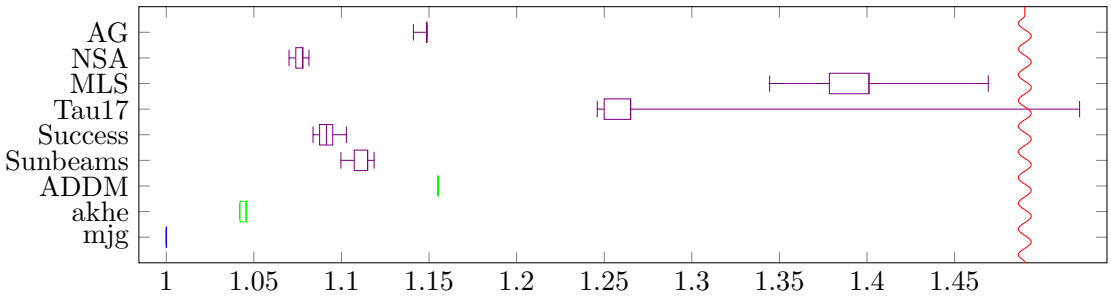


Figure 6.6: VeRoLog_late_r1000d25_1

Figure 6.11 shows the mean rank of the teams that submitted in the restricted resources challenge (on the so-called "ORTEC late instances"). The figure shows a large gap between the third and fourth place, which lead to the organizer's decision to allow precisely three participants in the finale.

Table 6.4 shows the characteristics of the hidden instances and the cost of the current best-known solutions. As before, the last integer in the instance name denotes the type of most penalized cost as described in 6.4.1.

Table 6.5 summarizes the results. We performed Mann-Whitney-Wilcoxon test [90, 53] with a 95% confidence level to compare pairwise performance variations of two given competing methods

[6] At the time of writing this article, after the all-time-best challenge has ended.

Table 6.2: The characteristics of the all-time-best instances and the cost of best obtained solutions

Instance	Customers	Tools	Capacity	Max distance	Best solution	Obtained by
VeRoLog_r100d5_1	100	3	50	20000	1,552,435,049	mjg
VeRoLog_r100d5_2	100	2	40	20000	996,709,544	mjg
VeRoLog_r100d5_3	100	4	40	20000	119,957,689	mjg
VeRoLog_r100d5_4	99	5	30	15000	1,359,088,350	Success
VeRoLog_r100d5_5	100	2	45	16000	300,125,049,016	mjg
VeRoLog_r100d10_1	100	3	50	16000	1,313,786,538	mjg
VeRoLog_r100d10_2	100	5	35	20000	1,555,438,898	mjg
VeRoLog_r100d10_3	100	2	40	17000	155,316,178	mjg
VeRoLog_r100d10_4	100	4	45	15000	1,114,888,217	Success
VeRoLog_r100d10_5	98	3	35	16000	43,871,262,004	Success
VeRoLog_r500d15_1	494	4	35	15000	3,256,089,143	mjg
VeRoLog_r500d15_2	491	3	35	20000	3,800,352,751	mjg
VeRoLog_r500d15_3	490	2	45	15000	402,839,699	mjg
VeRoLog_r500d15_4	487	3	35	17000	2,807,990,462	mjg
VeRoLog_r500d15_5	488	3	45	15000	251,378,880,010	mjg
VeRoLog_r1000d25_1	950	2	45	15000	7,004,087,706	mjg
VeRoLog_r1000d25_2	943	4	35	16000	6,486,405,100	mjg
VeRoLog_r1000d25_3	944	3	50	17000	207,087,083	mjg
VeRoLog_r1000d25_4	949	4	30	17000	5,598,405,178	mjg
VeRoLog_r1000d25_5	951	3	50	16000	161,446,120,006	mjg
VeRoLog_r1000d30_1	940	5	40	15000	5,220,068,560	mjg
VeRoLog_r1000d30_2	942	4	35	15000	5,181,409,255	mjg
VeRoLog_r1000d30_3	945	4	40	20000	187,843,389	mjg
VeRoLog_r1000d30_4	930	4	40	16000	4,562,837,156	mjg
VeRoLog_r1000d30_5	948	3	35	16000	257,497,762,007	mjg

statistically. The following notations are used: Given competing method A_1 versus competing method A_2, (i) $A_1 > (<) A_2$ denotes that A_1 (A_2) is better than A_2 (A_1) and this performance variance is statistically significant, (ii) $A_1 \simeq A_2$ indicates that there is no statistically significant difference between A_1 and A_2.

Overall, it is clear that there is an unambiguous hierarchy: mjg (mean rank 1.02) performs better than akhe (rank 2.16) and akhe performs better than ADDM (rank 2.82). However, there are some exceptions to this and we focus on some differences between the second and third place. For the instances of type 4 (highest cost are of the type vehicle day) and type 5 (highest cost are of the type distance) the difference in performance between akhe and ADDM is less significant. This can also be observed in the late instances submitted by the participants. This suggests that the submitted solvers performed in a stable and consistent manner. With regards to instances with high tool cost (represented by type 1 and 2), akhe performed better than ADDM. One particular characteristic of the challenge problem is that tool cost can be avoided if tools are not directly delivered to a customer from the depot but transferred between customers instead. The results of the late and hidden instances indicate that ADDMs solver prioritizes distance cost over tool cost and therefore generally performs worse in such cases.

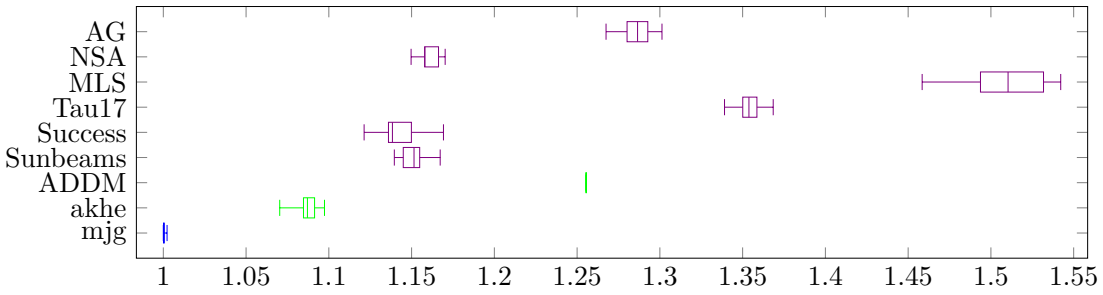


Figure 6.7: VeRoLog_late_r1000d25_2

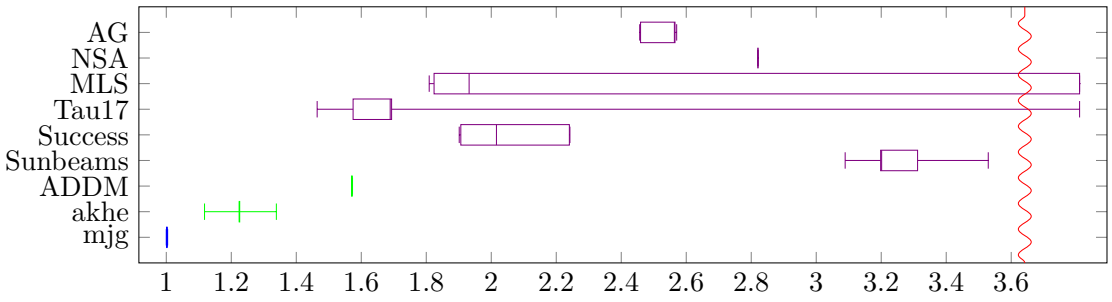


Figure 6.8: VeRoLog_late_r1000d25_3

6.4.3 Convergence comparison

In this section, we compare akhe and ADDM algorithms on all five versions of VeRoLog_late_r1000d25 dataset. Our experiments are performed on Intel(R) Core(TM) i7-6500U CPU with a 2.50GHz, 2.60GHz and 8.00GB of RAM. The convergence curves of the two algorithms on the selected instances is illustrated in Figure 6.12. In all the cases, the hyper-heuristic method (akhe) improves the quality of the candidate solutions at the beginning of the search process rapidly, and then the process slows down when reaching the local optimum. The employment of the sequence-based strategy seems to lead the search to jump from local optima in some cases (e.g. VeRoLog_late_r1000d25_5), allowing further improvement to the quality of the solutions. Note that the plotted objective values for ADDM start slightly later than akhe. This is due to the initial phase of the solver: before starting the genetic algorithm, the initial population is evaluated using the routing algorithm with different parameter sets. Each parameter set is geared towards different cost priorities. The parameter set with the best average score is then chosen for the rest of the evolution. As this takes up some time, the objective value curves start only once this phase is completed.

6.5 Conclusion

In this paper, two heuristic algorithms were proposed to solve a vehicle routing problem with inter-route and intra-route challenges. This problem was the topic of a recent competition, referred to as VeRoLog Solver Challenge 2016 - 2017. It is based on a real-life problem of a cattle improvement company that combines routing, scheduling and inventory aspects. Instances differed, apart from size, in cost penalties, making the problem potentially relevant from a multi-objective point of view. 28 teams participated worldwide in the all-time-best challenge that ran for 8 months and 9 teams participated in a restricted resources challenge.

Academic challenges, such as the one described in this paper, have the pleasant property that algorithms can be compared objectively. Since it is ensured that (1) all researchers are working

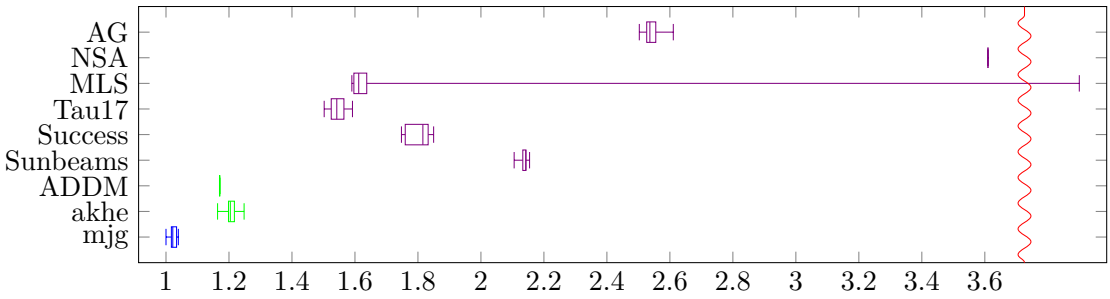


Figure 6.9: VeRoLog_late_r1000d25_4

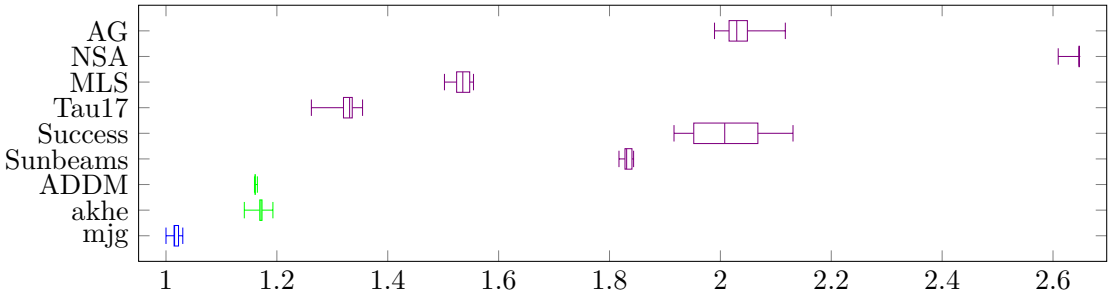


Figure 6.10: VeRoLog_late_r1000d25_5

on exactly the same, well-defined problem, (2) there is compensation for run times on different machines, and (3) each algorithm is applied to many different problem instances.

We described two different solution methods for the problem central to the challenge: the first method, by team akhe, is based on finding promising combinations of low level heuristics. These heuristics, such as *move, swap* or *reverse*, are combined in sequences that are randomly drawn with probabilities that are updated in a tuning process that depends on the problem instance. This algorithm is rather generic, in the sense that applying it to different problems would require relatively few changes (as long as it is easy to find initial feasible solutions for the problem).

The second method, by team ADDM, focuses on decomposing the problem. This means that the algorithm is explicitly tackling the problem of assigning tasks to days. It spends the first 80% of the computation time on finding good day to day schedules using a genetic algorithm. The last 20% of time is spent on Variable Neighborhood Descent in order to improve the routing given a certain day to day schedule. One might say this approach is intuitive, because the decomposition explicitly deals with the scheduling and the routing aspects of the problem.

We can observe differences and similarities between the two approaches. Let us focus on the most obvious difference first: team ADDM decomposed the problem whereas team akhe did not. While the two approaches appear quite different altogether, we can still find several similarities. First of all, the 'low level heuristics' as mentioned in akhe's approach overlap with the heuristics used in team ADDM's neighborhood search. Furthermore, both teams made use of the fact that initial feasible solutions were easy to find. Finally, both approaches were randomized and allowed for trying moves that appeared to be unlikely to improve the solution - albeit with a smaller probability than those moves that appeared promising.

The ability to compare algorithms objectively makes a challenge a valuable opportunity to gain insights into state-of-the-art solution techniques. In this paper, we demonstrated that the two solution approaches - although altogether different - were both effective in solving the $\mathcal{NP}$-hard optimization problem that underlined the VeRoLog Solver Challenge 2016 - 2017.

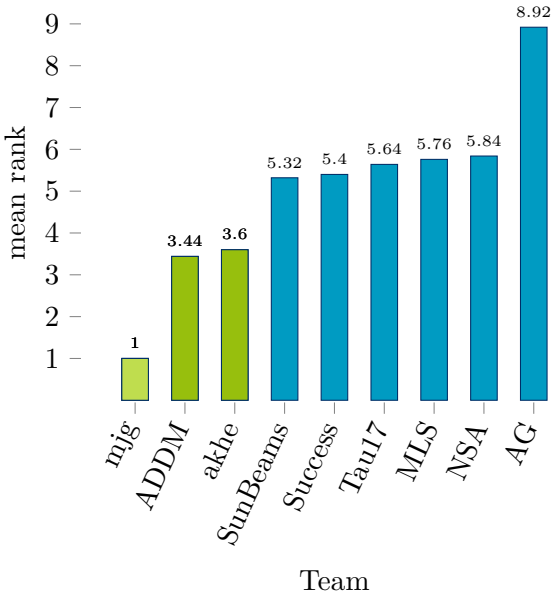


Figure 6.11: Mean rank per team, as computed according to the challenge rules. These results are based on the submissions of the restricted resources challenge (late instances). The leftmost three bars correspond to the teams that reached the finale.

6.6 Appendix: Challenge rules

We summarize the challenge rules, which were originally published in Dullaert et al. [47]. The challenge consisted of three parts, and the first two ran partially parallel in time.

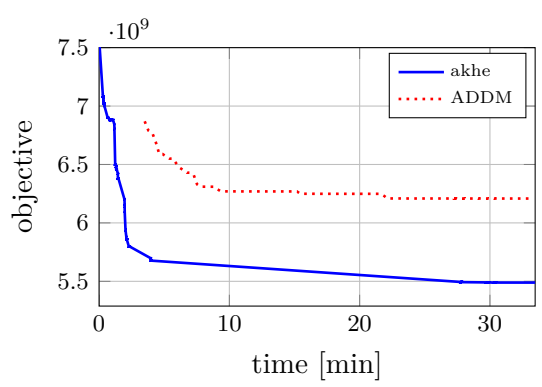
6.6.1 All-time-best challenge

The organizers disclosed 25 instances in December 2016: the "all-time-best instances". Participants were invited to submit a solution to an instance if it was better than the best solution submitted so far for this instance. Progress, i.e., the cost of the best solution over time, was shown to the participants. This information is still visible on the website, and it shows that different instances are won by different participants.

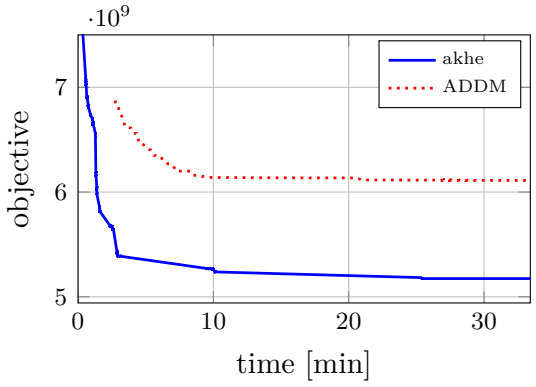
The all-time-best challenge ran till July 2, 2017 and the participants were rewarded in two ways: for every week during the all-time-best period that their solution was the best, and additionally for having the best solution at the end of the challenge. In this part of the challenge, any means, resources and time, were allowed.

6.6.2 Restricted resources challenge

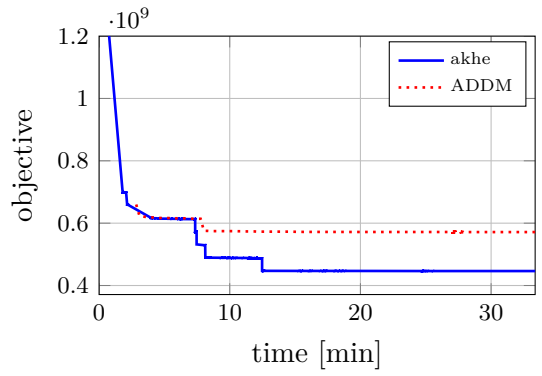
This challenge had a more "traditional" form: the resources were restricted, especially the computing time. The time T_{limit} (seconds) that each algorithm was allowed to run on the organizers' single core machine is limited by the formula $T_{limit} = 10 + 2|R|$. Here $|R|$ is the number of (delivery) requests in the instance. The organizers provided a calibration tool, so that each participant could estimate the equivalent time on his or her local machine. In addition, it was not allowed to use any software that is not freely available for commercial use. In particular, this means that for example the use of commercial MILP solvers was forbidden. Each algorithm had to run on a new set of 25 instances (available since April 1 2017). Furthermore, each solver



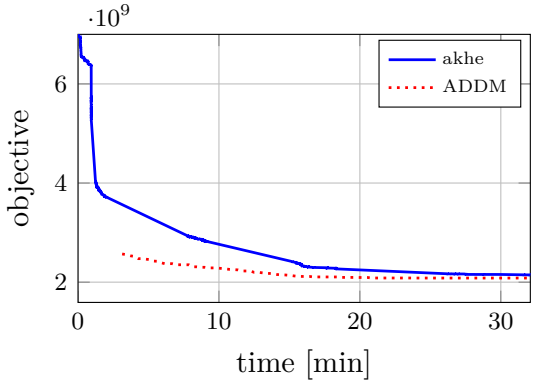
(a) VeRoLog_late_r1000d25_1



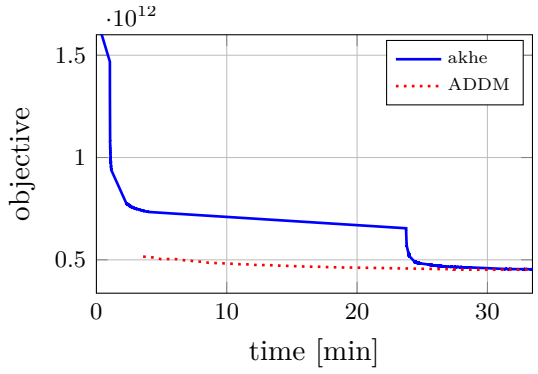
(b) VeRoLog_late_r1000d25_2



(c) VeRoLog_late_r1000d25_3



(d) VeRoLog_late_r1000d25_4



(e) VeRoLog_late_r1000d25_5

Figure 6.12: Comparison of the convergence profile of akhe and ADDM algorithms on VeRoLog_late_r1000d25 dataset

had to run on each instance, using nine different random seeds, in order to reduce the variance coming from randomized algorithms. Non randomized algorithms could also profit from the random seeds: they were known to be between 10^8 and 10^9 with a different starting digit for each seed, and hence it was possible to detect this and run 9 different deterministic algorithms. The corresponding results and solver binaries were submitted on May 8, 2017.

The evaluation of algorithms in the restricted resources challenge was done as follows. A rank score was calculated per instance for each solver. First, per instance, the two best solutions and the two worst solutions found by the solver were removed. The remaining five solutions were used to compute the score of the solver. If these five solutions were all feasible, their average counted as the score of the solver. Alternatively, if there were infeasible solutions among the middle 5 solutions, that solver was first ranked with respect to the number of feasible solutions, and secondary by the average cost of the feasible ones. Finalists were announced on June 1.

6.6.3 The finals

The finalists' solvers were run by the organizers on a set of 50 not previously disclosed (the so-called *hidden*) instances. Again, per solver per instance but equal for each finalist, nine runs with different random seeds were done, again with nine different starting digits. A solver ranking per instance was made with the same rules as above, as well as a ranking of the solvers based on these scores. The winner of the challenge was the participant whose solver had the lowest mean of the ranks.

Table 6.4: The characteristics of the hidden instances and the cost of best obtained solutions

Instance	Customers	Tools	Capacity	Max distance	Best known
VeRoLog_hidden_r500d10_1	485	4	50	17000	8365884102
VeRoLog_hidden_r500d10_2	492	2	35	16000	5770640038
VeRoLog_hidden_r500d10_3	491	3	40	20000	744495928
VeRoLog_hidden_r500d10_4	489	5	40	20000	3774764460
VeRoLog_hidden_r500d10_5	492	3	40	20000	166281961014
VeRoLog_hidden_r500d15_1	490	4	35	16000	2696765455
VeRoLog_hidden_r500d15_2	491	3	45	15000	5295158689
VeRoLog_hidden_r500d15_3	486	5	50	15000	216780767
VeRoLog_hidden_r500d15_4	492	3	35	17000	2019397633
VeRoLog_hidden_r500d15_5	493	2	50	16000	153864525012
VeRoLog_hidden_r1000d20_1	950	4	30	17000	8176085650
VeRoLog_hidden_r1000d20_2	950	3	50	20000	4487557411
VeRoLog_hidden_r1000d20_3	942	5	50	16000	452935897
VeRoLog_hidden_r1000d20_4	946	5	35	16000	1816133429
VeRoLog_hidden_r1000d20_5	950	3	35	17000	673881241010
VeRoLog_hidden_r1000d25_1	947	2	40	15000	2476429490
VeRoLog_hidden_r1000d25_2	961	4	40	16000	5953632945
VeRoLog_hidden_r1000d25_3	952	4	40	16000	176042639
VeRoLog_hidden_r1000d25_4	952	2	35	16000	7253854432
VeRoLog_hidden_r1000d25_5	952	2	45	20000	445754474005
VeRoLog_hidden_r1500d30_1	1379	4	40	17000	6446878020
VeRoLog_hidden_r1500d30_2	1372	5	40	16000	6483597038
VeRoLog_hidden_r1500d30_3	1373	2	45	20000	115134924
VeRoLog_hidden_r1500d30_4	1374	3	40	16000	10599076534
VeRoLog_hidden_r1500d30_5	1382	4	50	20000	1138250720007
VeRoLog_hidden_r1500d40_1	1375	3	40	15000	3598635808
VeRoLog_hidden_r1500d40_2	1370	5	40	20000	6035110304
VeRoLog_hidden_r1500d40_3	1367	3	50	16000	253836767
VeRoLog_hidden_r1500d40_4	1373	4	35	15000	2670813271
VeRoLog_hidden_r1500d40_5	1385	2	40	15000	1610263788013
VeRoLog_hidden_r2000d50_1	1785	2	45	16000	5029970838
VeRoLog_hidden_r2000d50_2	1785	5	30	16000	4290045748
VeRoLog_hidden_r2000d50_3	1791	5	40	16000	323926260
VeRoLog_hidden_r2000d50_4	1792	4	35	16000	18321969466
VeRoLog_hidden_r2000d50_5	1777	4	40	16000	1047956765006
VeRoLog_hidden_r2000d65_1	1776	2	40	16000	3989738288
VeRoLog_hidden_r2000d65_2	1773	4	50	16000	2457293730
VeRoLog_hidden_r2000d65_3	1782	5	40	16000	195384322
VeRoLog_hidden_r2000d65_4	1767	3	35	17000	15891623281
VeRoLog_hidden_r2000d65_5	1799	5	45	16000	1370590324007
VeRoLog_hidden_r2500d70_1	2166	3	45	16000	4751728213
VeRoLog_hidden_r2500d70_2	2164	5	35	16000	5599542429
VeRoLog_hidden_r2500d70_3	2181	2	35	15000	354579316
VeRoLog_hidden_r2500d70_4	2170	4	50	20000	14479189673
VeRoLog_hidden_r2500d70_5	2140	5	40	17000	1177525804008
VeRoLog_hidden_r2500d75_1	2160	4	40	15000	5591086150
VeRoLog_hidden_r2500d75_2	2160	3	35	16000	3520404346
VeRoLog_hidden_r2500d75_3	2147	5	50	15000	300634318
VeRoLog_hidden_r2500d75_4	2185	3	45	17000	15283897582
VeRoLog_hidden_r2500d75_5	2192	2	50	17000	1625596820006

Table 6.5: Summary of the competition results (hidden instances)

Instance	ADDM (A)		akhe (B)		mjg (C)		A vs B	A vs C	B vs C
	Avg. cost	Rank	Avg. cost	Rank	Avg. cost	Rank			
VeRoLog_hidden_r500d10_1	Infeasible	3	8543852756	2	8365899289	1	$<$	$<$	$<$
VeRoLog_hidden_r500d10_2	7301404930	3	5903916399	2	5771493723	1	$<$	$<$	$<$
VeRoLog_hidden_r500d10_3	863711081	3	814843760	1	819673854	2	$<$	$<$	$\simeq$
VeRoLog_hidden_r500d10_4	4561034407	2	4587216376	3	3799479893	1	$\simeq$	$<$	$<$
VeRoLog_hidden_r500d10_5	1.95044E+11	2	2.20579E+11	3	1.70088E+11	1	$\simeq$	$<$	$<$
VeRoLog_hidden_r500d15_1	3267429112	3	2838813151	2	2697367524	1	$<$	$<$	$<$
VeRoLog_hidden_r500d15_2	6264518370	3	5518848425	2	5311310871	1	$<$	$<$	$<$
VeRoLog_hidden_r500d15_3	300157427	3	258486750	2	217665073	1	$<$	$<$	$<$
VeRoLog_hidden_r500d15_4	2333125167	2	2603742796	3	2071346776	1	$>$	$<$	$<$
VeRoLog_hidden_r500d15_5	1.79542E+11	2	1.79855E+11	3	1.55013E+11	1	$\simeq$	$<$	$<$
VeRoLog_hidden_r1500d30_1	7848014014	3	6881717534	2	6467120269	1	$<$	$<$	$<$
VeRoLog_hidden_r1500d30_2	7788925107	3	7127091955	2	6489993680	1	$<$	$<$	$<$
VeRoLog_hidden_r1500d30_3	131376619	3	127751602	2	115315364	1	$\simeq$	$<$	$<$
VeRoLog_hidden_r1500d30_4	13173341132	3	13166641854	2	10703447930	1	$\simeq$	$<$	$<$
VeRoLog_hidden_r1500d30_5	1.31728E+12	2	1.34828E+12	3	1.15184E+12	1	$>$	$<$	$<$
VeRoLog_hidden_r1500d40_1	4320649929	3	3865210740	2	3599106687	1	$<$	$<$	$<$
VeRoLog_hidden_r1500d40_2	7316145898	3	6535928682	2	6081597171	1	$<$	$<$	$<$
VeRoLog_hidden_r1500d40_3	378641946	3	316000036	2	266249047	1	$<$	$<$	$<$
VeRoLog_hidden_r1500d40_4	3568036772	3	3269104825	2	2688601902	1	$<$	$<$	$<$
VeRoLog_hidden_r1500d40_5	1.92543E+12	2	2.03423E+12	3	1.62762E+12	1	$>$	$<$	$<$
VeRoLog_hidden_r2000d50_1	5810784719	3	5298648594	2	5030224679	1	$<$	$<$	$<$
VeRoLog_hidden_r2000d50_2	5328466675	3	4866236414	2	4308682151	1	$<$	$<$	$<$
VeRoLog_hidden_r2000d50_3	486474150	3	418201601	2	324847201	1	$<$	$<$	$<$
VeRoLog_hidden_r2000d50_4	25001890955	3	23252981136	2	18510562245	1	$<$	$<$	$<$
VeRoLog_hidden_r2000d50_5	1.28428E+12	3	1.25324E+12	2	1.05629E+12	1	$<$	$<$	$<$
VeRoLog_hidden_r2000d65_1	4810734214	3	4294615936	2	3990007875	1	$<$	$<$	$<$
VeRoLog_hidden_r2000d65_2	3083403319	3	2702857405	2	2465266227	1	$<$	$<$	$<$
VeRoLog_hidden_r2000d65_3	261437817	3	238093078	2	195919946	1	$<$	$<$	$<$
VeRoLog_hidden_r2000d65_4	20897565143	2	21949270729	3	16051327503	1	$>$	$<$	$<$
VeRoLog_hidden_r2000d65_5	1.64544E+12	3	1.61752E+12	2	1.38209E+12	1	$>$	$<$	$<$
VeRoLog_hidden_r2500d70_1	5923449374	3	5114335854	2	4751953813	1	$<$	$<$	$<$
VeRoLog_hidden_r2500d70_2	7171256068	3	6556981185	2	5643936333	1	$<$	$<$	$<$
VeRoLog_hidden_r2500d70_3	519502106	3	469897674	2	354745805	1	$<$	$<$	$<$
VeRoLog_hidden_r2500d70_4	18895390762	3	18836590052	2	14619186238	1	$\simeq$	$<$	$<$
VeRoLog_hidden_r2500d70_5	1.51888E+12	3	1.4329E+12	2	1.18559E+12	1	$<$	$<$	$<$
VeRoLog_hidden_r2500d75_1	6952945996	3	6156231838	2	5597113528	1	$<$	$<$	$<$
VeRoLog_hidden_r2500d75_2	4335142495	3	4135070761	2	3556430390	1	$<$	$<$	$<$
VeRoLog_hidden_r2500d75_3	428374927	3	383930752	2	301433094	1	$<$	$<$	$<$
VeRoLog_hidden_r2500d75_4	20347372369	3	19519419317	2	15493912178	1	$<$	$<$	$<$
VeRoLog_hidden_r2500d75_5	2.01467E+12	3	1.9737E+12	2	1.63357E+12	1	$<$	$<$	$<$
Average ranking	2.82		2.16		1.02				

Chapter 7

Total Distance Approximations for Routing Solutions

Published in: *Computers & Operations Research*
Total Distance Approximations for Routing Solutions.
D. Nicola, R. Vetschera & A.G. Dragomir.
https://doi.org/10.1016/j.cor.2018.10.008

Abstract In order to make strategic, tactical and operational decisions, carriers and logistic companies need to evaluate scenarios with high levels of accuracy by solving a large number of routing problems. This might require relatively high computational efforts and time. In this paper, we present regression-based estimation models that provide fast predictions for the travel distance in the Traveling Salesman Problem (TSP), the Capacitated Vehicle Routing Problem with Time Windows (CVRP-TW), and the Multi-Region Multi-Depot Pickup and Delivery Problem (MR-MDPDP). The use of general characteristics such as distances, time windows, capacities and demands, allows us to extend the models and adjust them to different problems and also to different solution methods. The resulting regression models in most cases achieve good approximations of total travel distances, except in cases where strong random noise is present, and outperform previous models.

7.1 Introduction

Many exact and heuristic algorithms have been developed to solve transportation problems in short computational time. However, there are situations in which it is not necessary or possible to apply such algorithms. For example, carriers who need to respond to an inquiry by a potential customer or who are involved in a competitive bidding process need rapid information about costs, but not about the actual routing. This information must often be obtained very quickly for a large number of transportation requests, for example in a combinatorial auction, when bids must be made for many bundles of requests. Combinatorial auctions were frequently proposed as part of collaboration mechanisms to improve the efficiency of the transportation system [15, 62].

Several approaches for approximating travel distances in transportation problems have been developed, which we will review in section 7.2. Previous methods were mostly based on analytically derived approximation formulas [e.g., 13], which were sometimes augmented by empirically estimated parameters [e.g., 25]. This approach limits the domain of problem classes for which approximations can be developed to comparatively simple problems, in which tour lengths can be approximated analytically. Only few authors [e.g., 21, 79] so far considered a more empirical approach, but also limited their work to problems like the traveling salesman problem or vehicle routing problems without complex, real world constraints.

Our work takes an empirical perspective, initially considering a large set of (possibly redundant) potential variables, and then with the help of statistical methods identifying variables

that are particularly useful in approximating total costs. Some of these variables, like distances between customers, are present in all routing problems. Other variables, like time windows and capacity constraints, are added to approximate the solution of problems containing those constraints. Previous studies have mostly reported the quality of approximations in terms of goodness of fit to the problem instances for which the model was estimated. We complement this in-sample evaluation with an out-of-sample evaluation, in which we analyze how well our models are able to predict total costs for new problem instances.

Our empirical approach requires actual solutions of some problems to estimate the parameter values. The approach therefore does not necessarily approximate the optimal solution to a problem, but the solution that can be found by the algorithm applied. For many of the applications envisioned, this is exactly the information that is required. For example, in a competitive bidding process, the price charged for performing a set of requests needs to cover the actual costs that will be incurred. If the algorithm used for generating the actual schedule delivers a solution that causes higher than optimal costs, these are the costs that need to be covered. Of course, having a better routing algorithm would make the carrier more competitive, but pricing should be based on the actual planning process.

Although this empirical approach is adaptable to different solution methods, this does not imply that the same empirical model can be applied to approximate solutions found by any algorithm. It is quite possible that the solution quality of different algorithms also depends on different characteristics of the problem, thus different empirical models will be needed to approximate solutions of the algorithms. However, the general method remains the same, even if other variables are included in the models.

In line with much of the existing literature [55, 37], we consider (variable) costs to be roughly equivalent to travel distance. We develop approximations for travel distance in different types of transportation problems of increasing complexity, ranging from a standard Traveling Salesman Problem (TSP) to the recently introduced [43] Multi-Region Multi-Depot Pickup and Delivery Problem (MR-MDPDP). This gradual approach allows us to study whether the presence of additional constraints like vehicle capacity or time windows influences the quality of approximations in general. We also study how such factors can be represented in the approximation to make it more accurate even for a complex setting. In our view, the ability to incorporate additional problem characteristics into the approximation in a rather straightforward manner is a considerable advantage of an empirical approach. Thus, we consider it important to determine the potential benefit of adding such additional parameters to the model.

The remainder of the paper is structured as follows: In Section 7.2, we present a literature review on approximation models for different types of transportation problems. Section 7.3 introduces the regression based approximation for the different types of logistic problems we study in this paper. Computational experiments and results are presented in Section 7.4, and Section 7.5 concludes the paper by summarizing its results and presenting an outlook onto future research.

7.2 Literature Review

Bearwood et al. [13] are among the first authors who developed a distance approximation. They demonstrated that for a set of n nodes in a compact and convex area A, the length of the TSP tour asymptotically converges to $c\sqrt{nA}$ when $n \to \infty$, c being a constant. Christofides and Eilon [25] approximated the average TSP route length by $100 \times c\sqrt{n}$. Further distance approximations for the TSP were developed by Chien [23], Kwon et al. [91], and Hindle and Worthington [79]. They all recognized the need for additional parameters representing the shape of the area or distances between customers. Chien [23] modified the area factor A by including the area of the smallest rectangle that covers all customers and also includes distance related measures like the average distance to the depot. Kwon et al. [91] used regression models and neural networks

to improve the TSP approximations. They considered a rectangular length/width ratio as well as a shape factor. Hindle and Worthington [79] used two different models depending on how customers are distributed over the area. Their first model considered a uniform distribution of locations. The second model used "demand surfaces" to represent different densities of customers across regions. In a recent paper on the approximation of TSP travel distances, Çavdar and Sokol [21] proposed an approximation based on coordinates of customers. Their approximation was based on the standard deviations of coordinates as well as the standard deviations of distances between customers and the average-point of the area. Stein [158] considered the bus problem, a TSP with additional constraints in which pickup and delivery nodes are paired. They extended the results of Bearwood et al. [13] and estimated the length of the optimal tour either for one or for several buses.

The first published approximations for the Capacitated Vehicle Routing Problem (CVRP) were developed in the 1960s by Webb [165], who studied the correlation between route length and customer-depot distance. Eilon et al. [50] proposed approximations to the length of the CVRP based on the shape and area of delivery, distances between customers and the depot, and capacity of vehicles. Daganzo [37] approximated CVRP tour length as

$$CVRP(n) \approx 2rn/Q + 0.57\sqrt{nA} \tag{7.1}$$

where n is the number of customers customers, r is the average distance between the customers and the depot and Q is the maximum number of customers that can be served by a vehicle. Robusté et al. [136] tested Daganzo's approximation and proposed adjustments based on the shape of the area, in particular for elliptic zones. Erera [51] extended Daganzo's approximation for stochastic versions of the CVRP. Langevin and Soumis [92] developed an approximate method for planning pickup and delivery zones assigned to vehicles. Their approach included an estimation of the increase in the number of vehicles used if zones are allowed to overlap.

Time window constraints were first considered in the late 1980s by Daganzo [38] by dividing the time horizon in periods and clustering the customers in rectangles. The problem was simplified by assigning customer time windows to a time period. A more recent contribution for VRP's with time windows is provided by Figliozzi [54], who approximated the distance for serving n customers using a known number m of routes as

$$VRP(n) \approx b\frac{n-m}{n}\sqrt{An} + m2r. \tag{7.2}$$

The parameter b was estimated by linear regression. Figliozzi [55] studied approximations to the average length of VRP's when the number of customers, time window constraints and demand levels vary. A detailed literature review of models that use continuous approximation in distribution management can be found in Franceschetti et al. [60].

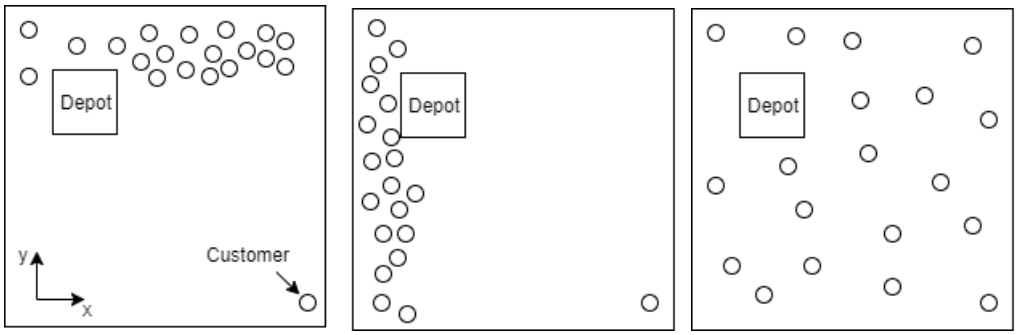
7.3 Regression Based Approximations

This section introduces the problems that are studied in this paper and describes the variables that are included in the estimation models presented in Section 7.4. We consider three classes of problems in increasing order of complexity: The Traveling Salesman Problem (TSP), the Capacitated Vehicle Routing Problems with Time Windows (CVRP-TW), and the Multi Region Multi-Depot Pickup and Delivery Problem (MR-MDPDP).

These problems are also different with respect to the solution methods that can be applied. For small TSP, an optimal solution can be found via exact methods, for the MR-MDPDP, only one heuristic is available. This allows us to study how well the regression approach can adapt to solutions and methods of different quality. We also study this question in more detail for the CVRP-TW, where we apply the regression approach both to known optimal solutions and to solutions obtained with a simple heuristic.

7.3.1 Traveling Salesman Problem

The first models for the TSP approximated the total tour length using the factor $\sqrt{nA}$. Since the total travel distance depends on the distance between nodes, we include different distance measures in the estimation model. To represent different distributions of nodes in the area, we also include dispersion measures such as the variance of distances and maximum distances to the average-node. Based on the approach proposed in [21], we also include the product of the variances of coordinates. Figure 7.1 illustrates the purpose of including this interaction term. The problems presented in figures 7.1a and 7.1b both have solutions with a small total travel distance, but variances in the two coordinates differ considerably. As Figure 7.1c shows, large travel distances will occur if variances in both coordinates are sufficiently high, this effect is covered by including the product.



(a) VarX>>VarY (b) VarX<<VarY (c) VarX ≈ VarY

Figure 7.1: TSP nodes distribution scenarios in a similar area. Figure 7.1a shows a distribution of nodes along the x axis. Figure 7.1b shows a distribution along the y axis. Figure 7.1c shows a distribution in which nodes are uniformly distributed along both axis, leading to similar values for their variances.

We therefore consider the following variables:

- Number of requests n (indicated by Req in the tables).
- Distances between nodes: These include the minimum (MinP) and maximum (MaxP) distance across all pairs of nodes and the variance of distances (VarP). Furthermore, we consider the sum of distances to the nearest (SumMinP) and to the farthest (SumMaxP) neighbor of each node.
- Distances to the average-node: Distances to the node located at the average x and y coordinates are aggregated across nodes by considering the minimum (MinM), the maximum (MaxM), the sum (SumM) and the variance (VarM).
- Product of variances of x and y coordinates of nodes (VarX×VarY).

7.3.2 Capacitated Vehicle Routing Problem with Time Windows

In the CVRP-TW, each customer has a known demand, which has to be served by a given fleet of homogeneous vehicles of limited capacity, so that each customer is visited once by one vehicle within a given time window. The presence of time windows can have a strong effect on the total travel distance. In one extreme case, customers could be visited in exactly the same sequence as in the optimal tour without time windows, if time windows are large enough or happen to correspond exactly to that sequence. In the other extreme, time windows may require to visit a customer that is far from the previous customer, and then come back to a customer that is located close to the first one. We use two types of measures to represent how much the presence of time windows constrains the problem: The first directly refers to the length of time

windows of individual customers, where we consider the sum and the variance. Furthermore, we use the average overlap and variance of overlaps of time windows between pairs of customers to measure how strongly time windows restrict routing possibilities.

The presence of time windows also influences which links between customers can actually be used. If it is not possible to visit customer i after customer j, then the travel distance from i to j cannot influence the solution. We therefore include travel distance d_{ij} between customers i and j in the calculation of distance-related measures only if the condition

$$STW_i + serv_i + d_{ij} \leq ETW_j \tag{7.3}$$

is fulfilled. In (7.3), $serv_i$ is the service time, and STW_i and ETW_i are the starting and ending times of the time window of customer i. A vehicle is allowed to arrive early to a node and wait until the customer's time window starts.

Since the capacity of each vehicle is limited, additional vehicles will be required once the capacity of a vehicle is exceeded. This increases the total tour length, and we therefore have to include these effects in the model. This is achieved by adding two more variables. The total-demand/vehicle-capacity ratio provides information on the minimum number of tours necessary to perform all requests. The vehicle-capacity/average-demand ratio represents the average number of requests that can be included in one tour.

Previous studies have included the number of requests, the average distance to the depot and the ratio between the number of requests and the vehicle capacity. In addition, we include the following variables:

- Sum (SumD) and variance (VarD) of distances between customers and the depot.
- Sum (SumTW) and variance (VarTW) of the lengths of time windows.
- Minimum (MinPosDist), maximum (MaxPosDist), sum (SumPosDist) and variance (VarPosDist) of distances between customers. Only distances that fulfill condition (7.3) are included in these calculations.
- Sum (SumOverlap), average (AvgOverlap) and variance (VarOverlap) of overlaps of time windows betwen customers.
- Total-demand/vehicles-capacity ratio (Q/Cap) and vehicles-capacity/average-demand ratio (Cap/AvQ).

7.3.3 Multi-Region Multi-Depot Pickup and Delivery Problem

The MR-MDPDP has not been studied as much in previous literature as the TSP and the CVRP-TW. It extends the other problems by providing a hierarchical structure of the transportation system. Customers (pickup and delivery nodes) are located in different regions. A request is fulfilled in three phases: pickup, performed by a short-haul vehicle, a long-haul trip, performed by a larger vehicle, and delivery, again performed by a short-haul vehicle. Requests have different demands and customers have time windows in which they can be visited. The short-haul routing for each region can therefore be classified as a Vehicle Routing Problem with Mixed Backhauls (VRPMB) according to the definitions by Parragh et al. [121]). The objective is to minimize total costs of short-haul and long-haul trips. Although the distance between regions is fixed and there is no routing involved, costs of long-haul trips can vary because different schedules require a different number of trips. Figure 7.2 shows a graphical representation of the problem. In the present paper, we consider only two regions. This problem already provides a sufficiently rich environment to test how well regression based models can approximate optimal solutions in such a complex transportation problem.

New and more complex problems like the MR-MDPDP include multi-modal transportation using different vehicles. Therefore, we have to relate demand to the capacities of all types of

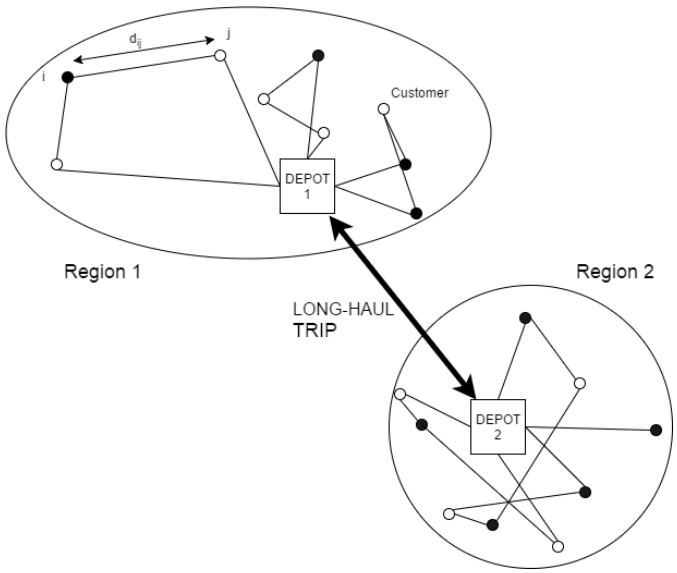


Figure 7.2: 2R-MDPDP Diagram. Requests have paired pickup (white) and delivery (black) nodes in opposite regions. To be serviced, they are part of a vehicle tour in every region starting and ending at the region depot. Both depots are connected by long-haul trips.

vehicles involved. Furthermore, timing of long-haul trips might influence the total distance. For example, if the scheduled departure times are fixed in a way that orders cannot be consolidated, more long-haul trips will be necessary than if the long-haul schedule can be flexibly adjusted.

Even small instances of this problem are quite difficult to solve exactly in reasonable time. Therefore, heuristic solution methods have to be used and the regression approximates the costs in the solutions obtained by these methods, which might be higher than the optimal costs. Different algorithms might find different solutions, but the regression would work in a similar way. However, as we already indicated, we cannot study differences between algorithms for this particular problem, since no alternative algorithms are available.

The dependent variable of the model is the total distance, including long-haul and short-haul trips. The model is very complex, so many properties of the problem can influence the objective value. In particular, we have to distinguish between variables related to long- and short-haul transportation, and variables that refer to single regions or both regions simultaneously. In addition to the variables considered in the previous models, we include the following variables in the initial regression model:

- Long-haul schedule type: a binary variable taking the value of 0 if the schedule is fixed and 1 if it is flexible (BinFlexFix).
- Distances to the depots: in addition to the sum and variance, we also consider the minimum (MinD) and maximum (MaxD).
- Sum of distances between customers (SumP).
- The sum of demands is zero, as it is a pickup and delivery problem. Therefore, we include the quantity-capacity information using the total-pickup-quantities/capacity ratio (SumQ/Cap). When both regions are considered together, the quantity-capacity information is introduced as the variance-of-quantities/capacity ratio (VarQ/Cap).

7.4 Computational Experiments and Results

Our computational experiments are based on instances that are either randomly generated or obtained from literature. Each set of instances was split into two parts, a training set and an evaluation set. Parameters of the regression models were first estimated for the training set, and then the resulting regression equations were applied to the evaluation set. Thus, we obtain data on the within-sample and out-of-sample fit of the model. In line with previous research [23, 91, 54, 55], the regression model does not contain a constant parameter, since a problem without customers obviously has an objective value of zero. After generating a model including all possible variables, we apply forward and backward stepwise regression to obtain models with a smaller set of parameters.

We apply two measures for the quality of approximation. The Mean Percentage Error (MPE) is defined as:

$$MPE = \frac{1}{k} \sum_{i=1}^{k} \frac{D_i - E_i}{D_i} \times 100\% \tag{7.4}$$

and the Mean Absolute Percentage Error (MAPE) as:

$$MAPE = \frac{1}{k} \sum_{i=1}^{k} \frac{|D_i - E_i|}{D_i} \times 100\% \tag{7.5}$$

where D_i is the total travel distance obtained by the solution method, E_i the approximation obtained by the regression models for observation i, and k is the number of observations.

7.4.1 Traveling Salesman Problem

For the TSP, the groups of instances, solution methods to obtain the travel distance and the estimation models with their results are presented in this section.

Instances and Solution Method

Instances for this problem contain between 25 and 1000 nodes located in a square of 200×200 units. Travel distance between nodes is assumed to be equal to the Euclidean distance. All instances for this problem were randomly generated. The first group of TSP problems consisted of 260 instances (130 in the training set and 130 in the evaluation set) with 25 to 50 customers. Problems were solved to optimality using the commercial solver IBM CPLEX version 12.6.3.

To test approximation to the TSP problem for larger instances, we generated three groups of 400 instances (200 for training, 200 for evaluation) with 50 to 1000 customers. The three groups differed in the locations of nodes. In one group (random instances, R), nodes were uniformly distributed across the entire area. The second group (C) contained clustered instances, where two clusters 1/9 the size of the entire area were created and located at opposite corners within the area and nodes were uniformly distributed within these clusters. The last group (random-clustered, RC) was a mixed group, in which the set of nodes was randomly fractioned in three smaller sets, two were uniformly distributed in each cluster and one was uniformly distributed across the entire area. As these instances are too large to solve optimally, we used the Lin-Kernighan (LK) implementation by Helsgaun [75].

Regression Parameters and Results

The models estimated for the small instances are presented in Table 7.1. Model 1 was created using a backward stepwise regression and Model 2 is based on a forward stepwise regression. In

both models, a threshold of $p = 0.05$ for entering/removal was used. The best models for each of the three groups of larger instances are presented in Tables 7.2, 7.3 and 7.4. For comparison, we also add the models of Hindle and Worthington [79] (HW) and the model of Çavdar and Sokol [21] (CS).

Parameter	Model 1	Model 2	HW	CS
Req	***7.077		6.98	
MinP				
MaxP	***1.309	***1.369		
SumMinP	***0.478	***0.388		
SumMaxP		***0.072		
VarP		***-7.4 E-5		
VarX × VarY	**3.23 E-6			
MinM				
MaxM		***-1.581		
SumM		***0.101		
VarM	***0.0899	***0.127		
Constant			59.18	
ln(Req)			186.48	
$\sqrt{Req(cstdev_x cstdev_y)}$				*** 2.879
$\sqrt{Req(stdev_x stdev_y)\frac{A}{C_x C_y}}$				4.4 E-5
R2 (adjusted)	0.9988	0.9989	0.7153	0.9958
MPE (in sample)	0.07%	0.07%	-0.33%	0.23%
MAPE (in sample)	2.76%	2.57%	4.79%	5.23%
MPE (out of sample)	0.64%	0.88%	0.78%	0.34%
MAPE (out of sample)	2.60%	2.73%	4.42%	5.55%

$*p < 0.1; **p < 0.05; ***p < 0.01$

Table 7.1: TSP Estimation Models for Small Instances

In sample, for the small instances, our regression models provide a considerably better approximation than the previous models. Out-of-sample, the MPE is comparable to the value of the model of Hindle and Worthington [79], and higher than in the model of Çavdar and Sokol [21]. However, the MAPE of our models is considerably better. Approximation errors in the model of Çavdar and Sokol [21] are more evenly distributed around zero, but on average, the absolute error is larger. For larger instances, all models perform similarly for the R instances, providing small estimation errors. However, our models outperform the other two models for the RC and C instances.

Parameter	Model 1	HW	CS
Req	0.198	*** 2.33	
MinP	***-33.818		
MaxP	***3.064		
SumMinP	***1.063		
SumMaxP	-0.003		
VarP			
VarX × VarY	** -1.3 E-5		
MinM	**-2.635		
MaxM	-0.652		
SumM	*0.0188		
VarM	-0.088		
Constant		***-1073.8	
ln(Req)		***539.392	
$\sqrt{Req(cstdev_x cstdev_y)}$			***2.70883
$\sqrt{Req(stdev_x stdev_y)\frac{A}{C_x C_y}}$			***-1.65 E-6
R2 (adjusted)	0.9998	0.9972	0.9996
MPE (in sample)	-0.11%	0.03%	0.54%
MAPE (in sample)	1.59%	1.85%	2.02%
MPE (out of sample)	0.12%	-0.01%	0.40%
MAPE (out of sample)	1.77%	1.79%	2.15%

*:$p < 0.1$; **:$p < 0.05$; ***:$p < 0.01$

Table 7.2: TSP Estimation Models for Large R Instances

Parameter	Model 1	HW	CS
Req		***1.573	
MinP	***-17.268		
MaxP	***2.107		
SumMinP	***1.1147		
SumMaxP			
VarP			
VarX × VarY			
MinM	***-6.597		
MaxM			
SumM	***0.012		
VarM	***-0.090		
Constant		***-1507.87	
ln(Req)		***623.38	
$\sqrt{Req(cstdev_x cstdev_y)}$			***2.33239
$\sqrt{Req(stdev_x stdev_y)\frac{A}{C_x C_y}}$			2.43 E-6
R2 (adjusted)	0.9996	0.8872	0.9779
MPE (in sample)	-0.23%	-1.29%	0.41%
MAPE (in sample)	1.81%	9.66%	13.25%
MPE (out of sample)	-0.37%	-0.38%	1.32%
MAPE (out of sample)	2.12%	10.32%	13.43%

*:$p < 0.1$; **:$p < 0.05$; ***:$p < 0.01$

Table 7.3: TSP Estimation Models for Large RC Instances

Parameter	Model 1	HW	CS
Req	***0.157	***1.228	
MinP	***-36.069		
MaxP	***1.194		
SumMinP	***0.987		
SumMaxP			
VarP	***-4.41 E-7		
VarX × VarY			
MinM			
MaxM			
SumM	***0.025		
VarM	***0.157		
Constant		*-757.049	
ln(Req)		***350.698	
$\sqrt{Req(cstdev_x cstdev_y)}$			***1.409
$\sqrt{Req(stdev_x stdev_y)\frac{A}{C_x C_y}}$			***9.9 E-6
R2 (adjusted)	0.9997	0.8798	0.9665
MPE (in sample)	-0.15%	-0.15%	1.20%
MAPE (in sample)	1.72%	11.16%	17.02%
MPE (out of sample)	0.56%	0.80%	-0.17%
MAPE (out of sample)	1.91%	11.62%	25.46%

*:$p < 0.1$; **:$p < 0.05$; ***:$p < 0.01$

Table 7.4: TSP Estimation Models for Large C Instances

7.4.2 Capacitated Vehicle Routing Problem with Time Windows

Section 7.4.2 describes the instances and solutions for the CVRP-TW. Estimation models and their results for optimal and Nearest Neighbor Solutions are presented in Section 7.4.2.

Instances and Solution Method

For this problem, the instances proposed in Solomon [150] for 25, 50 and 100 customers are used. These instances have a hierarchical objective, minimizing first the number of vehicles and then the total distance. The dependent variable corresponds to the travel distance reported for the optimal solutions. Instances and solutions can be found at *http://web.cba.neu.edu/~msolomon/home.htm*. Additional solutions were obtained from Baldacci et al. [8].

The Solomon data set contains six groups of instances, clustered instances (C), random instances (R) and random-clustered instances (RC), which are all further separated into instances with long and short planning horizons (subgroups C1 and C2 etc.). In works like [55], separate models were generated for each group of instances. This approach implicitly assumes that when applying the model to a new instance, one is able to correctly identify the level of clustering as well as whether the time horizon is long or short. In this paper, we present models for instance groups C, R and RC separately, as they differ considerably in the distribution of customers. Groups 1 and 2 are merged by including variables that relate to capacity.

Regression Parameters and Results

Table 7.5 presents the best performing approximation models for the optimal solutions of the three groups of problems. The model for instances of type C approximates the solution

with very small errors. For instances of type R, our results are comparable to results obtained in previous papers like [55]. The approximation error for problems of type RC is considerably larger. These problems seem to be particularly difficult, since instances that are very similar in all the variables considered sometimes have very different objective values. However, this effect can, to a certain extent, be corrected when calibrating the models by considering the in-sample approximation error of each instance. For the present estimation, we removed one instance (RC205) that had a particularly large in-sample error. RC205 represents a situation where, because of the arrangement of time windows across nodes, one would expect a large total travel distance. Time windows in this problem are rather short and have little overlap. Still the total travel distance is rather short because neighboring nodes also have time windows that allow to visit them within a short time span.

These results show that there might always be additional factors not included in the model that increase the variance of travel distance and thus make prediction more difficult. Such influences might come from the problem, but also from the solution algorithm used. In particular, simple algorithms tend to find good solutions in some instances, but also very bad solutions in other instances. This creates a large variance in solutions that is hard to predict. Table 7.6 illustrates this effect with models that approximate solutions obtained by a nearest neighbor insertion algorithm. As it was expected, a less powerful solution algorithm like this one leads to a high variability in objective values, which makes approximation difficult. Our models perform acceptably only for the R type instances. For the C and RC type, the approximation errors are considerably larger.

Parameter	R	RC	C
Req		**-18.713	***25.422
SumD	***0.901	***0.653	
VarD	**-0.168	***0.06	***-0.071
VarX ×VarY			***1.29 E-7
SumMinP			***4.86 E-4
SumTW	**4.393 E-3	***-7.79 E-3	***-9.06 E-5
MinPosDist	**-25.907		
MaxPosDist	***0.143		
SumPosDist	***-3.63 E-3	***-3.99 E-3	
AvgOverlap	*** -0.035		
VarOverlap		***4.12 E-7	
Q/Cap	***63.251	***57.212	***-12.963
Cap/AvQ		***4.612	
R2 (adjusted)	0.9980	0.9989	0.9999
MPE (in sample)	-0.26%	-0.20%	0.09%
MAPE (in sample)	3.22%	2.51%	0.37%
Out-of-Sample			
MPE (out of sample)	0.01%	3.38%	0.1%
MAPE (out of sample)	4.45%	6.47%	0.5%

*:$p < 0.1$; **:$p < 0.05$; ***:$p < 0.01$

Table 7.5: CVRPTW Estimation Models for Optimal Solutions

Parameter	R	RC	C
Req	***27.882	87.652	-23.252
SumD		-0.914	***2.439
VarD		-0.022	-0.035
VarX ×VarY			
SumTW		-0.032	*-0.014
VarTW		2.22 E-5	**3.21 E-6
MinPosDist	***-19.706		
MaxPosDist			
SumPosDist	***-3.33 E-3	-9.2 E-4	***-7.2 E-3
VarPosDist		-4.4 E-5	* -4.7 E-6
SumMinPosDist	***0.925		
SumOverlap		8.41 E-5	**9.62 E-5
AvgOverlap	***-0.218	-0.0281	0.156
VarOverlap		3.31 E-7	-3.47 E-9
Q/Cap	***30.763	-160.62	***-130.449
Cap/AvQ		-9.906	***7.624
R2 (adjusted)	0.9990	0.9930	0.9979
MPE (in sample)	9.94 E-3%	-0.81%	0.04%
MAPE (in sample)	3.12%	6.90%	3.31%
MPE (out of sample)	-0.96%	-4.11%	0.97%
MAPE (out of sample)	4.78%	12.06%	6.45%

*:$p < 0.1$; **:$p < 0.05$; ***:$p < 0.01$

Table 7.6: CVRPTW Estimation Models for Nearest Neighbor Solutions

7.4.3 Multi-Region Multi-Depot Pickup and Delivery Problem

The instances for this problem are described in Section 7.4.3. Since no standard method for this problem class is available, instances were solved using a specifically developed heuristic approach described in Section 7.4.3. The models, which estimate the solutions obtained by this particular solution method, and their results are presented in Section 7.4.3.

Instances

For this problem, instances with 10, 25, 50 and 100 customers were randomly created. We consider two regions with a size of 100×100 distance units. Each customer has a transportation request from a randomly located pickup node in one region to a randomly created delivery node in the other region. Demand values are randomly assigned to each customer. Two types of time windows are used: broad time windows, with a length of 75% of the planning period for all customers, and tight time windows, with a length of 6.25% of the planning period for all customers. For every instance, we also considered two types of long-haul schedules: a fixed schedule, in which long-haul vehicles leave at fixed points in time, and a flexible schedule, in which long-haul vehicles can leave at any time. A total of 444 instances is used for this problem, split evenly into training and evaluation sets.

Solution Method

Due to problem complexity, a decomposition approach is used. In the first step, an exact solution for the long-haul assignment is determined using the commercial solver IBM CPLEX version 12.6.3. The long-haul assignment fulfills all requests and minimizes long-haul costs. This long-haul information is used as input for the second step, in which the short-haul routing is

performed. The short-haul routing can be performed for each region independently and can therefore be classified as Vehicle Routing Problem with Mixed Backhauls (VRPMB) [121]. The solution method starts by assuming single-customer routes for each node. It then uses the pilot method [162] with an underlying savings algorithm [26]. Since we are dealing with a problem with time windows, the standard savings calculation

$$s_{ij} = d_{i0} + d_{0j} - d_{ij} \tag{7.6}$$

is changed to

$$s_{ij} = w_1 \cdot (d_{i0} + d_{0j} - d_{ij}) - w_2 \cdot waitingTime \tag{7.7}$$

where w_1 and w_2 are weights to account for distances and waiting time, referred as $waitingTime$. Waiting time is considered because the savings algorithm merges routes without re-evaluating sequence of nodes within the combined route. If a vehicle arrives at a node before the time window opens, waiting time occurs. It is time that is not spent productively and too much waiting time could lead to infeasible solutions. A long waiting time makes joining the two routes less attractive.

We use the pilot method to determine the order in which the savings are selected. This method "looks ahead" to determine which of the available immediate decisions is the best in the long run. In contrast to a greedy approach, which chooses the (locally) best savings, or a pure random approach, we try out the best m savings. This approach creates m solution candidates, which correspond to the m best savings in the first iteration and the best savings for the next n iterations. The best partial solutions after $n-1$ iterations are chosen for evaluation of the current step. This is iterated until the entire solution is built. Figure 7.3 shows the basic principle of the "looking ahead" approach.

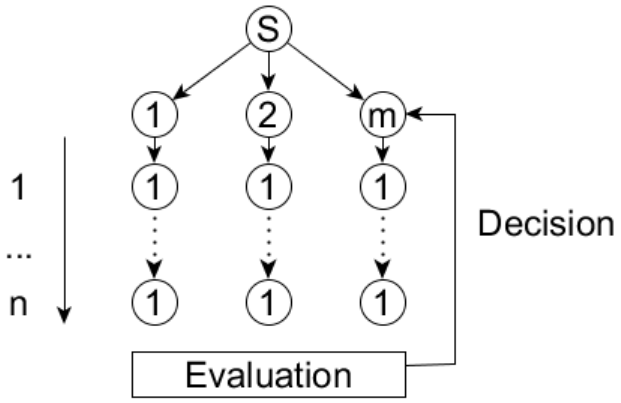


Figure 7.3: The basic principle of the pilot method showing $1..m$ solution copies of solution S with $1..n$ looking ahead steps.

Since a vehicle can leave the depot multiple times, the tours obtained by the pilot method and savings algorithm are combined to vehicle routes via a heuristic proposed by Battarra et al. [12]. This is also necessary if at this point the number of vehicles needed is greater than the number of available vehicles. All tours are initially sorted by non-decreasing starting time, breaking ties according to minimum route duration. Tours are appended in this order to vehicles that are already in use. If a tour cannot be added anywhere, a new vehicle is used. Vehicles are selected to minimize the idle time spent at the depot.

This solution method's average computation time for the instances described in Section 7.4.3 ranged from 21.13 seconds for problems with 10 customers to 162.17 seconds for problems with 100 customers.

Regression Parameters and Results

The results for models considering variables for each region separately are presented in Table 7.7. Model 1 is the full model that includes all variables. Models 2 and 3 are obtained by forward and backward stepwise procedures. In both cases, a threshold value of $p = 0.05$ was used. The models achieve a very good approximation with out-of-sample MAPE values below 2%.

Table 7.8 illustrates the advantage of considering each region individually. In that table, we combine distances from both regions into one variable, which results in a much higher MAPE value. Considering each region individually allows the model to evaluate the complexity of the VRP, and thus the likely contribution to the total objective value, in each region separately. In contrast, similar average values across both regions could be generated by very different scenarios in the two regions, leading to high variance of solutions and thus a considerably worse approximation overall.

7.5 Conclusions

A fast and accurate approximation of travel distances that can be obtained without solving a complex transportation problem is important for many applications. In this paper, we have presented an empirical approach to estimate total travel distances for different routing problems using different properties as predictors in regression models. Our models covered a wide range of transportation problems, ranging from conceptually simple traveling salesman problems to complex multi-region transportation problems involving a wide spectrum of real-life constraints. Although the quality of approximation varies between different problem classes, in many cases we obtain very good approximations, which are more accurate than previous models. In particular, we obtained good approximation results for a new class of problems, the 2R-MDPDP problem, for which no previous results were available. This shows that an empirical approach makes it possible to cover problems that include additional characteristics that could not be taken into account in previous models.

Furthermore, we were also able to show that models estimated on one set of instances can reliably predict travel distances for other, previously unknown problems. This is a particularly important result in view of potential applications of our models. Another important feature of the empirical approach for this type of applications is that our models can not only be fitted to optimal solutions, but also reliably predict the travel distances that can be obtained with heuristic methods like the one we have utilized for the 2R-MDPDP problem.

Apart from the fact that our models provide a very good approximation in many cases, the regression models also provide additional insight about which variables have a strong influence on total travel distances for the different types of transportation problems. By applying a stepwise regression approach, we were able to identify significant drivers of travel distances (and costs) out of a large set of possible influence factors. An interesting result here is that the number of requests and the area in which customers are located are not present in all the models, and that the total travel distance can also be estimated based mainly on information about distances between customers. Information about which problem characteristics are strong drivers of travel distances is not only important to obtain better approximations, but can also support carriers in their decision making. Knowing which factors influence total travel distances can help carriers to select those requests that can be fulfilled efficiently and thus to focus their acquisition efforts. Our models therefore can support carriers in many ways in their planning and decision making for efficient operations.

Our results indicate not only the usefulness, but also the limitations of such an empirical approach. For some types of problems, approximation errors were considerably larger than for other problems. An empirical approach relies on the stability of the relationship between explanatory variables and the value to be predicted. Any effect that is not explicitly captured in the model can be considered as noise that makes the empirical prediction more difficult. Apart

Parameter	Model 1	Model 2	Model 3
Req	**19.617		***17.017
BinFlexFix	***-214.185	***-210.349	***-213.9
SumTW	**-8.8 E-4		**-9.536 E-4
MinD1	-0.4448		
MaxD1	***2.964	***3.081	***3.495
SumD1	***1.936	***1.901	***1.887
VarD1	-3.598 E-3		
MinD2	*2.231		**2.656
MaxD2	-0.299		
SumD2	***1.917	***1.937	***1.907
VarD2	**1.008 E-2		***9.72 E-3
MinP1	-0.628		
MaxP1	*3.2	***2.318	
SumP1	-3.34 E-3		
VarP1	3.02 E-4		
MinP2	-0.394		
MaxP2	*2.808		***4.289
SumP2	1.339 E-3		
VarP2	-6.7 E-4		**-5.316 E-4
VarX × VarY1	2.36E-07		
VarX × VarY2	2.E-06	***3.69 E-7	***2.077
MinM1	1.071		
MaxM1	-1.491		
SumM1	*-0.409		
VarM1	-0.017		
MinM2	0.337		
MaxM2	-0.228	***4.744	
SumM2	*-0.539		***-0.741
VarM2	*-2.992 E-2		**-4.204 E-2
SumTW × Req	*1.13 E-5		***1.245 E-5
BinFlexFix × Req	***5.189	***5.014	***5.137
SumQ/Cap	-2.077		
R2(adjusted)	0.9998	0.9998	0.9998
MPE (in sample)	4.74 E-5%	0.12%	0.04%
MAPE (in sample)	1.44%	1.49%	1.46%
MPE (out of sample)	-0.32%	-0.12%	-0.21%
MAPE (out of sample)	1.57%	1.50%	1.55%

*:$p < 0.1$; **:$p < 0.05$; ***:$p < 0.01$

Table 7.7: 2R-MDPDP Estimation Models with variables for every region separately

Parameter	Model 1
Req	***118.303
BinFlexFix	-134.489
SumTW	***5.006 E-3
VarQ/Cap	**-289.528
FlexReq	**4.533
MinD	-0.639
MaxD	*-7.691
SumD	0.318
VarD	0.011
MinP	0.553
MaxP	**8.317
SumP	0.007
VarP	***-4.206 E-3
VarXVarY	5.71E-07
MinM	13.253
MaxM	-2.393
SumM	0.102
VarM	**0.079
R2(adjusted)	0.9916
MPE (in sample)	-0.67%
MAPE (in sample)	7.91%
MPE (out of sample)	-0.61%
MAPE (out of sample)	9.30%
*:$p < 0.1$; **:$p < 0.05$; ***:$p < 0.01$	

Table 7.8: 2R-MDPDP Estimation Model for two region-combined variables

from systematic factors omitted from the model specification, such noise can also result from some inherent randomness of the problem, which makes travel distances to a certain extent unpredictable. One possible source of such randomness might be the particular structure of the problem, as became evident in the case of the random-clustered CVRP problems. Here the partly systematic, partly random location of customers makes it difficult to approximate total costs. Another source of randomness may be the algorithm used to solve the problem, as became evident in the CVRP solved with the nearest neighbor heuristic. There, the highly variable performance of the algorithm makes prediction difficult. However, as the example of the 2R-MDPDP has shown, it is sometimes possible to overcome this high level of randomness by considering relevant variables at a more detailed level, here by considering information about both regions separately. This shows that a systematic analysis of possible variables to be included in such models is a worthwhile exercise, that can lead to a highly reliable prediction of travel distances and costs.

Chapter 8

Conclusion

This thesis studies multiple variants of the pickup and delivery problem (PDP), a subtopic of logistics and operations research. It has advanced the research base thematically and methodologically by addressing PDPs in new contexts and through various computational studies. This has been achieved by studying transportation networks that require transshipment and multi-modal transportation, or by examining the potential of new concepts that merge the wants of customers towards better service and the requirements of carriers toward less unsuccessful deliveries and shorter delivery routes. Methodologically, this thesis applied many well known heuristic solution methods and developed extensions and adaptations thereof, as well as developing new problem specific operators.

In Chapter 2 a new type of logistic problem is introduced: A PDP in multiple geographically separated regions based on real-world applications. The separated region make intra-region as well as inter-region transportation necessary. The regions are connected by long-haul (LH) vehicles of larger capacity, different speed and cost, in comparison to the short-haul (SH) vehicles used for intra-region transportation. The characteristics of these problems lead to the necessity of dealing with other issues like multi-modal transportation and multi-depot networks. Chapter 2 provides a literature review and step-by-step construction of mathematical models.

In Chapter 3 a two-region PDP with a LH connection is examined. A matheuristic algorithm was developed and the problem is decomposed into a LH assignment and the SH routing. The LH assignment is solved exactly and then provides input for the SH routing. To demonstrate the quality of the solution algorithm, it is compared to a similar problem from the literature. Additionally, two types of self-generated instances are provided, namely random ones and realistic ones. The methodological contribution consists of the implementation of the pilot insertion method (PI) and the development of the extended pilot insertion method (EPI). The comparison indicates that the EPI can be superior. Furthermore, we studied the effects of a more flexible LH connection on SH routing costs. When offering additional LH departure times, without increasing the amount of actual LH departures, we observe SH cost reductions of up to 22%. On average, more LH connections result in a 11% (for the PI) or 12% (for the EPI) cost decrease. Decision makers need to be aware of the importance of synchronization between transportation modes, as well as the need to remain flexible with respect to predefined schedules. However, a flexible LH schedule can have its drawbacks, especially if multiple SH routing problems depend on it, and the potential SH savings have to be balanced against the cost of flexibility.

Chapter 4 investigates a new potential product for the portfolio of transportation carriers to increase flexibility for customers without increasing cost for the carriers: the PDP with alternative locations (PDPAL). We include the possibility to add both roaming locations for pickup and deliveries as well as an alternative recipient and even 24-hour locker boxes. When comparing this flexible system with a traditional home delivery system, we generate potential cost savings up to almost 30%. This includes a detailed examination of the potential benefits of locker boxes (up to 6.4% cost savings) or the implications if only a part of the customer base is willing to participate in the system. We showed that allowing for roaming locations and alternative recipients is profitable even if only 25% of customers are participating in the system.

Chapter 5 compares different heuristics for the PDPAL. We compare five different construction heuristics for the routing: two nearest neighbor variations and three insertion variations. For instances with time windows neither of the nearest neighbor algorithms can be recommended. The potential of local search is examined, as well as the importance of adding a waiting time weight

when calculating the routing costs for problems with time windows. For the request-to-vehicle assignment we compare four different metaheuristics: an Adaptive Large Neighborhood Search (ALNS), a Genetic Algorithm (GA), a multi-start ALNS (MS-ALNS), and a combination of GA and ALNS (GA-ALNS). GA seems to be the worst method by far. The ALNS leads quickly to improvements but seems to become stagnant later in the run time. GA-ALNS improves quicker than MS-ALNS, however, at the end of the run time MS-ALNS seems to have outperformed the GA-ALNS.

In Chapter 6 the results of the VeRoLog Solver Challenge 2016 - 2017 are reported. The competition problem is based on a real-life problem of a cattle improvement company that combines routing, scheduling and inventory aspects. Instances were generated with wildly different cost penalties for different parts of the objective function, making the problem potentially relevant from a multi-objective point of view. For the challenge we used an intuitive approach and focused on decomposing the problem. A GA was used for the scheduling and a Variable Neighborhood Decent for the routing improvement. The presented solution approach was awarded third place in the challenge.

The work presented in Chapter 7 resulted from the necessity to obtain fast and accurate approximations of travel distances without necessarily solving a complex transportation problem. This is important for many applications, for example auction based carrier collaborations. Chapter 7 presents an empirical approach to estimate total travel distances for different routing problems using different properties as predictors in regression models. Our models covered a wide range of transportation problems, ranging from conceptually simple traveling salesman problems to complex multi-region transportation problems involving a wide spectrum of real-life constraints. An important feature of the empirical approach for this type of applications is that our models can not only be fitted to optimal solutions, but also reliably predict the travel distances that can be obtained with heuristic methods like the one we have utilized for the two-region multi-depot pickup and delivery problem (2R-MDPDP).

Bibliography

[1] Ahmed, L., Mumford, C., and Kheiri, A. (2019). Solving urban transit route design problem using selection hyper-heuristics. *European Journal of Operational Research*, 274(2):545–559.

[2] Al Karim, R. (2013). Customer satisfaction in online shopping: A study into the reasons for motivations and inhibitions. *IOSR Journal of Business and Management*, 11(6):13–20.

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

[4] Angelelli, E. and Speranza, M. G. (2002). The periodic vehicle routing problem with intermediate facilities. *European Journal of Operational Research*, 137(2):233–247.

[5] Arnold, D., Furmans, K., Isermann, H., Kuhn, A., and Tempelmeier, H., editors (2008). *Handbuch Logistik*, volume 3. Berlin: Springer.

[6] Asta, S. and Özcan, E. (2015). A tensor-based selection hyper-heuristic for cross-domain heuristic search. *Information Sciences*, 299:412–432.

[7] Bäck, T. (1996). *Evolutionary algorithms in theory and practice: Evolution strategies, evolutionary programming, genetic algorithms*. Oxford university press.

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

[9] Bard, J. F., Huang, L., Dror, M., and Jaillet, P. (1998). A branch and cut algorithm for the VRP with satellite facilities. *IIE Transactions*, 30(9):821–834.

[10] Barnhart, C. and Schneur, R. R. (1996). Air network design for express shipment service. *Operations Research*, 44(6):843–1019.

[11] Battarra, M., Cordeau, J.-F., and Iori, M. (2014). Pickup-and-delivery problems for goods transportation. In Toth, P. and Vigo, D., editors, *Vehicle Routing*, chapter 6, pages 161–191.

[12] Battarra, M., Monaci, M., and Vigo, D. (2009). An adaptive guidance approach for the heuristic solution of a minimum multiple trip vehicle routing problem. *Computers & Operations Research*, 36(11):3041–3050.

[13] Beardwood, J., Halton, J. H., and Hammersley, J. M. (1959). The shortest path through many points. *Mathematical Proceedings of the Cambridge Philosophical Society*, 55(4):299–327.

[14] Berbeglia, G., Cordeau, J.-F., Gribkovskaia, I., and Laporte, G. (2007). Static pickup and delivery problems: A classification scheme and survey. *TOP*, 15(1):1–31.

[15] Berger, S. and Bierwirth, C. (2010). Solutions to the request reassignment problem in collaborative carrier networks. *Transportation Research Part E: Logistics and Transportation Review*, 46(5):627–638.

[16] Bettinelli, A., Ceselli, A., and Righini, G. (2014). A branch-and-price algorithm for the multi-depot heterogeneous-fleet pickup and delivery problem with soft time windows. *Mathematical Programming Computation*, 6(2):171–197.

[17] Bontekoning, Y. M., Macharis, C., and Trip, J. J. (2004). Is a new applied transportation research field emerging?—A review of intermodal rail–truck freight transport literature. *Transportation Research Part A: Policy and Practice*, 38(1):1–34.

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

[19] Buijs, P., Alvarez, J. A. L., Veenstra, M., and Roodbergen, K. J. (2016). Improved collaborative transport planning at dutch logistics service provider Fritom. *Interfaces*, 46(2):119–132.

[20] Burke, E. K., Gendreau, M., Hyde, M., Kendall, G., Ochoa, G., Özcan, E., and Qu, R. (2013). Hyper-heuristics: A survey of the state of the art. *Journal of the Operational Research Society*, 64(12):1695–1724.

[21] Çavdar, B. and Sokol, J. (2015). A distribution-free TSP tour length estimation model for random graphs. *European Journal of Operational Research*, 243(2):588–598.

[22] Ceselli, A., Righini, G., and Salani, M. (2009). A column generation algorithm for a rich vehicle-routing problem. *Transportation Science*, 43(1):56–69.

[23] Chien, T. W. (1992). Operational estimators for the length of a traveling salesman tour. *Computers & Operations Research*, 19(6):469–478.

[24] Choubey, N. S. (2013). Moving target travelling salesman problem using genetic algorithm. *International Journal of Computer Applications*, 70(2):30–34.

[25] Christofides, N. and Eilon, S. (1969). Expected distances in distribution problems. *Journal of the Operational Research Society*, 20(4):437–443.

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

[27] Coelho, L. C., Cordeau, J.-F., and Laporte, G. (2014). Thirty years of inventory routing. *Transportation Science*, 48(1):1–19.

[28] Cordeau, J.-F., Gendreau, M., Laporte, G., Potvin, J.-Y., and Semet, F. (2002). A guide to vehicle routing heuristics. *Journal of the Operational Research Society*, 53(5):512–522.

[29] Cordeau, J.-F. and Laporte, G. (2007). The dial-a-ride problem: Models and algorithms. *Annals of Operations Research*, 153(1):29–46.

[30] 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*, pages 327–357. Springer.

[31] Crainic, T. G. and Kim, K. H. (2007). Intermodal transportation. *Handbooks in Operations Research and Management Science*, 14:467–537.

[32] Crainic, T. G., Mancini, S., Perboli, G., and Tadei, R. (2011). Multi-start heuristics for the two-echelon vehicle routing problem. In *European Conference on Evolutionary Computation in Combinatorial Optimization*, pages 179–190. Springer.

[33] Crainic, T. G., Perboli, G., and Rosano, M. (2018). Simulation of intermodal freight transportation systems: A taxonomy. *European Journal of Operational Research*, 270(2):401–418.

[34] Crevier, B., Cordeau, J.-F., and Laporte, G. (2007). The multi-depot vehicle routing problem with inter-depot routes. *European Journal of Operational Research*, 176(2):756–773.

[35] Croes, G. A. (1958). A method for solving traveling-salesman problems. *Operations Research*, 6(6):791–812.

[36] Currie, R. H. and Salhi, S. (2003). Exact and heuristic methods for a full-load, multi-terminal, vehicle scheduling problem with backhauling and time windows. *Journal of the Operational Research Society*, 54(4):390–400.

[37] Daganzo, C. F. (1984). The distance traveled to visit N points with a maximum of C stops per vehicle: An analytic model and an application. *Transportation Science*, 18(4):331–350.

[38] Daganzo, C. F. (1987). Modeling distribution problems with time windows: Part I. *Transportation Science*, 21(3):171–179.

[39] Dethloff, J. (2001). Vehicle routing and reverse logistics: The vehicle routing problem with simultaneous delivery and pick-up. *OR Spectrum*, 23(1):79–96.

[40] Detti, P., Papalini, F., and de Lara, G. Z. M. (2017). A multi-depot dial-a-ride problem with heterogeneous vehicles and compatibility constraints in healthcare. *Omega*, 70:1–14.

[41] Dondo, R. and Cerdá, J. (2007). A cluster-based optimization approach for the multi-depot heterogeneous fleet vehicle routing problem with time windows. *European Journal of Operational Research*, 176(3):1478–1507.

[42] Dragomir, A. G. and Doerner, K. F. (2020). Solution techniques for the inter-modal pickup and delivery problem in two regions. *Computers & Operations Research*, 113. available online.

[43] Dragomir, A. G., Nicola, D., Soriano, A., and Gansterer, M. (2018). Multidepot pickup and delivery problems in multiple regions: A typology and integrated model. *International Transactions in Operational Research*, 25(2):569–597.

[44] Dragomir, A. G., Van Woensel, T., and Doerner, K. F. (2020). The pickup and delivery problem with alternative locations and overlapping time windows. *Transportation Research Part B: Methodological*. submitted.

[45] Dror, M., Fortin, D., and Roucairol, C. (1998). Redistribution of self-service electric cars: A case of pickup and delivery. Technical Report RR-3543, INRIA.

[46] Dueck, G. (1993). New optimization heuristics: The great deluge algorithm and the record-to-record travel. *Journal of Computational Physics*, 104(1):86–92.

[47] Dullaert, W., Gromicho, J., van Hoorn, J., Post, G., and Vigo, D. (2017). The VeRoLog Solver Challenge 2016–2017. *Journal on Vehicle Routing Algorithms*, pages 1–3.

[48] Dumas, Y., Desrosiers, J., and Soumis, F. (1991). The pickup and delivery problem with time windows. *European Journal of Operational Research*, 54(1):7–22.

[49] eBay (2019). eBay Inc. reports fourth quarter and full year 2018 results and announces capital structure evolution. `https://www.ebayinc.com/stories/news/ebay-q4-2018-results/`.

[50] Eilon, S., Watson-Gandy, C. D. T., and Christofides, N. (1971). *Distribution management: Mathematical modelling and practical analysis*. Griffin, London.

[51] Erera, A. L. (2000). *Design of large-scale logistics systems for uncertain environments*. University of California, Berkeley.

[52] Experian (2019). 70% of consumers would share more data if there was a perceived benefit, with greater online security and convenience at the top of the list. https://www.prnewswire.com/news-releases/70-of-consumers-would-share-more-data-if-there-was-a-perceived-benefit-with-greater-online-security-and-convenience-at-the-top-of-the-list-300785756.html.

[53] Fagerland, M. W. and Sandvik, L. (2009). The Wilcoxon–Mann–Whitney test under scrutiny. *Statistics in Medicine*, 28(10):1487–1497.

[54] Figliozzi, M. (2008). Planning approximations to the average length of vehicle routing problems with varying customer demands and routing constraints. *Transportation Research Record: Journal of the Transportation Research Board*, 2089(1):1–8.

[55] Figliozzi, M. A. (2009). Planning approximations to the average length of vehicle routing problems with time window constraints. *Transportation Research Part B: Methodological*, 43(4):438–447.

[56] Fishman, E. (2016). Bikeshare: A review of recent literature. *Transport Reviews*, 36(1):92–113.

[57] Florio, A. M., Feillet, D., and Hartl, R. F. (2018). The delivery problem: Optimizing hit rates in e-commerce deliveries. *Transportation Research Part B: Methodological*, 117:455–472.

[58] Floyd, R. W. (1962). Algorithm 97: Shortest path. *Communications of the ACM*, 5(6):345.

[59] Forrest, S. (1996). Genetic algorithms. *ACM Computing Surveys (CSUR)*, 28(1):77–80.

[60] Franceschetti, A., Jabali, O., and Laporte, G. (2017). Continuous approximation models in freight distribution management. *TOP*, 25(3):413–433.

[61] Gambella, C., Naoum-Sawaya, J., and Ghaddar, B. (2018). The vehicle routing problem with floating targets: Formulation and solution approaches. *INFORMS Journal on Computing*, 30(3):554–569.

[62] Gansterer, M. and Hartl, R. F. (2016). Request evaluation strategies for carriers in auction-based collaborations. *OR Spectrum*, 38(1):3–23.

[63] Gendron, B. and Semet, F. (2009). Formulations and relaxations for a multi-echelon capacitated location–distribution problem. *Computers & Operations Research*, 36(5):1335–1355.

[64] Ghilas, V., Cordeau, J.-F., Demir, E., and Van Woensel, T. (2018). Branch-and-price for the pickup and delivery problem with time windows and scheduled lines. *Transportation Science*, 52(5):1035–1296.

[65] Ghilas, V., Demir, E., and Van Woensel, T. (2016). An adaptive large neighborhood search heuristic for the pickup and delivery problem with time windows and scheduled lines. *Computers & Operations Research*, 72:12–30.

[66] Goebel, P., Moeller, S., and Pibernik, R. (2012). Paying for convenience: Attractiveness and revenue potential of time-based delivery services. *International Journal of Physical Distribution & Logistics Management*, 42(6):584–606.

[67] Goel, A. and Gruhn, V. (2005). Large neighborhood search for rich VRP with multiple pickup and delivery locations. In *Proceedings of the 18th Mini EURO Conference on VNS (MEC-VNS)*. Citeseer.

[68] Goel, A. and Gruhn, V. (2008). A general vehicle routing problem. *European Journal of Operational Research*, 191(3):650–660.

[69] Golden, B., Bodin, L., Doyle, T., and Stewart Jr, W. (1980). Approximate traveling salesman algorithms. *Operations Research*, 28(3-part-ii):694–711.

[70] Gromicho, J. A., Haneyah, S., and Kok, L. (2015). Solving a real-life VRP with inter-route and intra-route challenges. `http://dx.doi.org/10.2139/ssrn.2610549`.

[71] Grünert, T. and Sebastian, H.-J. (2000). Planning models for long-haul operations of postal and express shipment companies. *European Journal of Operational Research*, 122(2):289–309.

[72] Guajardo, M. and Rönnqvist, M. (2016). A review on cost allocation methods in collaborative transportation. *International Transactions in Operational Research*, 23(3):371–392.

[73] Hansen, P. and Mladenović, N. (1999). An introduction to variable neighborhood search. In *Meta-heuristics*, pages 433–458. Springer.

[74] Hart, E., Ross, P., and Corne, D. (2005). Evolutionary scheduling: A review. *Genetic Programming and Evolvable Machines*, 6(2):191–220.

[75] Helsgaun, K. (2000). An effective implementation of the Lin-Kernighan traveling salesman heuristic. *European Journal of Operational Research*, 126(1):106–130.

[76] Helvig, C. S., Robins, G., and Zelikovsky, A. (2003). The moving-target traveling salesman problem. *Journal of Algorithms*, 49(1):153–174.

[77] Hemmelmayr, V. C., Cordeau, J.-F., and Crainic, T. G. (2012). An adaptive large neighborhood search heuristic for two-echelon vehicle routing problems arising in city logistics. *Computers & Operations Research*, 39(12):3215–3228.

[78] Hiermann, G., Puchinger, J., Ropke, S., and Hartl, R. F. (2016). The electric fleet size and mix vehicle routing problem with time windows and recharging stations. *European Journal of Operational Research*, 252(3):995–1018.

[79] Hindle, A. and Worthington, D. (2004). Models to estimate average route lengths in different geographical environments. *Journal of the Operational Research Society*, 55(6):662–666.

[80] Höller, H., Melián, B., and Voß, S. (2008). Applying the pilot method to improve VNS and GRASP metaheuristics for the design of SDH/WDM networks. *European Journal of Operational Research*, 191(3):691–704.

[81] Irnich, S. (2000). A multi-depot pickup and delivery problem with a single hub and heterogeneous vehicles. *European Journal of Operational Research*, 122(2):310–328.

[82] Iwan, S., Kijewska, K., and Lemke, J. (2016). Analysis of parcel lockers' efficiency as the last mile delivery solution – The results of the research in Poland. *Transportation Research Procedia*, 12:644–655.

[83] Jaw, J.-J., Odoni, A. R., Psaraftis, H. N., and Wilson, N. H. (1986). A heuristic algorithm for the multi-vehicle advance request dial-a-ride problem with time windows. *Transportation Research Part B: Methodological*, 20(3):243–257.

[84] Jian, N., Freund, D., Wiberg, H. M., and Henderson, S. G. (2016). Simulation optimization for a large-scale bike-sharing system. In *Proceedings of the 2016 Winter Simulation Conference*, WSC '16, pages 602–613, Piscataway, NJ, USA. IEEE Press.

[85] Johnson, D. S. (1973). *Near-optimal bin packing algorithms*. PhD thesis, Massachusetts Institute of Technology.

[86] Kheiri, A., Dragomir, A. G., Mueller, D., Gromicho, J., Jagtenberg, C., and van Hoorn, J. J. (2019). Tackling a VRP challenge to redistribute scarce equipment within time windows using metaheuristic algorithms. *EURO Journal on Transportation and Logistics*, 8(5):561–595.

[87] Kheiri, A. and Keedwell, E. (2015). A sequence-based selection hyper-heuristic utilising a hidden Markov model. In *Proceedings of the 2015 on Genetic and Evolutionary Computation Conference*, GECCO '15, pages 417–424.

[88] Kheiri, A. and Keedwell, E. (2017). A hidden Markov model approach to the problem of heuristic selection in hyper-heuristics with a case study in high school timetabling problems. *Evolutionary Computation*, 25(3):473–501.

[89] Kheiri, A., Keedwell, E., Gibson, M. J., and Savic, D. (2015). Sequence analysis-based hyper-heuristics for water distribution network optimisation. *Procedia Engineering*, 119:1269–1277. Computing and Control for the Water Industry (CCWI2015) Sharing the best practice in water management.

[90] Kruskal, W. H. (1957). Historical notes on the Wilcoxon unpaired two-sample test. *Journal of the American Statistical Association*, 52(279):356–360.

[91] Kwon, O., Golden, B., and Wasil, E. (1995). Estimating the length of the optimal TSP tour: An empirical study using regression and neural networks. *Computers & Operations Research*, 22(10):1039–1046.

[92] Langevin, A. and Soumis, F. (1989). Design of multiple-vehicle delivery tours satisfying time constraints. *Transportation Research Part B: Methodological*, 23(2):123–138.

[93] Laporte, G. (1992). The vehicle routing problem: An overview of exact and approximate algorithms. *European Journal of Operational Research*, 59(3):345–358.

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

[95] Lin, S. and Kernighan, B. W. (1973). An effective heuristic algorithm for the traveling-salesman problem. *Operations Research*, 21(2):498–516.

[96] Lombard, A., Tamayo-Giraldo, S., and Fontane, F. (2018). Vehicle routing problem with roaming delivery locations and stochastic travel times (VRPRDL-S). *Transportation Research Procedia*, 30:167–177.

[97] MA 23 (2019a). Beschäftigung - Statistiken.

[98] MA 23 (2019b). Bevölkerung nach Bezirken 2004 bis 2019.

[99] Martello, S. and Toth, P. (1990). Lower bounds and reduction procedures for the bin packing problem. *Discrete Applied Mathematics*, 28(1):59–70.

[100] Masson, R., Lehuédé, F., and Péton, O. (2013). An adaptive large neighborhood search for the pickup and delivery problem with transfers. *Transportation Science*, 47(3):295–454.

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

[102] McLeod, F., Cherrett, T., and Song, L. (2006). Transport impacts of local collection/delivery points. *International Journal of Logistics*, 9(3):307–317.

[103] Mendoza, J. E., Castanier, B., Guéret, C., Medaglia, A. L., and Velasco, N. (2011). Constructive heuristics for the multicompartment vehicle routing problem with stochastic demands. *Transportation Science*, 45(3):285–449.

[104] Min, H. (1989). The multiple vehicle routing problem with simultaneous delivery and pick-up points. *Transportation Research Part A: General*, 23(5):377 – 386.

[105] Min, H., Current, J., and Schilling, D. (1992). The multiple depot vehicle routing problem with backhauling. *Journal of Business Logistics*, 13(1):259.

[106] Mladenović, N. and Hansen, P. (1997). Variable neighborhood search. *Computers & Operations Research*, 24(11):1097–1100.

[107] Moccia, L., Cordeau, J.-F., Laporte, G., Ropke, S., and Valentini, M. P. (2011). Modeling and solving a multimodal transportation problem with flexible-time and scheduled services. *Networks*, 57(1):53–68.

[108] Montané, F. A. T. and Galvão, R. D. (2006). A tabu search algorithm for the vehicle routing problem with simultaneous pick-up and delivery service. *Computers & Operations Research*, 33(3):595–619.

[109] Montoya-Torres, J. R., Franco, J. L., Isaza, S. N., Jiménez, H. F., and Herazo-Padilla, N. (2015). A literature review on the vehicle routing problem with multiple depots. *Computers & Industrial Engineering*, 79:115–129.

[110] Montoya-Torres, J. R., Muñoz-Villamizar, A., and Vega-Mejía, C. A. (2016). On the impact of collaborative strategies for goods delivery in city logistics. *Production Planning & Control*, 27(6):443–455.

[111] Morganti, E., Seidel, S., Blanquart, C., Dablanc, L., and Lenz, B. (2014). The impact of e-commerce on final deliveries: Alternative parcel delivery services in France and Germany. *Transportation Research Procedia*, 4:178–190.

[112] Muter, I., Cordeau, J.-F., and Laporte, G. (2014). A branch-and-price algorithm for the multidepot vehicle routing problem with interdepot routes. *Transportation Science*, 48(3):425–441.

[113] Naccache, S., Côté, J.-F., and Coelho, L. C. (2018). The multi-pickup and delivery problem with time windows. *European Journal of Operational Research*, 269(1):353–362.

[114] Nagy, G. and Salhi, S. (2005). Heuristic algorithms for single and multiple depot vehicle routing problems with pickups and deliveries. *European Journal of Operational Research*, 162(1):126–141.

[115] Nguyen, V.-P., Prins, C., and Prodhon, C. (2012). Solving the two-echelon location routing problem by a GRASP reinforced by a learning process and path relinking. *European Journal of Operational Research*, 216(1):113–126.

[116] Nicola, D., Vetschera, R., and Dragomir, A. G. (2019). Total distance approximations for routing solutions. *Computers & Operations Research*, 102:67–74.

[117] Oliveira, A. S., Savelsbergh, M., Veelenturf, L., and Van Woensel, T. (2019). Crowd-based city logistics. In *Sustainable Transportation and Smart Logistics*, pages 381–400. Elsevier.

[118] OpenStreetMap contributors (2017). Planet dump retrieved from https://planet.osm.org . `https://www.openstreetmap.org`.

[119] Ozbaygin, G., Karasan, O. E., Savelsbergh, M., and Yaman, H. (2017). A branch-and-price algorithm for the vehicle routing problem with roaming delivery locations. *Transportation Research Part B: Methodological*, 100:115–137.

[120] Ozbaygin, G. and Savelsbergh, M. (2019). An iterative re-optimization framework for the dynamic vehicle routing problem with roaming delivery locations. *Transportation Research Part B: Methodological*, 128:207–235.

[121] Parragh, S. N., Doerner, K. F., and Hartl, R. F. (2008a). A survey on pickup and delivery models - Part I: Transportation between customers and depot. *Journal für Betriebswirtschaft*, 58(1):21–51.

[122] Parragh, S. N., Doerner, K. F., and Hartl, R. F. (2008b). A survey on pickup and delivery problems - Part II: Transportation between pickup and delivery locations. *Journal für Betriebswirtschaft*, 58(2):81–117.

[123] Perboli, G. and Rosano, M. (2019). Parcel delivery in urban areas: Opportunities and threats for the mix of traditional and green business models. *Transportation Research Part C: Emerging Technologies*, 99:19–36.

[124] Perboli, G., Rosano, M., Saint-Guillain, M., and Rizzo, P. (2018). Simulation–optimisation framework for city logistics: An application on multimodal last-mile delivery. *IET Intelligent Transport Systems*, 12(4):262–269.

[125] Perboli, G. and Tadei, R. (2010). New families of valid inequalities for the two-echelon vehicle routing problem. *Electronic Notes in Discrete Mathematics*, 36:639–646.

[126] Perboli, G., Tadei, R., and Vigo, D. (2011). The two-echelon capacitated vehicle routing problem: Models and math-based heuristics. *Transportation Science*, 45(3):364–380.

[127] Pfrommer, J., Warrington, J., Schildbach, G., and Morari, M. (2014). Dynamic vehicle redistribution and online price incentives in shared mobility systems. *IEEE Transactions on Intelligent Transportation Systems*, 15(4):1567–1578.

[128] Poikonen, S., Golden, B., and Wasil, E. A. (2019). A branch-and-bound approach to the traveling salesman problem with a drone. *INFORMS Journal on Computing*, 31(2):335–346.

[129] Polacek, M., Hartl, R. F., Doerner, K., and Reimann, M. (2004). A variable neighborhood search for the multi depot vehicle routing problem with time windows. *Journal of Heuristics*, 10(6):613–627.

[130] Rainer-Harbach, M., Papazek, P., Hu, B., and Raidl, G. R. (2013). Balancing bicycle sharing systems: A variable neighborhood search approach. In Middendorf, M. and Blum, C., editors, *Evolutionary Computation in Combinatorial Optimization*, pages 121–132, Berlin, Heidelberg. Springer Berlin Heidelberg.

[131] Rainer-Harbach, M., Papazek, P., Raidl, G. R., Hu, B., and Kloimüllner, C. (2015). PILOT, GRASP, and VNS approaches for the static balancing of bicycle sharing systems. *Journal of Global Optimization*, 63(3):597–629.

[132] Rand, G. K. (2009). The life and times of the savings method for vehicle routing problems. *ORiON*, 25(2):125–145.

[133] Raviv, T. and Kolka, O. (2013). Optimal inventory management of a bike-sharing station. *IIE Transactions*, 45(10):1077–1093.

[134] Renaud, J., Laporte, G., and Boctor, F. F. (1996). A tabu search heuristic for the multi-depot vehicle routing problem. *Computers & Operations Research*, 23(3):229–235.

[135] Reyes, D., Savelsbergh, M., and Toriello, A. (2017). Vehicle routing with roaming delivery locations. *Transportation Research Part C: Emerging Technologies*, 80:71–91.

[136] Robusté, F., Estrada, M., and López-Pita, A. (2004). Formulas for estimating average distance traveled in vehicle routing problems in elliptic zones. *Transportation Research Record: Journal of the Transportation Research Board*, 1873(1):64–69.

[137] Rohm, A. J. and Swaminathan, V. (2004). A typology of online shoppers based on shopping motivations. *Journal of Business Research*, 57(7):748–757.

[138] Røpke, S. (2006). *Heuristic and exact algorithms for vehicle routing problems*. PhD thesis.

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

[140] Rosenkrantz, D. J., Stearns, R. E., and Lewis, II, P. M. (1977). An analysis of several heuristics for the traveling salesman problem. *SIAM journal on computing*, 6(3):563–581.

[141] Rothlauf, F. (2011). *Design of modern heuristics: Principles and application*. Springer Science & Business Media.

[142] Salhi, S. and Nagy, G. (1999). A cluster insertion heuristic for single and multiple depot vehicle routing problems with backhauling. *Journal of the Operational Research Society*, 50(10):1034–1042.

[143] Salhi, S. and Sari, M. (1997). A multi-level composite heuristic for the multi-depot vehicle fleet mix problem. *European Journal of Operational Research*, 103(1):95–112.

[144] Sampaio, A., Kinable, J., Veelenturf, L. P., and Van Woensel, T. (2019). A scenario-based approach for the vehicle routing problem with roaming delivery locations under stochastic travel times. Working paper, Optimization Online.

[145] Savelsbergh, M. and Van Woensel, T. (2016). 50th anniversary invited article – city logistics: Challenges and opportunities. *Transportation Science*, 50(2):579–590.

[146] Savelsbergh, M. W. and Sol, M. (1995). The general pickup and delivery problem. *Transportation Science*, 29(1):17–29.

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

[148] Schuijbroek, J., Hampshire, R., and van Hoeve, W.-J. (2017). Inventory rebalancing and vehicle routing in bike sharing systems. *European Journal of Operational Research*, 257(3):992–1004.

[149] Shaw, P. (1998). Using constraint programming and local search methods to solve vehicle routing problems. In *International Conference on Principles and Practice of Constraint Programming*, pages 417–431. Springer.

[150] Solomon, M. M. (1987). Algorithms for the vehicle routing and scheduling problems with time window constraints. *Operations Research*, 35(2):254–265.

[151] Song, L., Cherrett, T., McLeod, F., and Guan, W. (2009). Addressing the last mile problem: Transport impacts of collection and delivery points. *Transportation Research Record*, 2097(1):9–18.

[152] Sörensen, K. and Glover, F. W. (2013). Metaheuristics. In Gass, S. I. and Fu, M. C., editors, *Encyclopedia of Operations Research and Management Science*, pages 960–970. Springer US.

[153] Spiegel, J. R., McKenna, M. T., Lakshman, G. S., and Nordstrom, P. G. (2011). Method and system for anticipatory package shipping. US Patent 8,086,546.

[154] Stadt Wien MA 18 - Stadtentwicklung und Stadtplanung, Bevölkerungsevidenz Oktober 2017, MA 14, MA 41 (2017). Einwohnerinnen-dichte 2017. `https://www.wien.gv.at/stadtentwicklung/grundlagen/stadtforschung/karten/bevoelkerung.html`.

[155] Statistik Austria (May 2019). Erwerbstätige und unselbständig Erwerbstätige nach Vollzeit/Teilzeit und Geschlecht seit 1994.

[156] SteadieSeifi, M., Dellaert, N. P., Nuijten, W., Van Woensel, T., and Raoufi, R. (2014). Multimodal freight transportation planning: A literature review. *European Journal of Operational Research*, 233(1):1–15.

[157] Steffen, A. (2017). Second-hand consumption as a lifestyle choice. *International Conference on Consumer Research (ICCR)*, pages 189–207.

[158] Stein, D. M. (1978). An asymptotic, probabilistic analysis of a routing problem. *Mathematics of Operations Research*, 3(2):89–101.

[159] Talbi, E.-G. (2009). *Metaheuristics: From design to implementation*, volume 74. John Wiley & Sons.

[160] Vidal, T., Crainic, T. G., Gendreau, M., Lahrichi, N., and Rei, W. (2012). A hybrid genetic algorithm for multidepot and periodic vehicle routing problems. *Operations Research*, 60(3):611–624.

[161] Vidal, T., Crainic, T. G., Gendreau, M., and Prins, C. (2013). Heuristics for multi-attribute vehicle routing problems: A survey and synthesis. *European Journal of Operational Research*, 231(1):1–21.

[162] Voß, S., Fink, A., and Duin, C. (2005). Looking ahead with the pilot method. *Annals of Operations Research*, 136(1):285–302.

[163] Wall, M. B. (1996). *A genetic algorithm for resource-constrained scheduling*. PhD thesis, Massachusetts Institute of Technology.

[164] Wang, X. and Kopfer, H. (2014). Collaborative transportation planning of less-than-truckload freight. *OR Spectrum*, 36(2):357–380.

[165] Webb, M. (1968). Cost functions in the location of depots for multiple-delivery journeys. *Operational Research Quarterly*, 19(3):311–320.

[166] Wieberneit, N. (2008). Service network design for freight transportation: A review. *OR Spectrum*, 30(1):77–112.

[167] Wilson, D., Rodrigues, S., Segura, C., Loshchilov, I., Hutter, F., Buenfil, G. L., Kheiri, A., Keedwell, E., Ocampo-Pineda, M., Özcan, E., Pena, S. I. V., Goldman, B., Rionda, S. B., Hernandez-Aguirre, A., Veeramachaneni, K., and Cussat-Blanc, S. (2018). Evolutionary computation for wind farm layout optimization. *Renewable Energy*, 126:681–691.

[168] Zäpfel, G., Braune, R., and Bögl, M. (2010). *Metaheuristic search concepts: A tutorial with applications to production and logistics*. Springer Science & Business Media.

Abstract

The thesis is comprised of topics dealing with variations of the pickup and delivery problem (PDP), a subfield of commercial transportation problems. The research on PDPs has been advanced thematically and methodologically by studying new topics and developing extensions of existing algorithms and new solution methods.

In particular, this thesis considers transportation networks operating in multiple regions that are linked by a long-haul connection. An extensive literature review for the multi-region multi-depot pickup and delivery problem (MR-MDPDP) and its preceding problems is provided as well as more detailed work on a three-part network structure to solve a PDP with long-hauls without direct shipments between regions while fulfilling capacity and time window constraints. The insights gained from this problem show that long-haul flexibility influences short-haul routing cost, such that large improvements are possible merely by increasing long-haul flexibility but not long-haul cost. This work was also extended as a basis to train a regression model to enable the fast estimation of cost for the MR-MDPDP.

The thesis also examines the pickup and delivery problem with alternative locations and overlapping time windows in one region. Each request may have multiple roaming pickup locations throughout the day with non-overlapping time windows. Requests may also have multiple roaming delivery locations and additionally an alternative recipient with its own set of roaming locations. The assumption is that it is no longer necessary to deliver to a single specific person. Multiple persons in different locations can be available simultaneously to accept a delivery. Additionally, recipients of parcels can use 24-hour locker boxes supplied by the logistics provider if they are located near their home. In particular, we explore the benefits of locker boxes, roaming locations, alternative recipients, and mixed customer profiles with different preferences concerning data sharing and convenience. We find that an increase in convenience for the customers and flexibility translates into large cost savings for the carriers.

A rich variation of the pickup and delivery problem was solved as part of the VeRoLog Solver Challenge 2016–2017. This problem required the redistribution of expensive and, therefore, scarce equipment over customer locations within time windows. The difficulty was in creating combinations of pickups and deliveries that reduce the amount of equipment needed to execute the schedule, as well as the lengths of the routes and the number of vehicles used. The developed algorithm was awarded third place in the challenge.

Methodologically, this thesis applied heuristics, metaheuristics, matheuristics, and decomposition approaches. Some established methods (adaptive large neighborhood search, variable neighborhood decent, genetic algorithms) are applied successfully, while others (extension of the pilot method, variations of the savings algorithm and the insertion algorithm) are adapted and extended for a better applicability on the problems studied. The quality of the solution methods is ensured by comparing with suitable previous results from the literature. Managerial insights are gained by solving realistic and real-life scenarios.

Zusammenfassung

Diese Dissertation befasst sich mit Variationen des Abhol- und Zustellproblems (engl. pickup and delivery problem - PDP), einem Teilbereich des gewerblichen Transports. Die Forschung wurde thematisch und methodisch vorangetrieben, indem neue Themen behandelt, bestehende Algorithmen erweitert und neue Lösungsmethoden entwickelt wurden.

Insbesondere befasst sich diese Dissertation mit Verkehrsnetzen, die in mehreren Regionen betrieben werden und über eine Fernverbindung miteinander verknüpft sind. Eine ausführliche Literaturrecherche zum Multi-Region Multi-Depot Abhol- und Zustellproblem (engl. Multi-Region Multi-Depot Pickup and Delivery Problem - MR-MDPDP) und seinen vorhergehenden Problemen wird präsentiert. Es wird eine dreiteiligen Netzwerkstruktur zur Lösung eines PDPs mit Transporten ohne direkte Verbindung zwischen den Regionen betrachtet, wobei Kapazitäts- und Zeitfensterbeschränkungen eingehalten werden müssen. Die aus diesem Problem gewonnenen Erkenntnisse zeigen, dass die Langstreckenflexibilität die Kosten der Kurzstreckenrouten beeinflusst, sodass große Verbesserungen ermöglicht werden wenn die Langstreckenflexibilität, nicht aber dessen Kosten, erhöht wird. Diese Methode wurde auch für ein Regressionsmodell verwendet, um eine schnelle Kostenabschätzung für das MR-MDPDP zu ermöglichen.

Des Weiteren wird das PDP in einer Region mit alternativen Standorten und Zeitfenstern, die sich überschneiden, untersucht. Jeder Transportauftrag kann den ganzen Tag über an mehreren Orten innerhalb von Zeitfenstern abgeholt werden. Der Auftrag kann auch an mehrere Orte zugestellt werden. Zusätzlich dazu gibt es einen alternativen Empfänger. Die Annahme ist, dass es nicht mehr notwendig ist, an eine bestimmte Person zu liefern. Es können mehrere Personen gleichzeitig verfügbar sein, um eine Lieferung entgegenzunehmen. Außerdem können Paketempfänger 24-Stunden Schließfächer verwenden, die vom Logistikdienstleister bereitgestellt werden. Insbesondere werden die Vorteile von Schließfächern, alternative Abhol- und Zustellorte, und alternative Empfänger mit gemischten Kundenprofilen untersucht. Wir stellen fest, dass eine Steigerung des Komforts und mehr Flexibilität für Kunden zu großen Kosteneinsparungen für Spediteure führen kann.

Im Rahmen der VeRoLog Solver Challenge 2016–2017 wurde eine umfangreiche Variante des PDPs gelöst. Dieses Problem erforderte die Umverteilung teurer und daher knapper Geräte über Kundenstandorte zu bestimmten Zeiten. Die Schwierigkeit bestand darin, Kombinationen von Abholungen und Lieferungen zu erstellen, die die Anzahl der Geräte sowie die Länge der Routen und die Anzahl der verwendeten Fahrzeuge reduzieren. Der entwickelte Algorithmus wurde in der Challenge mit dem dritten Platz ausgezeichnet.

Die in dieser Dissertation verwendeten Methoden sind Heuristiken, Metaheuristiken, Matheuristiken und Zerlegungsansätze. Einige etablierte Methoden (Adaptive Large Neighborhood Search, Variable Neighborhood Decent, Genetische Algorithmen) werden erfolgreich eingesetzt, während andere (Erweiterung der Pilot Method, Variationen des Savings Algorithmus und des Insertion Algorithmus) für eine bessere Anpassung an die Probleme erweitert wurden. Die Qualität der Lösungsverfahren wird durch Vergleiche mit Ergebnissen aus der Literatur sichergestellt. Einblicke in mögliche Managemententscheidungen wird durch das Betrachten realistischer und realer Szenarien gewonnen.