# Efficient Heuristics for Routing and Integrated Logistics

*Florian Arnold*

Supervisor: Prof. dr. Kenneth Sörensen

Doctoral dissertation submitted to obtain the degree of Doctor of Applied Economics

Faculty of Business and Economics - Antwerp, 2018

Universiteit Antwerpen

Faculty of Applied Economics

Dissertation

# Efficient Heuristics for Routing and Integrated Logistics

Thesis submitted in order to obtain the degree
of
Doctor in Applied Economics

*Author:*
Florian Arnold

*Supervisor:*
Prof. dr. Kenneth Sörensen

August, 2018

ISBN: 978-90-8994-182-4
Cover design: University of Antwerp, photo by Jasper Léonard
Printing and binding: Universitas, Antwerp

# Publications

This thesis is based on the following publications and working papers:

- Arnold, F. and Sörensen, K. (2018b). What makes a VRP solution good? The generation of problem-specific knowledge for heuristics. *Computers & Operations Research*. Advance online publication. doi:10.1016/j.cor.2018.02.007

- Arnold, F. and Sörensen, K. (2017). Knowledge-guided local search for the Vehicle Routing Problem. Working paper 12, University of Antwerp, Faculty of Applied Economics. Submitted to Computers & Operations Research

- Arnold, F., Gendreau, M., and Sörensen, K. (2017b). Efficiently solving very large scale routing problems. Working paper 75, Polytechnique Montreal, CIRRELT. Submitted to Computers & Operations Research

- Arnold, F., Sörensen, K., and Palhazi Cuervo, D. (2017c). From storage to shipment - The effect of ignoring inventory when planning routes. Working paper 2, University of Antwerp, Faculty of Applied Economics. Submitted to European Journal of Operations Research

- Arnold, F., Sörensen, K., and Springael, J. (2018). Vertical integration of distribution and inventory management with pooling. Working paper, University of Antwerp, Faculty of Applied Economics. Submitted to Omega

- Arnold, F. and Sörensen, K. (2018a). Efficiently solving location routing problems with routing heuristics. Working paper, University of Antwerp, Faculty of Applied Economics. Submitted to European Journal of Operations Research

- Arnold, F., Cardenas, I., Sörensen, K., and Dewulf, W. (2017a). Simulation of B2C e-commerce distribution in Antwerp using cargo bikes and delivery points. *European Transport Research Review*, 10(1):2. doi:10.1007/s12544-017-0272-6

Furthermore, publications of research not covered in this thesis include the following:

- Sörensen, K., Arnold, F., and Palhazi Cuervo, D. (2017). A critical analysis of the "improved Clarke and Wright savings algorithm". *International Transactions in Operational Research*. Advance online publication. doi:10.1111/itor.12443

- Jäger, G. and Arnold, F. (2015). SAT and IP based algorithms for magic labeling including a complete search for total magic labelings. *Journal of Discrete Algorithms*, 31:87–103

- Arnold, F., Guck, D., Kumar, R., and Stoelinga, M. (2014b). Sequential and parallel attack tree modelling. In *International Conference on Computer Safety, Reliability, and Security*, pages 291–299. Springer

- Arnold, F., Hermanns, H., Pulungan, R., and Stoelinga, M. (2014c). Time-dependent analysis of attacks. In *International Conference on Principles of Security and Trust*, pages 285–305. Springer

- Arnold, F., Gebler, D., Guck, D., and Hatefi, H. (2014a). A tutorial on interactive markov chains. In *Stochastic Model Checking. Rigorous Dependability Analysis Using Model Checking Techniques for Stochastic Systems*, pages 26–66. Springer

- Arnold, F., Pieters, W., and Stoelinga, M. (2013b). Quantitative penetration testing with item response theory. In *Information Assurance and Security (IAS), 2013 9th International Conference on*, pages 49–54. IEEE

- Arnold, F., Belinfante, A., Van der Berg, F., Guck, D., and Stoelinga, M. (2013a). DFTCalc: A tool for efficient fault tree analysis. In *International Conference on Computer Safety, Reliability, and Security*, pages 293–301. Springer

# Acknowledgments

It is not easy to describe what it feels like to pursue a PhD, so let me try by using the story of Lord of the Rings as an allegory. Frodo, a young cheerful fellow that does not yet know what to expect from life, is inspired by a wise wizard with a big reputation to take a novel journey. Without actually knowing what the journey is about, Frodo throws himself at it and is entrusted with the One project. Supported by dear friends, he sets out on an unexpected adventure. The path keeps changing and twisting, some obstacles can be avoided others have to be fought through. He masters many challenges, makes great new friends, and meets awe-inspiring people. Eventually, the weight of the One project becomes too heavy, Frodo disappears, and sets out on the perilous path towards Mordor to submit the One project to Mount Doom.

This page is devoted to those dear people that have helped me, directly and indirectly, to carry the One project for the past years. First and foremost, this thesis would never have been written without the trust and steady support of the wizard Kenneth Sörensen. Kenneth's inspiring aura, and calm eye for what really matters, motivated me to leave the beaten path of academic literature and challenge the norm. Kenneth, thank you for many thought-provoking discussions of spontaneous ideas, for putting things in the right perspective, and for offering good advice and cheerful company throughout the years (and for being the last remaining professor to swing the hip on the dance-floor after conference dinners). Many other people accompanied me for a while on my journey. I had the honor to visit Michel Gendreau in Montreal, one of the smartest, most experienced and yet relaxed people I have ever worked with. Johan Springael, in his outmost friendly and ever curious manner, introduced me to the secrets of inventory management. David Martens, with his fresh vigor and wide knowledge, draw me into the bright side of data mining. Daniel Palhazi Cuervo, your energetic attitude and scientific precision helped me to bring many thoughts into order, and Ivan Cardenas supported me to bring theoretical algorithms into the domain of urban distribution. The extraordinary support of An and Esmeralda allowed me to approach every new challenge with ease. My gratitude also goes to the jury members Kris Braekers, Daniele Vigo and Christian Prins for their thoughtful reading of this thesis and for their valuable feedback.

Every journey is only fun in good company. I had great luck to share it with the enthusiastic fellowship of floor 5. High-spirited discussions with Christof, Renata and Nicholas, turning conferences into parties with Floor, Stiene, Babiche, Lissa and Steven, energizing noon-runs with Jasmine and Tyché, evening talks with Trijntje, being a VIP at Tomorrowland thanks to Ellen, Marija, Julie and Jellis... These and so many more good memories made work a place to feel good at.

A journey this long and this intense cannot be completed without one fundamentally-important element: friends that distract you from work and show you that there is more to life. Alan, Cynthia, Amalie and Aman, the guardians, you guys have accompanied me along the whole journey in cheerful and stressful times. Gabi, Vasco and Loresel, you always made me smile with wonderful moments like marveling at the last sunrays at the MAS. Paul, Loïc, Jonas, and Karsten, sharing laughs, good stories and Belgian beers with you never failed to be inspiring. I am grateful for many after-work drinks with Sofia, Rita, Fabiola, Erin, Carlos and my gang in Montréal, and for throwing discs with the Schijnwerpers. Visits, holidays and skype calls with my long-time friends Dennis, Annika, Sascha, Theo, Jan, Andi, Nikolas, Michael, Sven and Enrico got me back on earth when my mind was flying in distant research spheres. And, of course, the most important anchor have been my parents Gundi and Siggi, and my sisters Mieke and Tinka. You provide the kind and loving stability that encourages me on my path.

Now that I am about to finish the One project at Mount Doom, who has this guy become on his journey? Am I smarter, wiser, or at least better looking than before? Or have I made earth a better place? I cannot say so for sure. But what I know is that plenty of magical moments, and the opportunity to believe in and realize my own ideas (and even write a book about them) have made every single bit of this journey worthwhile.

Florian Arnold
Antwerpen, August 2018

# English preface

Routing is a core activity in logistics and involves the transportation of goods from one place to another. One practical example for routing are the deliveries of parcels. Especially with the rise of e-commerce shopping, an increasing number of people order products online and have them delivered as parcels at home. Deliveries for numerous customers are usually bundled on delivery routes, on which one customer is visited after the other. Delivery routes should be planned as efficiently as possible to save costs, and reduce emissions and external effects. On the other hand, the planning of delivery routes is usually interconnected with other decisions in the logistic chain. The demanded products have to be on stock before they can be delivered, resulting in a dependency between inventory management and distribution. Likewise, the location of warehouses constitutes an important strategical decision that impacts delivery routes.

The algorithmic planning of delivery routes, let alone its interplay with other activities, is a difficult problem in the field of combinatorial optimization. Customers need to be assigned to routes, and the customers on the same route need to be visited in the best order. For the delivery of goods to 60 customers, the number of possible solutions is already larger than the number of observable atoms in the universe. Routing is therefore usually tackled with heuristics. Heuristics do not necessarily aim at finding the best routing solution, but rather at finding one that is good enough in a short time. However, many state-of-the-art heuristics are rather complex, can only solve problems up to a certain size, and it is usually difficult to re-implement and apply them in practice.

With this work, I approach the challenge to develop a heuristic that is easily understandable and yet highly-efficient by using two principles. Firstly, routing solutions are improved by analyzing and performing many small changes in a very short time. This technique is called local search, and I investigate various ideas to maximize its performance. Secondly, I investigate the characteristics of good routing solutions by utilizing ideas from data science. The insights can be used to determine where those changes should be made.

The heuristic's efficiency allows to study routing in the context of integrated logistics. Given that a logistic service provider is responsible for planning both, the inventory in warehouses and the deliveries from warehouses to customers, how can both activities be combined to maximize the overall efficiency? Based on previous findings, I hypothesize that the overall efficiency can be elevated if logistic chains are integrated, both vertically and horizontally, and even a strategic decision like the opening of warehouses can be improved by evaluating the corresponding delivery routes. Finally, I want to highlight the practical value of routing algorithms with a case study that tries to answer the question 'what is the best way to distribute parcels in Antwerp?'. I am hopeful that the contributions of this thesis can be used by developers to design simple and good heuristics, and by practitioners to maximize the performance of logistic chains.

# Dutch preface

Routing is een kernactiviteit in de logistieke sector en omvat het vervoer van goederen van de ene plaats naar de andere. Een praktisch voorbeeld van routing is het leveren van pakketten. Vooral met de opkomst van e-commerce, bestellen steeds meer mensen pakketten en laten deze vervolgens thuis leveren. Leveringen voor meerdere klanten worden gebundeld op routes. Er wordt een volgorde bepaald van welke klant wanneer zijn levering zal ontvangen. Deze routes moeten zo efficiënt mogelijk gepland worden om kosten te besparen, emissies te vermijden, en externe effecten te verminderen. De planning van de routes echter, is vaak ook verbonden met andere beslissingen in de logistieke keten. De bestelde producten moeten op voorraad zijn in het magazijn waaruit ze geleverd worden. Dit resulteert in een afhankelijkheid tussen voorraadbeheer en distributie. Evenzo is de locatie van magazijnen een belangrijke strategische beslissing die de routes beïnvloed.

Leveringsroutes plannen aan de hand van algoritmen is een complex probleem dat valt binnen het domein van van combinatorische optimalisatie. Klanten moeten toegewezen worden aan bepaalde routes, de klanten op dezelfde route moeten volgens een optimale volgorde bezocht worden. Leveren aan slechts 60 klanten geeft reeds meer mogelijke volgordes dan het aantal waarneembare atomen in het heelal. Daarom worden routingproblemen meestal opgelost met heuristieken. Heuristieken proberen niet noodzakelijk de beste oplossing te vinden, maar een oplossing die goed genoeg is.

In dit werk ontwikkel ik een uiterst efficiënte heuristiek die op twee principes is gebaseerd. Ten eerste worden oplossingen verbeterd door het analyseren en uitvoeren van veel kleine wijzigingen in een zeer korte tijd, een techniek die 'local search' genoemd wordt. Ik onderzoek verschillende ideeën om de performantie van local search te maximaliseren. Ten tweede onderzoek ik de karakteristieken van goede oplossingen met behulp van ideeën uit data science. De inzichten kunnen gebruikt worden om local search te leiden, en te bepalen waar veranderingen moeten worden aangebracht. De resulterende heuristiek behoort tot de meest efficiënte heuristieken in de literatuur en kan de grootste bestaande problemen aanpakken.

Met deze heuristiek onderzoek ik routing in de context van geïntegreerde logistiek. Gegeven dat een logistieke dienstverlener verantwoordelijk is voor de planning van voorraden en leveringen aan klanten, hoe kunnen beide operaties gecombineerd worden om de efficiëntie te maximaliseren?

Resultaten geven aan dat de globale efficiëntie kan verbeterd worden als de logistieke keten zowel verticaal als horizontaal wordt geïntegreerd. Zelfs een strategische beslissing als de opening van magazijnen kan worden verbeterd door een analyse van de routes. Ten slotte benadruk ik de waarde van routingalgoritmen met een praktische gevalstudie. Deze gevalstudie vergelijkt modellen voor pakketbezorging in Antwerpen en kwantificeert interne en externe kosten via een simulatieaanpak.

Ik hoop dat de contributies van deze thesis gebruikt zullen worden door ontwerpers om eenvoudige en goede heuristieken te ontwerpen, alsook door professionelen om de prestaties van logistieke ketens te maximaliseren.

# Contents

1 INTRODUCTION 1
1.1 Routing 1
1.2 Routing and integrated logistics 3
1.3 Solving routing problems with heuristics 5
1.4 Contributions of this thesis 8
2 THE GENERATION OF PROBLEM-SPECIFIC KNOWLEDGE FOR HEURISTICS 13
2.1 Introduction 13
2.2 From a problem to data 15
2.2.1 Instance generation 15
2.2.2 Computation of solutions 17
2.2.3 Metrics for a solution 20
2.2.4 Metrics for an instance 22
2.3 From data to problem-specific knowledge 24
2.3.1 Prediction accuracy 25
2.3.2 Explaining prediction 26
2.3.3 From explanations to rules 28
2.4 Application of problem-specific knowledge in the design of heuristics 29
2.5 Conclusions 32
3 KNOWLEDGE-GUIDED LOCAL SEARCH FOR THE VEHICLE ROUTING PROBLEM 35
3.1 Introduction 35
3.2 Local search 36
3.2.1 Lin-Kernighan heuristic 39
3.2.2 CROSS-exchange 40
3.2.3 Complementary compound moves - relocation chain 42
3.3 Guiding local search 44
3.4 What makes a local search effective? 47
3.4.1 Neighborhood size and perturbation 48
3.4.2 Penalization criterion 49
3.4.3 Pruning 51
3.4.4 The complete local search heuristic 52
3.5 Computational results 53
3.5.1 Test details 54
3.5.2 Results 54
3.6 Conclusions and future research 56
3.7 Appendix: Detailed benchmark results 58
4 EFFICIENTLY SOLVING VERY LARGE SCALE ROUTING PROBLEMS 63
4.1 Introduction 63

4.2 From large VRPs to very large VRPs . . . . . . . . . . . . . . . . . . . . 65
4.3 Effective local search for very large scale VRPs . . . . . . . . . . . . . . 68
4.3.1 Construction of a starting solution . . . . . . . . . . . . . . . . . 69
4.3.2 Complexity of operators . . . . . . . . . . . . . . . . . . . . . . . 70
4.3.3 Pruning . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 72
4.3.4 Memory usage . . . . . . . . . . . . . . . . . . . . . . . . . . . 73
4.3.5 An effective configuration for very large instances . . . . . . . 75
4.4 New instances . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 76
4.5 Conclusion . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 80
5 INTEGRATING INVENTORY LEVELS IN ROUTING PROBLEMS 81
5.1 Introduction . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 81
5.2 Integration of inventory constraints in routing problems . . . . . . . . 83
5.3 Design of a heuristic . . . . . . . . . . . . . . . . . . . . . . . . . . 84
5.4 Experimental setup . . . . . . . . . . . . . . . . . . . . . . . . . . . 88
5.4.1 Generation of instances . . . . . . . . . . . . . . . . . . . . . . 88
5.4.2 Generation of inventory situations . . . . . . . . . . . . . . . . . 89
5.4.3 Simulation setup . . . . . . . . . . . . . . . . . . . . . . . . . . 90
5.5 Experimental results . . . . . . . . . . . . . . . . . . . . . . . . . . 91
5.6 Conclusions and future research . . . . . . . . . . . . . . . . . . . . 93
5.7 Appendix: Definition of an MPMDVRPI and experimental results . . 95
6 VERTICAL INTEGRATION OF ROUTING AND INVENTORY MANAGEMENT 99
6.1 Introduction . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 99
6.2 Problem formulation . . . . . . . . . . . . . . . . . . . . . . . . . . . 101
6.2.1 Models to pool inventory management and routing . . . . . . . 101
6.2.2 Simplifying hypotheses . . . . . . . . . . . . . . . . . . . . . . . 103
6.3 No pooling . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 104
6.4 Pooling of inventory via lateral transshipments . . . . . . . . . . . . . 105
6.5 Pooling of inventory and distribution . . . . . . . . . . . . . . . . . . 106
6.6 Experimental results . . . . . . . . . . . . . . . . . . . . . . . . . . . 109
6.6.1 No pooling versus inventory pooling . . . . . . . . . . . . . . . 110
6.6.2 Pooling of inventory versus pooling of distribution . . . . . . . 111
6.6.3 The effect of distance between depots . . . . . . . . . . . . . . . 111
6.6.4 The effect of variance in demand . . . . . . . . . . . . . . . . . . 112
6.6.5 Asymmetric setups . . . . . . . . . . . . . . . . . . . . . . . . . . 113
6.7 Conclusion . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 114
7 SOLVING THE LOCATION–ROUTING PROBLEM 117
7.1 Introduction . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 117
7.2 Decomposition of location and routing decisions . . . . . . . . . . . . 119
7.2.1 Finding promising location options . . . . . . . . . . . . . . . . 120
7.2.2 Efficient approximation of routing solutions . . . . . . . . . . . 123
7.3 Solving location–routing problems through iterative filtering . . . . . 124
7.3.1 Capacitated location–routing problems . . . . . . . . . . . . . . 126
7.3.2 Large location–routing problems . . . . . . . . . . . . . . . . . . 126
7.4 Computational experiments . . . . . . . . . . . . . . . . . . . . . . . 128
7.5 Conclusion . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 130

7.6 Appendix: Placement algorithm and detailed benchmark results . . . 132
8 SIMULATION OF B2C E-COMMERCE DISTRIBUTION IN ANTWERP 137
8.1 Introduction . . . 137
8.2 Urban logistics and e-commerce deliveries . . . 138
8.3 Simulation . . . 140
8.4 A simulation study for B2C e-commerce distribution in Antwerp . . . 141
8.4.1 The distribution process in Antwerp . . . 141
8.4.2 Simulation of the distribution process . . . 142
8.4.3 Analysis of delivery costs . . . 145
8.5 Results . . . 147
8.5.1 Simulation of home deliveries by vans . . . 147
8.5.2 Simulation of self-pick-up . . . 149
8.5.3 Simulation of bike deliveries . . . 150
8.5.4 A compromise between low externalities and low operational costs . . . 152
8.5.5 Discussion of limitations . . . 153
8.6 Conclusion . . . 154
9 GENERAL CONCLUSIONS 157

1

# Introduction

## 1.1 Routing

Routing is a core activity in logistics and involves the transportation of goods or people from one place to another. It enables economic and social activities and impacts many aspects of our everyday life. In this thesis, I consider routes which visit multiple locations.Several destinations are given, and all of them have to be visited in an efficient way. Finding efficient routes is a key task especially in industry. Some examples inspired by Toth and Vigo (2002) and Golden et al. (2002) include the following.

**Example 1: Parcel distribution**
With the rise of e-commerce shopping, an increasing number of people order products online and have them delivered at home. In Belgium, for example, the daily quantity of delivered parcels correspond to about 1% of the population, so that in cities like Brussels 20,000 and more deliveries need to be carried out every day. Parcels are delivered by logistic service providers on delivery tours. A delivery tour starts at a distribution center, and delivers parcels to numerous customers or service points with vehicles. With parcel-delivery being a mostly cost-driven business, it is crucial to optimize this distribution. Having fewer delivery vans and fewer delivery kilometers can decrease significantly the costs of service providers, but it also reduces emissions and congestion and thereby increases the quality of life in cities. I will elaborate on this topic further in Chapter 4 and in Chapter 8.

**Example 2: Deliveries in supply chains**
Companies that produce, process and sell products often need to plan the transportation of the raw materials or finished products from one site to another. As an example, super market chains with multiple outlets usually supply these outlets from a central hub, and efficient deliveries from the hub to the outlets can be crucial to have sufficient amounts of products in the shelves. In many supply chain systems,

other activities have to be planned as well. The super market chain might have to decide on the best location for a new hub, and has to manage the inventory for all its products. In Chapter 5, Chapter 6 and Chapter 7 I will investigate in how far transportation decisions can be integrated with these supply chain decisions.

Routing problems are omnipresent in the field of combinatorial optimization and can be formally described as follows. Let $N$ denote the number of destinations that need to be visited from one point of origin. In the following, we will use the standard terminology and call destinations *customers* and the point of origin *depot*. Then we can model a routing problem as a graph $G = (V, E)$ where $V$ denotes the set of nodes and $E$ the set of edges. The $|V| = N + 1$ nodes reflect the customers and the depot ($v_0$). The edges $(i, j)$ reflect the connections between nodes $i$ and $j$ that can be taken. In the standard formulation of the problem, $G$ is complete and undirected, so that each node can be visited from each other node. Each edge $(i, j)$ is annotated with a cost $c(i, j)$ that is realized if the edge is taken. This cost usually corresponds to the distance between two nodes, however, it can also be replaced by other metrics. A *path* in $G$ is a sequence of adjacent edges $(i, j), (j, k), \ldots$ so that all visited nodes are distinct. A *route* in $G$ is then defined as a path that starts at $v_0$, and has an additional edge back to $v_0$, i.e., $(v_0, i), (i, j), \ldots, (k, l), (l, v_0)$.

**Definition: Travelling Salesman Problem (TSP)**
Given a complete graph $G$, find a route that visits all customers and minimizes the sum of the costs of the involved edges.

In many applications the routes have to respect limitations. For instance, a truck can only transport so many parcels. Formally, these constraints are modelled by allocating a demand $d_i$ to each customer $i$. The sum of the demand of all customers on a route may not exceed a certain value $Q$. This limitation is commonly called *capacity constraint*. For instance, $d_i$ could represent the number of parcels that have to be delivered to customer $i$ and $Q$ could define the limit of parcels that can be transported in one vehicle. Because of these limitations multiple routes might have to be planned, which leads to the definition of the Vehicle Routing Problem as introduced by Dantzig and Ramser (1959).

**Definition: Vehicle Routing Problem (VRP)**
Given a complete graph $G$, find a set of routes that visits all customers, respects capacity constraints and minimizes the sum of the costs of the involved edges.

The Travelling Salesman Problem (TSP) and the vehicle routing problem (VRP) are among the most intensively studied problems in the field of combinatorial optimization, and exemplary solutions are visualized in Figure 1.1. Typing 'Vehicle Routing Problem' in Google Scholar results in more than 600.000 findings, an overwhelming number for a single research problem. Much of this popularity can

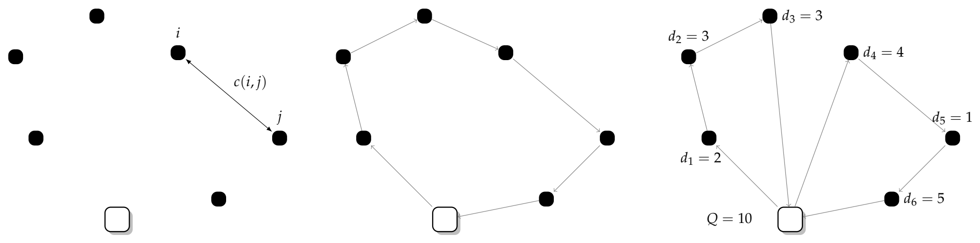


Figure 1.1: (Left) A routing problem is defined by a depot node and several customer nodes that need to be visited. Each edge between two nodes is annotated with a cost $c(i,j)$. (Middle) A solution for a TSP. (Right) A solution for a VRP, in which the capacity restriction has to be respected.

be attributed to its *simplicity*, *complexity* and *practical relevance*. The problem can be defined in one brief paragraph and, yet, it is NP-hard. Thus, problems beyond a certain size can no longer be solved to optimality within a reasonable computing time limit. These two attributes make the VRP highly accessible and yet challenging enough for cutting-edge research. At the same time, the problem is highly relevant in an increasingly interconnected world.

A fourth aspect that contributes to the pervasiveness of the VRP in combinatorial optimization is its *extensibility*. Real-world problems are typically governed by many constraints, conditions and possibly multiple objectives. *Multi-attribute vehicle routing problems* (MAVRP) capture these additional requirements of real-world applications. For instance, the daily working hours of a parcel-delivering driver are limited, and one could model this constraint by imposing a limit on the length of routes. Other variants consider elements that range from characteristics of the fleet of vehicles (i.e., capacity constraints, homogeneous or heterogeneous vehicles, loading policies, driving regulations), to different types of transport requests (i.e., pickup or delivery), storage facilities (i.e., single- or multi-depot - MDVRP), planning horizons (i.e., single- or multi-period) and operating environments (i.e., static, dynamic, deterministic or stochastic). The collection of extended problems has been thoroughly surveyed by Laporte (2009), Lahyani et al. (2015) and Braekers et al. (2016). The focus of this work is to develop solution methods for the standard VRP which can be readily extended to problems in integrated logistics.

## 1.2 Routing and integrated logistics

In 2018 we rely on everything being available very fast at minimal costs. If I forgot to buy a Christmas present in time, I can go online, place an overnight order and have the present delivered the next day at my doorstep, ready to be wrapped in shiny gift wrap paper. The same self-conception is omnipresent in industry. The increasing importance of e-commerce puts high requirements on logistic service providers, who are nowadays facing a higher degree of disintermediation and need to increase the efficiency and flexibility of their delivery activities (Delfmann et al., 2002). Likewise, if a car manufacturer wants to increase his production on short notice, he will demand from his suppliers to deliver more parts.

This efficiency can be realized through *vertical integration* of supply chains. With vertical integration, different activities of a supply chain are synchronized, rather than being planned in isolation. The potential of a holistic perspective on the analysis and planning of logistic processes has been demonstrated in many works, for instance for the integration of production and distribution (Reimann et al., 2014; Chandra and Fisher, 1994) or the integration of location decisions and distribution (Salhi and Rand, 1989; Prodhon and Prins, 2014). In this thesis, I have a particular look on how to combine routing with inventory management and location choices.

Drawing on Example 1 above, larger mail order companies operate in whole countries, continents or even worldwide, and their logistic systems are often comprised of several hierarchical levels (super hub, distribution center) to efficiently funnel the products from geographical wide areas down to granular regional levels and, finally, the customer. In these systems, it is not only important to plan the delivery routes to the customers but also to plan how much and which inventory to stock in which depot. Is it necessary to store all products in sufficient quantity (whatever this is) in all depots? From where are the customers delivered? Questions like these can only be answered if the routing perspective is combined with the inventory perspective.

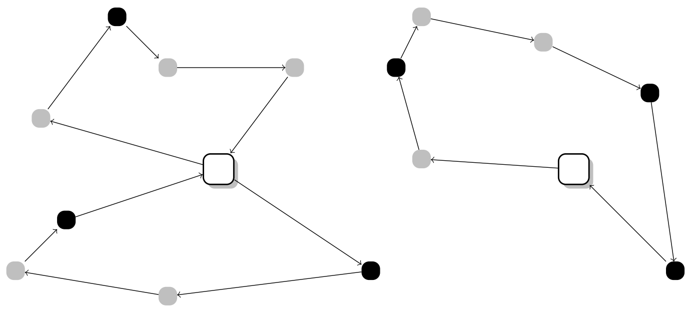


Figure 1.2: Illustration of a routing solution from two depots.

Figure 1.2 illustrates a routing solution in which customers are delivered from two depots. In the standard formulation of this problem, any customer can be delivered from any depot and, thus, we can usually find cost-efficient routes. In reality, the optimal distribution of goods from potentially multiple depots to customers depends on the inventory levels in the depots. When a depot runs out of stock and is not able to deliver the demanded product, delivery might have to be organized by another depot, and, thus, routing decisions have to be adapted. Such a situation is visualized in Figure 1.3. Two different products are delivered from the same depots, however, the right depot has run out of stock such that some deliveries have to be taken over by the left depot. The challenge of this integration of routing and inventory is to optimize the routing decisions taking into account the particular inventory situation in the various depots.

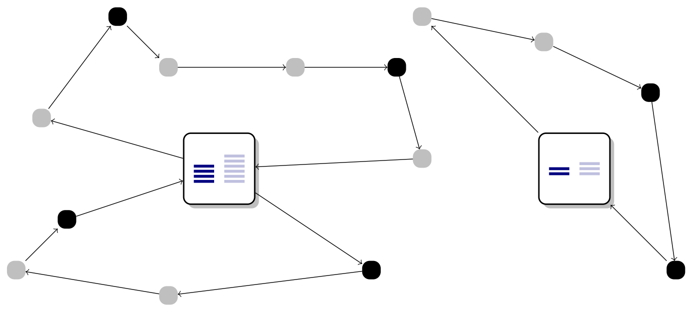


Figure 1.3: The routing solution changes if the inventory levels in the depots are considered.

On a strategic level, companies might consider to relocate, merge, or open new depots, as illustrated in Figure 1.4. This decision impacts the entire logistic chain, and therefore should be carefully analysed by considering its effects on inventory and routing decisions. In the second half of this thesis I investigate such problems and how they can be solved efficiently.

## 1.3 Solving routing problems with heuristics

Routing problems are NP-hard, which means that they can be reduced to any NP-problem, where NP stands for nondeterministic polynomial time. In other words, routing problems are at least as hard as any NP-problem, and there does not (yet) exist a deterministic algorithm which solves them in polynomial time.

A rough estimation of the complexity of a VRP can be done as follows. With complexity I hereby refer to the number of possible solutions for a given VRP

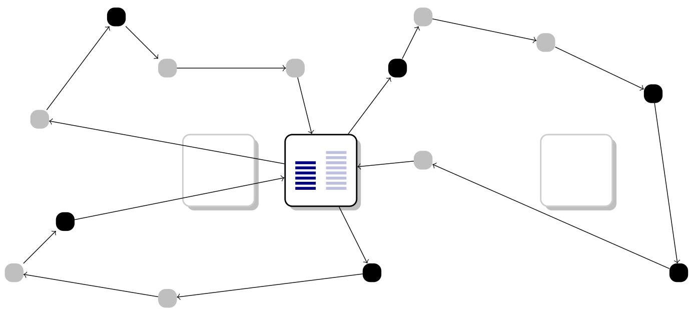


Figure 1.4: The entire supply chain changes if the two depots are replaced by a more central depot.

problem, which can also be called *solution space*. Starting from the depot $v_0$ we can start a route with the visit of one of the $N$ customers. The route can then be continued by visiting one of the remaining $N-1$ customers, and so forth. This results in $N \cdot (N-1) \cdot (N-2) \cdot \dots \cdot = N!$ possible combinations (factorial complexity). The number of possible solutions increases further, if we have to plan multiple routes and need to return to the depot between two customer visits. Problems with factorial complexity grow extremely fast in the input size (the number of customers). For $N = 20$ customers there are already more possible solutions for a VRP than there are cells in the human body (around $10^{14}$). For $N = 60$ the number of VRP solutions is larger than the number of observable atoms in the universe (around $10^{81}$). In Chapter 4 we will solve VRPs with up to 30.000 customers, and the sheer magnitudes of the resulting problem complexity are beyond any physical comparison in the real-world.

Searching through all of these solutions requires much time. A VRP can be modelled as a mixed integer linear problem (MILP), since the objective and constraints can be expressed by linear functions. State-of-the-art solvers use branch-and-cut and column generation techniques to explore the solution space of the problem in an effective way, but because of the rapidly increasing number of solutions, they can only solve VRP instances up to a certain size within a feasible time. Some of the largest instances solved with exact methods, given several hours of computing time, have up to 200 customers (Baldacci et al., 2007; Pecin et al., 2017).

If one wants to tackle larger problems in less computation time, another solution approach has to be chosen. For problems with a large solution space, it might be useful to only look at promising parts, rather than looking at the entire space. This is the basic idea of a *heuristic*. Given that resources are limited, or the problem too complex to consider every alternative, or both, heuristics try to take a shortcut and find a solution that is not necessarily the best one, but good enough. This idea is so fundamental that it is even wired in the human brain. Many decisions that we take in our daily life are not the product of a detailed rational analysis and comparison between all available options, but rather based on past experience and current emotion. In the field of psychology, two of the most famous heuristics are the availability and the representativeness heuristic (Tversky and Kahneman, 1974). When people estimate how likely it is that a certain event occurs, they often base their estimate on the availability of related memories. Even though this is generally a good approach, such events that can be easily brought to mind (e.g., a terror attack) tend to be overestimated. With the representativeness heuristic, people categorise objectives or other people according to similarities to a prototype of that category (e.g., an extrovert). This heuristic pays attention to a particular characteristic, and might ignore how common those categories are in the population. Despite these possible drawbacks, heuristics clearly have their practical benefits, both in real-life and in the design of algorithms.

In the context of the VRP, many impressive heuristics have been developed in the last decade, and VRPs with several hundred customers can be solved to near-optimality in a few minutes of computing time (Vidal et al., 2013a). This empirical success has lead to a race for ever better and faster algorithms and influenced the nature of the research field. Generally, a heuristic that perform well or better than other heuristics has a greater chance to get published than a heuristic that performs less well but offers other contributions, such as insights about a certain component or a study on how a given heuristic behaves in different situations. Measuring scientific contributions with empirical success certainly has its benefits. The bar is set higher and higher and inspires scientists to develop even better algorithms and continuously beat the state-of-the-art. The drawback of this competitiveness is that other desirable research directions are left untouched, such as developing simple and flexible heuristics as argued by Cordeau et al. (2002). Gaining insights into why and in which situation heuristic $A$ works better than heuristic $B$ could offer many insights for future research. Instead, much research has been spent on the extension of heuristic $A$ with component $a_1$, component $b_3$ and $c_6$, or even on the development of heuristic $AB$ which is then coined 'harmony search' or 'imperialist competitive algorithm', as discussed by Sörensen (2015). The final algorithm might perform well, but it possibly raises more questions than offering answers. Why did it work well? Which component worked well together and can be re-used in other heuristics and for other problems? As a result, state-of-the-art heuristics grow in complexity, often without offering sufficient explanations.

This complexity has two main disadvantages. First, studying the impact of the individual components and their parameters, which in turn allows to generalize the findings and generate a deeper understanding of why exactly the algorithm works well, becomes more difficult. Secondly, re-implementing heuristics (either to validate their results or to use them in other research or even in practice) becomes a near-impossible task. Not only does this limit the practical usability of most heuristics for the VRP, it also raises questions about the reproducibility of the presented results.

During the search for an efficient construction heuristic in the early stages of my research, we discovered a promising version of the popular heuristic by Clarke and Wright (1964). The reported results were astonishing for a construction heuristic, which additionally could be implemented in a matter of days. However, careful re-implementation showed that the results published in the paper could not have been produced by the described heuristic (Sörensen et al., 2017). We avoided the discussion on whether this publication was the result of fraud or of carelessness, but rather focused on what the research community could do to stop such obviously wrong results from being published. The solution is rather simple; source code should be disclosed to the reviewers. Of course, sharing code with the entire research community would be even better, as this would not only prevent erroneous of fraudulent results from reaching publication, it would benefit the community as a whole. Overall, this could lead to more comparable heuristics and would relieve the community of the burden of programming everything from scratch.

In summary, there are plenty of different heuristics for the VRP to choose from, and there are few guidelines as to which one works better for which problem type. The re-implementation of one of the suggested approaches is mostly an extremely difficult if not impossible task, at least for high-performing heuristics. This situation, in addition to few available source codes, sets a high entry level to the field. There is much to do in the future.

Sörensen et al. (2018) call this future of the field "the scientific period, during which the design of heuristics becomes a science instead of an art. [...] Most importantly, the change from a performance-driven community to a community in which scientific understanding is more important, will take place during the scientific period. Without doubt, this will lead to the development of even better heuristics, even more efficient, but it will also lead to heuristics that are usable outside of the developer's lab environment."

Some first steps in this direction have been taken, and I can only highlight some of the various excellent contributions. Survey papers summarize the extensive literature on different types of heuristics (Vidal et al., 2013a; Laporte et al., 2014), first libraries for heuristic components are being shared online (Groër et al., 2010), realistic problem instances for testing and benchmark purposes are available Uchoa et al. (2017) and shared on online platforms (Xavier, Ian, 2018; Mendoza et al., 2014), and frameworks for a thorough statistical analysis are being developed (McNabb et al., 2015; Drake et al., 2013; Corstjens et al., 2016).

## 1.4 Contributions of this thesis

The main contributions of this thesis are twofold. Firstly, I aim to contribute to the foundation of the dawning scientific period of heuristics, and generate insights into the VRP (Chapter 2) as well as into key components of VRP heuristics (Chapter 3). These insights are used to design a rather simple and efficient routing heuristic (Chapter 3 and Chapter 4). As a second main contribution, I extend the routing heuristic to solve integrated logistics problems, looking especially on the integration between routing and inventory management (Chapter 5 and Chapter 6), and routing and location decisions (Chapter 7). Finally, I demonstrate the potential of applying routing heuristics to real-world problems with a case study (Chapter 8). Thus, the first part of the thesis constitutes the more theoretical underpinning to efficiently address more practical relevant problems and derive useful guidelines in the second part. In more detail, this thesis provides the following contributions to the academic literature.

Chapter 2 : We argue that knowledge about the structural characteristics that distinguish good from not-so-good solutions of a combinatorial optimization problem, can be instrumental in designing efficient heuristics. We develop a data-mining based approach that can generate such knowledge and apply it to the vehicle routing problem. We define several metrics to characterize both a VRP solution and a VRP instance, and generate and classify 192.000 solutions for various instances. With these metrics we are able to distinguish between optimal and non-optimal solutions with an accuracy of up to 90%. We discuss the most distinguishing characteristics of good VRP solutions, and show how the knowledge thus generated can be used to improve the performance of an existing heuristic.

Chapter 3 : Local search has been established as a successful cornerstone to tackle the VRP, and many state-of-the-art heuristics use it as one ingredient. In this Chapter we aim to demonstrate that a well-implemented local search on its own suffice to create a heuristic that can compete with the best heuristics in the literature. To this end we combine three powerful local search techniques, and implement them in an efficient way that minimizes computational effort. We conduct a series of experiments to determine how local search can be effectively combined with perturbation and pruning, and make use of the problem-specific knowledge extracted in Chapter 2, to guide the search to promising solutions more effectively. The heuristic created in this way not only matches the performance of the best heuristics in the literature, it is also straightforward in its design and does not contain any components of which the contribution is unclear.

Chapter 4 : Almost all research for the VRP so far has been targeted to solve problems of several hundred customers, even though some applications like waste collection and parcel distribution can involve thousands or even tens of thousands of customers. We extend the heuristic from Chapter 3 in such a way that it is able to compute high-quality solutions for these very large scale instances. We demonstrate how to linearize the time and space complexity of a heuristic that is entirely based on a well-implemented local search. The resulting algorithm significantly outperforms a previously introduced heuristic and computes high-quality solutions for instances of 10,000 and more customers in less than an hour. In order to stimulate more research in this domain, we introduce a new set of benchmark instances that is based on a real-world problem.

Chapter 5 : In many situations, the optimal distribution of different goods from potentially multiple depots to customers depends on the inventory levels in the depots. Customers can only be served if sufficient inventory of the demanded product is available. In this Chapter, we present a model that captures inventory constraints in routing problems with multiple depots (MPMDVRPI). As the main contribution, we study the effect that different inventory levels have on the quality of the respective distribution routes. We observe increases of routing costs of up to 80% if inventory levels are sparse, which can be reduced by holding additional safety stock, reducing product variety or opening more depots.

Chapter 6 : From the findings in Chapter 5 we conclude that the optimal inventory strategy depends on the routing decisions, and vice-versa. In this Chapter, we develop an optimisation framework that enables the analysis of integrated decisions. The vertical integration of inventory management in depots and the delivery from depots to customers can improve the performance of a supply chain. Likewise, the collaboration among different depots has proven beneficial in many situations, especially the pooling of inventory via lateral transshipments. We investigate how to effectively combine both concepts, and set up a vertical integration of distribution and inventory decisions in a supply chain with multiple depots. We propose a simple simulation approach to derive the routing and inventory costs for a periodic review model with neglectable lead times, and perform a set of experiments to gain insights into the best supply chain configuration. We observe that in such systems the pooling of distribution activities can be a more cost-efficient way to balance inventory than the pooling via lateral transshipments, while more inventory should be stored in central depots and depots with a higher variance in demand.

Chapter 7 : Next to inventory management, a more strategic decision in supply chains concerns the locations of depots. On the one hand, the delivery of customers needs to be planned as effectively as possible, and on the other hand, the location of depots from where these deliveries are executed has to be determined carefully. In the last years many heuristic approaches have been proposed to tackle the resulting Location Routing Problems (LRP), and usually the computation of excellent solutions comes at the cost of a very intricate algorithmic design. In this Chapter, we introduce an efficient heuristic for LRPs that is almost entirely based on the routing heuristic in Chapter 3, under the premise 'if we can solve routing, we can solve LRPs'. Despite its simple design, it can compete with the best results in literature, and it can also be readily adapted to solve problems of very large scale.

Chapter 8 : Routing is of high practical relevance, but so far we have only modelled and solved abstract problems. In this Chapter, we aim to demonstrate how routing algorithms can be applied to problems in reality with a case study in parcel distribution. The growth of E-commerce is accompanied by an increasing distribution of parcels in cities. Online orders of books or clothing need to be shipped to people's houses, and this transportation of parcels results in externalities like traffic congestion or emissions which reduce the quality of life. As a consequence, other delivery concepts like bike deliveries or delivery points have been suggested and partly tested in practice. Naturally, companies will only accept these changes if they do not result in higher costs or reduced efficiency. However, it is difficult to predict the impact of a certain delivery concept in a certain city. We demonstrate how the effects of different delivery concepts can be quantified and compared with the help of a simulation study. We take care to accurately model the delivery processes and utilise a real-world dataset and realistic cost values. On the basis of these inputs, we simulate and analyse the current state-of-the-practice in the distribution of e-commerce goods in Antwerp and compare it to possible 'what-if' scenarios. The results highlight that the investigated delivery concepts can benefit either the companies or the quality of life in the city. Operational costs of companies can be reduced by stimulating customer self-pick-up, while externalities decrease with the implementation of a cargo bike distribution system. We demonstrate that both operational and external costs can be minimised, if involved stakeholders from industry and the public look for sustainable delivery solution jointly.

# 2 The generation of problem-specific knowledge for heuristics

## 2.1 Introduction

For many combinatorial optimization problems, heuristics based on either local search or constructive strategies have been demonstrated to provide the best trade-off between solution quality and computation time. In recent years, it has become apparent that the efficiency of a heuristic strategy depends to a large extent on how well it is able to exploit the characteristics of the problem being solved. In other words, to be as efficient as possible, heuristic search strategies require some knowledge on what distinguishes a good solution from a not-so-good solution. In many cases, however, this knowledge is limited to the trivial fact that the objective function of the former will be closer to optimal than that of the latter. A question that remains largely unanswered is whether it is possible to determine whether good solutions are *structurally different* from not-so-good solutions, i.e., whether we can distinguish close-to-optimal solutions from not-so-close-to-optimal ones by only looking at the solutions *themselves* and not at their objective function values. When available, this information can be used to *guide* the search, e.g., by removing solution attributes that do not frequently appear in close-to-optimal solutions and replacing them with attributes that do.

In this chapter, we study the capacitated vehicle routing problem (VRP) as introduced in Chapter 1. We investigate the characteristics of good solutions as opposed to those of not-so-good solutions of the VRP, as such information could be used to *guide* the search. If, for example, it is possible to identify edges that are unlikely to appear in good solutions, a strategy can be employed that attempts to remove these edges. The idea of *guiding* a heuristic search process and adapting it to the specific problem instance being solved, has attracted a large amount of attention. For instance, adaptive large neighborhood search (Pisinger and Ropke, 2010) tries to find the best operators during the solution process. In Sörensen et al. (2008) the authors argue in favour of a similar self-adaptive concept within the family of variable neighborhood search algorithms. Simheuristics combine heuristics with simulation techniques, which allows to use the feedback from the simulation to refine the setup of the heuristic (Juan et al., 2015). Instead of adapting the operators during the search, McNabb et al. (2015) use a thorough statistical analysis to identify good operators before the execution of the search.

Rather than adapting or tuning a given heuristic, hyper-heuristics (Burke et al., 2010; Marshall et al., 2014) go a step further by proposing a methodology to select and generate suitable heuristics automatically. For instance, Drake et al. (2013) automatically generate and improve the components of a variable neighbourhood search algorithm.

All of these approaches attempt to guide the search towards promising regions of the solution space by adapting a heuristic algorithm to the problem instance being solved. None of these approaches, however, precede the search with an analysis of which attributes contribute to the quality of a solution. That an analysis of the underlying problem is useful has already been demonstrated in the context of a scheduling problem (Aickelin and Dowsland, 2000). In this paper, the authors used empirical data to explore the structure of a practical nurse scheduling problem, and incorporated the findings successfully in their heuristic. To the best of our knowledge, such an analysis does not exist for the VRP (or any other routing problem).

Our methodology uses *data mining* techniques to find characteristics that distinguish good from not-so-good solutions for the VRP. The purpose of data mining is to find models that identify patterns and establish relationships in huge amounts of data. In contrast to statistics, the goal of these models is "not to infer the true distribution, but rather to predict future data as accurately as possible" (Shmueli et al., 2010). In other words, with data mining we aim to predict previously unseen data instead of explaining data at hand, the focus being on association rather than causation. We therefore also speak of predictive modeling. Even though predictive modelling can lack explanatory power, it is useful to find and test new theories in an exploratory fashion.

The input to any predictive model is a data set, and, as a rule of thumb, more available data points and richer information per data point leads to better models. The information per data point is called its *features*. For instance, we could predict whether a certain client will be persuaded by a new marketing campaign with the help of data about a previous campaign. The dataset of the previous campaign could be composed of the features gender, age, family status and postal code, as well as the information whether the person responded to it. A *classification learner* builds a model that tries to find patterns in the data that distinguish between different classes, e.g., those clients that responded to the previous campaign from those that did not. This process is also called *classification*. This model can then be used to predict future data points, i.e., whether potential addressees will respond to the new campaign. A large variety of classification learners is available, from simple decision trees, to random forests (Breiman, 2001) and Support Vector Machines (SVM) (Hsu et al., 2003).

The predictive model does not necessarily offer causal explanation. However, it is possible to derive some causal theory. If we use simple linear models as predicitive models, explanation about the relative importance of different features can be made by looking at their coefficients. If the internal model is non-linear, we can use *rule extraction* to obtain a set of rules that explain the predictions (Martens et al., 2007). In this way, we are able to deduce theories from data. Since these theories are generalized findings that can be readily applied to new data, we will in the following use the term *knowledge*. A problem is usually characterized by a set of relevant or representative instances, e.g., benchmark instances. If findings about the problem can be generalized to all relevant instances, we speak of *problem-specific knowledge*.

The remainder of this chapter is structured as follows. In Section 2 we describe how to define and generate datapoints for the VRP. The data is then used as the input to a classification learner which provides insights into the problem-structure. We present and discuss the corresponding results in Section 3. In Section 4 we demonstrate with a case-study that these insights can help to design more efficient heuristics, and conclude with a summary of our results in Section 5. Even though we describe the whole process from the perspective of the VRP, the described techniques and ideas should be applicable to a wide range of related and unrelated combinatorial optimization problems.

## 2.2 From a problem to data

Our methodology to generate knowledge on what makes a solution of a VRP instance good is composed of three major steps: (1) generate data, (2) build a predictive model, and (3) extract knowledge from the predictive model. While the latter two steps mainly involve technical challenges, the first one is highly exploratory and requires creativity and a focus on details. Since we can only deduce findings from the data that we create and the features we define, data generation is the most important and the most difficult step. In the following, we describe our data generation process for the VRP. Firstly, we generate random and diverse instances, for which we compute solutions, which we, finally, transform into a feature space.

### 2.2.1 *Instance generation*

The VRP is a complex problem with a great variety of instances that can be varied according to the following attributes (Uchoa et al., 2017): (1) the positioning and number of customers, (2) the positioning of the depot, (3) the distribution of demand, and (4) the average route size or the number of routes, defined by the vehicle capacity. We try to sample a representative set of instances, by randomizing the choice of

Table 2.1: Instance parameters for the different instance classes in the experiments.

| Class | Number of Customers | Depot position | Demand | Routes |
|---|---|---|---|---|
| 1 | 20-50 | Center | [1,1] | 3-6 |
| 2 | 20-50 | Center | [1,10] | 3-6 |
| 3 | 20-50 | Edge | [1,1] | 3-6 |
| 4 | 20-50 | Edge | [1,10] | 3-6 |
| 5 | 70-100 | Center | [1,1] | 6-10 |
| 6 | 70-100 | Center | [1,10] | 6-10 |
| 7 | 70-100 | Edge | [1,1] | 6-10 |
| 8 | 70-100 | Edge | [1,10] | 6-10 |

attributes for a particular instance within reasonable bounds. By the definition of these bounds, we automatically arrive at the definition of different instance classes.

We define a set of small instances with 20-50 customers, and a set of bigger instances with 70-100 customers. Hereby, the customers are located randomly and uniformly on a squared 1000x1000 grid. We do not explicitly consider the clustering of customers, even though customer clusters can form by chance. The clustered VRP is an important problem variant, but we focus on the standard VRP in the following experiments. We expect that solutions for clustered VRPs contain mostly edges that connect customers within the same cluster, and thus, it might be more difficult to reveal differences between solutions of different quality.

We also define two classes for the position of the depot and the demand distribution, respectively. The depot is either located at the center of the grid at position 500x500, or at its edge, i.e., one coordinate is chosen randomly and the other set to 0. Likewise, the variance in customer demand is either rather high (randomly chosen between 1 and 10), or all customers have the same demand. For each combination of these three classes we generate one dataset and, thus, we generate eight datasets in total as presented in Table 2.1. Finally, we define the vehicle capacity randomly in such a way, that we have between 3 and 6 routes in the smaller classes and 6-10 routes in the bigger class. We remark that we could have introduced two more classes, one that contains instances with fewer and one with more routes. However, that would have increased the number of experiments beyond feasibility.

The separate analysis of different instance classes has two advantages. Firstly, differences in these datasets could point to insights, that only holds for certain instance characteristics. Reversely, if similar findings occur in all datasets we can interpret them as problem-specific-knowledge. Secondly, the instances within a dataset are expected to be more homogeneous since they share some attributes, which might enhance predictions on these datasets.

In all of the instances we solve the standard variant of the VRP. The cost of taking a certain edge between two customers is defined by the Euclidean distance and we aim at minimizing the total distance driven on all routes (Laporte, 2007). This standard definition of distance imposes an important assumption on the metrics defined below. For instance, it is straight-forward to define whether two edges form an intersection since every edge is a straight line. On real-world road networks this assumption might no longer hold and it could be more intricate to define accurate metrics.

### 2.2.2 *Computation of solutions*

We want to understand what characterizes a good solution. Formulated differently, we can ask what distinguishes an optimal or near-optimal solution from a worse solution. If we find such distinguishing properties, they could help in finding those good solutions more effectively, being the goal of every heuristic. Thus, we need to compute and compare solutions of different quality.

#### 2.2.2.1 *Computation of near-optimal solutions*

A comprehensive study on the basis of data mining requires sufficient data points. In our case, a data point is derived from a solution for an instance, and thus, for the generation of a dataset we need to compute many solutions in a short time. Since MILP solvers like CPLEX can require several minutes to compute a solution even for a small instance, we use a heuristic. With the use of a heuristic, we cannot guarantee that the computed solutions are always optimal, and thus we will use the terminology *near-optimal* in the following.

The heuristic is a basic version of the Guided Local Search (GLS) introduced in Chapter 3, and does not rely on any problem-specific knowledge. The starting solution is constructed with the heuristic by Clarke and Wright (1964) (CW), and improved with three local search operators. Improvements between routes are executed with an ejection chain (EC) (Glover, 1996) and the CROSS-exchange operator (CE) (Taillard et al., 1997), while a route itself is optimised with the Lin-Kernighan heuristic (Lin and Kernighan, 1973). After an initial local search, the heuristic iteratively penalizes the longest edge, like in Mester and Bräysy (2007). The local search operators then try to remove the penalized edge. After $\frac{N}{10}$ local search changes without improvements we apply the local search to the entire solution, and after $10N$ changes without improvement we remove all penalties and reset the solution to the previously best found solution. Despite its simple design and its deterministic behaviour, the heuristic is able to compute high-quality solutions in very short computation times.

Even though we have no guarantee that the computed solutions are optimal, we can get an intuition about the quality of the solutions with publicly-available benchmark sets for which the optimal solutions are known. We computed solutions for the 27 instances of the Augerat-A benchmark set (Augerat et al., 1995). This set comprises instances with $N \in [31, 79]$ customers and, therefore, corresponds to the size of our generated random instances. We use the heuristic above and allow a maximum runtime of $\frac{N}{20}$ seconds, e.g. 1 second for an instance with 20 customers and 5 seconds for an instance with 100 customers. Given this very limited runtime, we obtain solutions with an average gap of 0.20% to the optimal solution of those instances, and for 63% of the instances we compute the optimal solutions. On the basis of these results we can be reasonable certain that the computed solutions for randomly-generated instances, given a similar size and the same computation time, have a similar high quality.

#### 2.2.2.2 *Computation of non-optimal solutions*

Given a particular instance, we want to compare the above computed near-optimal solutions to solutions of lower quality. In the following, we will call these solutions *non-optimal*. Of course, the term 'lower quality' demands a precise definition, since solutions can vary across a wide range. We decided to generate, for each instance, three solutions of different quality. We first compute a near-optimal solution as defined above with solution value $S^*$. We then generate a solution with value $S^1$ that is about 2% worse, i.e., $\frac{S^1}{S^*} \approx 1.02$, and a solution with value $S^2$ that is about 4% worse. Note that it can be very difficult (or even impossible) to obtain solutions with an exactly specified value, and thus, we accept solutions within a 1% range of the specified gaps (e.g. for $S^1$ we accept a solution that is between 1% to 3% worse than $S^*$). We found that, despite this allowed variability, the average gap of the computed solutions are about 2% and 4% worse, respectively. We choose these two gaps on the intuition that a solution with a gap of 4% can usually be obtained with an effective construction heuristic and is, thus, not especially good. A solution with a 2% gap is significantly better, but there is usually still much computational effort necessary to get from there to a near-optimal solution.

These two different classes of non-optimality allow us to investigate if and in how far the degree of non-optimality impacts differences in the solutions. Intuitively, better solutions should exhibit less structural differences to near-optimal ones. Reversely, there can be many solutions of a similar low quality that have a completely different structure, or in short, the variability in structural characteristics (as defined below) could increases with lower quality. Whereas there might be only a few or even only a unique optimal solution for an instance, there are many more solutions that exhibit a similar gap. Therefore, the structure and characteristics of the non-optimal solution might be biased towards the solution process. For instance, if the solution

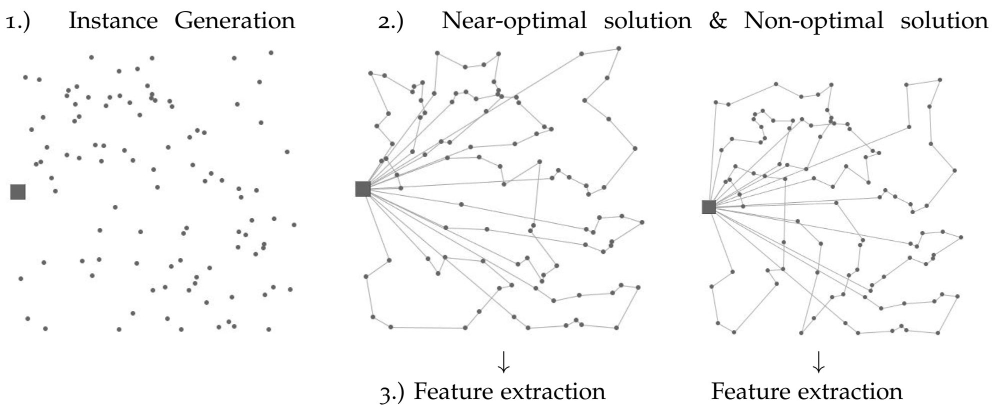


Figure 2.1: Visualization of the data generation process. In a first step we create a random instance (in this case for class 7), and then compute a near-optimal and a non-optimal solution (the right solution is 2.4% worse). For the human eye, it is difficult to recognize and generalize distinguishable pattern between both solutions, whereas classification learners are designed for this purpose. A classification learner requires a numeric input, and we need to transform a solution into a feature space.

process focuses on removing long edges, the computed solutions could differ from those that are computed by focusing on intra-route optimisation. We addressed this possible issue by utilizing two different heuristics H1 and H2 to generate non-optimal solutions.

Heuristic H1 is the same that we use in the computation of the near-optimal solution. It simply stops after it obtains a solution with the aspired quality $S^1$ or $S^2$, e.g. if the solution after the initial local search already meets the required quality we return this solution.

The second heuristic H2 should create solutions in a different manner. For simplicity, we chose the heuristic of Pichpibul and Kawtummachai (2012) because it is easy to implement, does not guide the solution process in any way, and can produce solutions of moderate quality. The heuristic is based on a randomized extension of the parallel Clarke and Wright heuristic (PCW). PCW starts from a solution in which each of the $N$ customers is visited in a separate tour. For each customer pair, the algorithm then determines the saving that would result from connecting these customers directly, and creates a savings list by sorting these savings in decreasing order. Before each run, the algorithm randomizes the current savings list to a certain extent and accepts it, if PCW produces a solution that is better or the same than the previous best solution. Once again the algorithm is aborted if it has found a solution with the specified quality.

Both heuristics were implemented by the authors in Java. All experiments were executed on a Dell E5550 laptop with an Intel Core i7-5600 CPU working at 2.60GHz. The process of instance generation and solution computation is exemplary visualized in Figure 2.1. In the next steps, we need to transform a solution into measurable metrics.

### 2.2.3 *Metrics for a solution*

To discover characteristics that are typical for good solutions, we need to 'measure' a solution with defined metrics, i.e., we need to transform the structure of a solution into quantitative metrics, which later serve as input to the predictive model. This step is highly exploratory, since there are no guidelines about which metrics should be included. As a rule of thumb, the more metrics we can define, the better the chance that the classification learner detects patterns.

For a formal definition of our metrics, we utilize the following notation (for a more intuitive visualization of the metrics we refer to Figure 2.2). Let $R$ denote the set of all routes of a VRP solution, then a route $r_i = \{I_1, I_2, \dots, I_{|r_i|}\} \in R$ is defined by the ordered set of all customers $I_k$ that are visited on this route, starting and ending at depot $D$. Thus, a route $r_i$ has the edges $(D, I_1), (I_1, I_2), \dots, (I_{|r_i|}, D)$. The Euclidean distance between two nodes $I_j$ and $I_k$ is denoted $d(I_j, I_k)$, and $x(I_k)$ and $y(I_k)$ define the $x$ and $y$-coordinate of a node, respectively. The $x$-coordinate of the center of gravity $G_{r_i}$ of a route $r_i$ is then defined as $x(G_{r_i}) = \frac{\sum x(I_k)+x(D)}{|r_i|+1}$, the y-coordinate is computed accordingly. Let $L_{G_{r_i}}$ be the line through $D$ and $G_{r_i}$, then $d(L_{G_{r_i}}, I_k)$ denotes the distance between this line and a customer $I_k$ (an example is visualized in the right graph in Figure 2.2). The distance is positive, if the customer is on the right side of the line and negative otherwise. We denote the absolute value of a number $x$ by $x^+$. Similarly, let $rad(I_j, I_k)$ denote the difference in radians between two nodes, with respect to the depot $D$ (i.e., the angle that is spanned between the lines connecting each of them with the depot). Finally, Let $I(r_1, r_2)$ denote the number of intersections between routes $r_1$ and $r_2$. Then we define the solutions metrics as following:

S1 - Average number of intersections per customer

$$\frac{1}{N} \sum_{i=1}^{|R|-1} \sum_{j=i+1}^{|R|} I(r_i, r_j)$$

S2 - Longest distance between two connected customers, per route

$$\frac{1}{|R|} \sum_{r_i \in R} \max_{k \in \{1, \dots, |r_i|-1\}} d(I_k, I_{k+1})$$

S3 - Average distance between depot to directly-connected customers

$$\frac{1}{2|R|} \sum_{r_i \in R} \left( d(D, I_1) + d(I_{|r_i|}, D) \right)$$

S4 - Average distance between routes (their centers of gravity)

$$\frac{1}{|R| \cdot (|R| - 1)} \sum_{r_1 \in R} \sum_{r_2 \in R \setminus r_1} d(G_{r_1}, G_{r_2})$$

S5 - Average width per route

$$\frac{1}{|R|} \sum_{r_i \in R} \left( \max_{k \in \{1,\ldots,|r_i|\}} d(L_{G_{r_i}}, I_k) - \min_{k \in \{1,\ldots,|r_i|\}} d(L_{G_{r_i}}, I_k) \right)$$

S6 - Average span in radian per route

$$\frac{1}{|R|} \sum_{r_i \in R} \max_{j,k \in \{1,\ldots,|r_i|\}} rad(I_j, I_k)$$

S7 - Average compactness per route, measured by width

$$\frac{1}{N} \sum_{r_i \in R} \sum_{k=1}^{|r_i|} \left( d(L_{G_{r_i}}, I_k) \right)^+$$

S8 - Average compactness per route, measured by radian

$$\frac{1}{N} \sum_{r_i \in R} \sum_{k=1}^{|r_i|} rad(G_{r_i}, I_k)$$

S9 - Average depth per route

$$\frac{1}{|R|} \sum_{r_i \in R} \max_{k \in \{1,\ldots,|r_i|\}} d(I_k, D)$$

S10 - Standard deviation of the number of customers per route

$$\sqrt{\frac{\sum\limits_{r_i \in R} (|r_i| - \frac{N}{|R|})^2}{|R|}}$$

These metrics and their measurement are visualized in Figure 2.2. The first two metrics reflect observations in the field of Operations Research, especially with regard to the famous Traveling Salesman Problem. We expect that good solutions tend to have fewer intersections and less extremely long edges. In comparison, the impact of the other metrics seems less well-established. Metric S3 is exploratory, and we hypothesize that in good solutions, edges connecting the depot and customers should be rather short. Intuitively, routes should be clearly separated (S4) and not overlap too much, both in width (S5) and in depth (S9). Thus, we expect good

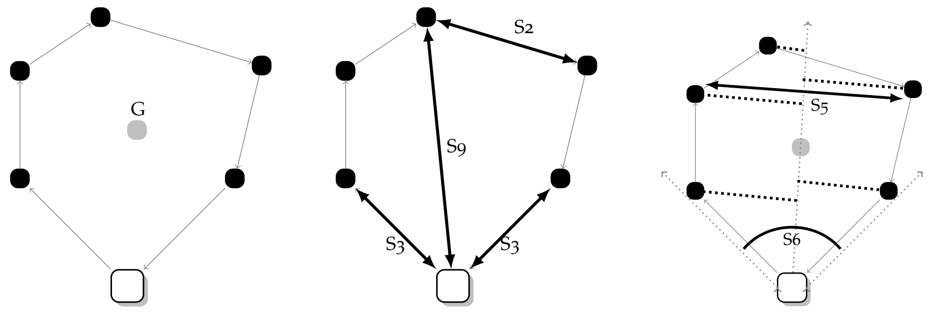


Figure 2.2: Visualization of solution metrics. A route with a center of gravity *G* (left) is measured with the longest edge (S2), the edges connected to the depot (S3) and its depth (S9). Furthermore, we determine its width (S5) and compactness (S7) by measuring the distance of each node towards the central line, and derive the span in radian (S6).

solutions to have less wide and deeper routes on average, resulting in a relatively high distance between them. The width of a route can also be interpreted by looking at the customers' radians toward the depot (S6). Narrow routes also tend to have a smaller difference in the radians of its customers, which is used in the popular sweep heuristic. Routes can be shaped more like a line (very compact) or more circular (less compact), which we measure with S7. Note that the metric compactness is quite similar to the metric width, as compact routes tend to be less wide. Then, compactness can also be measured by the variation in radians of a route's customers (S8). Finally, with S10 we measure how balanced the routes are in terms of the number of delivered customers.

The number of routes is usually also part of the solution in the standard VRP. However, for simplicity we assume that the number of routes is defined beforehand, and the number of routes in the near-optimal and respective non-optimal solution is always the same. Practically, whenever we compute for a particular instance a pair of near-optimal and non-optimal solutions in which the number of routes differ, we simply ignore this datapoint (this only happens relatively rarely). The reason for this assumption is that we investigate the impact of the structure of a solution on its quality, and a different number of routes might change the solution structure significantly.

### 2.2.4 *Metrics for an instance*

The values of the above solution metrics are dependent on the respective instance. Consider Figure 2.3 for an example. For both instances, the average width per route is lower in the optimal solution, from which one could conclude that low width is a predictor for good solutions. However, the values are not comparable between the two instances, since the instances vary in the number of routes and the number and

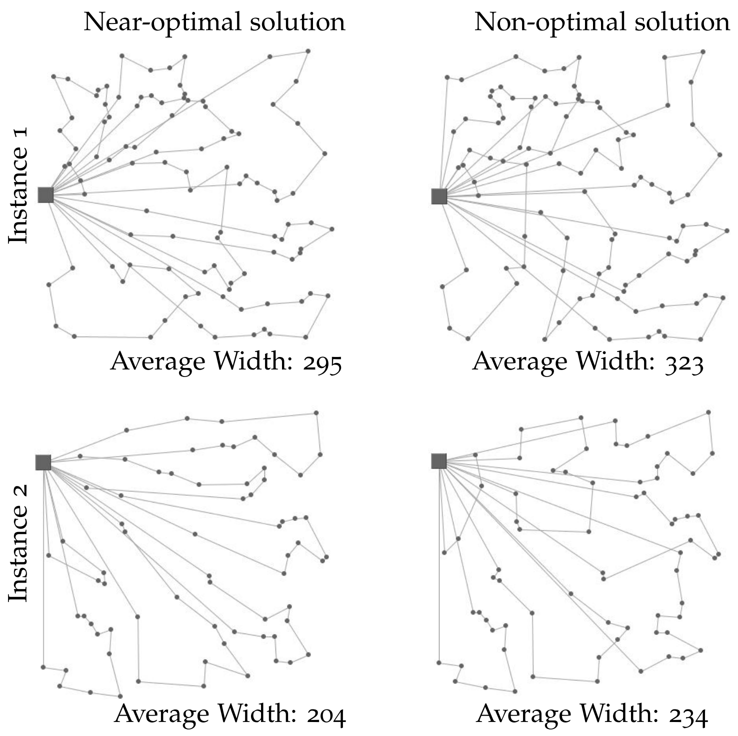


Figure 2.3: Instance metrics are necessary to normalize solution metrics between instances. In this example the width is smaller in the near-optimal solution in both instances, however, the absolute numbers are much higher in instance 1 even though both instances are from the same class.

location of the customers. The classification learner does not know whether two sets of solution metrics (for the near-optimal and the non-optimal solution) stem from the same instance and are paired, and only comparing the absolute values will lead to wrong or no predictions. Thus, we need to normalize the solution metrics with respect to the respective instance. This requires the definition of relevant metrics that capture the characteristics of an instance. We utilized the following metrics:

I1 - Number of customers

I2 - Number of routes

I3 - Degree of capacity utilisation

I4 - Average distance between each pair of customers

I5 - Standard deviation of the pairwise distance between customers

I6 - Average distance from customers to the depot

I7 - Standard deviation of the distance from customers to the depot

I8 - Standard deviation of the radians of customers towards the depot

Metrics I1 and I2 are basic instance parameters. The degree of capacity utilisation (I3) is given by the sum of demand divided by the available capacity in all vehicles. This value is the same for all computed solutions per instance, since we fix the number of vehicles. The last five metrics indicate the relative position of customers to each other and to the depot. For instance, I4 is lower if the customers are more clustered, and higher if they are uniformly spreaded. Likewise, I7 is lower if the depot is in a more central position. For each customer we also determine the radian from the perspective of the depot. Then, I8 has a low value, if the depot is located at the fringe and the customers are more clustered.

We achieve normalization with respect to these instance metrics within the data mining framework. The instance metrics form, along with the solution metrics, the input of the classification learner. Then, the interdependencies between solution and instance metrics are learnt. This approach might require larger datasets since it comes at the cost of more features and more interdependencies thus need to be learnt.

## 2.3 From data to problem-specific knowledge

With the above process we want to obtain problem-specific knowledge in the following way. We generate instances, and for each instance we compute a near-optimal and a non-optimal solution. From both solutions we then extract the defined features, and use a classification learner to find distinguishing patterns between both solution types.

In total, we generate four datasets for each of the eight instance classes. In each of these four datasets we vary the generation of the non-optimal solution with respect to the gap to the near-optimal solution (either 2% or 4%) and the heuristic used (either H1 or H2). For the smaller instance classes (20-50 customers) we generate about 5,000 instances with about 10,000 solutions per dataset. Larger instances take longer to solve and, thus, for the instance classes with 70-100 customers we limit the data to about 1,000 instances and 2,000 solutions. Altogether, we computed about 192,000 VRP solutions in the course of this study. For each computed solution we derive the corresponding solution metrics S1-S10 and instance metrics I1-I8, which, together with a flag for whether the solution is near-optimal or non-optimal, constitute one data-point of the dataset. The non-normalized raw-data generated with heuristic H1 is publicly available at `http://antor.uantwerpen.be/problem-knowledge`.

Table 2.2: Prediction accuracies with linear SVM for each dataset.

| | | | 2% gap | | 4% gap | |
|---|---|---|---|---|---|---|
| | | #data points | H1 | H2 | H1 | H2 |
| 20-50 cust. | Class 1 | 10.000 | 65% | 62% | 76% | 64% |
| | Class 2 | 10.000 | 67% | 61% | 77% | 63% |
| | Class 3 | 10.000 | 67% | 68% | 76% | 75% |
| | Class 4 | 10.000 | 66% | 65% | 74% | 71% |
| 70-100 cust. | Class 5 | 2.000 | 81% | 81% | 89% | 89% |
| | Class 6 | 2.000 | 80% | 80% | 89% | 89% |
| | Class 7 | 2.000 | 85% | 85% | 90% | 91% |
| | Class 8 | 2.000 | 81% | 82% | 88% | 89% |

We analyse each dataset with the classification learner application of MATLAB, using 5-fold cross validation. In short, the learner uses a certain percentage of randomly-selected data points (training set) to train an internal predictive model which tries to distinguish between near-optimal and non-optimal solution. This predictive model is then validated with the help of the remaining data points (validation set). The quality of the predictive model can be quantified by its *prediction accuracy*, which is defined by the percentage of correctly classified data points in the validation set. A prediction accuracy of higher than 50% reveals that there are predictive patterns within the data, since there are as many data points associated with near-optimal solutions as with non-optimal solutions (and one could obtain an accuracy of 50% by randomly guessing which data point corresponds to which solution quality).

### 2.3.1 *Prediction accuracy*

As classification learners we use simple decision trees, random forests and linear SVMs. Hereby, linear SVMs (with default settings: box constraint level = 1 and automatic kernel scale) performed best for all datasets. Prior to using linear SVMs we normalized all features to the same range (0-1). All learners can be trained in a matter of minutes (SVM) or even seconds (random forests and decision trees). We want to remark that larger accuracies can possibly be achieved with a prior feature selection and more sophisticated learners such as stacked models, however, in the context of this study we are interested in the general magnitude of prediction accuracy and kept the learning process simple. In Table 2.2 we report the corresponding prediction accuracies for each dataset. From these result we can make the following observations.

For all datasets we obtain prediction accuracies that are significantly higher than 50%. In other words, if we give the predictive models any solution for any instance within the defined parameter bounds, it can tell us whether the solution is near-optimal or non-optimal with a probability of up to 90%. This reveals that there is predictive power in at least some of the solution metrics we defined above. Thus, some values for the characteristics S1 to S10 are, alone or in interplay with others, more characteristic for a near-optimal solution than for a non-optimal solution. Hereby, the way in which we generate the non-optimal solution, either by H1 or H2, does not seem to affect the prediction accuracy much. We only observe differences for smaller instances with a centric depot location (Class 1 and Class 2), for which the learner seems to struggle with finding distinctive characteristics of non-optimal solutions generated with H2.

In general, the predictive power appears to depend on the gap between non-optimal and near-optimal solutions. A higher gap results in a higher prediction accuracy. This was to be expected, since, with a higher difference in solution value, the two solutions are also more likely to be more different in structure.

Finally, for instance classes 5-8 we obtain a higher prediction accuracy than for their counterparts 1-4. Since these problems exhibit a higher complexity with more customers connected on more routes, they are also richer in information. A resulting hypothesis is that, with increasing instance size, it is easier to extract and learn distinctions between solutions of different quality. Similarly, but much less strongly, the predictability seems to be slightly higher for problem instances where the depot is located at an edge (e.g., Class 3 versus Class 1). On the other hand, there are no systematic differences between instance classes that vary in the distribution of customer demand (e.g., Class 5 versus Class 6). It is possible that this observation is related to the fixation of the number of routes. If the variance in demand is high, it is typically more difficult to minimize the number of routes, and worse solutions tend to have more routes than better ones. Since we fix the number of routes beforehand, we do not capture this possible effect. Overall, the effect of the instance parameters depot location and distribution of customer demand appears less impactful than the number of customers.

### 2.3.2 *Explaining prediction*

These results raise the question 'What are predictable differences between solutions of different quality?'. To obtain some explanations for why a classification learner predicted the way it did, we can estimate the individual contribution of each metric to the prediction model.

For each dataset and solution metric we build a predictive model with a simple decision tree, using only the respective solution metric together with the eight instance metrics as features. We allow up to 9 splits in the tree, since we have 9 features in this model. The prediction accuracy of this model is an estimator for the predictive power of the respective solution metric.

Table 2.3: Solution metrics with an individual prediction accuracy of higher than 55% per instance class. The highest prediction accuracies per instance class and gap are highlighted in bold.

| | 2% gap | | | | | | | | | | 4% gap | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | S1 | S2 | S3 | S4 | S5 | S6 | S7 | S8 | S9 | S10 | S1 | S2 | S3 | S4 | S5 | S6 | S7 | S8 | S9 | S10 |
| Class 1 | | | | | | | | **58** | | | 57 | 56 | 56 | | 56 | 59 | 57 | **61** | | |
| Class 2 | | | | | | | | **57** | | | 57 | 57 | 56 | | | 56 | 56 | **62** | | |
| Class 3 | 58 | | | | **60** | | 60 | 57 | | | 61 | | 56 | | **65** | 59 | 64 | 60 | | |
| Class 4 | 57 | | | | **58** | | 58 | 56 | | | 59 | | 56 | | **62** | 57 | 62 | 61 | | |
| Class 5 | | | 62 | | 67 | **68** | 67 | 67 | | | 60 | | 71 | | 78 | 77 | **79** | 76 | 59 | |
| Class 6 | 57 | | 62 | | 65 | 66 | 68 | **70** | | | 60 | | 67 | | 74 | 73 | 74 | **75** | | |
| Class 7 | 66 | 57 | 60 | | **79** | 65 | 75 | 65 | | | 71 | | 66 | | **84** | 72 | 80 | 72 | | |
| Class 8 | 64 | | | | **72** | 61 | 70 | 66 | | | 68 | | 58 | | **79** | 67 | 77 | 72 | | |

We found that the results of the datasets in which we generated the non-optimal solutions with H1 are similar to the results of the corresponding datasets in which we used H2. In other words, H1 and H2 compute non-optimal solutions that are similar in structure. Thus, in the following we report the results as the average over both corresponding datasets.

Table 2.3 highlights for each instance class the most predictive individual solution metrics and their prediction accuracy if they are higher than 55%. As above, it seems that also the individual prediction accuracies are higher for larger gaps, and for larger instances. From these results we can observe the following findings.

Some of our solution metrics explain most of the prediction accuracy, while others have no or only little predictive power on their own. Surprisingly, longest edges that are not connected to a depot (S2) have only little predictive value. Likewise, the distance between routes (S4), their depths (S9) and the variance in the number of customers per route (S10) do not seem to differ too much between near-optimal and non-optimal solutions.

On the contrary, the length of edges that are directly connected to the depot (S3) are a more distinguishing characteristic. The shorter they are, the better the solution tends to be. Similarly, better solutions have fewer intersections (S1) and can be classified as such with up to 71% accuracy. However, the most predictive metrics in distinguishing near-optimal from non-optimal solutions are those associated with width, radiants and compactness of routes (S5-S8). In general, the width (S5) seems to be the best indicator whether a solution is near-optimal or not. By only considering width of routes we can correctly classify a non-optimal solution with up to 84% accuracy. Also the compactness and radiants, which measure similar structural elements, are highly predictive. In summary, the four most characteristic properties of near-optimal VRP solutions are narrow, compact routes with few intersections and

Table 2.4: Extracted rules from the classification tree for the dataset of class 6 with a 2% gap.

| Solution Metrics | | | | Instance metrics | | |
|---|---|---|---|---|---|---|
| S1 | S3 | S6 | S8 | I4 | I8 | |
| | $> 167$ | | $> 0.23$ | | | $\rightarrow$ Non-optimal |
| | $> 199$ | $> 0.97$ | $> 0.27$ | | $< 1.6$ | $\rightarrow$ Non-optimal |
| $> 0.26$ | $> 222$ | $> 0.73$ | $> 0.27$ | $> 152$ | | $\rightarrow$ Non-optimal |
| | $> 240$ | $> 0.74$ | $> 0.23$ | | | $\rightarrow$ Non-optimal |
| | $> 294$ | | $< 0.23$ | | | $\rightarrow$ Non-optimal |
| ... | ... | | | | ... | |

short outgoing edges from the depot. These findings hold for all instance classes, and can therefore be considered as problem-specific knowledge.

We also observe some differences between the classes. The radiants seems to be a slightly better metric for instances where the depot is in the center. On the contrary, if the depot is located at an edge, the distinction between near-optimal and non-optimal solutions is greater with respect to the width and the number of intersections. On the other hand, the demand distribution does not appear to have any significant effect on differences in route structure.

Another interesting observation is that the relative prediction strength of metrics does not change with different gaps. Even though the absolute prediction accuracy is lower for 2% gaps compared to 4% gaps, the ranking of most predictive metrics is almost the same. A possible explanation is that these metrics are predictable in most cases, independent of the gap. Another explanation could be that the considered gaps are too close to each other to observe significant differences.

### 2.3.3 *From explanations to rules*

So far, we only considered solution metrics in isolation to explain results. In the following, we also want to consider the interaction between metrics and arrive at rules that tell us if and why a certain solution is classified as non-optimal. For each dataset we again build a simple decision tree, this time using all features. A classification tree branches according to the most predictive metrics, and its leaves contain the class label 'near-optimal' or 'non-optimal' . The branching towards each leaf can be transformed into a rule, represented by a set of value ranges for certain features. If a solution fulfills these conditions, it is classified according to the defined class label. Table 2.4 displays extracted rules from the dataset of instance class 6 with a 2% gap.

These rules constitute more detailed knowledge about the differences between near-optimal and non-optimal solutions, and might be of great value in practice. Imagine we had a non-optimal solution for a certain instance, and we would like to know, how this solution could be further improved. We simply compute the respective solution metrics and instance metrics, and find the rule, or rules, that classifies the solution as non-optimal (if the solution is already optimal or close to optimal, there might not be such a rule). The conditions of this rule then indicate likely reasons for why this solution is non-optimal.

As an example, consider the first row in Table 2.4. If this rule applies, our solution is likely to be non-optimal because the outgoing edges from the depot are too long and the routes are not compact enough. With this information we can guide the search process into a promising direction. In the case of a local search, we could select operators that tend to make routes more compact and, thereby, focus especially on the edges connected to the depot. In the case that the routes are sufficiently compact, but the edges to the depot are too long (fifth rule in Table 2.4), we can take even more targeted action and focus specifically on those edges. Reversely, it might be difficult to infer any meaningful actions if the rules are too complex. For instance, the third rule applies, if our solution has too many intersecting edges while the routes are too wide and non-compact, and edges to the depot are too long. There is probably no operator that can deal with all of these issues at once.

These rules could also be used to develop new operators. With regard to the fifth rule, we could try to build a local search operator which exchanges customers that are connected to the depot while keeping the remaining parts of the routes fixed.

We do not argue that this process works for any instance or at any stage of the search, nor that rules like 'make routes more compact' can be readily translated into actions. However, in general it has proven to be the most distinguishing property over a large sets of solutions, given the respective instance and solution. At this point, it also becomes clear how far we can go when we speak of knowledge: It is not a provable if-then cooking recipe, but rather *a set of classification rules that hold with an above-random-chance probability for similar types of instances*. Due to the complexity of the VRP, this might present the most granular form of knowledge that can be found.

## 2.4 Application of problem-specific knowledge in the design of heuristics

In the sections above we have derived insights into the nature of good VRP solutions. In this section we describe, using a case study, how such knowledge can help to design better heuristics in practice, i.e., how problem-specific knowledge can be used as guidance in the search process. A solution for a VRP instance can be represented as a set of edges traversed by the vehicles. Knowledge on which edges appear more frequently in near-optimal solutions as opposed to non-optimal solutions can be used in several heuristics to guide the search.

In this section, we study a guided local search (GLS) (Voudouris and Tsang, 2003) heuristic, and adapt it so edges are penalized and removed based on the information generated above. GLS has already been successfully applied to the TSP (Voudouris and Tsang, 2003) and the VRP (Mester and Bräysy, 2007) and stands out due to its simplicity and effectiveness. Its idea is to penalize a 'bad' edge by changing the cost $c(i,j)$ of that edge between customers $i$ and $j$ to $c^g$:

$$c^g(i,j) = c(i,j) + \lambda p_{(i,j)} c(i,j), \tag{2.1}$$

where $p_{(i,j)}$ counts the number of penalties of edge $(i,j)$ during the search, and $\lambda$ (usually chosen to be in $[0.1, 0.3]$ as in Kilby et al. (1999)) controls the impact of penalties. After the penalization of an edge, we try to remove this edge by means of a local search. Since the local search only focuses on a certain area of a solution, i.e., the penalized edge, it can be executed very efficiently, and thus, the two steps of edge penalization and local search can be iterated thousands of times.

Naturally, the effectiveness of this heuristic approach depends on the ability to detect (and remove) 'bad' edges. One of the most efficient heuristics to date penalizes the longest edges (Mester and Bräysy, 2007), a straight-forward idea since longer edges contribute more weight to the objective function. Indeed, our study confirmed that longer edges, especially those connected to the depot (S3), have a certain predictive power to classify good solutions. Even though the length of the longest edge per route did not appear to have much predictive power, it seems generally advisable to remove long edges to minimize the objective function. Even more important, it appeared that the most predictive metrics were the width and compactness of routes (S5-S8), i.e., solutions of higher quality tend to have narrower and more compact routes. Wider and less compact routes tend to have wider edges, and thus, it seems straight-forward to penalize and thereby avoid *wide* edges. Hereby, the width of an edge $(i,j)$ in route $r$ is defined, on the basis of Section 2.3, as the difference $w = \left(d(L_{G_r}, i) - d(L_{G_r}, j)\right)^+$ of the distances to the central line between the two adjacent nodes.

We aggregated these findings in a function $b^k(i,j)$ that measures the 'badness' of an edge $(i,j)$:

$$b^k(i,j) = \frac{c(i,j) + w(i,j)}{1 + p_{(i,j)}}, \tag{2.2}$$

where $c(i,j)$ indicates the original cost value or length of edge $(i,j)$. We want to remark that we also tried to incorporate the penalization of intersecting edges (since

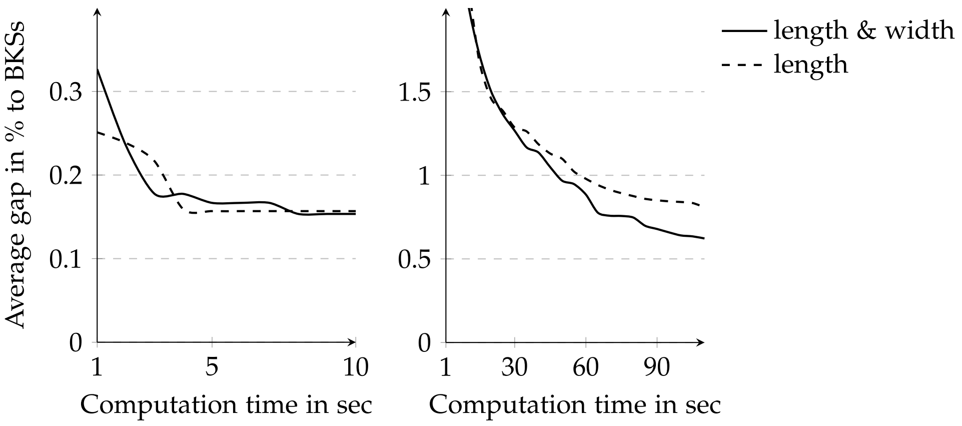


Figure 2.4: Average Improvement of the GLS over time on the smaller A-instances by Augerat (left) and larger instances by Golden (right). The dotted line presents the performance of the heuristic with traditional edge penalization, the continuous line also considers the width of edges in the penalization.

S1 was among the most predictive metrics), however, we did not find any significant improvements. Thus, for simplicity we decided not to consider this metric in the penalization function.

In this way, the edge in the current solution with the highest function value is penalized according to (2.2), and the succeeding local search tries to remove it. The denominator makes sure that we do not penalize the same edge over and over, especially when it reappears in a solution, but rather spread penalization among the entire search space.

We demonstrate that this simple extension of an already powerful heuristic framework with problem-specific knowledge enhances the framework's efficiency even further. Our methodological approach is the following. We use the A-instances of Augerat et al. (1995) and instances 9-20 without route-length constraints of Golden et al. (1998) to compare the effectiveness of a GLS with traditional edge penalization $b^t$ (which corresponds to the heuristic H1 that we described in Section 3.2) and a GLS with knowledge-driven penalization $b^k$. The instances by Augerat are relatively small with 32 to 80 customers, while the instances by Golden include between 200 and 488 customers. As stated above, the traditional GLS penalizes the longest edges, which have the highest value in the function

$$b^t(i,j) = \frac{c(i,j)}{1 + p_{(i,j)}}. \tag{2.3}$$

The graphs in Figure 2.4 visualize for both GLS versions the average improvement with increasing computation time over all instances for the two benchmark sets. The improvement is given with respect to the average gap in % to the best known solutions (BKS). While the differences in the set of smaller instances are almost negligible, in the set of larger instances the knowledge-based version clearly outperforms the traditional version. Note that the solutions for the smaller instances are almost immediately very close to the optimal ones (less than 0.15%), whereas there is more potential to improve the solutions of the larger instances. Since the only difference between both heuristics is the criteria to penalize edges, we can conclude that the criteria 'penalize wide edges' enhances the effectiveness of the search, especially for larger instances. This is remarkable given that the instances by Golden (up to 488 customers) are larger than the ones on which we conducted the proceeding study (up to 100 customers). Thus, the results of the study appear to also hold for larger instances.

These findings suggests that problem-specific knowledge becomes more valuable when problem instances grow in complexity. With a higher complexity, the number of possible options at each step of the search grows as well, and thus, even a limited amount of knowledge can help to take better decisions.

## 2.5 Conclusions

In this chapter, we have introduced a framework to derive problem-specific knowledge for combinatorial optimization problems and applied it to the vehicle routing problem. The core of the framework is data mining, a technique to find and extract pattern and generalize it to new data. Hereby, the most challenging step is the definition of relevant metrics to measure the structure of a solution and that of an instance. This is a exploratory task which requires a deeper understanding about the problem as well as some creativity.

Furthermore, the framework requires a random instance generator as well as solution techniques to generate solutions of varying quality. Since we need to generate a huge amount of data points for the classification learner to work, the solution technique should be efficient and preferably be able to produce almost optimal solutions.

We demonstrated that the approach is able to find simple characteristics that can distinguish with great accuracy between good solutions and not-so-good solutions, and thereby explain what makes a solution good. In the example of the VRP, we found that the compactness, the width and the span in radians of routes, as well as the number of intersecting edges and the distances of connecting edges to the depot are distinguishing features between near-optimal and non-optimal solutions. Interestingly, those findings hold for a variety of instance classes which suggests that structural differences in near-optimal and non-optimal solutions are similar across a wide range of instances.

Finally, we showed that these characteristics can be used to make a good heuristic even better by guiding the search process. In how far problem-specific knowledge can be incorporated in a particular heuristic, might depend on its design. In our case, we simply adapted the evaluation function for edges and obtained better results. In general, it seems reasonable to build a heuristic around problem-specific knowledge, rather than the other way around. If we know the structural characteristics of good solutions, it appears sensible to design operators and functions to lead us there.

In this work we only scratched at the surface of what might be possible. With the help of the deduced set of rules, the guidance could be designed in a much finer case-by-case fashion. In each step of the heuristic we could identify relevant rules and adapt parameters and operators accordingly, e.g., the prioritization of the elimination of certain intersections over the reduction of route width. This process could be the first step in the automation of heuristic design. Given a particular problem together with a database of metaheuristic designs, operators and problem-specific knowledge, an efficient heuristic could be developed almost autonomously, complemented with the creativity and experience of developers.

# 3 Knowledge-guided local search for the Vehicle Routing Problem

## 3.1 Introduction

In the last decade many successful heuristics for the VRP have been developed, that can solve instances of several hundred customers to near-optimality in a few minutes of computing time (Vidal et al., 2013a). Most heuristics are based upon metaheuristic designs, and a comprehensive survey about the plethora of heuristic designs is provided by Salhi (2014). Many authors have shown that high-quality results can be achieved with different designs, most notably with genetic algorithms (Vidal et al., 2013a), a mix of exact solvers and heuristics (Subramanian et al., 2013) and memetic algorithms (Nagata and Bräysy, 2009). However, one component that probably all good heuristics for the VRP have in common is local search.

Local search has proven to be the cornerstone of many solution techniques for various combinatorial optimisation problems. Complete metaheuristic designs revolve around an effective local search, e.g., variable neighborhood search (Mladenović and Hansen, 1997) or guided local search (Voudouris and Tsang, 2003), and it has been particularly successful to solve the Traveling Salesman Problem (TSP) (Lin and Kernighan, 1973). In the context of the VRP, many effective local search operators have been developed over the years, and have been condensed in an online library (Groër et al., 2010).

Despite its known success, it appears that most state-of-the-art VRP heuristics use local search as an ingredient to boost a heuristic framework, rather than the other way around. As a consequence, less research has been devoted to a deeper analysis of how succesful local search operators can be setup and implemented effectively on its own. That such an analysis and refinement of local search can result in impressive solution methods has already been shown for the TSP by Helsgaun (2000).

In this chapter we aim to demonstrate that a well-implemented local search suffices to create a heuristic that can compete with the best heuristics in the literature. To this end we combine three powerful local search techniques, and implement them in an efficient way that minimizes computational effort. Furthermore, we make use of problem-specific knowledge, to guide the search to promising solutions more effectively. The heuristic created in this way not only matches the performance of the best heuristics in the literature, it is also straightforward in its design and does not contain any components of which the contribution is unclear. The heuristic can also be applied to variants of the VRP, which we show using the example of the VRP with multiple depots (MDVRP).

We introduce local search components in Section 2, and show, how they can be implemented efficiently. In Section 3 we demonstrate how local search can be guided by the penalization of edges. Through various experiments we analyse different components of the resulting local search framework in Section 4. In Section 5 we conduct detailed testing on the CVRP and MDVRP, and we conclude with a brief summary of our findings in Section 6.

## 3.2 Local search

Local search is one of few general approaches to combinatorial optimization problems with empirical success (Johnson et al., 1988). The basic idea underlying local search is that high-quality solutions of an optimization problem can be found by iteratively improving a solution using small (local) modifications, called *moves*. A *local search operator* specifies a move *type* and generates a *neighborhood* of the current solution. Given solution $s$, the neighborhood of a local search operator is the set of solutions $\mathcal{N}(s)$ that can be reached from $s$ by applying a single move of that type.

After generating the neighborhood of the current solution, the neigborhood is evaluated, and local search uses a move *strategy* to select at most one solution from the neighborhood $\mathcal{N}(s)$ to become the next current solution. The most commonly used move strategy is called steepest descent, that selects the solution from the neighborhood with the best objective function value, provided that this solution is better than the current solution. If no improving solution is found in the neighborhood of the current solution, a *local optimum* has been reached.

In general, local search operators for the VRP can be distinguished between operators for *intra-route optimization* and operators for *inter-route optimization*. These two operator types reflect the two tasks that one has to solve in a VRP: (1) The allocation of customers to routes (inter-route optimization), and (2) the optimization of each route in itself (intra-route optimization). These two tasks do not necessarily have to be solved in this order, nor sequentially. For instance, an *Ejection Chain* (Glover, 1996) solves both tasks simultaneously. However, intra-route optimization can be executed rather efficiently, since it corresponds to solving a TSP with relatively few customers. Therefore, it seems sensible to optimize the routes in themselves, before they are optimised jointly.

The most commonly used operator for intra-route optimization is *2-opt*. The idea of 2-opt is to remove two edges from the considered route and replace them by two new edges, such that a new route is formed. The same idea can be extended to three edges (3-opt) and four edges (4-opt). However, the size of the considered neighborhood increases quickly in the number of considered edges. Lin and Kernighan (1973) (LK) have generalized this idea for the TSP in a computationally feasible way. LK has proven to be highly effective to solve TSPs, and therefore constitutes an ideal operator for intra-route optimization.

On the other hand, operators for inter-route optimization try to change the allocation of customers to routes. Let a route $r_i$ be defined by the string (an ordered set) of customers that are visited on this route $r_i = \{I_1, I_2, \ldots, I_{|r_i|}\}$. For ease of notation, let the neighbours of a node $I_k$ be denoted as $\underline{I_k} = I_{k-1}$ and $\overline{I_k} = I_{k+1}$. The neighborhood of the operator *relocate* contains all solutions, in which a single customer is removed from its route and inserted into another route. More formally, it looks for a substring $\hat{r}_i$ of a route $r_i$, which is inserted into another route $r_j$, where the substring only contains a single element $|\hat{r}_i| = 1$. The operator *Or-exchange* generalizes this idea by removing and re-inserting longer substrings $|\hat{r}_i| \geq 1$. A *swap* tries to exchange two customers from two different routes. A substring $\hat{r}_i$ is exchanged with a substring $\hat{r}_j$ with $|\hat{r}_i| = |\hat{r}_j| = 1$. This idea can again be generalized by exchanging larger substrings $|\hat{r}_i|, |\hat{r}_j| \geq 1$. In a *Crossover* both exchanged substrings are connected to the depot, and thus contain $I_1$ or $I_{|r_i|}$. Hereby, the substrings have possibly to be inverted. Finally, *CROSS-exchange* (CE) exchanges any two substrings of any size. Again, the substrings can be inverted (which is called I-CROSS, we denote both variants with CE in the following). Note that $|\hat{r}_i| = |r_i|$ and $|\hat{r}_j| = |r_j|$ would result in an exchange of two entire routes and the solution remains unchanged. These operators are the most commonly used and they are illustrated in Figure 3.1. Further operators include the ejection chain and the GENI insertion operator (Gendreau et al., 1992). Given this large pool of local search operators, there are some established principles as to which operators to choose in practice.

*Complementarity*
An important observation is that a local optimum for one local search operator is generally not a local optimum for another one. For this reason, it is sensible to use several local search operators in a single algorithm. This is the underlying principle of the *variable neighborhood search* metaheuristic framework (Mladenović and Hansen, 1997), but it is extremely common in the area of vehicle routing (Sörensen et al., 2008). For example, the active-guided evolution strategies of Mester and Bräysy (2007) use relocate, swap, 2-opt and Or-exchange, the hybrid genetic algorithm of Vidal et al. (2013b) uses relocate, swap and 2-opt, and the iterated local search of

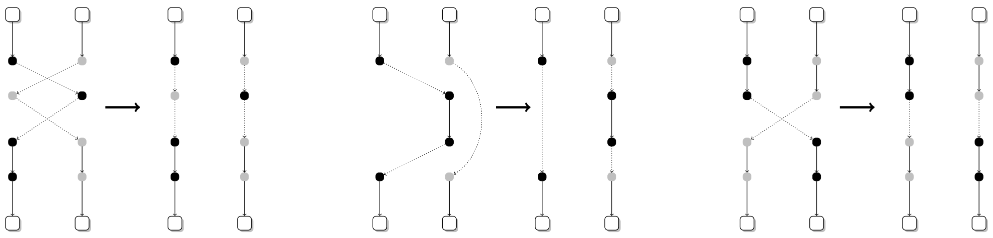


Figure 3.1: Illustrations of different local search moves. The routes are visualized vertically, the depot nodes are pairwise the same. From left to right: a swap (4 edges are exchanged), an Or-exchange (3 edges are exchanged), a Crossover (2 edges are exchanged).

Subramanian et al. (2013) uses relocate, 2-opt, Or-exchange and CROSS-exchange. However, the usage of several local search operators is only beneficial as long as they explore different neighborhoods. In the best case, the pairwise intersections of the considered neighborhoods are empty, i.e., given two operators $o_1$ and $o_2$ we have $\mathcal{N}^{o_1}(s) \cap \mathcal{N}^{o_2}(s) = \emptyset$ for every solution $s$.

*Complexity versus neighborhood size*
In general, the computational complexity of a local search operator depends on the size of its neighborhood. The larger the neighborhood, the more solutions need to be generated and evaluated. On the other hand, larger neighborhoods also come with a larger probability of finding improvements. Consequently, there is a trade-off between computational complexity and the probability of improvement. This trade-off presents one of the greatest challenges in the design of an efficient local search. A balance needs to be reached between the size of the neighborhood and the computational effort required to generate it. Operators with relatively small neighborhoods such as a swap can be easily implemented and quickly executed. However, they might not explore a sufficiently large neighborhood to find improvements. Reversely, operators with large neighborhoods such as CROSS-exchange exhibit a high complexity ($O(n^4)$). For larger instances already a quadratic complexity might be computationally too expensive. A commonly used tool to reduce the complexity of local search operators is *heuristic pruning*. Rather than generating the entire neighborhood for each operator, pruning tries to limit the considered neighborhood to promising options. As an example, CE usually only considers substrings up to a certain length, or a relocate-move can be restricted in such a way that only insertions next to relatively close nodes are considered. A metaheuristic that drastically restricts neighborhoods is, e.g., *granular tabu search* (Toth and Vigo, 2003). Another very effective approach to prune the neighborhood of more complex operators is sequential search (Irnich et al., 2006), which constitutes a generalization of the partial gains criterion in LK. The idea of sequential search is to split complex operators into exchanges of edges, and only consider subsequent exchanges as long

as overall improvements are obtained. We provide a more elaborate description of sequential search below.

In summary, we argue that a successful local search for the vehicle routing problem contains a small yet diverse set of well-chosen local search operators for intra- and inter-route optimization that have sufficiently large neighborhoods, but are able to find improvements without exploring the neighborhood exhaustively. In the following, we outline how such an efficient implementation of local search operators can be realized, using the example of LK and CE. Both operators generate and evaluate rather large neigborhoods (LK incorporates 2-opt and 3-opt moves, CE includes insert, swap, and crossover). These operators can be complemented by an operator that can improve more than two routes simultaneously, and we introduce such an operator using the idea of embedded neighborhoods.

### 3.2.1 *Lin-Kernighan heuristic*

Intra-route optimisation corresponds to solving a TSP in which the nodes are composed of the depot and all customers on the respective route. One of the most effective heuristics for the TSP is the heuristic by Lin and Kernighan (1973) (LK), which has been further refined by Helsgaun (2000) and Applegate et al. (2003) to solve instances with more than 100.000 customers. LK looks for $\lambda$-opt moves by exchanging $\lambda$ existing edges with $\lambda$ new edges. Since the complexity of finding $\lambda$-opt moves increases rapidly for larger $\lambda$, usually 2-opt and 3-opt are used in heuristics. LK introduces some clever ideas to also generate and evaluate more complex moves in a feasible runtime.

Starting with the longest edge in the considered route, edges are iteratively removed and added, such that the pair of removed and added edge shares an endpoint, fulfills the *partial gain criterion* and results in a feasible tour if the tour would be closed. With the partial gain criterion only those pairs of edges are considered for removal and insertion, such that the overall sequence of exchanges results in an overall improvement. This criterion reduces the neighborhood drastically by restricting the search to promising options for larger $\lambda$. If, for instance, there is no sequential removal and insertion of two pairs of edges with a positive gain, LK does not continue to remove and add more pairs. An example is provided in Figure 3.2. As soon as an improving move is found with the closure of the tour, the move is executed and LK is restarted. The heuristic stops, if for each initially removed edge no improving move can be found. For a more elaborate description of LK we refer the interested reader to Helsgaun (2000).

As in previous LK implementations, we prune the neighborhood by only considering new edges between a node and its ten nearest nodes within the considered route. Routes in VRPs tend to be rather small TSPs with few customers so that we slightly adapt the standard version. Instead of executing the first improving move, we

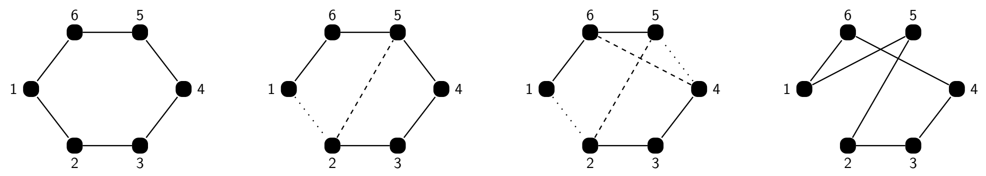


Figure 3.2: Illustration of the sequential removal and adding of edges in LK. Starting with the removal of one edge (1,2), an edge (2,5) starting from one of the endpoints is added, such that $c(1,2) > c(2,5)$. Because of the positive gain, the process continues by removing an edge incident to node 5, and adding an edge, e.g., (4,5) and (4,6). A continuation is considered, if $c(1,2) + c(4,5) > c(2,5) + c(4,6)$. The tour could be closed by removing edge (5,6) and adding edge (1,5) to obtain a 3-opt move.

generate and evaluate all feasible $\lambda$-opt moves (starting with the removal of one particular edge) and execute the best improving move. The upper limit for $\lambda$ is set to four.

### 3.2.2 *CROSS-exchange*

The CROSS-exchange operator (CE) is a generic local search operator that tries to exchange two substrings $\hat{r}_i$ and $\hat{r}_j$ of two different routes $r_i$ and $r_j$ (Taillard et al., 1997). An example of such a move is visualized in Figure 3.3. The generations of all substring of a route has the theoretical complexity $O(n^2)$, and thus matching two substrings amounts to $O(n^4)$. To tame this complexity, the length of considered substrings is usually limited. In the following we adopt the idea of partial sums from LK and sequential search to efficiently prune the neighborhood of CE.

The evaluation of a CE move can be deconstructed into two parts: (1) The identification of the starts of two substrings and (2) The determination of a suitable length of both substrings. To find the start of a substring that is potentially removed from route $r_i$, we first need to determine an edge that is removed. Let this edge be $(I_k, \overline{I_k})$. For $I_k$ we need to find a new neighbour in another route. Using the idea of heuristic pruning, we only consider potential neighbours among the $C$ closest nodes. Let $J_l$ be such an option, a node in a different route that is among the $C$ closest of $I_k$. If we add $(I_k, J_l)$ as a new edge, we need to remove one of the edges $(J_l, \underline{J_l})$ or $(J_l, \overline{J_l})$, since $J_l$ can only have two adjacent nodes. Finally, we reconnect the adjacent node of the removed edge $\overline{J_l}$ or $\underline{J_l}$ to $\overline{I_k}$. A visualization is given in Figure 3.3. In total, we evaluate at most $4C$ starts for a CE move which starts with the removal of one edge $(I_k, \overline{I_k})$ ($2C$ for all nearest nodes of $I_k$ and $2C$ for all nearest nodes of $\overline{I_k}$).

For each start generated this way, we only continue with (2) the determination of substrings, if the start does not increase solution costs, e.g., $c_1 = c(I_k, J_l) + c(\overline{I_k}, \underline{J_l}) - c(I_k, \overline{I_k}) - c(J_l, \underline{J_l}) \leq 0$. The start generates a first 'cross' between two

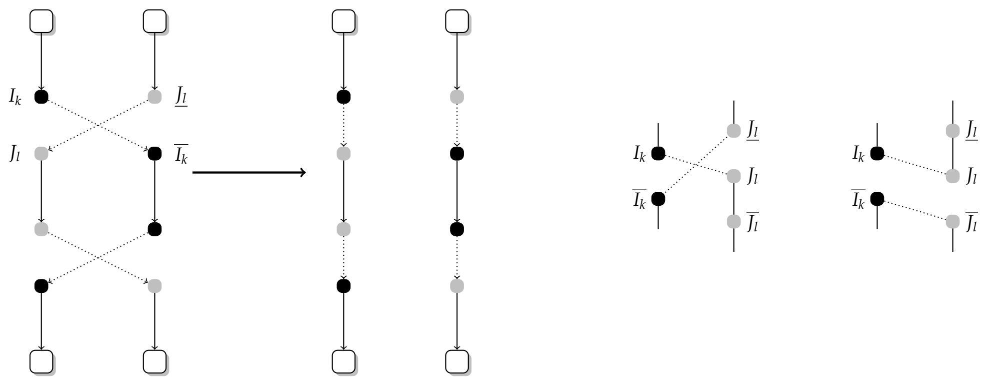


Figure 3.3: (Left) Illustration of a CE move with substrings of two customers. (Right) Two possibilities to generate starts for a CE. Starting with the removal of $(I_k, \overline{I_k})$, we add an edge $(I_k, J_l)$. We can either complete the start by removing $(J_l, \underline{J_l})$ and adding $(\overline{I_k}, \underline{J_l})$, or by removing $(J_l, \overline{J_l})$ and adding $(\overline{I_k}, \overline{J_l})$. Starts are considered, if the sum of the costs of added edges is not higher than the sum of costs of removed edges. In both cases, the substrings can start from $\overline{I_k}$ and $J_l$ (see left), or from $I_k$ and $\underline{J_l}$.

routes, and we can already store it as a candidate move if route constraints are not violated (a crossover). For a CE move we need to determine a second 'cross' between the two routes. Starting form the first 'cross', substrings can be constructed in two ways. The start of the substring to be inserted into $r_j$ can either be $I_k$ or $\overline{I_k}$. Let $\hat{r}_i = \{\overline{I_k}\}$ be the substring to be inserted into $r_j$ and $\hat{r}_j = \{J_l\}$ be the corresponding substring to be inserted into $r_i$ (the description for $\hat{r}_i = \{I_k\}$ and $\hat{r}_j = \{\underline{J_l}\}$ follows analogously). Then the costs of the second 'cross' are $c_2 = c(\overline{I_k}, \overline{J_l}) + c(J_l, I_{k+2}) - c(\overline{I_k}, I_{k+2}) - c(J_l, \overline{J_l})$. If $c_1 + c_2 \leq 0$ we have found a candidate move (a swap). We repeat this evaluation for different lengths of the substrings. Keeping $\hat{r}_i = \{\overline{I_k}\}$ fixed, we extend $\hat{r}_j = \hat{r}_j \cup \overline{J_l}$, recompute $c_2$ with $l = l + 1$, and add it as candidate move if $c_1 + c_2 \leq 0$. We continue this extension of $\hat{r}_j$ until we have reached the depot node, or until we violate the constraints of $r_i$. We then iteratively extend $\hat{r}_i$, starting with $\hat{r}_i \cup I_{k+2}$, and repeat the above process. In this way, we determine all combinations of substrings starting from $\overline{I_k}$ and $J_l$ whose exchange result in a non-increasing solution value while meeting route constraints.

The special cases $\hat{r}_i = \emptyset$ or $\hat{r}_j = \emptyset$ (an Or-exchange) can be generated and evaluated analogously with different functions $c_1^*$ and $c_2^*$. Using the example above with $\hat{r}_i = \emptyset$, we have $c_1^* = c(I_k, J_l) - c(I_k, \overline{I_k})$. If $c_1^* \leq 0$, we again evaluate all extensions of $\hat{r}_j$ until the depot node is reached or constraints in $r_i$ are violated. For $\hat{r}_j = \{J_l\}$ we have $c_2^* = c(\overline{I_k}, J_l) + c(\underline{J_l}, \overline{J_l}) - c(J_l, \underline{J_l}) - c(J_l, \overline{J_l})$.

In summary, we suggest the following approach to iteratively generate and evaluate the neighborhood of the CE operator, starting with the removal of edge $(I_k, \overline{I_k})$:

1) Determine the nearest nodes of $I_k$ and $\overline{I_k}$ that are in a different route $r_j$. This information can be preprocessed.

2) Determine all starts with $c_1 \leq 0$ and $c_1^* \leq 0$.

3) For each remaining start, iteratively evaluate and extend the substrings. If the exchange results in feasible routes and $c_1 + c_2 \leq 0$, store the move as a candidate move (CROSS-exchange), analogously for $c_1^* + c_2^* \leq 0$ (Or-exchange).

4) Execute the candidate move with the steepest descent.

This approach generates a potentially very large neighborhood, especially since we do not impose a restriction on the length of substrings. However, the evaluation of starts runs in linear time, and is likely to remove many options, especially in good VRP solutions.

### 3.2.3 *Complementary compound moves - relocation chain*

CE generates large neighborhoods to improve a pair of routes simultaneously, while LK optimises the individual routes. These two operators can be complemented by an operator that affects more than two routes simultaneously. Such an operator can be constructed by using the ideas of an *embedded neighborhood*. In an embedded neighborhood, several simple moves are combined to form one compound move (Ergun et al., 2006). An example for a compound move is the ejection chain introduced in Glover (1996) and formalized in Rego (2001) for the VRP.

An ejection chain can induce changes in several routes simultaneously, however, it also generates intra-route moves. With LK we have a dedicated operator for intra-route optimisation, and thus a complementary operator should focus exclusively on changes between routes. We build such an operator by combining several relocation moves. This idea is similar to the concept of an ejection chain, however, we only allow inter-route relocations. A relocation is simple to implement and to evaluate, and thus, it constitutes a suitable building block for a more complex compound move. In the following we will call this compound move *relocation chain* (RC).

A RC starts with a relocation of a customer node from route $r_i$ into route $r_j$. This relocation is followed by a relocation of a customer node from route $r_j$ into route $r_k$ (where $i = k$ is possible). This process can be repeated until an upper limit of relocations is reached. The motivation for such an operator is to change the solution while maintaining feasibility. Especially in tightly-constrained VRPs, many pairwise exchanges between routes will violate constraints, so that the neighborhood of feasible solutions can be relatively small. A relocation of a node from $r_i$ into $r_j$ might improve the solution, but exceed the capacity constraints of $r_j$. However, the

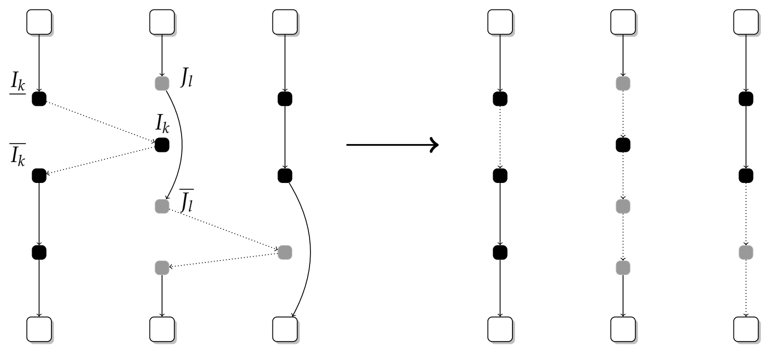


Figure 3.4: Illustration of a relocation chain with two relocations.

move might become feasible, if we make space, by simultaneously relocating a node from $r_j$ into another route. An example is visualized in Figure 3.4.

The complexity of this compound move is relatively high. Let us assume that we start a RC with node $I_k$. We can relocate this node into any position of any other route ($O(n)$). Let $r_j$ be an option for a destination route (there might be several options). Then we can continue the RC by relocating any node in $r_j$ into any other position of any other route ($O(n^2)$). Thus, with every continuing relocation, the complexity of the operator grows exponentially.

This complexity can be reduced with ideas from sequential search, pre-processing and heuristic pruning. Let a RC start with node $I_k$. Then the costs of a relocation of $I_k$ next to $J_l$ can be computed as the difference between the minimal costs of inserting $I_k$ either in front or behind of $J_l$, and the detour induced by the visit of $I_k$ in its current route. The insertion costs can be computed as $c_I = \min_{J_l^* \in \{\underline{J_l}, \overline{J_l}\}} \{c(I_k, J_l) + c(I_k, J_l^*) - c(J_l, J_l^*)\}$ and the costs of the detour as $c_D = c(I_k, \overline{I_k}) + c(I_k, \underline{I_k}) - c(\underline{I_k}, \overline{I_k})$.

This cost information can be preprocessed and updated when necessary. If the relocation does not increase solution costs and we have $c_I - c_D \leq 0$, we consider it as start of a RC. This corresponds to the idea of partial gains in LK and the basic idea of sequential search. If the relocation keeps $r_j$ feasible, we store it as candidate move. Otherwise, we try to extend the RC by a relocation of a node in $r_j$. A subsequent relocation is considered, if it restores feasibility in $r_j$, and the overall costs of both relocations are non-positive. If the destination route of the second relocation is feasible, we store the set of both relocations as a candidate move. Otherwise, we continue the same process until an upper bound of relocations is reached.

In other words, we sequentially relocate nodes in such a way that the aggregated costs are non-positive and all but the destination route of the last relocation are feasible. For every relocation there might be several options for a continuation move, and thus, information about moves and their continuations is stored in a tree-structure. In order to avoid further cost computations, we do not allow an insertion into the position of a previously ejected customer node in the same RC. As in CE, we further reduce complexity by only considering insertions of a node next to its $C$ closest customer nodes. The more of those nodes are in the same route, the less options remain. Additionally, we only consider one relocation position per destination route. If, for instance, two of the nearest nodes of $I_2$ are $J_3$ and $J_5$, we only consider the relocation of $I_2$ into the position of route $r_j$ that results in the lowest insertion costs.

After, for each starting relocation, we have explored an upper limit of following relocations, we execute one of the candidate moves. A simple idea to improve efficiency and make use of the possibly quite long list of generated candidate moves, is to execute multiple moves. We first execute the candidate move with the largest decrease in costs (steepest descent), and then remove all other candidate moves that 'interfere' with the executed move. A move interferes with another move, if it includes a relocation of a node that is also relocated in the other move, or a relocation of a node that is a new or old neighbor of a node relocated in the other move. This interference changes the cost evaluation of the respective moves and would require additional cost computations. Among all non-interfering moves we again execute the one with the largest decrease in costs, and so forth.

We observed during the experiments described in Section 4 that the computation of RCs with a maximal length of four relocations requires up to 10 times more computation time than RCs with maximal three relocations. To keep computations efficient, we therefore limit RCs to three sequential relocations.

## 3.3 Guiding local search

The three operators LK, CE and RC generate large and diverse neighborhoods. Yet, at some point they will inevitably reach a local optimum $s^*$ for which the generated neighborhoods $\mathcal{N}(s^*)$ do not contain a better solution. Heuristics usually use *perturbation* to escape this local optimum. The idea of perturbation is to change solution $s^*$ where, in contrast to local search, this change does not necessarily have to result in an improving solution. Perturbation can appear in many different forms. *Large neighborhood search* and *ruin & recreate* destroy and repair parts of the solution. Methods based on simulated annealing accept, with a certain probability, local search moves that worsen the solution. Generally, most heuristics utilize some form of randomization to achieve a perturbed solution, or a pool thereof.

One approach that does not rely on randomization nor on additional algorithmic components is *guided local search* (GLS) (Voudouris and Tsang, 2003). Even though GLS is not as popular as other metaheuristics (for instance tabu search or variable neighborhood search), it has been successfully applied to the TSP (Voudouris and

Tsang, 2003), the VRP (Mester and Bräysy, 2007), and arc routing problems (Beullens et al., 2003).

The idea behind GLS is to change the cost evaluation of $s^*$, rather than $s^*$ itself. Features that are considered bad are penalized. In the context of the VRP, those features are edges, and the cost of bad edges is increased. More formally, we change the cost $c(i,j)$ of an edge in the current solution between customers $i$ and $j$ to

$$c^g(i,j) = c(i,j) + \lambda p(i,j)L, \tag{3.1}$$

where $p(i,j)$ counts the number of penalties of edge $(i,j)$, $L$ is the average cost of an edge in the starting solution and $\lambda$ controls the impact of penalties. Kilby et al. (1999) experimentally found $\lambda \in [0.1, 0.3]$ to be a good choice, and we will use $\lambda = 0.1$ in the remainder of the paper.

After the penalization of an edge, local search can be used on the adapted search space to potentially remove this edge. With increasing costs, the probability of finding an $c^g$-improving move increases. Note that a $c^g$-improving move does not have to correspond to a $c$-improving move.

The effectiveness of this penalization strategy hinges on the identification of 'bad' edges. An obvious idea is to focus on the most expensive edges as in Mester and Bräysy (2007), since these contribute more weight to the objective function. This simple observation seems to constitute the entire body of knowledge on which features of a VRP solution should be considered 'bad', or, more precisely, which edges should be removed preferentially.

In order to shed some light on this issue, we performed an exploratory study in Chapter 2, and found that the most predictive metrics, as to whether a solution is near-optimal or non-optimal, were the width and compactness of routes. Solutions of higher quality appear to have narrower and more compact routes. Hereby, the *width* of a route is defined as the maximum distance between any pair of customers, measured along the axis perpendicular to the line connecting the depot and the center of gravity of a route. The coordinates of the center of gravity of a route are calculated as the average of the coordinates of all nodes in that route. The *compactness* of a route is the average distance to the line connecting the depot and the center of gravity of all customers in that route.

In other words, moves that reduce the width of a route or increase its compactness are likely to make the solution better. Within the context of a GLS, we need to translate these general findings on bad solutions into guidelines on bad edges. Firstly, wider and less compact routes tend to have wider edges, and thus, it seems straight-forward to penalize and thereby avoid *wide* edges as shown in Section 2.4. Secondly, the penalization of long edges, i.e., edges with a high *cost*, seems reasonable since it has already proven to work well in the context of the VRP and

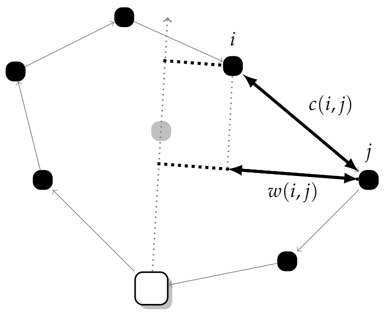


Figure 3.5: Metrics determining the 'badness' of edge $(i,j)$. The *width* $w(i,j)$ is computed as the distance between nodes $i$ and $j$ measured along the axis perpendicular to the line connecting the depot and the route's center of gravity (gray dot). The *cost* $c(i,j)$ is equal to the length of the edge.

the TSP. We confirmed with the above study that especially those edges that are connected to a depot tend to be shorter in high-quality solutions.

These two metrics *width* and *cost* are illustrated in Figure 3.5, and they can be readily transformed into functions that measures the 'badness' of an edge $(i,j)$:

$$b^{w}(i,j) = \frac{w(i,j)}{1+p(i,j)} \qquad b^{c}(i,j) = \frac{c(i,j)}{1+p(i,j)} \qquad b^{w,c}(i,j) = \frac{w(i,j)+c(i,j)}{1+p(i,j)} \tag{3.2}$$

where $w(i,j)$ denotes the width of edge $(i,j)$, and $c(i,j)$ its cost. The division by the number of previously received penalties is a standard approach in GLS, and should enhance the diversification of the penalization, so that the same edge is not penalized over and over again. A set of implemented local search operators (LS) together with a penalization strategy are sufficient to build a GLS heuristic for the VRP, as outlined in Algorithm 1.

Hereby, 'apply LS on a certain search space' means that we only consider moves originating from this space, e.g., if the search space is a single edge, we only consider CE moves that start with the removal of this edge, or a RC that starts with a relocation of one of the incident nodes. This heuristic framework is entirely based on local search, and does not depend on other components. Perturbation is achieved by changing the evaluation criterion and thereby inducing changes in the local optimum. The more moves are executed during this phase, the more the solution is perturbed. Note that a move is only executed if it improves the solution under $c^{g}(\cdot)$, and thus, not every penalization has to result in a subsequent move. Finally, the perturbed solution is optimised under normal evaluation $c(\cdot)$. We only apply LS on the part of the search space that changed during the previous perturbation phase, since routes that were not affected are unlikely to be the starting point of improving moves.

**Algorithm 1** A simple GLS framework

```
1: *Construction*. Construct a starting solution.
2: *Initial optimisation*. Apply LS on starting solution.
3: while not abortion criterion is reached do
4:     *Perturbation*:
5:     while not $P$ moves have been made do
6:         Penalize edge $(i, j)$ with the highest value $b(i, j)$ by incrementing $p(i, j)$.
7:         Apply LS on $(i, j)$, using $c^g(\cdot)$ as evaluation criterion.
8:     end while
9:     *Optimisation*. Apply LS on all routes that were changed during perturbation
10:    (using $c(\cdot)$ as evaluation criterion).
11: end while
```

Table 3.1: Local search operators used in the different setups.

| | Intra-route LS | Inter-route LS |
|---|---|---|
| $LS_1$ | 2-opt | relocate, swap, Or-exchange |
| $LS_2$ | LK | CE |
| $LS_3$ | LK | CE, RC |

This framework poses some interesting research questions which we will investigate in the following. Which operators should LS contain, how much perturbation is necessary and which penalization function $b(\cdot)$ is the most effective one?

## 3.4 What makes a local search effective?

In the following we conduct a series of experiments to explore various aspects of local search within the above GLS. We compare different neighbourhoods, different magnitudes of perturbation as well as different strategies to penalize edges. Three different sets of complementary local search operators are used as shown in Table 3.1. Set $LS_1$ is composed of relatively simple operators and generates the smallest neighboorhods of all sets. We mirror these operators with our CE implementation, where only exchanges of substrings of length one are considered (swap) or exchanges in which one substring is empty (relocate and Or-exchange). For the intra-route optimisation we use our LK implementation but only consider 2-opt moves. $LS_2$ generates larger neighborhood for both, intra-route and inter-route optimisation and uses the operators as described in the previous section. In $LS_3$ we additionally use the relocation chain as complementary neighborhood, so that this set should generate the largest neighborhoods.

Each of the three LS variants is used within the GLS framework. A starting solution is constructed with the popular and relatively simple heuristic by Clarke and Wright

(1964). During the optimisation phase, we first apply the inter-route operators on the considered search space, execute the move with the most beneficial evaluation (steepest descent), and then use the intra-route operator to optimise the changed routes. We iteratively apply this combination of inter- and intra-route optimisation until in one iteration the solution is not improved.

During the perturbation phase we iteratively penalize an edge and apply the inter-route operators on this edge, i.e., only moves which start with the removal of the respective edge are considered. Since the perturbation should trigger changes in the allocation of customers to routes, we apply the intra-route operator only at the end of the phase.

All experiments are executed on instances 26 to 55 by Uchoa et al. (2017). These instances have a reasonable size with 219 to 367 customers and are diverse with respect to all important characteristics of a VRP instance: clustering of customers (strong clustering versus uniform distribution), variation in customer demand ($[1, 1]$ to $[1, 100]$), the average length of routes (from 3 customers per route up to 24 customers per route) and depot location (central location versus eccentric location). Thus, experiments on this diverse set of instances should reduce the danger of overfitting the heuristic to particular types of instances. We report the results as performance over time on all 30 instances. The performance is expressed as the average gap between the best solutions found within a specific time horizon and the best known results, as reported by Uchoa et al. (2017). The average gap of the starting solutions computed with Clarke and Wright is about 6.4%.

### 3.4.1 *Neighborhood size and perturbation*

We investigate the impact of different neighborhoods generated by $LS_1$, $LS_2$ and $LS_3$ in interplay with different magnitudes of perturbation $P \in \{10, 100, 1000\}$. The edge penalization strategy $b^c$, and the degree of pruning for inter-route operators $C = 30$ are kept fixed. From the results in Figure 3.6 we can make the following observations.

Firstly, larger neighborhoods find better solutions. This seems a straight-forward statement to make, however, note that usually larger neighborhoods come at the expense of higher computational effort. Thus, smaller neighborhoods should generally find improving moves faster, at least at the start of the search. Since we do not observe this effect, we can conclude that the larger neighborhoods are pruned effectively and exhibit a good time-quality trade-off. On the other side, the results show that even a minimal local search implementation of four simple operators can obtain solutions within a 2% range in a very short computation time. Thus, gaps of this size should constitute a minimal standard when evaluating the performance of a new heuristic.

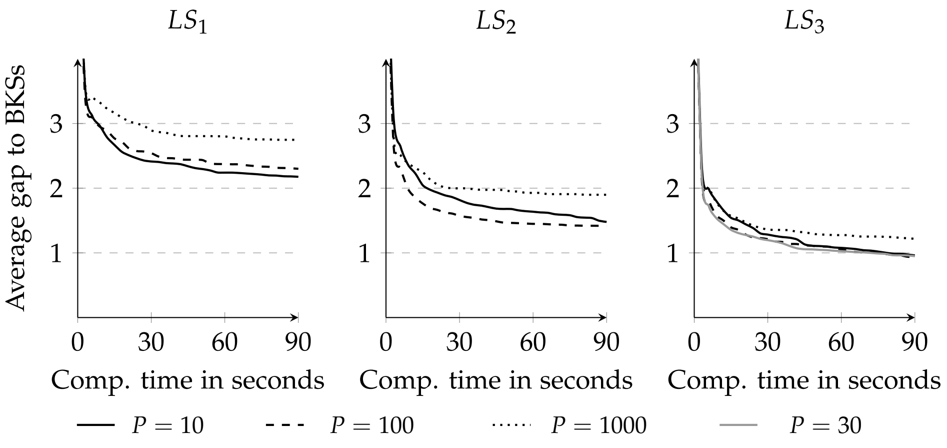


Figure 3.6: Average gap (in %) to the best known solutions (BKSs) over time on 30 instances by Uchoa et al. (2017) for different setups.

Secondly, RC seems to successfully complement CE, since the addition of this operator results in significant performance improvements. This is especially remarkable given that RC requires the most computation time of all three operators. Depending on the average length of routes, RC requires between 40% (on instances with longer routes) up to 80% (on instances with shorter routes) of the entire computation time. This additional computational effort seems to be beneficial.

Finally, the degree of perturbation should be chosen within reasonable bounds. Even though little perturbation ($P = 10$) seems to work well for smaller neighborhoods, for larger neighborhoods slightly more perturbation appears to be a better choice. An overly large perturbation ($P = 1000$) appears to disintegrate the previous local optimum for all considered neighborhoods, and results in a worse performance. Interestingly, the differences between different parameter choices become smaller with larger neighborhoods, as if larger neighborhoods compensate for a 'poorer choice'. We conclude that good results can be obtained with rather little but not too little perturbation. In more granular tests we found the best results with $P = 30$ for $LS_3$, and we will use this value in the remainder of the paper.

### 3.4.2 *Penalization criterion*

Keeping the best local search setup $LS_3$ with $P = 30$ fixed, we investigate which edges should be penalized. We compare five different strategies. The first three strategies correspond to $b^w$, $b^c$, and $b^{w,c}$. The fourth strategy diversifies the penalization, and deterministically changes the penalization criterion after every perturbation phase, in this order (rotation). Finally, we investigate the impact of the guided penalization strategies above by comparing them to an unguided strategy, in which the penalized

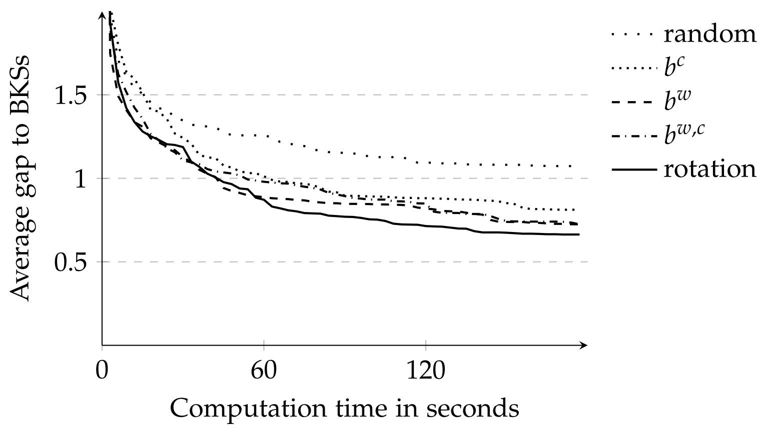


Figure 3.7: Average gap (in %) to the best known solutions (BKSs) over time on 30 instances by Uchoa et al. (2017) for different setups.

edge is chosen randomly (the seed of the random generator is fixed to keep the deterministic behaviour of the heuristic).

The results in Figure 3.7 reveal that the choice of edge penalization has less impact on performance than the choice of local search operators. Even with a randomized penalization, solutions with an average gap of about 1% can be computed. However, guidance can still improve this already good performance of the heuristic. Hereby, width appears to be a slightly better criterion to detect bad edges than cost, which confirms our findings in Chapter 2. The combination of both features $b^{w,c}$ yields similar results, while the best performance can be obtained if the penalization is diversified by using all three functions. A possible explanation for the success of this rotation strategy is that the best choice might be instance-dependent. For some instances it might be better to remove long edges, while for other ones it might be more important to obtain tight routes. One could attempt to investigate this relationship more closely, however, based on the findings in Figure 3.7, this is unlikely to result in significant performance gains. Another explanation could be that the exclusive focus on one criterion narrows the search efforts, and some edges are targeted excessively while others remain almost untouched. In other words, the degree of diversification might matter. When we tried the rotation strategies $b^w - b^c$ (less diversification) and $b^w - b^c - random$ (more diversification) we obtained worse results, and thus, the rotation strategy above appears to be a sweet-spot, at least for the tested instances.

In conclusion, the penalization strategy has less impact on performance than the local search, which suggests that most of the heuristic's efficiency is driven by a well-implemented local search. Nevertheless, guidance with problem-knowledge still elevates the overall performance, especially if the guidance is diversified.

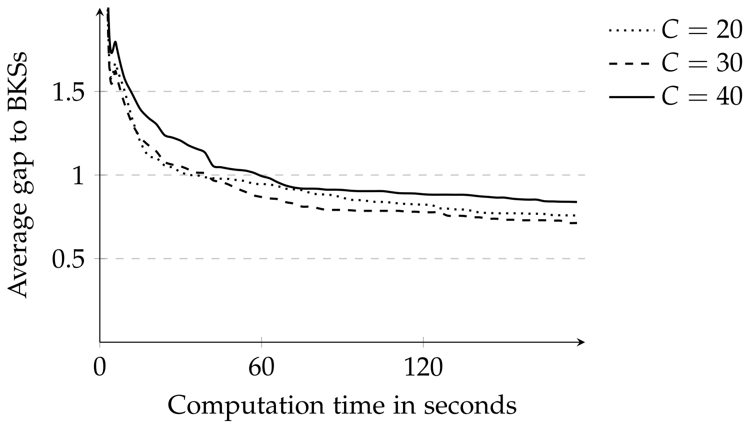


Figure 3.8: Average gap (in %) to the best known solutions (BKSs) over time on 30 instances by Uchoa et al. (2017) for different setups.

### 3.4.3 *Pruning*

For the inter-route operators we only consider new edges between a node and its $C$ closest nodes. For the experiments above we chose $C = 30$, since this value appeared to yield good result in pre-tests. In the following, we compare different pruning setups. If the value for $C$ is lower, then the pruning is tighter and the local search operators should be faster so that the GLS can execute more iterations in the same amount of time. On the other hand, a looser pruning increases the size of the considered neighborhoods at the expense of higher computational effort.

From the results in Figure 3.8 we observe that a tighter pruning is effective within short computation times, while $C = 30$ seems to be the better choice for longer runtimes. $C = 40$ appears to be too loose on the tested instances, and the size of the explored neighborhoods become unnecessarily large. We also observed some minor difference with respect to instance features. Generally, a looser pruning appeared to work better for instances with relatively small routes or instances with a large variation in demand. However, these effects were rather weak, and for simplicity we choose the same degree of pruning for all instances in the following.

Instead of considering the $C$ nearest nodes in terms of cost $c(\cdot)$, one could also use a different metric to define nearness. A good example is the $\alpha$-nearness defined in Helsgaun (2000) for the TSP. The $\alpha$-nearness $c^{\alpha}(\cdot)$ between two nodes $i$ and $j$ is defined as the cost difference between the optimal 1-tree of all nodes and the minimum 1-tree that contains edge $(i, j)$. Since every TSP solution is a 1-tree, both problems have a certain degree of similarity, but a minimal 1-tree can be computed more efficiently than a TSP solution. We implemented this idea for the VRP, and construct the 1-trees out of all customer nodes. We found that on average 96% of the $C = 30$ nearest nodes in terms of $c(\cdot)$ were also nearest nodes in terms of $c^{\alpha}(\cdot)$.

As a consequence, the local search considers the same edges and obtains the same performance. We also tested other metrics based on the idea of width. Given two nodes, we compute the distance between one node and the line connecting the depot and the other node, and add this 'width' to the cost $c(\cdot)$. The intuition behind this metric is to connect customers that are on the same line outgoing from the depot, and thus to obtain little detours. We found that this metric yielded good performances, however, it did not perform better than the simple $c(\cdot)$ metric. In conclusion, the traditional pruning with respect to costs appears to be an efficient pruning strategy for the VRP.

### 3.4.4 *The complete local search heuristic*

On the basis of the experimental results we identified the following local search setup as the most effective one.

**Algorithm 2** Knowlege-guided local search heuristic

```
1:  *Construction*. Construct a starting solution with the heuristic by Clarke and
    Wright and optimise the individual routes with LK.
2:  *Initial optimisation*. Apply CE and RC on starting solution. Whenever a route is
    changed, re-optimise it with LK. Set $b(\cdot) = b^w(\cdot)$.
3:  **while** time limit not reached **do**
4:      *Perturbation*:
5:      **while** not $P = 30$ moves have been made **do**
6:          Penalize edge $(i, j)$ with the highest value $b(i, j)$ by incrementing $p(i, j)$.
7:          Apply CE and RC on $(i, j)$, using $c^g(\cdot)$ as evaluation criterion.
8:      **end while**
9:      *Optimisation*. Change $b(\cdot)$. Apply LK, and then iteratively CE and RC on all
10:     routes that were changed during perturbation ($c(\cdot)$ as evaluation criterion).
11:     Whenever a route is changed with CE or RC, re-optimise it with LK.
12: **end while**
```

We assume that larger instances with more customers require more computation time and, thus, we define the abortion criterion as a maximum runtime with respect to the number of customers. The heuristic is entirely based on a local search with the steepest descent acceptance criterion, and thus, it works in a completely deterministic fashion. This determinism simplifies its analysis and the evaluation of its performance. In contrast, most state-of-the-art heuristics utilize randomness to diversify the search and escape local minima (Gutjahr, 2010). The exact role and the benefits of including random elements are rarely investigated, notwithstanding the fact that reliance on randomness has some significant disadvantages and, according to some researchers, can prevent the development of better, deterministic search components, see e.g., Glover (2007).

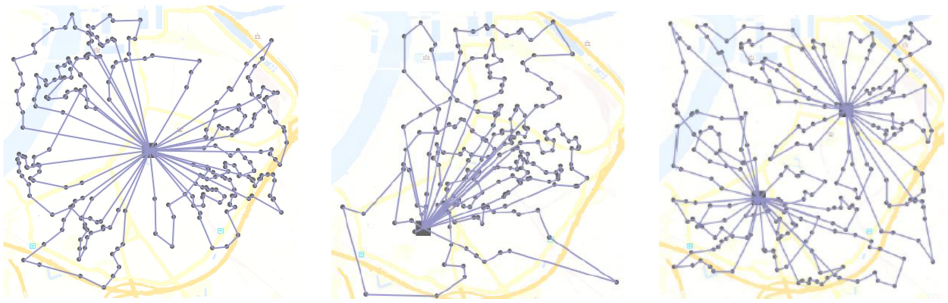


Figure 3.9: Benchmark instances from the 𝕌-set and the ℂ-set.

Finally, its minimalistic design can be utilized to apply it to variants of the standard VRP, for instance the Multi-Depot Vehicle Routing Problem (MDVRP). In the MDVRP customers can be delivered from a given set of depots. This adds another decision level: not only does the heuristic have to assign and sequence customers in routes, it also needs to determine which routes start and end at which depot. Notwithstanding this additional decision level, we can readily apply the heuristic to the MDVRP. The initial solution is constructed with a greedy approach, where each customer is assigned to its closest depot, and the routes per depot are then computed with the Clarke and Wright heuristic. The MDVRP involves to determine a suitable number of routes for each depot, and thus it it important to allow the heuristic to dynamically remove and add routes for each depot. The removal of routes is carried out by the local search, but the above heuristic never creates a new route. This is sufficient for VRP instances in which one usually wants to minimize the number of routes to obtain good solutions. For MDVRP instances, we add one empty dummy route per depot, and allow RC to relocate customers into these empty routes, if this move improves the solution.

## 3.5 Computational results

We test the performance of the heuristic on the entire instance set by Uchoa et al. (2017) (𝕌). As explained above, this instance set is diverse and considers different degrees of clustering customers, variation in customer demand, and different depot locations. For the MDVRP we perform the experiments on the benchmark set by Cordeau (ℂ) (Cordeau et al., 1997). Examples of solutions for both instance sets are visualized in Figure 3.9.

We compare the performance of our heuristic with some of the most efficient VRP heuristics in the literature: the hybrid genetic algorithm (HGSADC) of Vidal et al. (2013b), the iterated local search (ILS) of Subramanian et al. (2013), and the adaptive large neighborhood search (ALNS) of Pisinger and Ropke (2007).

### 3.5.1 *Test details*

The above heuristic has been implemented in Java and all tests have been performed within the Eclipse development environment on an AMD Ryzen 3 1300X CPU working at 3.5GHz on Windows 10, using a single thread. According to PassMark Software (2018) this setup is about 25% faster than the setup used to test HGSADC and ILS (Xeon CPU 3.07GHz). On each instance, we run the GLS up to a defined time limit of $3\frac{N}{100}$ minutes, i.e., we allow 3 minutes of computation time per 100 customers. We observed in pre-tests that larger time limits yield only marginal improvements in solutions quality, whereas most solutions are found in significant less time. When we double the time limit, we observe an average improvement by 0.05% on the $\mathbb{U}$-instances. For each instance we then report the best solution found within this time horizon as well as the required computation time to obtain this solution. We used the same configuration as described in Section 3.4.4 with the penalization strategy 'rotation'. On the whole instance set, this knowledge-guided penalization improves the performance by about 0.08% with respect to the classical penalization strategy 'cost'.

In the implementation, each node is represented by an object which contains distance information $d(\cdot)$, $d^g(\cdot)$, and penalty information to all other nodes, pre-processed information about the $C$ nearest nodes as well as the respective insertion costs, alongside basic information such as demand, the current route as well as the position in its current route. Routes are represented by a list of references to the respective nodes. For the encoding of routes, and also for other purposes such as the encoding of candidate moves, we used the datatype *ArrayList*, since it has a dynamic size and elements can be accessed in constant time. Even though insertions and deletions of elements require linear time (for instance to execute a move), they are executed not as often as accessing elements (for instance to evaluate a move).

### 3.5.2 *Results*

The aggregated results on the VRP benchmark set for the above GLS (denoted A&S) are presented in Table 3.2, detailed results per instance can be found in the appendix of this chapter. All other heuristics include random components, and present their results as the average performance over a number of runs, whereas the GLS produces the same solution in every run[1].

We observe that the GLS computes competitive solutions within a 1% range of the best known solution (BKS) for almost all $\mathbb{U}$-instances. On average the quality

1 An argument could be made that "the average cost over $n$ runs" is not a reasonable way to represent the results of a stochastic optimization algorithm. Surely, many practitioners will not accept a claim like "we are 50% sure that the solution quality will not be worse than X", but rather demand more certainty. The solution occupying the 5th percentile, e.g., would make for a more useful claim "we are 95% sure that the solution quality will not be worse than X". This is akin to statistical hypothesis testing, where the null hypothesis is only rejected when $p < .5$, i.e., when we are 95% sure of our claim.

Table 3.2: Results on the 100 $\mathbb{U}$ -instances, as average over all instances and for instances with a certain number of customers $N$. The gap is reported in % with respect to the BKS, and the time is reported in minutes, based on Uchoa et al. (2017).

| | ILS | | HGSADC | | A&S | |
|---|---|---|---|---|---|---|
| $N$ | Gap | Time | Gap | Time | Gap | Time |
| 100-250 | 0.31% | 2.4 | 0.07% | 6.0 | 0.45% | 2.2 |
| 250-500 | 0.53% | 23.1 | 0.24% | 30.3 | 0.59% | 4.7 |
| 500-1000 | 0.72% | 195.7 | 0.24% | 268.6 | 0.65% | 12.2 |
| | 0.53% | 71.7 | 0.19% | 98.8 | 0.57% | 6.3 |

Table 3.3: Results on the 23 MDVRP $\mathbb{C}$-instances, as average over all instances. The gap is reported in % with respect to the BKS, and the time is reported in minutes, based on Vidal et al. (2013b).

| ALNS | | HGSADC | | A&S | |
|---|---|---|---|---|---|
| Gap | Time | Gap | Time | Gap | Time |
| 0.40% | 3.8 | 0.06% | 4.1 | 0.08% | 0.7 |

of the computed solutions is comparable to those of the ILS and within a 0.40% range of those of HGSADC. It is especially successful on instances with many customers per route, since the local search puts a strong emphasis on intra-route optimization and the exchanges of (possibly quite long) sequences. Reversely, the GLS has more difficulties on instances with very short routes. On those instances, intra-route optimization and sequence exchanges are rarely possible, and most of the changes are triggered by RC. Likewise, some $\mathbb{U}$-instances with a very large variance in customer demand are difficult to solve, since they enforce the occurrence of extremely long edges. The results in Table 3.3 highlight that the GLS performs similarly to the best MDVRP heuristic in the literature and outperforms the ALNS.

These competitive results are computed in short computing times, especially on instances of larger size. In summary, it does not require more than a few minutes to compute solutions with very small gaps for VRP as well as MDVRP instances with several hundred customers, resulting in a remarkable time-quality trade-off. This time-quality trade-off opens a lot of potential to further improve the heuristic, and can be attributed to its scalability. For instance, we observed that resets can further increase the quality of computed solutions. If the GLS failed to find improvements for some time, the current solution can be reset to the previously best found solution while all accumulated penalties are removed.

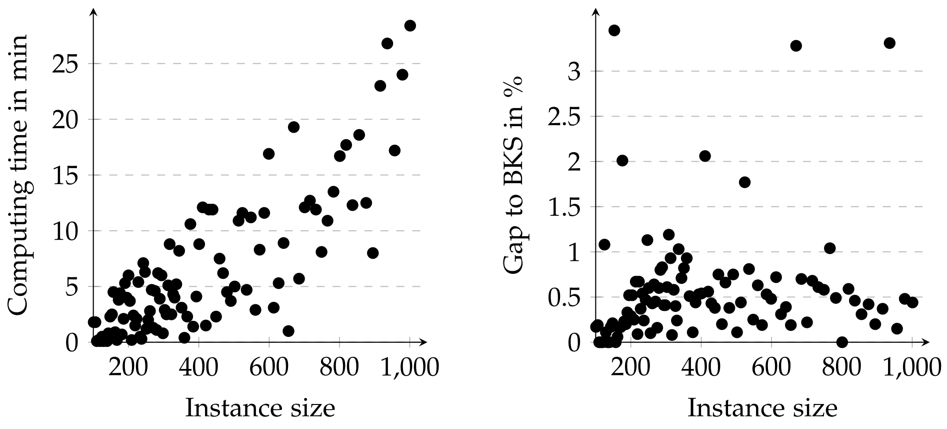


Figure 3.10: Computation time (left) versus performance (right) in dependence of instance size on the 100 𝕌-instances. Solutions of a similar high quality are found within computation times that grow approximately linear in the instance size.

In Figure 3.10 (left) we visualize the scaling of the heuristic's computing time and performance with growing instance size. As can be seen, the GLS computes solutions of similar quality, while the required computation time grows more or less as a linear function of the instance size. This linear scaling has its origin in the efficient design of the local search operators, which, despite all the heuristic pruning, generates high-quality solutions.

All in all, the presented GLS is one of the most efficient heuristics in the routing literature. It consistently produces competitive solutions for a large variety of instances in a very short time.

## 3.6 Conclusions and future research

In this chapter we demonstrated how to build a heuristic for the VRP around a well-implemented local search. We implemented three complementary local search operators using ideas from sequential search and pruning. This local search was embedded in a guided local search framework that penalizes and removes bad edges. To detect bad edges more accurately, we used insights from an exploratory study on properties of VRP solutions. Thorough experimentation on all key components showed that relatively little perturbation combined with large neighborhoods results in an effective local search.

The resulting heuristic is able to compete with the best heuristics in the literature on a wide range of benchmark instances. It computes high-quality solutions for large-scale instances in a few seconds or minutes. Therefore, it presents a suitable solution approach to routing problems that are restricted with respect to computation time. On the other hand, it might also be a starting point for further extensions to generate even better solutions, if more computation time is available. It can be extended by, for instance, adapting the edge penalization through learning mechanisms, or by embedding the heuristic into another metaheuristic. At the same time, we demonstrated that the heuristic can be used to solve other problem variants like the MDVRP.

From a more general perspective, we have shown that the development of a successful heuristic for the VRP does not require new operators or complex frameworks. Using local search operators that have been proven to work well, paired with an efficient implementation and the utilization of problem-knowledge, is sufficient to create powerful heuristics that work well on a wide range of instances and problem variations. Rather than through the development of new heuristic operators, let alone new metaheuristic frameworks, algorithms can be improved by putting more research emphasis on how to use *existing* methods in a more efficient and intelligent way. The utilization of problem-specific knowledge is a promising starting point for this. Knowing the structural properties that distinguish a near-optimal solution from a suboptimal solution allows an algorithm to prioritize certain features in the search process, and to develop operators that tackle a problem more efficiently. This chapter has presented one of the first approaches in this challenging research direction.

## 3.7 Appendix: Detailed benchmark results

Table 3.4: Results on the MDVRP ℂ-instances (Cordeau et al., 1997).

| Instance | D | ALNS | | | HGSADC | | | A&S | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | | Value | Gap | Time | Value | Gap | Time | Value | Gap | Time |
| P01 (50) | 4 | 576.87 | 0.00 | 0.5 | 576.87 | 0.00 | 0.2 | 576.87 | 0.00 | 0.1 |
| P02 (50) | 4 | 473.53 | 0.00 | 0.5 | 473.53 | 0.00 | 0.2 | 473.53 | 0.00 | 0.1 |
| P03 (75) | 2 | 641.19 | 0.00 | 1.1 | 641.19 | 0.00 | 0.4 | 641.19 | 0.00 | 0.2 |
| P04 (100) | 2 | 1006.9 | 0.49 | 1.5 | 1001.23 | 0.00 | 1.9 | 1001.06 | 0.00 | 1.0 |
| P05 (100) | 2 | 752.3 | 0.31 | 2 | 750.03 | 0.00 | 1.1 | 750.59 | 0.07 | 0.9 |
| P06 (100) | 3 | 883.01 | 0.74 | 1.6 | 876.50 | 0.00 | 1.1 | 876.70 | 0.02 | 0.9 |
| P07 (100) | 4 | 889.36 | 0.84 | 1.5 | 884.43 | 0.28 | 1.6 | 883.91 | 0.22 | 1.4 |
| P08 (249) | 2 | 4421.03 | 1.10 | 5.6 | 4397.42 | 0.56 | 10.0 | 4390.22 | 0.40 | 0.1 |
| P09 (249) | 3 | 3892.50 | 0.88 | 6.0 | 3868.59 | 0.26 | 9.5 | 3866.95 | 0.21 | 1.7 |
| P10 (249) | 4 | 3666.85 | 0.98 | 6.1 | 3636.08 | 0.14 | 9.8 | 3658.64 | 0.76 | 0.2 |
| P11 (249) | 4 | 3573.23 | 0.77 | 6.0 | 3548.25 | 0.06 | 7.1 | 3550.88 | 0.14 | 4.0 |
| P12 (80) | 2 | 1319.13 | 0.01 | 1.3 | 1318.95 | 0.00 | 0.5 | 1318.95 | 0.00 | 0.1 |
| P13 (80) | 2 | 1318.95 | 0.00 | 1.0 | 1318.95 | 0.00 | 0.6 | 1318.95 | 0.00 | 0.1 |
| P14 (80) | 2 | 1360.12 | 0.00 | 1.0 | 1360.12 | 0.00 | 0.6 | 1360.12 | 0.00 | 0.1 |
| P15 (160) | 4 | 2519.64 | 0.57 | 4.2 | 2505.42 | 0.00 | 1.9 | 2505.42 | 0.00 | 0.1 |
| P16 (160) | 4 | 2573.95 | 0.07 | 3.1 | 2572.23 | 0.00 | 2.0 | 2572.23 | 0.00 | 0.1 |
| P17 (160) | 4 | 2709.09 | 0.00 | 3.0 | 2709.09 | 0.00 | 2.1 | 2709.09 | 0.00 | 0.1 |
| P18 (240) | 6 | 3736.53 | 0.91 | 7.0 | 3702.85 | 0.00 | 4.5 | 3702.85 | 0.00 | 3.4 |
| P19 (240) | 6 | 3838.76 | 0.31 | 5.3 | 3827.06 | 0.00 | 4.2 | 3827.06 | 0.00 | 0.0 |
| P20 (240) | 6 | 4064.76 | 0.16 | 5.0 | 4058.07 | 0.00 | 4.4 | 4058.07 | 0.00 | 0.3 |
| P21 (360) | 6 | 5501.58 | 0.46 | 9.7 | 5476.41 | 0.00 | 10.0 | 5482.47 | 0.11 | 1.0 |
| P22 (360) | 6 | 5722.19 | 0.35 | 7.7 | 5702.16 | 0.00 | 10.0 | 5702.16 | 0.00 | 0.1 |
| P23 (360) | 6 | 6092.66 | 0.23 | 7.4 | 6078.75 | 0.00 | 10.0 | 6078.75 | 0.00 | 1.0 |
| | | | 0.40 | 3.8 | | 0.06 | 4.1 | | 0.08 | 0.7 |

Table 3.5: Results on the VRP 𝕌-instances (Uchoa et al., 2017).

| Instance | ILS | | | HGSADC | | | A&S | | |
|---|---|---|---|---|---|---|---|---|---|
| | Value | Gap | Time | Value | Gap | Time | Value | Gap | Time |
| P01 (101) | 27591.0 | 0.00 | 0.1 | 27591.0 | 0.00 | 1.4 | 27637 | 0.17 | 1.8 |
| P02 (106) | 26375.9 | 0.05 | 2.0 | 26381.8 | 0.08 | 4.0 | 26411 | 0.19 | 1.8 |
| P03 (110) | 14971.0 | 0.00 | 0.2 | 14971.0 | 0.00 | 1.6 | 14971 | 0.00 | 0.1 |
| P04 (115) | 12747.0 | 0.00 | 0.2 | 12747.0 | 0.00 | 1.8 | 12747 | 0.00 | 0.2 |
| P05 (120) | 13337.6 | 0.04 | 1.7 | 13332.0 | 0.00 | 2.3 | 13332 | 0.00 | 0.1 |
| P06 (125) | 55673.8 | 0.24 | 1.4 | 55542.1 | 0.01 | 2.7 | 56140 | 1.08 | 0.5 |
| P07 (129) | 28998.0 | 0.20 | 1.9 | 28948.5 | 0.03 | 2.7 | 28973 | 0.11 | 0.1 |
| P08 (134) | 10947.4 | 0.29 | 2.1 | 10934.9 | 0.17 | 3.3 | 10916 | 0.00 | 0.2 |
| P09 (139) | 13603.1 | 0.10 | 1.6 | 13590.0 | 0.00 | 2.3 | 13590 | 0.00 | 0.1 |
| P10 (143) | 15745.2 | 0.29 | 1.6 | 15700.2 | 0.00 | 3.1 | 15726 | 0.17 | 0.8 |
| P11 (148) | 43452.1 | 0.01 | 0.8 | 43448.0 | 0.00 | 3.2 | 43539 | 0.21 | 2.3 |
| P12 (153) | 21400.0 | 0.85 | 0.5 | 21226.3 | 0.03 | 5.5 | 21953 | 3.45 | 2.5 |
| P13 (157) | 16876.0 | 0.00 | 0.8 | 16876.0 | 0.00 | 3.2 | 16876 | 0.00 | 4.5 |
| P14 (162) | 14160.1 | 0.16 | 0.5 | 14141.3 | 0.02 | 3.3 | 14147 | 0.06 | 0.9 |
| P15 (167) | 20608.7 | 0.25 | 0.9 | 20563.2 | 0.03 | 3.7 | 20589 | 0.16 | 0.2 |
| P16 (172) | 45616.1 | 0.02 | 0.6 | 45607.0 | 0.00 | 3.8 | 45684 | 0.17 | 3.8 |
| P17 (176) | 48249.8 | 0.92 | 1.1 | 47957.2 | 0.30 | 7.6 | 48772 | 2.01 | 4.4 |
| P18 (181) | 25571.5 | 0.01 | 1.6 | 25591.1 | 0.09 | 6.3 | 25628 | 0.23 | 0.7 |
| P19 (186) | 24186.0 | 0.17 | 1.7 | 24147.2 | 0.01 | 5.9 | 24192 | 0.20 | 2.1 |
| P20 (190) | 17143.1 | 0.96 | 2.1 | 16987.9 | 0.05 | 12.1 | 17036 | 0.33 | 5.3 |
| P21 (195) | 44234.3 | 0.02 | 0.9 | 44244.1 | 0.04 | 6.1 | 44453 | 0.52 | 4.0 |
| P22 (200) | 58697.2 | 0.20 | 7.5 | 58626.4 | 0.08 | 8.0 | 58747 | 0.29 | 6.0 |
| P23 (204) | 19625.2 | 0.31 | 1.1 | 19571.5 | 0.03 | 5.4 | 19666 | 0.52 | 3.7 |
| P24 (209) | 30765.4 | 0.36 | 3.8 | 30680.4 | 0.08 | 8.6 | 30733 | 0.25 | 0.2 |
| P25 (214) | 11126.9 | 0.25 | 2.3 | 10877.4 | 0.20 | 10.2 | 10929 | 0.67 | 2.4 |
| P26 (219) | 117595.0 | 0.00 | 0.9 | 117604.9 | 0.01 | 7.7 | 117696 | 0.09 | 1.5 |
| P27 (223) | 40533.5 | 0.24 | 8.5 | 40499.0 | 0.15 | 8.3 | 40708 | 0.67 | 2.1 |
| P28 (228) | 25795.8 | 0.21 | 2.4 | 25779.3 | 0.14 | 9.8 | 25838 | 0.37 | 5.4 |
| P29 (233) | 19336.7 | 0.55 | 3.0 | 19288.4 | 0.30 | 6.8 | 19333 | 0.54 | 0.5 |
| P30 (237) | 27078.8 | 0.14 | 3.5 | 27067.3 | 0.09 | 8.9 | 27108 | 0.24 | 0.3 |
| P31 (242) | 82874.2 | 0.15 | 17.8 | 82948.7 | 0.24 | 12.4 | 83144 | 0.47 | 7.1 |
| P32 (247) | 37507.2 | 0.63 | 2.1 | 37284.4 | 0.03 | 20.4 | 37697 | 1.13 | 6.3 |
| | | 0.31 | 2.4 | | 0.07 | 6.0 | | 0.45 | 2.2 |

Table 3.6: Results on the VRP 𝕌-instances (Uchoa et al., 2017).

| Instance | ILS | | | HGSADC | | | A&S | | |
|---|---|---|---|---|---|---|---|---|---|
| | Value | Gap | Time | Value | Gap | Time | Value | Gap | Time |
| P33 (251) | 38840.0 | 0.40 | 10.8 | 38796.4 | 0.29 | 11.7 | 38916 | 0.60 | 1.2 |
| P34 (256) | 18883.9 | 0.02 | 2.0 | 18880.0 | 0.00 | 6.5 | 18899 | 0.10 | 2.0 |
| P35 (261) | 26869.0 | 1.17 | 6.7 | 26629.6 | 0.27 | 12.7 | 26672 | 0.43 | 2.8 |
| P36 (266) | 75563.3 | 0.11 | 10.0 | 75759.3 | 0.37 | 21.4 | 75964 | 0.64 | 4.7 |
| P37 (270) | 35363.4 | 0.21 | 9.1 | 35367.2 | 0.22 | 11.3 | 35451 | 0.45 | 1.2 |
| P38 (275) | 21256.0 | 0.05 | 3.6 | 21280.6 | 0.17 | 12.0 | 21280 | 0.16 | 4.6 |
| P39 (280) | 33769.4 | 0.80 | 9.6 | 33605.8 | 0.31 | 19.1 | 33704 | 0.60 | 1.1 |
| P40 (284) | 20448.5 | 1.10 | 8.6 | 20286.4 | 0.30 | 19.9 | 20388 | 0.80 | 6.2 |
| P41 (289) | 95450.6 | 0.28 | 16.1 | 95469.5 | 0.30 | 21.3 | 95971 | 0.83 | 3.9 |
| P42 (294) | 47254.7 | 0.19 | 12.4 | 47259.0 | 0.20 | 14.7 | 47360 | 0.41 | 6.0 |
| P43 (298) | 34356.0 | 0.37 | 6.9 | 34292.1 | 0.18 | 10.9 | 34371 | 0.41 | 0.8 |
| P44 (303) | 21895.8 | 0.69 | 14.2 | 21850.9 | 0.49 | 17.3 | 21877 | 0.61 | 2.9 |
| P45 (308) | 26101.1 | 0.94 | 9.5 | 25895.4 | 0.14 | 15.3 | 26166 | 1.19 | 2.5 |
| P46 (313) | 94297.3 | 0.27 | 17.5 | 94265.2 | 0.24 | 22.4 | 94919 | 0.93 | 5.1 |
| P47 (317) | 78356.0 | 0.00 | 8.6 | 78387.8 | 0.04 | 22.4 | 78414 | 0.08 | 8.8 |
| P48 (322) | 29991.3 | 0.42 | 14.7 | 29956.1 | 0.30 | 15.2 | 30038 | 0.58 | 2.5 |
| P49 (327) | 27812.4 | 0.93 | 19.1 | 27628.2 | 0.26 | 18.2 | 27667 | 0.40 | 4.3 |
| P50 (331) | 31235.5 | 0.43 | 15.7 | 31159.6 | 0.18 | 24.4 | 31178 | 0.24 | 4.0 |
| P51 (336) | 139461.0 | 0.19 | 21.4 | 139534.9 | 0.24 | 38.0 | 140635 | 1.03 | 5.2 |
| P52 (344) | 42284.0 | 0.44 | 22.6 | 42208.8 | 0.26 | 21.7 | 42398 | 0.71 | 8.2 |
| P53 (351) | 26150.3 | 0.79 | 25.2 | 26014.0 | 0.26 | 33.7 | 26159 | 0.82 | 3.1 |
| P54 (359) | 52076.5 | 1.10 | 48.9 | 51721.7 | 0.41 | 34.9 | 51988 | 0.93 | 0.4 |
| P55 (367) | 23003.2 | 0.83 | 13.1 | 22838.4 | 0.11 | 22.0 | 22930 | 0.51 | 2.3 |
| P56 (376) | 147713.0 | 0.00 | 7.1 | 147750.2 | 0.03 | 28.3 | 147879 | 0.11 | 10.6 |
| P57 (384) | 66372.5 | 0.44 | 34.5 | 66270.2 | 0.29 | 40.2 | 66372 | 0.44 | 1.4 |
| P58 (393) | 38457.4 | 0.49 | 20.8 | 38374.9 | 0.28 | 28.7 | 38471 | 0.53 | 4.1 |
| P59 (401) | 66715.1 | 0.71 | 60.4 | 66365.4 | 0.18 | 49.5 | 66602 | 0.54 | 8.8 |
| P60 (411) | 19954.9 | 1.20 | 23.8 | 19743.8 | 0.13 | 34.7 | 20125 | 2.06 | 12.1 |
| P61 (420) | 107838.0 | 0.04 | 22.2 | 107924.1 | 0.12 | 53.2 | 108400 | 0.56 | 1.5 |
| P62 (429) | 65746.6 | 0.37 | 38.2 | 65648.5 | 0.23 | 41.5 | 65784 | 0.43 | 11.9 |
| P63 (439) | 36441.6 | 0.13 | 39.6 | 36451.1 | 0.15 | 34.6 | 36534 | 0.38 | 11.9 |
| P64 (449) | 56204.9 | 1.53 | 59.9 | 55553.1 | 0.35 | 64.9 | 55772 | 0.75 | 2.3 |
| P65 (459) | 24462.4 | 1.16 | 60.6 | 24272.6 | 0.38 | 42.8 | 24230 | 0.20 | 7.5 |
| P66 (469) | 222182.0 | 0.12 | 36.3 | 222617.1 | 0.32 | 86.7 | 223378 | 0.66 | 6.2 |
| P67 (480) | 89871.2 | 0.38 | 50.4 | 89760.1 | 0.25 | 67.0 | 89871 | 0.38 | 4.5 |
| P68 (491) | 67226.7 | 0.89 | 52.2 | 66898.0 | 0.40 | 71.9 | 67132 | 0.75 | 3.7 |
| | | 0.53 | 23.1 | | 0.24 | 30.3 | | 0.59 | 4.7 |

Table 3.7: Results on the VRP 𝕌-instances (Uchoa et al., 2017).

| Instance | ILS | | | HGSADC | | | A&S | | |
|---|---|---|---|---|---|---|---|---|---|
| | Value | Gap | Time | Value | Gap | Time | Value | Gap | Time |
| P69 (502) | 69346.8 | 0.14 | 80.8 | 69328.8 | 0.11 | 63.6 | 69329 | 0.11 | 5.0 |
| P70 (513) | 24434.0 | 0.96 | 35.0 | 24296.6 | 0.40 | 33.1 | 24307 | 0.44 | 10.9 |
| P71 (524) | 155005.0 | 0.27 | 27.3 | 154979.5 | 0.25 | 80.7 | 157338 | 1.77 | 11.6 |
| P72 (536) | 95700.7 | 0.61 | 62.1 | 95330.6 | 0.22 | 107.5 | 95888 | 0.81 | 4.7 |
| P73 (548) | 86874.1 | 0.19 | 64.0 | 86998.5 | 0.33 | 84.2 | 86929 | 0.25 | 11.2 |
| P74 (561) | 43131.3 | 0.88 | 68.9 | 42866.4 | 0.26 | 60.6 | 43024 | 0.63 | 2.9 |
| P75 (573) | 51173.0 | 0.77 | 112.0 | 50915.1 | 0.27 | 188.2 | 50874 | 0.19 | 8.3 |
| P76 (586) | 190919.0 | 0.20 | 78.5 | 190838.0 | 0.15 | 175.3 | 191553 | 0.53 | 11.6 |
| P77 (599) | 109384.0 | 0.52 | 73.0 | 109064.2 | 0.23 | 125.9 | 109335 | 0.48 | 16.9 |
| P78 (613) | 60444.2 | 1.11 | 74.8 | 59960.0 | 0.30 | 117.3 | 60208 | 0.72 | 3.1 |
| P79 (627) | 62905.6 | 0.87 | 162.7 | 62524.1 | 0.25 | 239.7 | 62560 | 0.31 | 5.3 |
| P80 (641) | 64606.1 | 1.20 | 140.4 | 64192.0 | 0.55 | 158.8 | 64086 | 0.39 | 8.9 |
| P81 (655) | 106782.0 | 0.00 | 47.2 | 106899.1 | 0.11 | 150.5 | 106987 | 0.19 | 1.0 |
| P82 (670) | 147676.0 | 0.66 | 61.2 | 147222.7 | 0.35 | 264.1 | 151519 | 3.28 | 19.3 |
| P83 (685) | 68988.2 | 0.82 | 73.9 | 68654.1 | 0.33 | 156.7 | 68904 | 0.70 | 5.7 |
| P84 (701) | 83042.2 | 0.91 | 210.1 | 82487.4 | 0.24 | 253.2 | 82477 | 0.22 | 12.1 |
| P85 (716) | 44171.6 | 1.49 | 225.8 | 43641.4 | 0.27 | 264.3 | 43820 | 0.68 | 12.7 |
| P86 (733) | 137045.0 | 0.50 | 111.6 | 136587.6 | 0.16 | 244.5 | 137194 | 0.61 | 11.9 |
| P87 (749) | 78275.9 | 0.74 | 127.2 | 77864.9 | 0.21 | 313.9 | 78151 | 0.58 | 8.1 |
| P88 (766) | 115738.0 | 0.92 | 242.1 | 115147.9 | 0.41 | 383.0 | 115872 | 1.04 | 10.9 |
| P89 (783) | 73722.9 | 1.37 | 235.5 | 73009.6 | 0.39 | 269.7 | 73084 | 0.49 | 13.5 |
| P90 (801) | 74005.7 | 0.57 | 432.6 | 73731.0 | 0.20 | 289.2 | 73580 | 0.0 | 16.7 |
| P91 (819) | 159425.0 | 0.51 | 148.9 | 158899.3 | 0.18 | 374.3 | 159553 | 0.59 | 17.7 |
| P92 (837) | 195027.0 | 0.39 | 173.2 | 194476.5 | 0.11 | 463.4 | 195155 | 0.46 | 12.3 |
| P93 (856) | 89277.6 | 0.24 | 153.7 | 89238.7 | 0.20 | 288.4 | 89336 | 0.31 | 18.6 |
| P94 (876) | 100417.0 | 0.70 | 409.3 | 99884.1 | 0.17 | 495.4 | 100130 | 0.42 | 12.5 |
| P95 (895) | 54958.5 | 1.45 | 410.2 | 54439.8 | 0.49 | 321.9 | 54280 | 0.20 | 8.0 |
| P96 (916) | 330948.0 | 0.33 | 226.1 | 330198.3 | 0.11 | 560.8 | 331043 | 0.37 | 23.0 |
| P97 (936) | 134530.0 | 1.07 | 202.5 | 133512.9 | 0.31 | 531.5 | 137510 | 3.31 | 26.8 |
| P98 (957) | 85936.6 | 0.31 | 311.2 | 85822.6 | 0.18 | 432.9 | 85799 | 0.15 | 17.2 |
| P99 (979) | 120253.0 | 0.89 | 687.2 | 119502.1 | 0.26 | 554.0 | 119761 | 0.48 | 24.0 |
| P100 (1001) | 73985.4 | 1.71 | 792.8 | 72956.0 | 0.29 | 549.0 | 72943 | 0.28 | 17.7 |
| | | 0.72 | 195.7 | | 0.24 | 268.6 | | 0.65 | 12.2 |

# 4 Efficiently solving very large scale routing problems

## 4.1 Introduction

Recent research on the VRP and variants has focused on developing algorithms that are able to solve routing problems of at most a few hundred customers. The reason for the focus on instances of this size is unclear. One potential explanation is that it is a result of the size of most popular benchmark instances. The often used instances by Golden et al. (1998) contain up to 483 customers, and the largest problem in the more recent benchmark set by Uchoa et al. (2017) involves 1001 customers. Since a good performance on the standard benchmark instance sets significantly increases a paper's publication potential, it is natural to see heuristics developed that specifically target this problem size.

Even though routing problems of up to 1000 customers cover a wide range of real-world applications, there are routing problems of significantly larger sizes that need to be tackled on a daily basis. One example presented in Kytöjoki et al. (2007) is waste collection, where trucks have to collect waste from tens of thousands of customers. With more and more information, about, e.g., the amount of waste in each container becoming available and with increasing standards in service level and cost pressure, there is a need to model and optimize these large-scale systems more accurately. Another example is the distribution of parcels. The continuing growth of e-commerce goes hand in hand with a steady increase in the number of parcels that need to be delivered to customers every day. In Belgium, for example, the daily quantity of delivered parcels correspond to about 1% of the population, so that in cities like Brussels 20,000 and more deliveries need to be carried out every day. Given that the delivery business is largely cost-driven, these deliveries have to be planned and executed as efficiently as possible.

Consequently, there is a strong practical motivation to develop solution methods that can tackle much larger vehicle routing problems than those available in the standard benchmark instance sets. In the following we will use the term *'very large scale'* to denote such instances since *'large scale'* is commonly used to describe problems of several hundred customers. A first, and so far only, step in this direction has been taken a decade ago in Kytöjoki et al. (2007), who develop a variable neighborhood search algorithm that is able to solve instances of up to 20,000 customers. This limited amount of research is especially striking given that the literature on the TSP has covered very large instances since several decades (Reinelt, 1991), solving instances with up to 25,000,000 nodes. Of course, the development and testing of heuristics that are able to tackle very large instances requires that a benchmark set with such instances is available.

Efficiently solving instances of this size puts additional requirements on heuristics, both in terms of computational complexity and memory usage. A widely-used approach to reduce computational complexity is *heuristic pruning*. With pruning only promising parts of the solution space are investigated. For instance, the famous heuristic by Lin and Kernighan (1973) only consider local search moves that connect relatively close nodes. Helsgaun (2000) bases the decision of whether to consider a certain edge in a TSP solution on its $\alpha$-nearness, defined as the cost difference between the optimal 1-tree of all nodes and the minimum 1-tree that contains this edge. Since every TSP solution is a 1-tree, the cost differences in 1-trees provide a good intuition about the likelihood that an edge will be included in an optimal TSP solution. Similar approaches can be used to reduce memory usage. Helsgaun (2000) only stores distance entries of candidate edges computed with the approach above, and all other requested distance values are computed and cached during runtime. Applegate et al. (2003) minimize the representation of large TSPs by building a neighbour graph. Rather than limiting the amount of stored information, information can also be stored in a highly compressed format as proposed by Kytöjoki et al. (2007). They replace distance values by a more condensed representation of speed and twisty-road-curve factors, and the exact distances are recomputed during runtime.

In this chapter, we explore how such concepts can be applied in the context of local search to compute high-quality solutions for VRPs with several thousands or more customers in a reasonable amount of computing time. In Chapter 3 we have demonstrated that a well-implemented and well-configured local search is sufficient to compete with state-of-the-art heuristics for VRPs with up to 1000 customers, and the question arising is whether similar ideas can be used to tackle instances of larger magnitudes. On the basis of our findings about local search for smaller instances (Section 2), we outline the challenges of very large scale VRPs, and propose corresponding solutions in Section 3. These ideas are tested in various experiments in Section 4 to shed light on the questions 'how complex should local search operators be', 'how much pruning is necessary' and 'how much information should be stored'. As the second main contribution, we generate 10 very large scale VRP instances in Section 5. These instances are based on parcel demand data in Belgium and, thus, reflect real-world problems. The instance set is publicly available to stimulate more research in the domain of very large scale VRPs. Finally, we conclude with a summary of our findings in Section 6.

## 4.2 From large VRPs to very large VRPs

In Chapter 3, we developed a local-search based heuristic that is able to compete with the best heuristics in the literature on a wide range of benchmark instances, computing competitive results extremely fast. For most instance of up to 1001 customers it does not require more than a few minutes to compute a solution with a gap of 1% and less to the best known solutions, as demonstrated in Figure 3.10. This effective time-quality trade-off opens a lot of potential to further improve the heuristic, and can be attributed to a good scalability. In the following, we investigate how this scalability is a suitable basis to extend the heuristic to solve instances of larger magnitudes.

Solving very large instances with several thousand or tens of thousand of customers poses additional challenges to heuristics. Firstly, the computational complexity might grow rapidly in problem size. In Figure 4.1 we plot the computation times of two state-of-the-art heuristics, the hybrid genetic heuristic (HGSADC) by Vidal et al. (2013b) and the iterated local search (ILS) by Subramanian et al. (2013) against problem size, as reported in Uchoa et al. (2017). If we were to extrapolate these trajectories, we would conclude that the solution of a problem of, e.g., 10,000 customers (while keeping a similar solution quality) would require several months of computation time. Consequently, tackling very large scale VRPs demands heuristics to scale well in problem size. Even though the local search outlined above appears to have a good scaling for instances of up to 1000 customers, care must be taken to reduce the size of the considered solution space as much as possible.

Secondly, the amount of memory used (RAM) has to be limited. For instance, storing the distance between each pair of nodes in order to compute the length of edges and routes during the execution of the algorithm requires $N^2$ entries, where $N$ is the number of nodes. If a distance entry is represented by a float (4 bytes), a complete distance matrix for a 20,000 customer instance would require almost 3GB memory. In addition to the distance matrix, we also need to store information about penalized cost values and received penalties per edge, represent a solution and store potential local search moves. All of this data might increase memory usage way beyond available memory.

Both of these issues can be addressed by ignoring those parts of the solution space that appear unpromising, which is the idea of pruning. In other words, the higher complexity of larger instances can be tackled by restricting the considered solution space even more. If long edges do not appear in high quality solutions, as generally assumed, then there is no need to evaluate moves that contain such edges. Going one step further, it might not even be necessary to store information about such

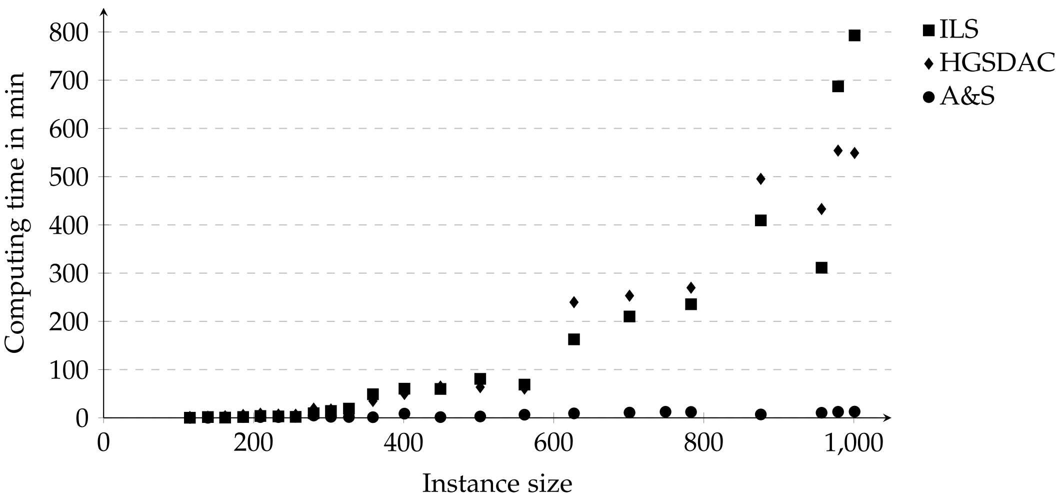


Figure 4.1: Computation time as a function of instance size on the instances of Uchoa et al. (2017) for two state-of-the-art heuristics from the literature (ILS and HGSADC) and the heuristic presented in Chapter 3 (A&S). The data shows that the runtime of both heuristics grows in a nonlinear fashion, and much faster than the runtime of the presented heuristic. All three heuristics produce high-quality solutions on these instances (with average gaps for HGSADC 0.19%, ILS 0.53%, A&S 0.57%).

edges. If distance entries between far-away nodes are never retrieved because the corresponding edges will not be taken into consideration, then there is no need to store their value. In short, we can drastically reduce computational complexity and the amount of used memory by asking the question 'Which edges should be considered during runtime?'.

We approach this question by investigating the relative 'closeness' of connected customers in high-quality VRP solutions. Let $r_i(j)$ denote the rank of customer $j$ in the sorted distance list of customer $i$ which contains the distances to all other customers. If $j$ is the closest customer of $i$ we have $r_i(j) = 1$. Given a VRP solution, we determine for each customer $i$ the closeness $o_i = \max(r_i(n_1), r_i(n_2))$ as the maximal 'distance rank' of its two connected neighbors $n_1$ and $n_2$. If one of the neighbors is the depot, we automatically choose the rank of the other neighbor. By looking at the closeness of all nodes in a high-quality solution, we get an intuition about how many neighbors need to be considered. We performed this analysis on various instances of the instance set of Uchoa et al. (2017), computing and then analysing a solution obtained with the heuristic in Arnold and Sörensen (2018b). All of these solutions have a gap of around 1% or less to the best known solutions.

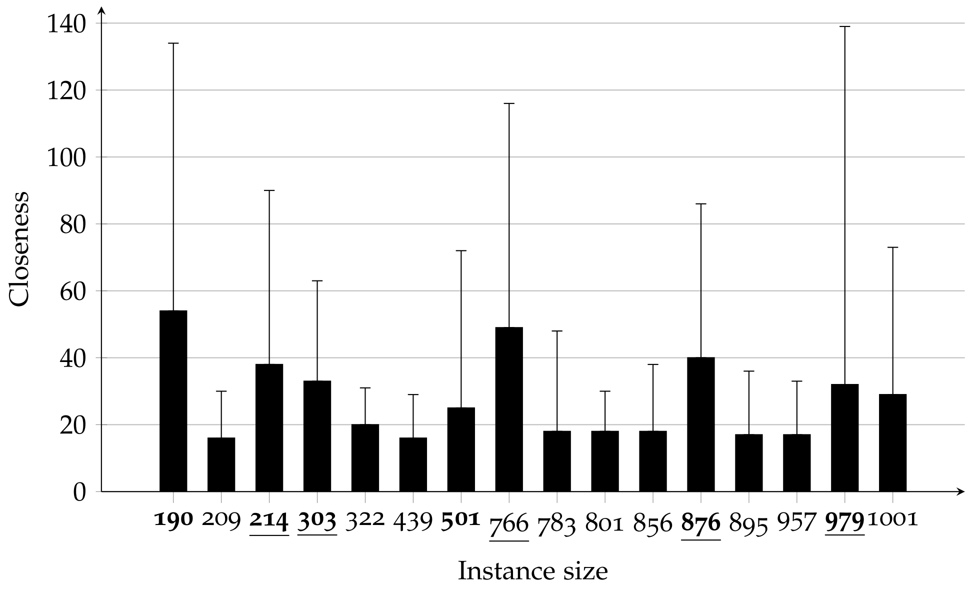


Figure 4.2: Maximal closeness of 95% of the customers (bars), as well as maximal closeness of 99% of the customers (error bars) for different instances. Instances with a high variety in demand (1-100) are underlined, instances with clusters are highlighted in bold.

The results in Figure 4.2 indicate that, for many instances, the connections for 95% of the nodes are with some of the 20 closest neighbors. Furthermore, 99% of the customers have two (or one, if connected to the depot) of their nearest 40 nodes as route neighbors. Only for instances that contain clusters or that have a high variance in demand do we observe a lower closeness. Instances with customer clusters require long edges that lead out of a cluster, and a high variance in demand results in more chaotic routes with longer edges. These findings are consistent for different instance sizes, which demonstrates that we do not necessarily need to consider more neighbors when facing larger instances.

Moreover, these result suggest that, for most nodes in high-quality solutions, it is sufficient to store information only about the nearest neighbors. Of course, there are exceptions. For instance, we have found good solutions in which some nodes had a very poor closeness of up to 300. However, it is reasonable to assume that we can obtain solutions of similar quality if we exclude those edges since (1) they occur very rarely, and (2) they are relatively long, so it is likely that similar or only slightly worse alternatives are available. These observations can be used to adapt the local search to instances of larger size.

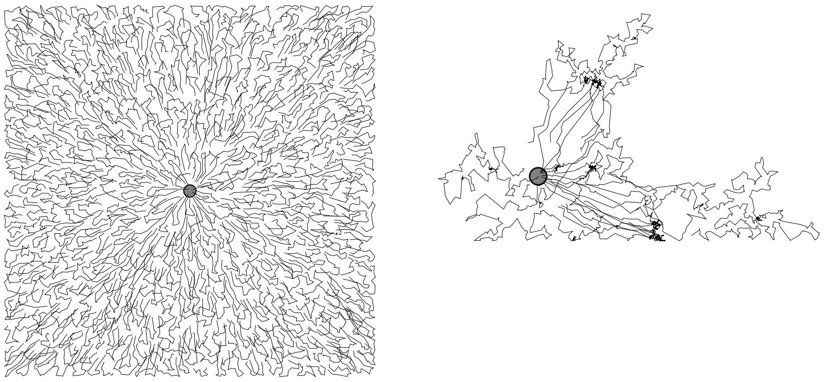


Figure 4.3: Solutions for the very large scale instances introduced in Kytöjoki et al. (2007). Random placement of customers (left) and waste collection (right). For a better visibility, the edges connected to the depot are not displayed.

## 4.3 Effective local search for very large scale VRPs

In the following, we use the observations from the previous section to investigate the functioning of local search for very large scale instances. We hereby hypothesize that an effective local search on larger instances requires a more drastic reduction of the considered neighbourhoods. The size of neighborhoods can be reduced by using less complex local search operators, or by pruning the neighborhoods of those operators more effectively. Finally, we investigate in how far different ways to store information impacts algorithmic performance.

We perform all experiments on the eight instances introduced by Kytöjoki et al. (2007). The first group of instances ($\mathbb{R}$-instances) contains randomly and uniformly distributed customers that are visited from a centrally located depot. Those instances have a size between 3000 and 12,000 customers, and the capacity is defined in such a way that a route visits 15 customers on average, which is more typical for short-haul problems. The second group of instances ($\mathbb{W}$-instances) were extracted from a real-life waste collection application in Eastern Finland. Outgoing from the depot, waste collection trucks need to visit between 7798 and 10,227 specific points to pick up waste bins. The capacity of the trucks is quite large, and one tour can cover 500 and more customers. Therefore, the large number of rather short routes in the $\mathbb{R}$-instances requires a heavy inter-route optimisation, wheres the long routes in the $\mathbb{W}$-instances put a stronger focus on intra-route optimisation. Examples of both instance types are displayed in Figure 4.3.

In both cases, inital solutions are constructed with CW and then improved with the local search. Since the depot is rather far away from the customers on the $\mathbb{W}$-instances, and routes are extremely long, it is crucial to minimize the number of routes during the construction phase. In CW we impose a limit on the number of routes that are constructed, allowing one more route than the (theoretical) minimal number of routes at each point in time. Saving entries that would exceed this limit by creating a new route are ignored but stored for later iterations.

We test different configurations of the heuristic above to solve each instance, allowing up to 15 minutes of computation time. The performance of a certain configuration is expressed as the average gap between the best solutions found within a specific time horizon and the best known results, as reported by Kytöjoki et al. (2007). The heuristic has been implemented in Java and all tests have been performed within the Eclipse development environment on an AMD Ryzen 3 1300X CPU working at 3.5GHz on Windows 10, using a single thread. According to PassMark Software (2018) this setup is about 4 times faster than the setup used in Kytöjoki et al. (2007) (AMD Athlon64 3000+).

### 4.3.1 *Construction of a starting solution*

In Chapter 3 we used the heuristic of Clarke and Wright (1964) to construct a starting solution for a VRP. This heuristic builds routes on the basis of a savings list that contains, for each pair of nodes, the savings that could be obtained when connecting these nodes. As a consequence, the savings list grows quadratically in the instance size, which might render this standard version inefficient for larger problems. An idea to circumvent this quadratic complexity is to only compute the savings between a node and its 100 nearest nodes, thereby linearizing the length of the list and the number of computational steps. Another construction heuristic with a linear number of steps is the famous sweep heuristic. This heuristic transforms the Cartesian coordinates of nodes into polar coordinates (described by angular and radial coordinates), where the depot is the center of the polar coordinate system. The nodes are then sorted according to their angular coordinates, and the node with the smallest angular coordinate (or any other one) is connected to the node with the next larger angular coordinate. This process is iterated until the capacity limit is reached, in which case a new route is started. When all customers are assigned to routes, each route is optimised in itself. In reality, this process resembles districting. Companies typically divide the area of interest into subareas, and each vehicle delivers the customers in a certain subarea.

In the following, we compare the suitability of these three construction heuristics $CW$, $CW^{100}$ and *Sweep* for very large instances. Since *Sweep* requires a post-optimisation step, we apply intra-route optimisation with LK on all constructed starting solutions.

From the results in Table 4.1 we observe that CW is a powerful heuristic for very large instances, and can compute reasonable solutions within seconds. However, the size of the entire savings list can quickly exceed the available heap space (in our case 4GB). This problem can be circumvented with $CW^{100}$, which only considers the

Table 4.1: Results of the construction heuristics on the very large scale instances by Kytöjoki et al. (2007). The gaps are reported in % to the results in the same paper, along with the time $T$ in seconds.

| Instance (N) | *CW* | | | *$CW^{100}$* | | | *Sweep* | | |
|---|---|---|---|---|---|---|---|---|---|
| | Value * | Gap | T | Value | Gap | T | Value | Gap | T |
| W (7,798) | 4,600,457 | 0.89 | 68 | 4,594,582 | 0.76 | 8 | 4,929,615 | 8.11 | 23 |
| E (9,516) | 4,758,438 | 0.02 | 85 | 4,771,803 | 0.30 | 13 | 5,100,387 | 7.21 | 36 |
| S (8,454) | 3,370,304 | 1.10 | 72 | 3,467,753 | 4.02 | 9 | 3,650,841 | 9.51 | 24 |
| M (10,217) | 3,162,326 | -0.27 | 116 | 3,396,401 | 7.11 | 10 | 3,360,857 | 5.99 | 31 |
| R3 (3,000) | 188,650 | 1.30 | 6 | 188,523 | 1.24 | 1 | 232,212 | 24.70 | 3 |
| R6 (6,000) | 355,361 | 0.75 | 23 | 355,481 | 0.79 | 1 | 462,084 | 31.01 | 3 |
| R9 (9,000) | 521,311 | 0.75 | 60 | 521,646 | 0.81 | 1 | 696,312 | 34.57 | 3 |
| R12 (12,000) | - | - | - | 685,975 | 0.76 | 4 | 931,042 | 36.75 | 3 |

* For some instances, we could not compute a solution with a heap space of 4GB.

savings between a node and its 100 nearest neighbours. This version also speeds up the construction process while the solution quality is only marginally affected for most instances. On the other hand, it appears that *Sweep* struggles to compute good starting solutions for the $\mathbb{R}$-instances with many short routes. On those instances, the computed routes resemble long lines originating from the depot, and almost every route visits a customer that is far away from the depot. For those instance types, we can certainly find better districting strategies, while on instances with long routes *Sweep* performs quite well. Overall, $CW^{100}$ appears to be well-scaling construction heuristic for very large instances, and we will use it in the following.

## 4.3.2 *Complexity of operators*

We have shown that a local search composed of LK, CE and RC functions effectively for instances of up to 1001 customers. However, the inherit complexity of these operators might be too large if facing instances of greater magnitude. This raises the question whether the local search needs to be adapted to successfully solve very-large scale instances.

LK has an excellent scaling performance, and is used as the basis to tackle large scale TSPs of tens of thousands of customers (see, for instance, (Applegate et al., 2003) and (Helsgaun, 2000)). In the context of the VRP multiple routes have to be planned, and if these routes are longer, the corresponding TSPs become larger in size. On the other hand, given a fixed route length, the consideration of more customers results in more routes and thus, more TSPs of the same size. Since LK exhibits a non-linear time complexity, the scaling of LK in the context of VRPs is determined by the length of routes. An important lever to adjust the runtime behavior of LK is the maximum number $\lambda$ of edge exchanges per move.

Table 4.2: Local search operators used in the different setups.

| | Intra-route LS | Inter-route LS |
|---|---|---|
| $LS_1$ | LK ($\lambda = 5$) | CE |
| $LS_2$ | LK ($\lambda = 5$) | CE, RC ($\rho = 2$) |
| $LS_3$ | LK ($\lambda = 5$) | CE, RC ($\rho = 3$) |
| $LS_4$ | LK ($\lambda = 4$) | CE, RC ($\rho = 3$) |

CE exchanges substrings in two different routes. Similarly to above, more combination of substrings have to be considered if routes increase in length. This complexity can be tackled by imposing a limit on the length of considered substrings, as suggested by Taillard et al. (1997). We found that the preceding sequential search already sufficiently reduces the neighborhood, and a restriction on the length of substrings does not yield performance gains in our implementation of CE.

Finally, we observed that RC requires the most runtime of the three operators, and the number of considered moves grows rapidly in the number of subsequent relocations. Therefore, it is crucial to define an upper bound $\rho$ for the number of combined relocations. For smaller instances we found $\rho = 3$ to be a good choice.

On the basis of these ideas, we compare the performances of different local search configurations, as shown in Table 4.2. $LS_3$ generates the largest neighborhoods and corresponds to the local search configuration that we used to tackle instances of up to 1000 customers. We use this configuration as the baseline to analyse potential alternatives. For the $\mathbb{R}$-instances it might be beneficial to reduce the complexity of intra-route operators and especially that of the computational demanding RC operator. We consider a configuration with shorter relocation chains ($LS_2$), and a configuration without this operator ($LS_1$). To tackle the long routes in the $\mathbb{W}$-instances we test a configuration with a less complex LK operator ($LS_4$). In all configurations we keep the degree of inter-route pruning ($C = 30$) as well as the memory usage (for each node, we store the distances to the 100 nearest nodes and compute all other distances when needed) fixed.

The experimental results in Figure 4.4 demonstrate that the more complex local search operators work surprisingly well on both benchmark sets, and each configuration improves upon the best known solution of each instance within a few minutes of computation time. On the $\mathbb{R}$-instances, we observe that the configurations with a less heavy inter-route optimisation $LS_1$ and $LS_2$ perform well within a short time horizon. Given more computation time, the configurations with larger neighborhoods perform better, with $LS_4$ computing the best results within 15 minutes. This setup is also by far the best configuration on the $\mathbb{W}$-instances. The reduction of intra-route optimisation appears to be especially important on instances with extremely long routes, otherwise too little runtime remains for inter-route operations. We can conclude that less complex local search operators that generate smaller neighborhoods are beneficial

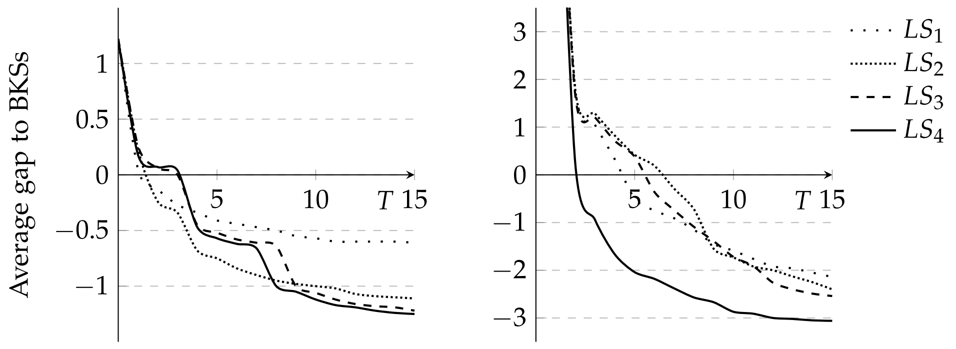


Figure 4.4: Average gap (in %) to the best known solutions (BKSs) over time ($T$ in minutes) on the $\mathbb{R}$-instances (left) and $\mathbb{W}$-instances (right) for different local search configurations.

to solve very large instances in a short time, while complex local search operators, if well-implemented, have more potential to improve solutions in the longer run. Overall, care should be taken to not dedicate too much computation time for intra-route optimisation, especially if routes visit many customers.

### 4.3.3 *Pruning*

The idea behind heuristic pruning is to limit the explored neighborhoods of the local search to those moves that are promising. Thus, we should expect lower runtimes per iteration if we apply more restrictive pruning strategies. However, at the same time the local search will explore less possible moves, which might prevent it from finding improvements. We should therefore observe a trade-off between runtime and solution quality. If the pruning is too restrictive, the local search is fast but potentially fails to find improving moves, and if the pruning is too loose, the local search might investigate too many alternatives at the cost of significantly higher computation times.

In the following experiments we investigate this trade-off, using the best configurations $LS_4$ from above. The inter-route operators CE and RC only consider moves in which a node is connected to one of its $C$ closest nodes. We investigate the effect of pruning on performance by varying $C \in \{10, 20, 30, 50\}$. From our observations in Figure 4.2 we would expect that a more restrictive pruning can be beneficial, since edges in high-quality solutions are typically between spatially close nodes.

The results in Figure 4.5 reveal that the degree of pruning affects the performance of the heuristic. On the $\mathbb{R}$-instances, a restrictive pruning ($C = 10$) appears to work well in the short run, while a looser pruning ($C = 30$) yields better results after about 15 minutes of computation time. These observations can be seen analogously to the ones in the previous section, since a more restrictive pruning corresponds to smaller

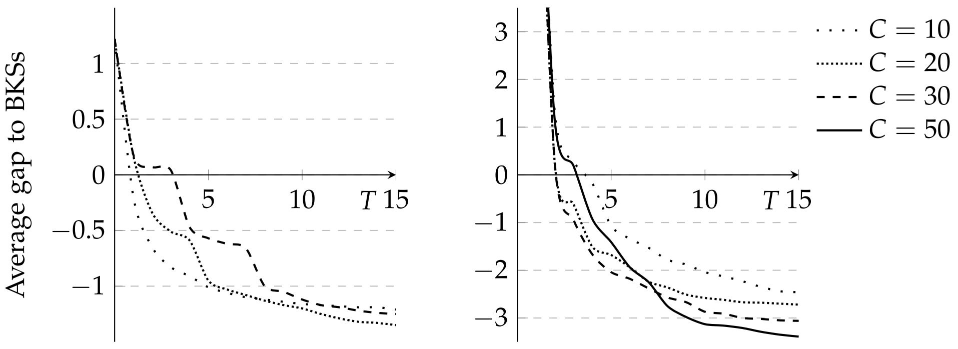


Figure 4.5: Average gap (in %) to the best known solutions (BKSs) over time ($T$ in minutes) on the $\mathbb{R}$-instances (left) and $\mathbb{W}$-instances (right) for different pruning strategies. For the larger $\mathbb{R}$-instances, the inital optimisation for $C = 50$ requires too much time and we do not report the corresponding results.

neighborhoods. $C = 20$ appears to be a good compromise and computes the best solutions after a few minutes. These results demonstrates that a tighter pruning can be beneficial for very large instances (for instances of up to 1000 customers we obtained the best results with $C = 30$). The gain in computation speed through a tighter pruning seems to compensate for the reduction in neighborhood size, so that more iterations can be executed in the same amount of computing time.

On the other hand, for the $\mathbb{W}$-instances a looser inter-route pruning ($C = 50$) appears to be the best choice in the long run. As seen above, long routes put more emphasis on intra-route optimisation, and a looser inter-route pruning establishes a better balance between both optimisation tasks. Additionally, with longer routes, many of a customer's nearest neighbors are likely to be in the same route and cannot be used as the starting point for inter-route moves.

### 4.3.4 *Memory usage*

Finally, we investigate the effect of limiting the amount of stored information. We have shown in Figure 4.2 that most customers are connected to relatively close neighbors in high-quality solutions and, thus, it might be sufficient to only store the distance information between close customers. More precisely, we only store the distance between a customer and its $S$ closest nodes as well as the distance to the depot. The distance between any other pair of customers outside of this range is automatically set to infinity. Since none of the local search operators will consider a move that involves such an edge, we thereby indirectly restrict the set of edges that will be taken into consideration. Note that the determination of the $S$ nearest neighbours per customer requires the computation and sorting of possibly

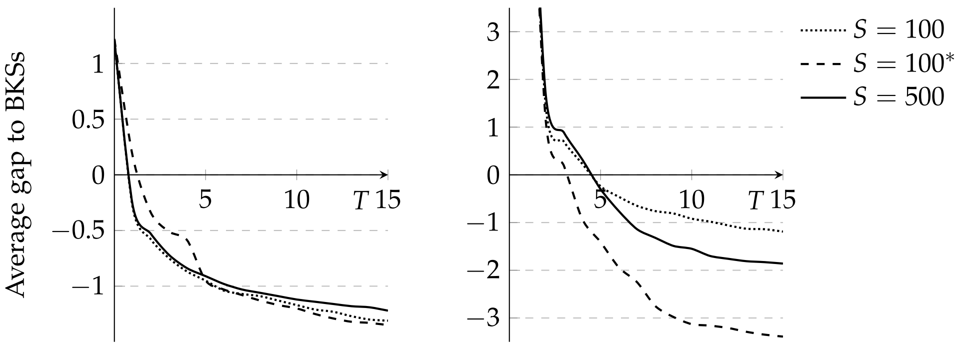


Figure 4.6: Average gap (in %) to the best known solutions (BKSs) over time ($T$ in minutes) on the $\mathbb{R}$-instances (left) and $\mathbb{W}$-instances (right) for memory usage strategies.

all distance values. We then store the tuples $< neighbour, distance\_value >$ for the $S$ nearest neighbours per customer node in a Hash-Map.

We compare the performance for $S \in \{100, 500\}$, keeping the pruning $C = 20$ ($\mathbb{R}$) and $C = 50$ ($\mathbb{W}$), and the best local search configuration $LS_4$ fixed. Additionally, we also investigate a setup marked as $100^*$, in which we store the 100 shortest distances prior to the start of the heuristic, but compute all other requested distance values during runtime (Helsgaun, 2000). In other words, in this final strategy distances between further-away nodes are not set to infinity, but computed on request only. For $S = 500$ our implementation already uses up to 2.8GB of RAM during runtime for the largest instance with 12,000 customers.

We observe from the results in Figure 4.6 that the amount of stored information has only a marginal effect on performance on the $\mathbb{R}$-instances; in other words, the storage of more information, beyond a certain point, does not seem to benefit performance. The two configurations $S = 100$ and $S = 500$ exhibit almost the same performance curves and are slightly better than $S = 100^*$ within the first minutes of runtime. Since the first two configurations do not consider any edges despite those being pre-processed, they prune the search space more drastically. The conclusion is the same as in the previous experiments; a more drastic pruning is beneficial for short computation times, while more information and larger neighborhoods enrich the local search for longer time horizons. In contrast, on the $\mathbb{W}$-instances the consideration of very long edges appears to be crucial to obtain high-quality results. In summary, the storage of often-requested information about short edges, complemented with a computation of the remaining edge information on request appears to be a good approach to tackle the memory challenges of very large instances.

### 4.3.5 *An effective configuration for very large instances*

From the above findings we can extract the following local search heuristic for very large instances. The starting solution is constructed with CW, by only considering savings between a customer and its 100 nearest nodes. Memory usage is reduced by storing those distance values between a customer and its 100 nearest neighbours, while all other distance values are computed on request during runtime. The local search configuration $LS_4$ with relatively complex inter-route operators, and a slightly less complex intra-route operator is effective for very large instances. Those operators span relatively large neighborhoods, and need to be pruned. A tighter pruning ($C = 20$) appears to be generally more effective for VRP instances with many routes, while for instances with very long routes a looser pruning ($C = 50$) is beneficial.

Table 4.3: Results of the presented heuristic on the very large scale instances by Kytöjoki et al. (2007). The gaps are reported in % to the results in the same paper, given different computation times ($T$ in minutes).

| Instance (N) | GVNS | | | A-G-S (short runtime) | | | A-G-S (long runtime) | | |
|---|---|---|---|---|---|---|---|---|---|
| | Value | Gap | T* | Value | Gap | T | Value | Gap | T |
| W (7,798) | 4,559,986 | 0.00 | 34.5 | 4,321,319 | -5.23 | 8.6 | 4,258,811 | -6.60 | 39.0 |
| E (9,516) | 4,757,566 | 0.00 | 83.9 | 4,710,134 | -1.00 | 21.0 | 4,691,189 | -1.40 | 47.5 |
| S (8,454) | 3,333,696 | 0.00 | 56.2 | 3,223,007 | -3.32 | 14.1 | 3,153,025 | -5.42 | 42.5 |
| M (10,217) | 3,170,932 | 0.00 | 77.6 | 3,055,020 | -3.66 | 19.4 | 3,038,053 | -4.19 | 51.0 |
| R3 (3,000) | 186,220 | 0.00 | 4.8 | 183,249 | -1.60 | 1.2 | 182,318 | -2.10 | 15.0 |
| R6 (6,000) | 352,702 | 0.00 | 24.4 | 348,204 | -1.28 | 6.1 | 347,708 | -1.42 | 30.0 |
| R9 (9,000) | 517,443 | 0.00 | 57.7 | 511,924 | -1.07 | 14.4 | 511,831 | -1.08 | 45.0 |
| R12 (12,000) | 680,833 | 0.00 | 108.4 | 673,542 | -1.07 | 27.1 | 673,268 | -1.11 | 60.0 |
| Average | | 0.00 | 55.8 | | -2.28 | 14.0 | | -2.91 | 41.3 |

* Based onPassMark Software (2018), this machine is about 4 times slower.

With this setup, we compute solution for the $\mathbb{R}$-instances and the $\mathbb{W}$-instances, and compare the performance to the results of the guided variable neighborhood search (GVNS) introduced by Kytöjoki et al. (2007). To get a deeper insight into the trade-off between computation time and solution quality, we provide results after differing running times of the heuristic. For the 'faster' runtime configuration we allow the same computation time than in the previous paper, given that our setup is about 4 times faster. For the 'longer' runtime configuration we allow 5 minutes per 1,000 customers.

From the results in Table 4.3 we can conclude that the presented heuristic outperforms the GVNS by a significant margin and computes high-quality, and so far best known, solutions for instances of 10,000 and more customers in less than an hour. These results suggest that the implemented local search works effectively for very large instances. To gain further insights into the nature of this effectiveness, we investigated how many local search moves are evaluated and executed during runtime. A move is defined to be 'evaluated' if all involved costs computations have been completed. Note that due to sequential search and preceding feasibility checks even more moves might be evaluated partially. We conduct this analysis for two instances of Uchoa et al. (2017) as well as R3 and R9. All of these instances have a similar structure with randomly distributed customers being delivered from a central depots on routes that visit around 15 customers. However, the instances vary in size, and we investigate in how far size impacts the behavior of the local search.

Table 4.4: Performance statistics for each local search operator on four instances of different size $N$. The number of evaluated moves per second Ev./s, the number of executed moves per second Ex./s, and the proportion of runtime required by this operator %T.

| N | LK | | | CE | | | RC | | |
|---|---|---|---|---|---|---|---|---|---|
| | Ev./s | Ex./s | %T | Ev./s | Ex./s | %T | Ev./s | Ex./s | %T |
| 137 | 1,276,388 | 1224 | 18% | 724,152 | 2219 | 30% | 11,552,304 | 1221 | 52% |
| 732 | 535,509 | 10,297 | 2% | 153,388 | 1640 | 21% | 3,333,494 | 328 | 77% |
| 3000 | 1,240,171 | 567 | 6% | 229,582 | 137 | 40% | 8,266,432 | 127 | 54% |
| 9000 | 1,058,961 | 535 | 1% | 259,481 | 16 | 42% | 10,886,467 | 13 | 57% |

Table 4.4 displays the average number of evaluated and executed moves per second, as well as the overall proportion of time that is required by each operator. We observe that the number of evaluated moves does not seem to depend on instance size. On an instance with 9000 customers the local search evaluates almost as many moves as on an instance with 137 customers. This demonstrates the effectiveness of all above mechanisms which enable the heuristic to work equally fast for instances of different magnitude. The heuristic performs about 10 millions RC evaluations per second, 1 million evaluations per second with LK, and a few hundred thousand evaluations with CE (notably, LK requires the least time out of all operators on those instances, since routes tend to be rather short). On the other hand, we observe that the number of executed moves per second decrease with larger instances, especially for the two inter-route operators. This observation can be attributed to the larger neighborhoods that occur in larger instances. In larger neighborhoods more moves have to be evaluated, so that overall less neighborhoods can be analysed, and less moves performed within the same time horizon. In conclusion, the ratio of executed moves to evaluated moves decreases with growing instance size, which is the main reason for a higher computational effort.

## 4.4 New instances

Most popular benchmark sets for the VRP comprise instances of several hundred customers, and a few include instances with up to 1,000 customers. As a result, newly developed heuristics are exclusively tuned to tackle instances of this size, whereas problems of higher magnitudes are usually entirely neglected. The instances by Kytöjoki et al. (2007) are the only very large scale ones. Given that the instances are a decade old, it is perhaps surprising that our paper is only the second one to tackle them. An explanation could be that, in the race for better performances,

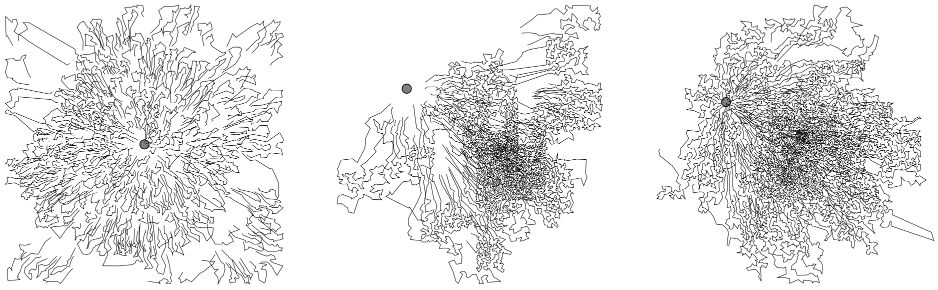


Figure 4.7: Solutions for the new instances based on parcel distribution in Belgium: Antwerp (central depot), Ghent (edge depot), and Brussels (edge depot). The edges connected to the depot are not displayed.

researchers have resorted to instances that are well-explored and that enable a more accurate comparison. An additional explanation is that, as we have demonstrated, very large scale problems pose additional challenges that make it more difficult to develop suitable heuristics.

To foster research in the area of very large scale vehicle routing we develop an additional set of very large scale and realistic benchmark instances. These instances are based on parcel distribution in Belgium. Logistic service providers face the daily task to deliver parcels via vans or trucks from a distribution center to their customers' doorsteps. With the rise of e-commerce in recent years, the number of delivered parcels has grown at a steady pace. In a relatively large city like Brussels (about 1.2 million inhabitants) more than 20,000 parcels have to be delivered every day. With parcel-delivery being a mostly cost-driven business, it is crucial to optimize this distribution. Having fewer delivery vans and fewer delivery kilometers can decrease significantly the costs of service providers, but it also reduces emissions and congestion and thereby increases the quality of life in cities.

On the basis of a dataset containing deliveries in the Flemish and Brussels regions of Belgium, we were able to extract the density of parcel demand for different zones, i.e., the number of delivered parcels per $km^2$. We use this demand density to sample the location of customers for one delivery day for the cities of Leuven, Antwerp, Ghent and Brussels as well as for the whole region of Flanders. For each of these geographic areas we fix the number of customers and locate them according to demand density. Each customer is assigned a random demand of either 1 (50%), 2 (30%) or 3 (20%) parcels. Furthermore, we consider two delivery scenarios for each of these five regions.

In the first scenario, the customers are delivered by vans from a distribution center outside of the city, where the capacity of a van is, depending on the instance, set to 100-200 parcels. In the second scenario, the distribution center is located within the city, and customers are delivered from there via cargo-bikes or electric tricycles that

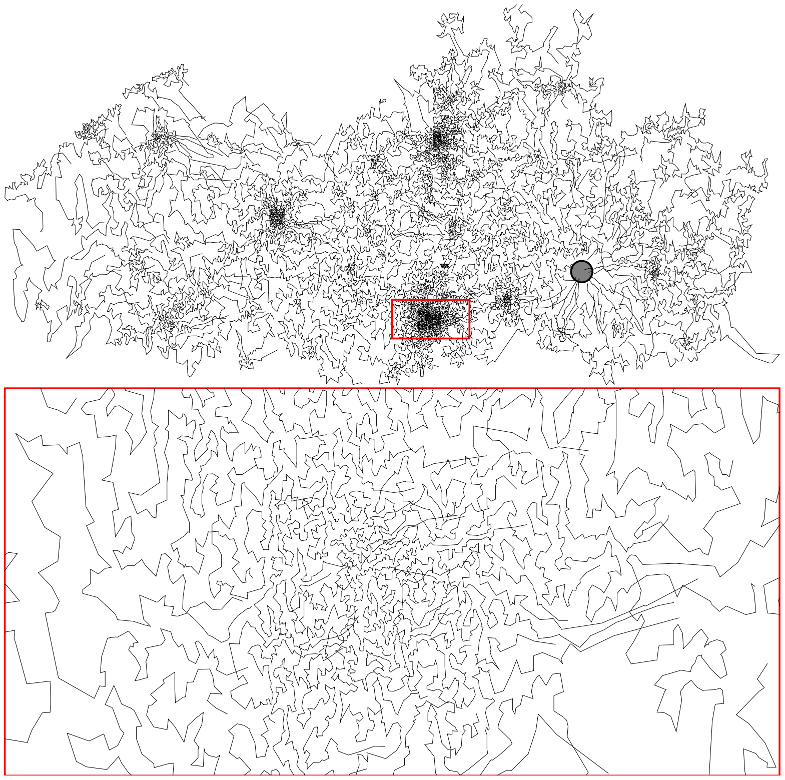


Figure 4.8: A solution to distribute parcels within the whole region of Flanders (Instance F2 - 30,000 customers), with a zoom-in on Brussels.

can carry 20-50 parcels at a time. Thus, the routes in the first case are rather long (long-haul) and in the second case rather short (short-haul). We obtain 10 instances that are diverse with respect to customer placement (more clustered versus more evenly distributed), depot location (center versus fringe), and route length (many customers versus few customers). All distances are Euclidean and rounded to the nearest integer.

These instances complement the $\mathbb{W}$ and $\mathbb{R}$-instances in several ways: (1) The instances have a greater variety and also include instances with more clustered customers that are delivered from a central depot, (2) they comprise instances of up to 30,000 customers, and (3) they have more moderate route lengths which are typical for parcel distribution. The new instances are, along with an explanation of the format, publicly available at `http://antor.uantwerpen.be/xxlrouting`, and examples are displayed in Figure 4.7 and Figure 4.8.

For each of the new instances we compute solutions to serve as a first benchmark, given 3 minutes of computation time per 1000 customers. We use the best configu-

Table 4.5: Results on the new very large scale instances for A-G-S and LKH-3 for different computation times $T$. The depot location ($D$ is either in the center (C) or at the edge (E)), and the average route length (L) vary across instances.

| Instance (N) | D | L | CW | | | A-G-S | | | LKH-3 | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | Value | Gap | T | Value | Gap | T | Value | Gap | T* |
| L1 (3,000) | C | 14.8 | 200,971 | 3.21 | 0.1 | 194,714 | 0.00 | 15.0 | 194,381 | -0.17 | - |
| L2 (4,000) | E | 85.1 | 125,613 | 8,68 | 0.1 | 115,577 | 0.00 | 20.0 | 113,484 | -1.81 | - |
| A1 (6,000) | C | 17.5 | 498,422 | 3.42 | 0.1 | 481,918 | 0.00 | 30.0 | 481,338 | -0.12 | - |
| A2 (7,000) | E | 58.3 | 322,902 | 7.63 | 0.1 | 300,023 | 0.00 | 35.0 | 297,478 | -0.85 | - |
| G1 (10,000) | C | 20.6 | 490,783 | 3.46 | 0.1 | 474,388 | 0.00 | 50.0 | 474,164 | -0.05 | - |
| G2 (11,000) | E | 100 | 287,371 | 7.33 | 0.1 | 267,751 | 0.00 | 55.0 | 265,763 | -0.74 | - |
| B1 (15,000) | C | 29.3 | 532,558 | 4.34 | 0.1 | 510,416 | 0.00 | 75.0 | 509,457 | -0.19 | - |
| B2 (16,000) | E | 87.9 | 386,048 | 6.71 | 0.1 | 361,762 | 0.00 | 80.0 | 357,382 | -1.21 | - |
| F1 (20,000) | C | 29.2 | 7,525,575 | 2.71 | 0.1 | 7,326,975 | 0.00 | 100.0 | 7,300,772 | -0.36 | - |
| F2 (30,000) | E | 117.2 | 4,805,608 | 5.24 | 0.1 | 4,566,287 | 0.00 | 150.0 | 4,499,422 | -1.46 | - |
| Average | | | | 5.27 | 0.1 | | 0.00 | 61.0 | | -0.70 | - |

* Results were obtained after several days up to a month of computation time, the exact times could not be retrieved.

ration derived from Section 4: $LS_4$ as local search, $C = 20$ for intra-route pruning and $S = 100^*$ to handle memory. The results are displayed in Table 4.5. We observe that a well-implemented CW algorithm with a pruned savings-list computes reasonable solutions almost instantaneously. Given a few minutes up to a few hours of computation time, those initial solutions can be improved by about 5% with local search.

We further obtained results from Helsgaun (2017) (LKH-3). This heuristic is an extension of a heuristic for the TSP (Helsgaun, 2000) and transforms a VRP into a standard TSP, while checking for constraint violations after each $\lambda$-opt move. Whereas a move is performed in $O(\sqrt{n})$ time, a constraint violation check takes $O(n)$ time. The heuristic focuses on solution quality rather than speed and, thus, the results were obtained after several days up to a month of computation time. LKH-3 produces high-quality results for the new instances, which suggests that heuristics that work well for TSPs can also be adapted to effectively solve VRPs, especially if variation in demand is low (between 1 and 3) and routes visit many customers. On instances with shorter routes (L1, A1, G1, B1, and F1) we observed that A-G-S reaches the same solutions as LKH-3 if we allow five to ten times more time, while the same could not be achieved for the respective instances with longer routes. LKH-3 also computed excellent solutions for the $\mathbb{W}$-instances with very long routes which were more than 2% better than the one reported above. While A-G-S provides good benchmark results for heuristics that focus on performance over time, LKH-3 provides a good benchmark for overall solution quality.

## 4.5 Conclusion

In this work we have developed a heuristic that is able to solve VRP instances of several thousands and even tens of thousand of customers within a feasible amount of computing time. We outlined the main challenges of very large scale problems — reduction of time and space complexity — and demonstrated how to tackle them in the context of the VRP using heuristic pruning and a limited information storage. These ideas were embedded in an already very efficient heuristic that uses well-implemented local search operators as its key driver. We further investigated the effect of these pruning techniques, and demonstrated the advantages of a stricter pruning in the context of very large scale problems. The resulting heuristic significantly improves the results of a previous heuristic in relatively short computation times. Finally, we used real-world data to generate a set of very-large scale instances that reflect problems in parcel distribution to promote more research in this domain.

# 5 Integrating inventory levels in routing problems

## 5.1 Introduction

After more than 50 years of research, the original VRP has evolved into a diverse collection of variants that attempt to better model the transportation scenarios found in the real world. These variants consider elements that range from characteristics of the fleet of vehicles (i.e., capacity constraints, homogeneous or heterogeneous vehicles, loading policies, driving regulations), to different types of transport requests (i.e., pickup or delivery), storage facilities (i.e., single-depot or multi-depot), planning horizons (i.e., single- or multi-period) and operating environments (i.e., static, dynamic, deterministic or stochastic). Such extended problems are called Rich Vehicle Routing Problems, and have been thoroughly surveyed by Laporte (2009), Lahyani et al. (2015) and Braekers et al. (2016).

Almost all variants of the VRP share the common assumption that inventory levels at the depot(s) are infinite or, at the very least, sufficiently available. This assumption stems from both practical and technical reasons. In many supply-chain environments, transportation planning involves daily decisions at an operational level, while inventory control requires tactical decisions that span over multiple days or weeks (Min and Zhou, 2002). Both activities involve, by themselves, solving complicated optimisation problems. Therefore, the traditional approach has been to tackle these problems sequentially and avoid any compounding complexity that would arise from their integration in a single optimisation problem. As pointed out by Andersson et al. (2010), in the vast majority of industrial scenarios, systems for inventory control produce order quantities and schedules, which are then used as input for transportation-planning tools.

Despite its prevalence, the sequential approach has substantial room for efficiency gains. It has been shown that, even in relatively simple supply-chain environments (involving, for instance, a single depot and a single product), there is a strong

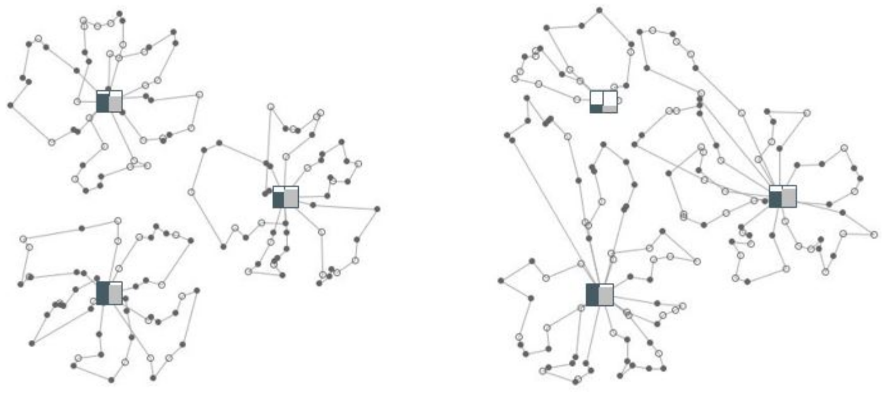


Figure 5.1: An example of how inventory levels can impact solutions for routing problems. The demand of the customers is fulfilled from three different depots. Each customer either demands product 1 (black) or product 2 (grey). In one case (left), all depots have sufficient inventory to supply all nearby customers, in the second case (right), the upper depot has run out of inventory and nearby customers need to be served from another depot. The latter situation might result in a more expensive routing.

interdependence between inventory control and transportation planning, see, for example, Federgruen and Zipkin (1984) and Sarmiento and Nagi (1999). This relationship has been the main motivation behind the Inventory Routing Problem (IRP) which combines vendor-managed inventory and transportation decisions (Coelho et al., 2013).

The study of these interdependencies between inventory control and routing is becoming more and more important for practical applications. The growth of e-commerce forces logistic service providers to increase the efficiency and flexibility of their delivery activities, while facing a higher degree of disintermediation (Delfmann et al., 2002). At the same time, mail order companies or large retailers have multiple depots to provide faster deliveries to their customers. In some cases, an order involving multiple products might even be split, and fulfilled by two or more depots. Systems like this, despite offering a major degree of flexibility, make inventory control even more critical. An inadequate inventory allocation that requires orders to be fulfilled by depots located far away from the customers can lead to substantial increases in the overall transportation cost. On the other hand, stocking too much inventory can result in significant inventory costs, considering that approximately 33 % of logistics costs can be attributed to holding inventory (Wilson, 2006). An example of this trade-off between inventory levels and delivery routes is visualized in Figure 5.1.

In this chapter, we explore the benefits of integrating the decisions for inventory and routing in logistic systems with multiple depots. As a first contribution, we propose a routing model that captures inventory constraints:

- Multiple products are distinguished and stored in multiple depots.
- Customers can only be served from a depot, if the demanded products are stored and available in sufficient quantity.

To solve the resulting routing problem we develop an effective heuristic. The heuristic optimises the delivery routes of products from depots to customers given certain inventory levels in the depots, rather than optimising the inventory levels or their allocation themselves. As a second contribution, we use the developed heuristic to investigate the effects of inventory levels on routing costs. This study follows the idea of Salhi and Rand (1989) in which the authors quantified the effect of ignoring transportation planning in strategic facility location decisions. We carry out an extensive simulation study involving a diverse range of scenarios with varying characteristics, such as the number of depots, the number of products, how clustered the customers are around the depots, and how much safety stock there is in each depot. For each scenario, we compare the transportation cost resulting from an "ideal" inventory allocation, i.e., that in which all the customers can be served by the closest depots, to those resulting from "non-optimal" allocations. By aggregating this data, we quantify to which extent each characteristic contributes to an increase in routing costs. Our ultimate goal is to provide practical insights that can help to mitigate the effects of inadequately allocating inventory in such complex environments.

The chapter is structured as follows. In Section 2 we introduce the planning problem which we call multi-product multi-depot vehicle routing problem with inventory restrictions (MPMDVRPI). In Section 3 we present an effective heuristic to solve the problem within reasonable computing times. The experimental setup of the simulation study is outlined in Section 4. Section 5 discusses the most important findings and Section 6 summarizes our contributions.

## 5.2 Integration of inventory constraints in routing problems

We extend the basic MDVRP (Montoya-Torres et al., 2015) by introducing inventory levels at the depots and distinguishing between different products. The resulting problem is called the multi-product multi-depot vehicle routing problem with inventory restrictions (MPMDVRPI). The MPMDVRPI is defined as follows: given a set of customers with known demands for several products, given several depots, each with their stock levels defined, find the lowest-cost routes starting and ending at one of the depots for a fleet of vehicles such that all demands are satisfied, the capacities of the vehicles are not exceeded, and the total demand for each product served from each depot does not exceed the inventory of that product at the depot. In contrast to the IRP (Moin and Salhi, 2007), the MPMDVRPI considers the inventory levels at the supplier's depot, rather than those at the customers. It could also be defined as a capacitated VRP with one capacity per product. The integration of inventory levels and the consideration of different products in the MDVRP is modelled by additional constraints.

More formally, let us consider a routing problem with $N$ customers. Each customer $i \in \{1, \ldots, N\}$ has a demand $d_{ip}$ for each product $p \in \{1, \ldots, P\}$, which might potentially be 0. The demand can be supplied from any depot $j \in \{1, \ldots, M\}$, but only if depot $j$ has sufficient inventory level $I_{jp}$ to cover the customer's demand for product $p$. If a customer orders two products $p_1$ and $p_2$, and no depot has sufficient inventory to deliver both products simultaneously, the customer needs to be supplied by two different depots, e.g., $p_1$ is delivered by depot $d_1$ and $p_2$ by $d_2$. The possibility of visiting a customer more than once is modelled by allocating a customer's product demand to a depot, rather than the customer itself. More specifically, let $x_{ijp}$ denote whether the demand of customer $i$ for product $p$ is satisfied by depot $j$, then

$$\sum_{i=1}^{N} d_{ip} x_{ijp} \leq I_{jp} \qquad \forall j = 1 \ldots M, \ \ p = 1 \ldots P. \tag{5.1}$$

Even though a customer's demand for different products can be fulfilled on different routes from different depots, we assume that the demand for one specific product is satisfied on one route and cannot be split. Additionally, we require that the inventory levels are sufficient to satisfy all demand. In other words, there exists a feasible solution in which all demand is met.

As in the MDVRP, the objective is to minimize the total distance traveled. Each vehicle starts and ends its delivery route at the same depot, has a certain capacity limit and can transport different products on the same route. The complete mixed-integer programming problem can be found in the appendix.

## 5.3 Design of a heuristic

The MPMDVRPI is a special case of the MDVRP, and, thus, it is an NP-hard problem for which commercial MIP solvers can find the optimal solution only for small instances within a reasonable time. In the last decades, heuristics have been proven to be a more tractable approach to obtain high-quality solutions for instances of realistic size (Laporte, 2009). Therefore, we introduce a heuristic which can solve MPMDVRPI instances effectively. The heuristic extends our heuristic introduced in Chapter 3.

**Algorithm 3** Knowlege-guided local search heuristic

```
1:  Construction. Allocate each customer to the nearest depot and construct a starting
    solution with the heuristic by Clarke and Wright (1964), for each depot separately.
    Optimise the individual routes with LK.
2:  Initial optimisation. Apply CE and RC on starting solution. Whenever a route is
    changed, re-optimise it with LK. Set $b(\cdot) = b^{w}(\cdot)$.
3:  while time limit not reached do
4:      Perturbation:
5:      while not $P = 30$ moves have been made do
6:          Penalize edge $(i,j)$ with the highest value $b(i,j)$ by incrementing $p(i,j)$.
7:          Apply CE and RC on $(i,j)$, using $c^{g}(\cdot)$ as evaluation criterion.
8:      end while
9:      Optimisation. Change $b(\cdot)$. Apply LK, and then iteratively CE and RC on all
10:     routes that were changed during perturbation ($c(\cdot)$ as evaluation criterion).
11:     Whenever a route is changed with CE or RC, re-optimise it with LK.
12: end while
```

We integrate inventory levels and product distinction into the heuristic by slightly adapting the construction mechanism as well as the local search operators. Initially, multiple products are distinguished by assigning a demand value for each customer and product. If a customer has demand for more than one product, it might be served by two different depots as noted above. We handle this special case by defining a new virtual customer for each additional ordered product, e.g. if a customer demands two products, we split it into two customers (with the same location) that both order one product. Note that even though this approach can be implemented without any effort, it increases the number of customers and, therefore, the complexity of the problem. In summary, we transform the given MPMDVRPI into a MPMDVRPI in which each customer has demand for exactly one product.

We then generate a feasible starting solution in a greedy fashion, by firstly allocating customers to depots and then computing initial routes for each depot. Initially, the remaining inventory in all depots for all products is set to the given levels. For each customer we then compute the difference between the distances to the second-closest and the closest depot, and store these potential savings in a list. We sort the list in descending order, and, starting from the top, iteratively try to allocate a customer to its closest depot. If the closest depot still has sufficient inventory to deliver the respective customer, we assign the customer to the depot and update the remaining inventory in the depot. Otherwise, we recompute the customer's saving as the difference in the distances between the two closest depots with sufficient inventory, and reinsert it into the list. In this way we allocate each customer to a depot. This regret heuristic minimizes the opportunity cost that are incurred when the closest depot does not have sufficient inventory to serve the customer. Finally, we construct a VRP solution for each depot and its assigned customers with the heuristic by Clarke and Wright (1964).

Table 5.1: Average gap in % for solutions for MPMDVRPI instances to the corresponding solutions for MDVRP instances. The inventory constraints are defined in such a way that similar solutions are possible.

| Number of depots | 1 product | 2 products | 3 products |
|---|---|---|---|
| 2 | 0.16 | 0.17 | 0.41 |
| 3 | 0.24 | 0.26 | 0.59 |
| 4 | 0.51 | 0.68 | 1.39 |

The subsequent local search operators CE and RC keep the solution feasible with respect to the inventory constraints. We keep track of the remaining inventory at each depot, and only allow local search moves that do not violate these constraints. These feasibility checks can be implemented without much effort. In short, the heuristic never generates an infeasible solution at any stage of the search. Also note that inventory constraints can only be violated by a local search move which changes two or more routes from different depots simultaneously. The penalization of edges does not directly result in any changes of the routing solution and, thus, does not require any adaptation. In summary, there are only few changes necessary to incorporate inventory constraints into the MDVRP heuristic.

Even though we have shown that the heuristic works well for standard VRP and MDVRP instances, this does not automatically imply that it also computes high-quality solutions if inventory levels are restricted. Since there are no benchmark instances for the MPMDVRPI available, we approach this question in the following way. We construct MDVRP instances according to the description in Section 4 (100 customers are uniformly distributed in a circular area around a specified number of depots) and generate a solution without inventory constraints. In this MDVRP solution, each customer is served by one depot, and on the basis of this assignment we can compute how much of each product is delivered by each depot. We then construct the corresponding MPMDVRPI where the inventory levels in the depots equal the delivered demand in the MDVRP solution (adding 1% extra inventory per product and depot to obtain a feasible starting solution).

These constraints, theoretically, allow the heuristic to compute a similar solution for the MPMDVRPI as it has computed for the MDVRP. In practice, however, the additional inventory constraints should limit the number of feasible local search moves, and thereby make it more difficult to find improvements. The closer the two computed solutions are, the better the heuristic should be able to handle the inventory constraints.

We conduct these experiments for a different number of depots $M$ and products $P$, giving 5 seconds of computation time for instances with 100 customers. For each pair $M$ and $P$ we generate 100 MDVRP and corresponding MPMDVRPI instances and compute the average gap between the solutions. From the results in Table 5.1 we

Table 5.2: Gaps (in % with respect to the best known solutions of the MDVRP instances) and Computation times (in min) on the first 11 MDVRP instances by Cordeau et al. (1997) and on the new MPMDVRPI instances.

| Instance | M | MDVRP | | | MPMDVRPI Set 1 | | | MPMDVRPI Set 2 | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | | Value | Gap | Time | Value | Gap | Time | Value | Gap | Time |
| P01 (50) | 4 | 576.87 | 0.00 | 0.1 | 576.87 | 0.00 | 0.1 | 659.21 | 14.27 | 0.1 |
| P02 (50) | 4 | 473.53 | 0.00 | 0.1 | 473.53 | 0.00 | 0.1 | 556.88 | 17.60 | 0.3 |
| P03 (75) | 2 | 641.19 | 0.00 | 0.2 | 641.19 | 0.00 | 0.1 | 730.85 | 13.98 | 0.7 |
| P04 (100) | 2 | 1001.06 | 0.03 | 1.0 | 1008.12 | 0.69 | 1.0 | 1155.50 | 15.41 | 0.2 |
| P05 (100) | 2 | 750.59 | 0.07 | 0.9 | 750.53 | 0.07 | 1.3 | 795.35 | 6.04 | 0.5 |
| P06 (100) | 3 | 876.70 | 0.02 | 0.9 | 875.27 | -0.14 | 1 | 1028.51 | 17.34 | 1.3 |
| P07 (100) | 4 | 883.91 | 0.22 | 1.4 | 881.79 | -0.02 | 0.2 | 981.25 | 11.25 | 1.3 |
| P08 (249) | 2 | 4390.22 | 0.40 | 0.1 | 4451.87 | 1.81 | 0.5 | 5168.4 | 18.41 | 3.3 |
| P09 (249) | 3 | 3866.96 | 0.21 | 1.7 | 3934.67 | 1.97 | 1.4 | 4430.27 | 14.81 | 1.4 |
| P10 (249) | 4 | 3658.64 | 0.76 | 0.2 | 3749.64 | 3.26 | 2.6 | 4060.71 | 11.83 | 2.8 |
| P11 (249) | 4 | 3550.88 | 0.14 | 4.0 | 3681.47 | 3.82 | 2.8 | 4103.90 | 15.73 | 3.3 |
| | | | 0.16 | 0.9 | | 1.04 | 1.0 | | 14.22 | 1.4 |

observe that the heuristic is able to find solutions for MPMDVRPI instances that are of similar quality than those of the respective MDVRP instances. With more depots, the heuristic seems to have more difficulties to find high-quality solutions, because the inventory levels in each individual depot are lower and, thus, the problem is more constrained. A similar observation can be made for the consideration of multiple products. With each additional product more constraints are added to the problem, which impedes the local search.

Finally, we generated 22 MPMDVRPI instances on the basis of the first 11 MDVRP instances by Cordeau et al. (1997) for further validation and for future benchmarks. In each instance we distinguish between three products, where the demand of each customer on the original instances is either assigned to one of these products or split among two products. In the first set of instances (Set 1) the inventory levels are, as above, defined in such a way that the computed solution for the respective MDVRP instance is also a feasible solution in the corresponding MPMDVRPI instance. In the second set of instances (Set 2), we slightly perturb these inventory levels in the way described in Section 4.2 (50% of the inventory of one depot is re-distributed among the other depots). As a result of this perturbation, the routing solutions are likely to be worse. All benchmark instances are publicly available at `http://antor.uantwerpen.be/MPMDVRPI`.

The computed results on all these instances are displayed in Table 5.2. As in Table 5.1, we observe that for many instances of Set 1 the heuristic is able to compute results similar to those of the corresponding MDVRP instances, in some cases the splitting of demand among multiple products even results in better solutions. As expected, the results on Set 2 are significantly worse due to the perturbed inventory levels. Note that this does not imply that the heuristic performs much worse on these instances, as we have confirmed with the experiments in Table 5.1, but rather that the inventory constraints are more restrictive. We investigate this effect of inventory levels on the quality of the routing in the following Section.

## 5.4 Experimental setup

In the introduction we gave a preliminary example of how inventory levels can affect routing decisions. The basic assumption is that the theoretical best routing plan is not always feasible in practice when considering inventory levels in the depots. Inventory can be either sparse or badly allocated and, thus, result in non-optimal routing decisions in which customers cannot be supplied by their closest depot. In reality, inventory levels which prevent optimal delivery routes, in the following we will call them *non-optimal* inventory levels, occur for various reasons.

Firstly, inventory planning is usually a tactical decision that is taken over a period of several days or weeks. In contrast, transportation decisions are operational and taken on a day-by-day basis. Synchronization between both decision levels can therefore be difficult, since the inventory plan needs to anticipate the demand and deliveries for a longer time period. Secondly, inventory levels are generally not accurately predictable. Stochastic lead times, unforeseen fluctuations in demand or perishable products make it difficult to optimise inventory levels on their own, let alone the synchronization with delivery plans. Finally, inventory holding is expensive (capital lockup, storage costs, costs of spoilage, depot costs), and the amount of stored inventory will often be kept at a minimum. In summary, inventory plans might not always be synchronized with anticipated delivery plans, and even if they are, inventory levels are usually low and subject to stochastic effects. Consequently, the available inventory in the depots might not be sufficient to realize an optimal or almost optimal routing from depots to customers.

With the following experiments we investigate the effect of such non-optimal inventory levels on the routing costs. In other words, we aim at answering the question 'How much additional routing costs is incurred, if the inventory is insufficient or badly allocated'. We generate MPMDVRPI instances, compute a routing solution without inventory constraints and compare the costs of this solution to the routing costs that we obtain when we consider a non-optimal inventory situation. Thus, for the experiments we need to (1) generate instances and (2) generate different inventory situations.

### 5.4.1 *Generation of instances*

The key characteristic of an MDVRP (and thus also MPMDVRPI) instance is the location of the depots and the location of customers. In reality, one can observe a plethora of different setups, and it is impossible to study all possible locational scenarios. Therefore, in this study we focus on a simple and sensible scenario: The

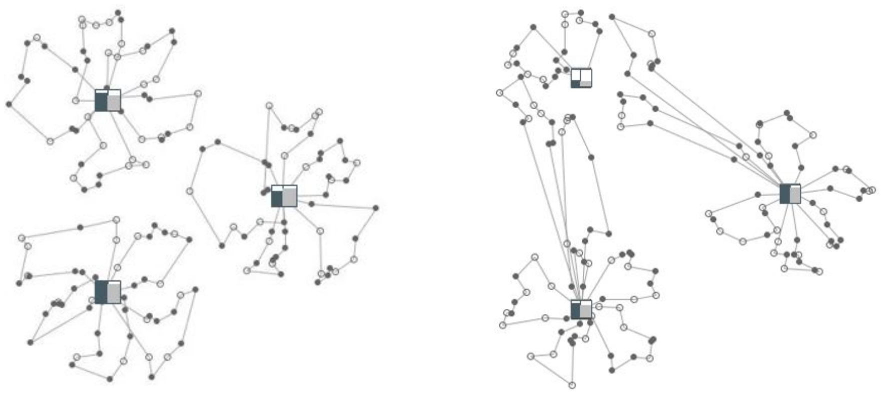


Figure 5.2: Locational setup of the MPMDVRPI instances. Customers can be less clustered (left) or more clustered (right) around the depots. At the same time, inventory levels can be defined to allow an optimal routing (left), or one depot is defined to have a shortage (right).

depots are located in such a way that the customers are clustered around them. We locate the depots evenly and equidistantly on a plane, define a circular area around each depot, and place the customers randomly into those circles. Hereby, we vary the radius of the circular areas as independent variable, where a smaller radius corresponds to a higher degree of clustering. An example of this locational setup is visualized in Figure 5.2.

We keep the number of customers $N = 100$ fixed (this is small enough to be computational feasible and large enough for realistic instances), while the number of depots is a second independent variable $M \in \{2, 3, 4\}$. The capacity limit of the vehicles is defined in such a way that each vehicle can deliver about 10 customers on average. For simplicity, each customer has a random demand $d_{ip} \in [1, 10]$ for one randomly selected product. Two underlying assumptions of this setup are that each depot has a similar number of customers in 'its area' and that the customers are distributed uniformly within the circular areas.

### 5.4.2 *Generation of inventory situations*

We generate inventory situations, in which high-quality routing solutions might become infeasible, in the following way. Given an instance from the generation procedure above, we compute a high-quality routing solution without inventory constraints with the MDVRP heuristic described in Section 3. From this solution we derive the optimal stock levels $I^O_{jp}$ for each depot $j$ by aggregating the demand for product $p$ over the allocated customers. For each product $p$, we then reduce the stock level of depot $j^*$ with the highest demand $I^O_{j^*p}$ by a factor $w \in \{0.25, 0.5, 0.75, 1\}$ and redistribute this amount equally among the other depots. Thus, higher values for $w$

correspond to worse inventory situations, and in the case of $w = 1$, one depot has completely run out of stock of product $p$. So far, the available inventory in all depots equals exactly the demand, while in reality, companies usually keep safety stocks to prevent stock-outs. We model these additional stocks by increasing the original stock levels by a factor $s \in \{0.05, 0.1, 0.2, 0.4, 0.6, 1\}$.

More formally, let $D_p = \sum_i d_{ip}$ denote the total demand for product $p$. Each depot is expected to have a similar number of nearby customers, and uniformly receives an additional stock $a = s\frac{D_p}{M}$. The stock levely of depot $j^*$ with the highest demand of product $p$ are set to $I^O_{j^*p} \cdot (1 - w) + a$, while the stock levels of the other depots are set to $I^O_{jp} + \frac{r}{M-1} + a$, with $r = I^O_{j^*p} \cdot w$. We do not argue that this way of generating an inventory situation is the most realistic. In fact, an accurate modelling of real inventory scenarios will depend on many aspects like the nature of the product, the ability to forecast demand, and the underlying logistic processes. However, the above generation procedure is relatively simple, and the assumption that the depot with the highest demand is most likely to suffer from shortages seems to be a reasonable one.

### 5.4.3 *Simulation setup*

We consider five independent variables. The first two variables, number of depots $M$ and the degree of clustering $C$, are associated with geographical properties of the problem, while the other three variables, number of products $P$, the degree of worsening $w$, and the amount of additional stock $s$, are related to the inventory situation. We perform a full-factorial $3 \times 2 \times 3 \times 4 \times 5$ design ('#depots' $\times$ 'degree of clustering' $\times$ '#products' $\times$ 'inventory situation' $\times$ 'additional stock'), and for each combination of variables we generate 100 MPMDVRPI instances. For each instance we compute a high-quality routing solution $S^O$ without inventory constraints, as well as a routing solution $S^I$ that we obtain when satisfying the respective inventory constraints, allowing 5 seconds of computation time in both cases. Our dependent variable is then the average increase in routing costs $\frac{S^I}{S^O}$ that we observe when considering the available inventory.

One might argue that this experimental setup does not reflect a real-world situation: Inventory levels are typically planned for several days or weeks, so just looking at a single delivery day rather than a multi-period model does not capture processes in reality. However, in some situations inventory is only ordered for a single period, e.g., if the considered products are perishables. Additionally, one instance could also represent the routing solution for multiple days, where the solutions for different days are virtually put on top of each other (assuming that each customer can be served at any day). With this in mind, the results are not only meaningful for a single day, but can also be interpreted from a multi-period perspective.

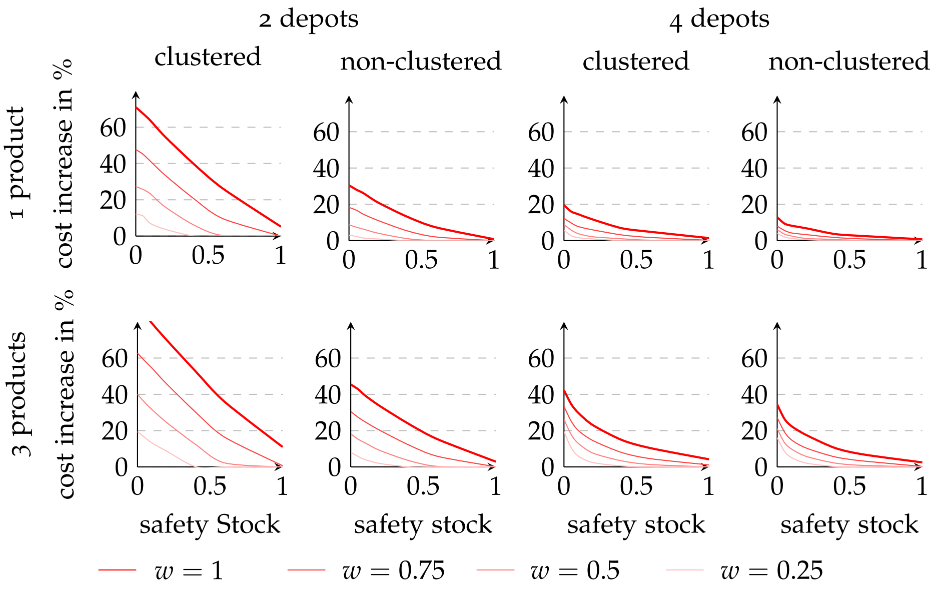


Figure 5.3: Experimental results for 2 and 4 depots, 1 and 3 products and two degrees of clustering, as a function of the amount of available inventory. The shaded curves present different inventory scenarios, where darker shades correspond to worse inventory scenarios. The increase in routing costs due to inventory limitations is highest, if many products are delivered from few depots to clustered customers.

## 5.5 Experimental results

We visualize the most insightful results in Figure 5.3, the complete results can be found in the appendix. For each combination of the variables '#depots', 'clustering' and '#products', we condense the results in one graph. Each graph depicts the increase in routing costs $\frac{S^I}{S^O}$ due to the respective inventory situation. The different inventory situations (where higher values for $w$ correspond to a worse inventory situation) are depicted by the different curves, in dependence of the available inventory and safety stock. From these results we can make the following observations.

*Inventory constraints*: The consideration of inventory constraints can have a significant effect on the routing costs. Depending on the allocation and the amount of additional stock, we observe an increase of routing costs of up to 80%. This cost increase is highest, if the inventory allocation is worse ($w = 1$) and there is no additional safety stock available. The significant increases in routing costs cannot be explained by a bias of the heuristic to produce better solutions if facing less constraints, as we have shown in Table 5.1. They are rather a direct consequence of the interdependency between routing and inventory.

*Additional stocks and safety stocks*: Additional stocks can cushion the effects of non-optimal inventory situations, and even in extreme situations ($w = 1$) they can reduce the additional routing costs by a large margin. Hereby, the value of additional stocks decreases with higher stock levels. If a depot has some inventory, it can at least supply the closest customers on very efficient delivery routes, which is significant better than if the depot is completely out of stock and all deliveries have to be performed from further away. However, these benefits fade with higher stock levels, since the distance between the depot and customers that could be served from this depot becomes larger and larger and thus, deliveries from other depots present a viable alternative.

*Inventory situation*: The results show that a worse inventory situation results in higher routing costs. Even a small deviation from the optimal stock levels ($w = 0.25$) can result in significant higher routing costs of up to 20% (in the case of 2 depots and 3 products). Little deviations can be remedied by holding some additional stock, while this becomes a less effective tool for worse inventory situations.

*Number of depots*: Additional depots appear to diminish the impact that inventory constraints have on additional routing costs, and routing costs increase less severely in non-optimal inventory situations. In other words, logistic systems with more depots are more flexible in reacting to non-optimal inventory situations. If a customer can no longer be served by its nearest depot, there are more options to deliver it from somewhere else.

*Customer location and clusters*: The spatial distribution of customers has a significant effect on the interdependencies between inventory and routing. If the customers are tightly clustered around their respective depot, the additional routing costs due to inventory constraints increases more drastically. Intuitively, if a customer from a cluster cannot be served by its closest depot, the detour that has to be taken to deliver the customer from another depot (cluster) becomes much longer, and, thus, the routing much worse.

*Product diversity*: The distinction between more products results in higher additional routing costs for non-optimal inventory situations. This can be considered to be a direct consequence of our definition of non-optimal inventory situations. We worsen the inventory situations for each product separately, and thereby possibly reduce stock levels in multiple depots. However, this effect will also occur in reality whenever products have individual and independent inventory policies, and it becomes more likely to face non-optimal inventory situations for at least one product.

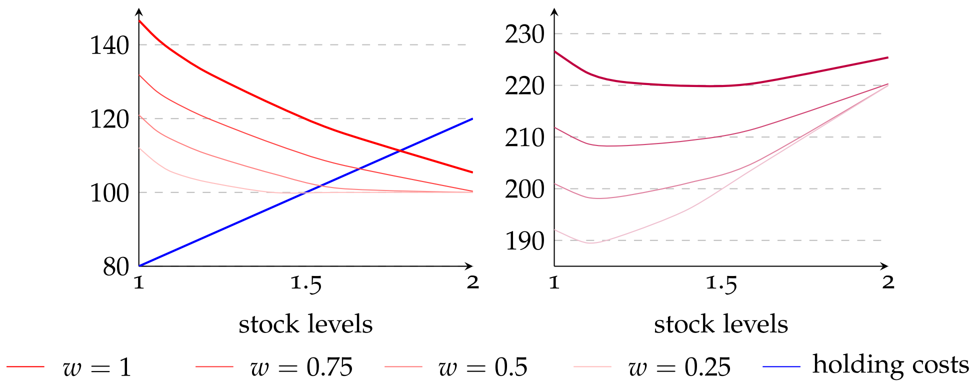


Figure 5.4: Hypothetical routing and inventory holding costs as a function of available inventory for $M = 3$ and $P = 2$ (left), as well as the sum of both costs (right).

The main implication of these results is that routing costs can increase significantly, if inventory is sparse and not well-allocated. A possible quick fix to this problem is to simply store more inventory in the depots. However, higher inventory levels come at the expense of higher holding costs, leading to a trade-off between inventory and routing costs. In Figure 5.4 we visualize such a hypothetical cost trade-off, assuming that holding costs increase linearly in the amount of stored inventory. The joint costs of transportation and inventory holding seem to be minimal with rather little safety stock. However, the determination of optimal stock levels requires knowledge about the likelihood of facing non-optimal inventory situations. To unfold the full potential of an integrated analysis, we need a framework that, on the basis of historic data, simulates numerous inventory scenarios and derives the probability of their occurrence, together with the corresponding routing solutions.

## 5.6 Conclusions and future research

In this chapter we extended the classical MDVRP to consider multiple products and inventory limitations, and formulated the resulting planning problem MPMDVRPI. We further introduced an effective heuristic to solve instances of this problem type and tested it on a new set of benchmark instances. To demonstrate its usability in a practical context, we used the model to investigate the impact that inventory constraints have on routing costs. The results highlight that non-optimal inventory situations can result in significant higher routing costs, especially when inventory is sparse or badly allocated. The magnitude of this effect depends on several factors like the number of depots, customer distribution and the product diversity.

The most important findings are simple and intuitive. Not only do safety stocks guard against stockouts, they can also improve delivery routes in non-optimal inventory situations. Consequently, a cost-efficient determination of stock levels for multiple depots should consider the subsequent routing from depots to customers. Likewise, the opening of additional depots can decrease additional routing costs in non-optimal inventory situations. Of course, opening depots comes at a considerable cost that should be weighed against its benefits.

In conclusion, these results help to shed some light on the interdependencies between inventory control and transportation planning. They can be used to further develop insights into optimal vertically-integrated logistic chains. In this respect, one of the most interesting questions is as to how inventory control and routing can be optimised jointly. The presented work lays the foundation by quantifying the effects that a certain inventory situation has on routing costs. A next step could be to shift the focus towards inventory, and optimise inventory decisions in such a way that the costs of both inventory and corresponding routing is minimal.

## 5.7 Appendix: Definition of an MPMDVRPI and experimental results

| | |
|---|---|
| $N$ | Number of customers |
| $M$ | Number of depots |
| $P$ | Number of products |
| $K$ | Number of vehicles |
| $C$ | Capacity limit of each vehicle |
| $d_{ip}$ | Demand of customer $i$ for product $p$ |
| $I_{jp}$ | Inventory level of product $p$ at depot $j$ |
| $c_{ij}$ | Travel cost between vertex $i$ and vertex $j$ (symmetric) |
| $b(S)$ | lower bound on the number of vehicles to serve all customers of $S$ |

$$x_{ijk} = \begin{cases} 1 & \textit{if a visit to vertex i is followed by a visit to vertex j in the route of vehicle k} \\ 0 & \textit{otherwise} \end{cases}$$

$$s_{ikp} = \begin{cases} 1 & \textit{if vehicle k delivers product p to customer i} \\ 0 & \textit{otherwise} \end{cases}$$

$$a_{jk} = \begin{cases} 1 & \textit{if vehicle k is allocated to depot j} \\ 0 & \textit{otherwise} \end{cases}$$

$$\min \left[ \sum_{i=1}^{N+M} \sum_{j=1}^{N+M} \sum_{k=1}^{K} c_{ij} x_{ijk} \right]$$

Subject to

1. Route continuation

$$\sum_{i=1}^{N+M} x_{imk} = \sum_{j=1}^{N+M} x_{mjk} \qquad \forall m = 1 \dots N+M, \forall k = 1 \dots K$$

2. Routes start and end at a depot

$$\sum_{i=N+1}^{N+M} \sum_{j=1}^{N} x_{ijk} = 1 \qquad \forall k = 1 \dots K$$

$$\sum_{i=1}^{N} \sum_{j=N+1}^{N+M} x_{ijk} = 1 \qquad \forall k = 1 \dots K$$

3. Synchronization of decision variables

$$\sum_{i=1}^{N} x_{ijk} = a_{jk} \qquad \forall j = N+1 \dots N+M, k = 1 \dots K$$

$$\sum_{p=1}^{P} s_{ikp} \leq P \sum_{j=1}^{N+M} x_{jik} \qquad \forall i = 1 \dots N, k = 1 \dots K$$

$$\sum_{p=1}^{P} s_{ikp} \geq \sum_{j=1}^{N+M} x_{jik} \qquad \forall i = 1 \dots N, k = 1 \dots K$$

$$\mathbb{I}_{d_{ip}>0} = \sum_{k=1}^{K} s_{ikp} \qquad \forall i = 1 \dots, N, p = 1 \dots P$$

4. Vehicle capacity constraints

$$\sum_{i=1}^{N} \sum_{p=1}^{P} d_{ip} s_{ikp} \leq C \qquad \forall k = 1 \dots K$$

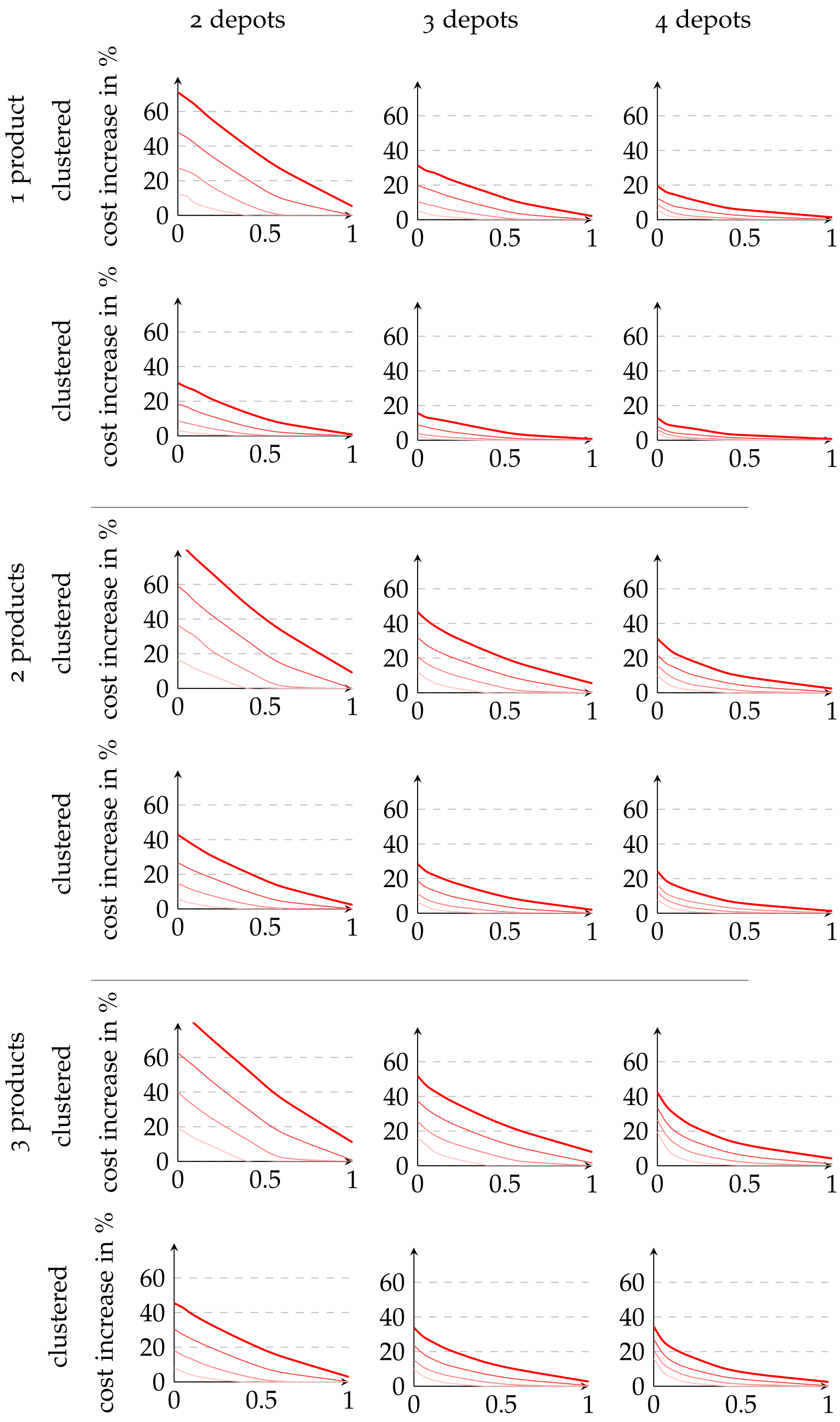
5. Inventory constraints

$$\sum_{k=1}^{K} a_{jk} \sum_{i=1}^{N} d_{ip} s_{ikp} \leq I_{jp} \qquad \forall j = N+1 \dots N+M, p = 1 \dots P$$

6. Subtour elimination

$$\sum_{i \in S, j \notin S} \sum_{k=1}^{K} x_{ijk} \geq 2b(S) \qquad \forall S \subset \{1, \dots, N\}$$

$$x_{ijk}, s_{ikp}, a_{jk} \in \{0, 1\}$$



Experimental results for 2, 3 and 4 depots; 1, 2 and 3 products and two degrees of clustering, as a function of the amount of available inventory.

6

# Vertical integration of routing and inventory management

## 6.1 Introduction

The integration of different activities within a logistic chain, called *vertical integration*, is a well-studied concept to elevate the performance in complex supply chain environments (Christopher, 2016). For logistic systems in which deliveries have to be planned, the potential of an integrative analysis between the distribution activities and other supply chain activities has been demonstrated for many problems, for instance the integration of production and distribution (Reimann et al., 2014; Chandra and Fisher, 1994) and the integration of location decisions and distribution planning (Salhi and Rand, 1989; Prodhon and Prins, 2014). The integration of inventory management and routing from a single depot is usually studied in the context of vendor-managed inventory (location routing problems) (Andersson et al., 2010; Moin and Salhi, 2007). However, the vertical integration between the supplier's inventory management in several depots and the distribution from the depots to the customers is less well-explored. The traditional approach has been to tackle these two problems sequentially and avoid any compounding complexity (Andersson et al., 2010). Some first work in this direction include location-inventory-routing problems, which additionally consider the location decision and therefore often simplify one component, e.g., static routes and customers are considered (Liu and Lee, 2003), or demand is assumed to be deterministic (Hiassat et al., 2017).

In the inventory literature, the best inventory policy is usually determined under the assumption that each depot faces stochastic demand. Multiple depots can collaborate and share their inventory by *pooling* their inventory. The pooling of inventory has been shown to be highly cost-efficient in many scenarios (Bimpikis and Markakis, 2015; Berman et al., 2011) since the risk of shortage can be balanced more effectively among the collaborating depots. These benefits increase with the number of depots sharing their inventory (Tagaras, 1999), while the benefits of pooling also depend on

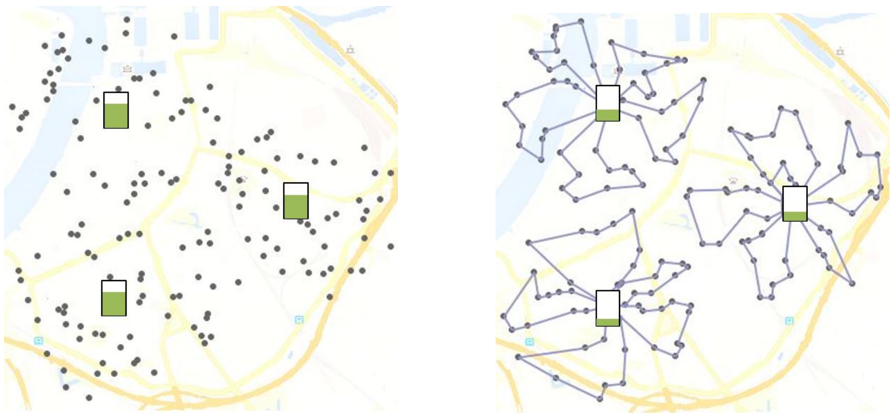


Figure 6.1: Integration of inventory management and routing. A company with several depots has to plan the inventory in the depots, as well as to plan delivery routes to fulfill customers' demand.

the variability of demand (Gerchak and He, 2003). The standard approach to pool inventory are *lateral transshipments*. With lateral transshipment, inventory can be transported from one depot to another to prevent shortage. These transportations can either be carried out at predetermined times before all demand is realised (proactive transshipment), or they can take place at any time to respond to shortage or potential shortage (reactive transshipment or emergency transshipment) (Paterson et al., 2011).

Reversely, in the routing literature a general assumption of multi-depot vehicle routing problems (MDVRP) is that any customer can be delivered by any depot (Montoya-Torres et al., 2015). The distribution activities are carried out jointly by all depots and pooling occurs naturally to obtain highly-efficient delivery routes. However, routing problems largely ignore the inventory perspective, assuming a sufficient availability of inventory in all depots.

Consequently, the fields of inventory management and distribution take different perspectives and make different assumptions. In inventory management, a limited amount of available inventory can be pooled to supply customers in such a way that inventory costs are minimised. In distribution, unlimited inventory is delivered flexibly to customers in such a way that routing costs are minimised. The question arises how both activities can be effectively integrated in a multi-depot context. In reality such problems can occur, for instance, for a logistic service provider or retailer that owns multiple depots and is responsible for planning its inventory in the depots as well the delivery of goods from the depots to fulfill the demand of customers. Given that future demand is stochastic, how can the company minimize the total costs, as visualized in Figure 6.1.

Since the pooling of inventory has been shown to be beneficial, what is the best strategy when integrating inventory and routing. Should the depots pool their inventory but not the routing, or should the depots collaborate on both levels? Given an periodic review model, we propose a simple simulation framework that enables the integrated analysis of inventory policies for multiple depots by taking the distribution into account.

We outline the problem in Section 2. In Section 3 we use previous findings in literature to compute an optimal inventory policy for the case that depots do not collaborate, and in Section 4 for the case that depots share their inventory via lateral transshipments. In Section 5 we propose a model in which depots can pool both, their inventory and routing plans. All three strategies are compared in a set of experiments in Section 6, and Section 7 summarizes our findings.

## 6.2 Problem formulation

### 6.2.1 *Models to pool inventory management and routing*

We consider a logistic system with multiple depots in which the inventory is planned on the basis of a periodic $(S-1, S)$ review model like in Tagaras (1999). The $(S-1, S)$ model is among the most frequent ones to be used in the context of pooling (Paterson et al., 2011), and Herer et al. (2006) have shown that there always exist an optimal $(S-1, S)$ policy within the class of non-shortage inducing replenishment policies for multiple depots with transshipments.

At the start of a period, each depot $i$ has an order-up-to level $S_i$ on stock and faces stochastic demand from customers. The customers have to be visited on delivery routes and are located in an area of interest around each depot. The aggregated customer demand $D_i$ in each area is stochastic and can be modelled with a distribution function, for instance, on the basis of historical data and forecasts. In the remainder of the chapter we assume that $D_i$ is normally distributed and independent of time. If $S_i - D_i$ is negative, not all customers can be delivered in this period and shortage cost incur. Otherwise, additional holding costs have to be paid. At the end of a period an order is placed such that the inventory is replenished to $S_i$ before the start of the next period. This 'immediate' replenishment assumes that lead times are negligible.

In terms of inventory management, the goal is to determine order-up-to levels $S_i$ that minimizes the sum of shortage costs and holding costs. Usually both cost types have different coefficients to reflect the additional cost of not fulfilling demand. In the following, let the holding costs be defined as one per unit of inventory on stock at the end of a period (the inventory levels at the start of a period are always the same and thus a constant), and let $s > 1$ be the *shortage factor*, the costs for each unit of demand that cannot be fulfilled. With larger $s$, there is an increasing incentive to avoid shortage which results in higher service levels.

Next to planning an adequate inventory policy, the demand needs to be fulfilled on efficient delivery routes from the depots to the customers. Planning delivery routes from a depot is formalized in the VRP and MDVRP as defined in Chapter 1.

We assume that the customers in each period can be different ones, and have different locations. Thus, the logistic system is highly dynamic and in each period new delivery routes have to be planned. Such a dynamic and complex system can be analysed via simulation. Given order-up-to levels $(S_i, S_j, \dots)$ in the depots and a certain routing strategy, simulate for $T$ periods

1. For each depot, generate demand $D_i$
2. Per depot, place $D_i$ customers in the respective area of interest
3. Compute a routing solution from the depots to customers
4. Update shortage cost $c_s = s \cdot \max(0, D_i - S_i)$, and holding costs $c_h = \max(0, S_i - D_i)$
5. Replenish the inventory to $S_i, S_j, \dots$

The result of one simulation run is the average routing cost and the average inventory cost per period. We consider an infinite planning horizon and want to minimize the long-run average costs, and thus, $T$ can be chosen large to obtain good estimates. We report routing cost and inventory cost separately, since both cost types have a different scale and might vary in magnitude. While routing costs are computed in terms of distance driven, inventory costs are expressed in terms of holding costs, and the transformation from one metric into the other depends on the considered logistic systems and the considered product.

The computation of routing solutions is the most computationally demanding step in this simulation framework. Routing problems are NP-hard and their complexity grows rapidly in problem size. For this reason, a significant research effort has been devoted to the design of heuristics that try to find good solutions quickly, but do not guarantee to find the optimal solution in a finite amount of time, as outlined in Chapter 1. If we want to improve the solution quality to a gap of 1% and less, we require at least a few seconds of time. Assuming we invest one second of computation time for the computation of a routing solution, then we would prolongate the time required for a simulation of $T = 10,000$ periods to several hours. Therefore, we utilise efficient construction heuristics in the remainder of this chapter to approximate the routing costs.

The depots can collaborate and organize these two logistic activities jointly. In the following, we compare three different strategies as to how this collaboration can be set up.

- No pooling, each depot plans inventory and distribution autonomously
- Depots pool their inventory, but not their distribution
- Depots pool their inventory and their distribution

For each strategy we show how to compute optimal order-up-to levels $(S_i, S_j, \ldots)$, and how to estimate the resulting routing costs.

### 6.2.2 *Simplifying hypotheses*

The above model is kept simple to allow a thorough investigation of the considered strategies and their interaction with different parameters. In the following, we outline the simplifying assumptions made, and briefly discuss how further elements could be incorporated in the model.

*Constant order interval.* The $(S-1, S)$ model assumes that an order is placed every period, and thus, corresponding order costs are a constant and do not need to be considered. In other inventory models, like for instance the $(r, Q)$ model, an order is only placed if a certain inventory level is reached, to minimize the sum of order costs, holding costs and shortage costs. Whereas in a $(S-1, S)$ model the system is reset after each period, a $(r, Q)$ model requires additional considerations. The routing has to be planned for possibly multiple periods until the inventory is restocked. Finding efficient multi-period routing solutions from multiple depots with inventory constraints is beyond the scope of this work.

*Negligible lead times.* If lead time are not negligible, an order arrives one or several periods after it has been placed. As above, this results in a situation where the system is not reset after each period, and complicates the planning of delivery routes. Depending on the inventory levels at the start of a period, one might change the strategy of how to plan the delivery routes (e.g., depots with more inventory available supply more customers). This dynamic component adds another decision level to the considered problem and might be a starting point for further research.

*Lost sales.* We assume that unmet demand results in lost sales. If demand exceeds inventory levels in a certain period, one could also choose to deliver some customers when inventory becomes available (backorders). Backorders can be incorporated in the model by storing the coordinates and demand of non-delivered customers, and adding them as customers in the following period. These deliveries could also be modelled as express deliveries. Backorder could be planned to improve the routing, e.g., an isolated customer is only delivered once a nearby customer can be delivered as well.

*Constant customer demand.* We assume that each customer demands one unit of the considered product, and thus, demand corresponds to the number of customers. Alternatively, customers could be allocated a random demand within a certain range. Then the total demand per period and depot could be distributed among a fixed or randomly-chosen number of customers.

## 6.3 **No pooling**

In the first strategy we assume that the depots do not collaborate, and each depot plans its inventory and the distribution to customers separately. This situation occurs in systems with a decentralized decision-making. The areas of interest for each depot are clearly defined and do not overlap, and thus, each depot only delivers to customers in its region of interest. As a consequence, we can compute the optimal inventory policy and routing plan for each depot independently.

Let the demand $D_i$ of a depot be normally distributed $D_i = \Phi(\mu_i, \sigma_i^2)$ with mean $\mu_i$ and variance $\sigma_i^2$. Then the optimal order-up-to level $\overline{S_i}$ for an individual depot with neglectable lead times can be computed analytically as $\overline{S_i} = \mu_i + \Phi^{-1}(\frac{s+1}{s}, 1)\sigma_i$ (Tagaras, 1999). Hereby, $\Phi^{-1}(\frac{s+1}{s}, 1)\sigma_i$ denotes the safety stock that is kept to reduce the probability of shortage. Generally, with higher values for $s$, we will obtain more risk-averse policies with higher order-up-to levels $\overline{S_i}$.

The routing plan for each depot and period can be computed by solving a VRP, for each depot separately. The allocation of customers to depots is fixed via the definition of the areas of interest, and depots cannot deliver customers that are located in the area of another depot. To solve the corresponding VRPs efficiently we use the heuristic by Clarke and Wright (1964) (CW). This construction heuristic can compute routing solutions for 100 customers in a few milliseconds. If a depot faces shortage in a period, we choose not to deliver those customers that are furthest away from the respective depot.

In summary, the costs of the strategy 'no pooling' can be determined in the following way. The optimal inventory policy $(\overline{S_i}, \overline{S_j}, \dots)$ is computed analytically, followed by a simulation to determine the corresponding inventory cost and routing costs. For $T$ periods,

1. For each depot, generate demand $D_i$
2. Compute a routing solution with CW, for each depot separately
3. Update $c_h = \min(0, S_i - D_i)$ and $c_s = s \cdot \max(0, D_i - S_i)$

4. Replenish the inventory to $\overline{S_i}, \overline{S_j}, \ldots$

## 6.4 Pooling of inventory via lateral transshipments

In the second strategy, the depots collaborate and share their inventory. Inventory sharing, also called *inventory pooling*, is realized via *lateral transshipments*, and assumes that decision-making is centralized. We consider emergency transshipments that are executed practically instantaneously. This assumption is justified when the collaborating depots are situated in close proximity. More formally, if depot $i$ is experiencing a shortage $S_i - D_i < 0$ at the start of a period, and another depot $j$ has remaining inventory $S_j - D_j > 0$, a transshipment from $j$ to $i$ is organized. The transshipment is carried out immediately and transports $\min(D_i - S_i, S_j - D_j)$ units of inventory. Thus, the shortage of depot $i$ is reduced or possibly prevented (less shortage costs), while the remaining inventory of $j$ is put to good use (less holding costs). Tagaras (1999) has shown that the type of transshipment policy does only have a minor effect on the system's performance. In this work we will use a simple policy that minimizes the distance driven to carry out the transshipments. The determination of such a transshipment plan corresponds to solving a transportation problem, where depots with positive inventory levels can supply depots with a shortage. If the pairwise distances between depots are the same, this problem can be solved by minimizing the number of transshipments in the following way. If a depot experiences shortage, the depot with the highest inventory position transships as much goods as possible to eliminate the shortage. If some shortage remains, a second transshipment can be executed from a third depot. If more than one depot has a shortage, the depot with the most shortage is considered first. This policy implies that transshipments eliminate as much shortage as possible in the system.

The pooling of inventory results in stochastic interdependencies between the depots complicating the system and, as a consequence, optimal order-up-to levels $(S_i^*, S_j^*, \ldots)$ can no longer be computed analytically, but have to be determined with simulation optimisation. Given an upper bound for the optimal order-up-to levels, we estimate the corresponding inventory costs by simulating for $T$ periods:

1. For each depot $i$, generate demand $D_i$
2. If $D_i > S_i$ for some $i$, execute transshipments
3. Update $c_h = \min(0, S_i - D_i)$ and $c_s = s \cdot \max(0, D_i - S_i)$
4. Replenish the inventory to $S_i, S_j, \ldots$

The optimal order-up-to levels with pooling will never be larger than the optimal order-up-to levels without pooling, and thus $(\overline{S_i}, \overline{S_j}, \ldots)$ constitutes an upper bound for the optimal inventory policy. Starting from this upper bound, we perform a grid search and iteratively decrease the order-up-to levels and estimate the inventory costs of the resulting policy via simulation. The policy with the lowest cost estimate is then chosen to be the optimal policy $(S_i^*, S_j^*, \dots)$. A similar approach has been suggested by Tagaras (1999) for inventory systems with positive lead times.

As in the strategy 'no pooling', the distribution from depots to customers is handled by each depot autonomously, and thus, the routing decisions do not affect the inventory levels (each depot attempts to supply the customers is its area). Consequently, the optimal inventory policy is independent of the delivery routes, and we do not need to compute any routing solutions to determine the optimal inventory policy in the simulation above. Since the simulation of one period is not computationally demanding, we can execute simulations with $T > 10.000$ periods without any computational effort while obtaining accurate estimates for the inventory costs.

However, we need to define a cost value for transshipments. Since a transshipment is basically a return trip from one depot to another depot, it can be expressed with the same cost type as that of the delivery routes: distance. One might argue that transshipments, and especially emergency transshipments, trigger further costs, e.g., for loading, unloading, organizing a spare vehicle, or finding an additional driver. For simplicity, we will just consider the distance driven, which corresponds to two times the distance between the depots. This distance is then added to the distance of the delivery routes. In other words, transshipments are considered to be part of the routing.

In short, the costs for the strategy 'inventory pooling' can be estimated with the following approach. Determine the optimal inventory policy $(S_i^*, S_j^*, \dots)$ via simulation, and estimate the resulting distribution costs by simulating $T$ periods

1. For each depot, generate demand $D_i$
2. If $D_i > S_i^*$ for some $i$, execute transshipments, and compute distance
3. Compute a routing solution with CW, for each depot separately
4 Update $c_h = \min(0, S_i^* - D_i)$ and $c_s = s \cdot \max(0, D_i - S_i^*)$
5. Replenish the inventory to $S_i^*, S_j^*, \dots$

## 6.5 Pooling of inventory and distribution

As a third strategy we consider a scenario in which the depots share their inventory and also coordinate their delivery routes jointly. A joint planning of deliveries implies that any customer can be delivered by any depot. Thus, the allocation of customers to depots is no longer fixed as in the first two strategies, and depots can

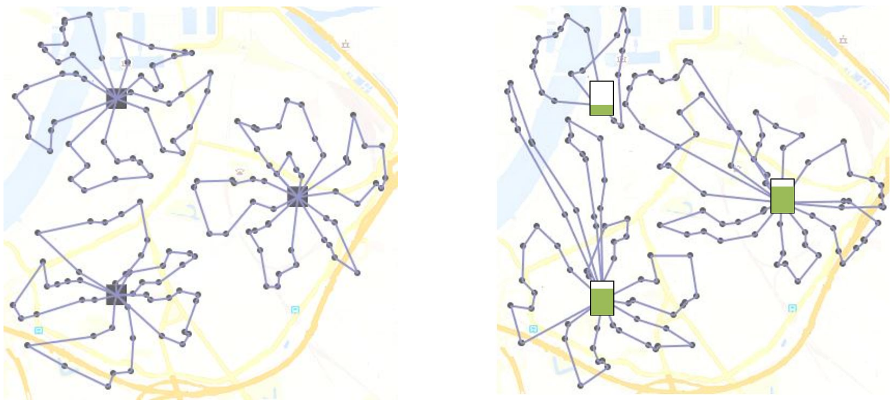


Figure 6.2: Routing solutions in a multi-depot context can be computed in different ways. (Left) Each depot only delivers customers in 'its area'. (Right) Each depot can deliver any customer as in classical MDVRPs. If we consider the inventory levels as constraint in MDVRPI, we might arrive at a slightly worse routing.

also visit customers in the area of another depot. This assumption is commonly used in MDVRPs, and allows to plan the routing in such a way that the distribution costs of the entire system are minimized, rather than those of an individual depot. The practical implication of this model assumption is that the routing can be planned flexibly. A driver is no longer just responsible for a certain area, but might also deliver far-away customers.

However, the standard formulation of the MDVRP does not consider the inventory levels in the depots and, thus, customers are usually supplied from the nearest depot on efficient delivery routes. If this depot no longer has the demanded product on stock, the delivery might have to be planned from a depot further away. This interaction between inventory levels and delivery routes is visualized in Figure 6.2. In Chapter 5 we incorporated these inventory constraints in the classical MDVRP. Given a set of customers with known demands, given several depots, each with their stock levels defined, find the lowest-cost routes such that the total demand served from each depot does not exceed its inventory levels (MDVRPI). On the basis of this extended model we investigated the effect that different inventory levels have on the quality of the respective delivery routes, and found that non-optimal inventory situations can result in significantly higher routing costs, especially when inventory is sparse or badly allocated. In other words, the interaction between routing and inventory management can have a relevant effect on the costs of the logistic system.

In the MDVRPI the globally available inventory in all depots is used to satisfy the demand of possibly all customers. Therefore, the inventory is pooled without using transshipments, at the expense of higher requirements for routing flexibility. The allocation of which depot delivers which customers is no longer spatially constrained, but rather depends on the computed delivery routes. As a result the routing decisions impacts the inventory levels in the depots at the end of a period, in contrast to the previous strategies. This interdependency appears to complicate the determination of optimal order-up-to levels, however, with some assumptions the inventory policy can be computed independently of the routing plans.

**Proposition 1.** If inventory is pooled, the sum of remaining inventory in all depots after a period is the same, independent of how the pooling is realized.

This proposition formalizes the assumption that pooling will eliminate as much shortage as possible. A depot can only have shortage, if all other depots have run out of stock as well. The case that one depot has a shortage while another depot has inventory on stock cannot occur.

**Corollary 1.** If inventory is pooled and depots have the same cost structure, then all inventory policies in which the sum of the order-up-to levels $S_i + S_j + \dots$ is the same, will result in the same inventory costs $c_h + c_s$.

Corollary 1 simplifies the determination of optimal inventory policies for the case that distribution is pooled. The sum of the order-up-to levels $S_i + S_j + \dots$ can be determined computationally cheap via simulation optimisation as in Section 2 without computing any routes. The remaining task is to distribute this global order-up-to-level among the collaborating depots in such a way that delivery costs are minimized. Two cases can be distinguished. The logistic system can be setup as in Figure 6.2 so that the pairwise distance between depots is the same and each area of interest has the same size and demand function. In this case, all depots are interchangeable and we call the system *symmetric*. In a symmetric system there is no reason to store more inventory in one depot than in another, and we will obtain $S_i = S_j = \dots$ as the optimal inventory policy in terms of routing costs.

Thus, we only need to further optimise the inventory policy if the system is *asymmetric*. Asymmetry can occur if either the distances between depots differ or the demand distributions in the regions of interest differ. Because of the asymmetry, there might be a better option than distributing the global-up-to-order levels equally among all depots to minimize routing costs (the inventory costs will remain the same independent of the distribution). To find the best distribution, we propose a grid search, keeping the sum $S_i + S_j + \dots$ fixed while changing the individual values, starting from $S_i = S_j = \dots$. A certain policy is again evaluated by simulating $T$ periods

1. For each depot, generate demand $D_i$

2. Determine the routing by solving an MDVRPI

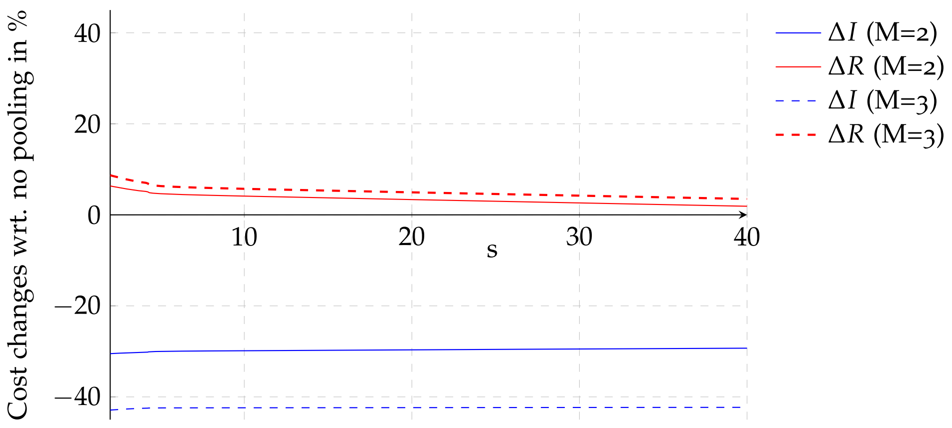


Figure 6.3: Comparison between the strategies 'no pooling' and 'inventory pooling'. Inventory pooling results in reduced inventory costs at the expense of a slight increase in routing costs.

3 Update $c_h = \min(0, S_i^* - D_i)$ and $c_s = s \cdot \max(0, D_i - S_i^*)$

4. Replenish the inventory to $S_i, S_j, \ldots$

We use the regret-heuristic introduced in Chapter 5 to compute MDVRPI solutions in a short time. In a first step, the heuristic allocates customers to depots in such a way that the opportunity cost, incurred when the closest depot does not have sufficient inventory to serve the customer, is minimized. If demand exceeds inventory in a period, the customers with the maximum distance to any depot are not delivered. In a second step a VRP solution for each depot and its assigned customers is constructed with the heuristic by Clarke and Wright (1964).

In summary, we can determine the optimal inventory policy for a symmetric logistic system in which deliveries are pooled without computing routing plans. In this case, the order-up-to levels are the same as in the strategy 'pooling inventory', and we only need to simulate the integrated system once to obtain the routing costs. If the system is asymmetric, the optimisation of order-up-to levels is more involved since they depend on the respective routing solutions.

## 6.6 Experimental results

On the basis of the models described above, we compare inventory and routing costs for the three different strategies. In the first experiments we assume that the depots are symmetric and have the same demand distribution with $\mu = 50$ and $\sigma = 10$. Each depot is surrounded by a circular area in which the customers are placed randomly. Hereby, the circular areas of each depot touch each other, i.e.,

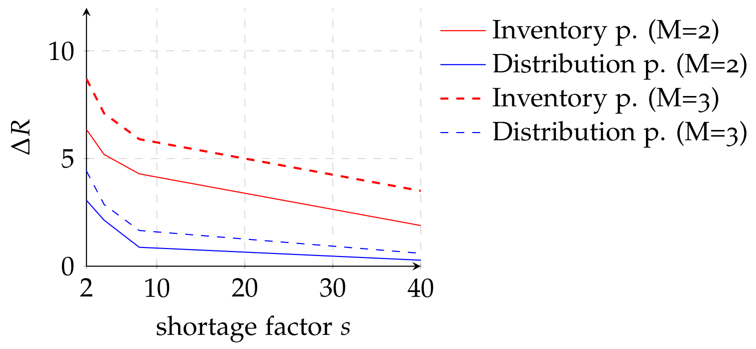


Figure 6.4: Comparison between the strategies 'inventory pooling' and 'distribution pooling' in terms of routing costs. We observe that the pooling of distribution results in a more effective way to balance inventory than via transshipments. The inventory costs are the same for both strategies.

the distance between two depots correspond to the diameter of the circular areas. Each customer has a demand of one unit, and each delivery route can visit up to 10 customers. We vary the shortage factor $s \in [2, 40]$, where higher values for $s$ reflect more risk-averse policies (remember that one cost unit corresponds to the holding costs). Each depot has the same cost structure with the same shortage factor and holding costs.

### 6.6.1 *No pooling versus inventory pooling*

First, we compare the first strategy (no pooling) and the second strategy (inventory pooling) for $M \in \{2, 3\}$ depots. Figure 6.3 displays the cost differences $\Delta I = \frac{I_2}{I_1} - 1$ between the inventory costs $I_1$ of the optimal inventory policy in the no pooling scenario and the inventory costs $I_2$ of the optimal inventory policy in the pooling scenario, respectively for the distribution costs $\Delta R = \frac{R_2}{R_1} - 1$. Details about the optimal policies can be found in Table 6.1.

In line with previous studies (Bimpikis and Markakis, 2015; Berman et al., 2011; Tagaras, 1999), we observe a reduction in inventory costs of up to 30% for two depots, and of up to 42% for 3 depots if inventory is pooled. This cost reduction appears to be independent of the shortage factor. It can be attributed to a reduction in shortage, and a reduction in order-up-to levels. This decrease in inventory costs is brought about by transshipments between the depots, which result in additional routing costs of up to 12%. These additional routing costs decline for higher shortage factors $s$, since the optimal inventory policy becomes more risk-averse and we have to carry out less transshipments.

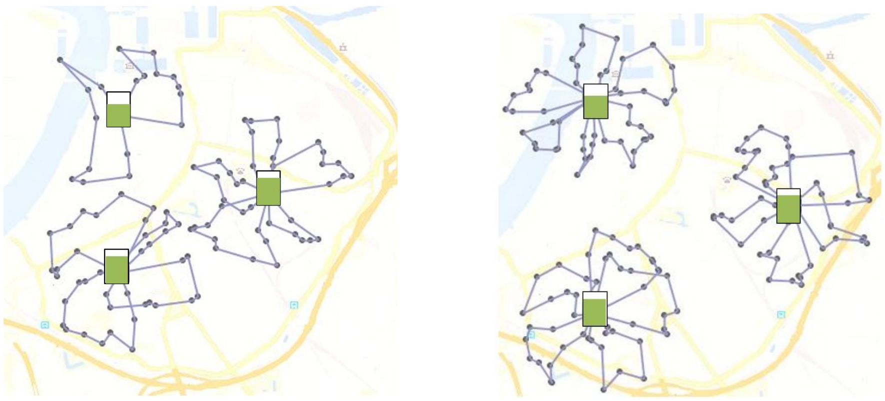


Figure 6.5: The regions of interest of the depots can almost overlap (left), or be further apart (right). In the latter case we would expect higher additional routing costs if depots pool their inventory.

### 6.6.2 *Pooling of inventory versus pooling of distribution*

On the basis of Corollary 1 we obtain the same optimal inventory policy with the same inventory costs for strategy 2 (pooling inventory) and strategy 3 (pooling distribution). Therefore, strategy 3 will yield the same significant reduction in inventory costs. The question arising is whether the same holds for the distribution costs. From the results in Figure 6.4 it appears that the pooling of inventory via a joint distribution is more cost-effective than the pooling via transshipments, and several percentages of routing costs can be saved. This finding can be explained by the fact that a transshipment is a very inefficient route. The distances between two depots are rather long in comparison to the distance between two successive customers on a delivery route. Thus, it is more effective to add one or several customers from another depot to a route, instead of executing an entire transshipment, if both costs are expressed in terms of distance. The detailed results for all three strategies can be found in Table 6.1.

### 6.6.3 *The effect of distance between depots*

So far we have considered logistic systems in which the areas of interest for each depot are relatively close to each other. If the distance between depots is rather high, for instance because they are located in different cities or even in different countries, we would expect that any interaction between them becomes more costly. In the following, we investigate the effect that this relative distance between depots has on pooling. With relative distance we denote the fact that depots are further apart while the distance to their customers remains the same. We compare the additional routing costs due to pooling of the previous experiment with the additional routing costs

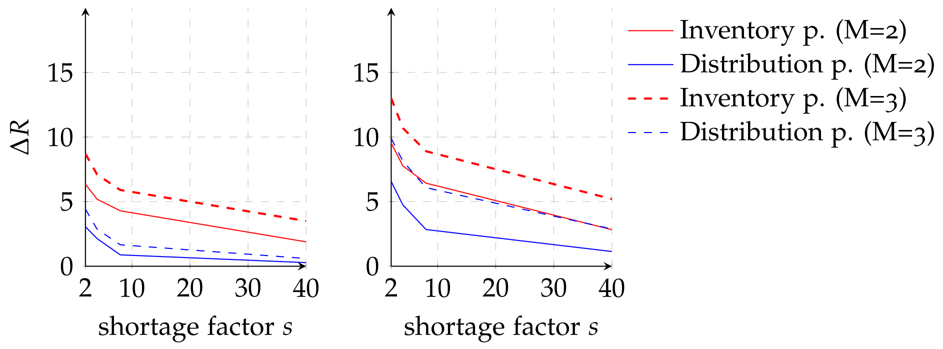


Figure 6.6: Comparison between the strategies 'inventory pooling' and 'distribution pooling' in terms of routing costs for the previous setup (left) and the setup where depots are further apart (right). We observe that a greater relative distance between distance results in higher additional routing effort, however, pooling inventory via a joint distribution is remains the superior strategy.

that incur if the areas of interest are spaced further apart. We increase the distance between the areas by the length of their radius, as visualized in Figure 6.5.

The results in Figure 6.6 confirm the initial intuition. With a greater relative distance between depots of about 50%, the additional distribution costs of pooling inventory increase by about the same margin. Strategy 3 still appears to be superior to strategy 2 even if relative distances are higher.

### 6.6.4 *The effect of variance in demand*

In the following we investigate the effect that the variance in demand has on the integrated logistic system. We repeat the experiment from Section 5.1 with different values $\sigma_A = \sigma_B = \cdots = 5$ and $\sigma_A = \sigma_B = \cdots = 20$. Since we observed that strategy 3 (pooling of distribution) dominates strategy 2 (pooling via transshipments), we compare strategy 1 (no pooling) with strategy 3. Generally, with a higher variance we obtain higher up-to-order levels in the optimal inventory policy to account for the greater risk of shortage.

We observe in Figure 6.7 that pooling results in the same reduction in inventory costs independent of variance. However, this reduction comes at the expense of additional routing efforts, as already seen in Section 5.1. If the variance is low, the additional routing costs are almost neglectable at about 1% and less, whereas they increase to 5% and more if the variance is higher. We conclude that in logistic systems with a low variance in demand pooling is more effective than in logistic systems with a lower predictability of future demand.

Table 6.1: Optimal up-to-order inventory policies for symmetric logistic systems with neglectable lead times. Order-up-to level $S$ per depot, average inventory costs $I$ per depot and period, average service level $SL$ per depot, average number of outgoing transshipments $T$ per depot and period and changes in routing costs ΔR.

| | | | | No Pooling | | | Inventory Pooling | | | | | Distribution Pooling | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| $\mu$ | $\sigma$ | M | s | S | I | SL | S | I | SL | T | ΔR | S | I | SL | ΔR |
| 50 | 10 | 2 | 2 | 54 | 10.9 | 95.5% | 53 | 7.8 | 96.9% | 0.22 | +6.35% | 53 | 7.8 | 96.9% | +3.06% |
| 50 | 10 | 2 | 4 | 58 | 14.0 | 97.7% | 56 | 9.9 | 98.4% | 0.18 | +5.19% | 56 | 9.9 | 98.4% | +2.14% |
| 50 | 10 | 2 | 8 | 62 | 17.2 | 98.9% | 58 | 12.1 | 99.1% | 0.15 | +4.26% | 58 | 12.1 | 99.1% | +0.88% |
| 50 | 10 | 2 | 40 | 70 | 23.5 | 99.8% | 64 | 16.4 | 99.9% | 0.07 | +1.89% | 64 | 16.4 | 99.9% | +0.28% |
| 50 | 10 | 3 | 2 | 54 | 10.9 | 95.5% | 52 | 6.3 | 97.0% | 0.30 | +8.69% | 52 | 6.3 | 97.0% | +4.42% |
| 50 | 10 | 3 | 4 | 58 | 14.0 | 97.7% | 55 | 8.1 | 98.8% | 0.25 | +7.14% | 55 | 8.1 | 98.8% | +2.87% |
| 50 | 10 | 3 | 8 | 62 | 17.2 | 98.9% | 57 | 9.9 | 99.4% | 0.21 | +5.92% | 57 | 9.9 | 99.4% | +1.66% |
| 50 | 10 | 3 | 40 | 70 | 23.5 | 99.8% | 61 | 13.5 | 99.9% | 0.13 | +3.47% | 61 | 13.5 | 99.9% | +0.60% |

### 6.6.5 *Asymmetric setups*

So far we assumed that all depots are interchangeable and the distance between each pair of depots is the same. Thus, in the optimal inventory policy we will obtain the same order-up-to levels $S_i = S_j = \dots$ for each depot. This situation might change if we consider asymmetric logistic systems, and in the following we compute the optimal inventory policy with the simulation approach in Section 5, assuming that depots pool their distribution.

**Asymmetry in distance**
We consider a system in which all depots are located next to each other on a line. Thus, in this setup we have a central depot $S_2$ with one depot on either side of it. Intuitively, a more central depot should hold more inventory since the customers of the other depots can be reached quite efficiently. For $\mu = 50$, $\sigma = 10$ and $s = 2$ we confirm this intuition, and compute the optimal order-up to levels as $S_1 = S_3 = 50$ and $S_2 = 56$. However, this policy results in only about 0.4% lower routing costs than in the equal distribution policy $S_1 = S_2 = S_3 = 52$. Higher values for $s$ yield even smaller differences. In summary, more central depots should generally stock more inventory to deliver the customers more efficiently, however, this asymmetry in distance has only a minor effect on the optimal inventory policy and the resulting routing costs.

**Asymmetry in variance**
Let the variance in demand of two depots be rather low $\sigma_1 = \sigma_2 = 5$, and for one

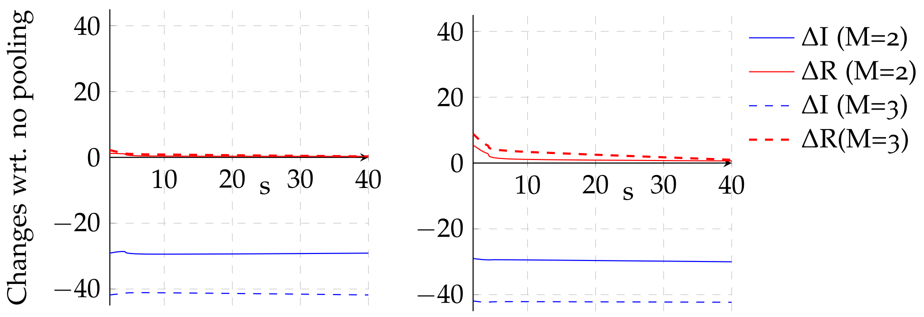


Figure 6.7: Comparison between the strategies 'no pooling' and 'distribution pooling' for different demand variance $\sigma = 5$ (left) and $\sigma = 20$ (right). Pooling results in a similarly high reduction in inventory costs independent of variance, whereas with higher variance the additional routing costs of pooling increase.

depot rather high $\sigma_3 = 20$. We would expect that a higher variance should be accompanied with a higher up-to-order level. The respective depot has a higher likelihood to experience shortage and balancing shortage through other depots results in additional routing costs. For $s = 2$ we compute the optimal inventory policy as $S_1 = S_2 = 50$ and $S_3 = 59$. This policy reduces the routing costs by about 0.5% compared to the policy $S_1 = S_2 = S_3 = 53$, which confirms our hypothesis that depots with a higher variance in demand should be assigned higher up-to-order levels to minimize routing efforts.

## 6.7 Conclusion

In this chapter we proposed a simple model and simulation framework that enables the integrated analysis of distribution activities and inventory policies. Different depots can collaborate and pool their inventory or their distribution. If the depots have the same cost structure, the optimal inventory policy for the pooling of distribution corresponds to the optimal inventory policy if inventory is pooled via transshipments. This observation allows to determine the optimal inventory policy without the computation of routing solutions.

In a set of experiments we compared the different collaboration strategies. Pooling of inventory via transshipments results in a significant reduction in inventory costs at the expense of a slight increase in routing costs. As observed in previous studies, these effects intensify with more collaborating depots. If the distribution activities are pooled as well, the additional routing costs can be reduced by several percent. In the context of the the defined setup, pooling inventory via joint distribution plans appears more cost-effective than pooling inventory via transshipments, independent of the relative distance between the depots and the shortage factor. Furthermore, the variance in demand impacts the effectiveness of pooling in terms of routing costs. A higher variance in demand corresponds to a higher routing effort to effectively balance inventory levels. Thus, pooling is more cost-effective in logistic systems in which variance is low. Finally, we can obtain slightly lower routing costs if the inventory policies in asymmetric logistic systems are adapted. Central depots and depots with a higher variance in demand should be assigned higher up-to-order levels.

The presented model captures the basic concepts of inventory management, and future research efforts could be devoted to adding more details and adapting more parameters. A positive and maybe even stochastic lead time could impact the decision-making, and the distinction between multiple products would further enrich the considered system. From the routing side, we assumed that routing decisions are extremely flexible and not time-dependent whereas in reality new demand might occur while other deliveries are already on the way. This would further complicate the analysis of pooling distribution.

# 7

# Solving the location–routing problem

## 7.1 Introduction

So far, we assumed that the location of depots is fixed. In this chapter we study the problem that results when the depots can be chosen among a number of options: The location–routing problem (LRP). The LRP is a well-established combinatorial optimization problem that combines two important decisions in the design of a supply chain: the decision where to open facilities and the decision how to set up the distribution from facilities to customers. In practice, both decisions are usually taken for different time horizons. While the opening of facilities affects the long-term, the distribution, also called routing, is mostly planned on an operational day-by-day basis. Despite the different planning horizons, Salhi and Rand (1989) have shown that the incorporation of distribution planning into a facility location decision can significantly improve the overall costs of the supply chain, and thereby established a practical motivation for solving LRPs.

The LRP is a generalization of standard routing problems like the Vehicle Routing Problem (VRP) or the Multi-Depot Vehicle Routing Problem (MDVRP) and the facility location problem (FLP). Given a set of customers with known demand and a set of potential facility locations, one has to to determine how many and which facilities to open (as in a FLP), as well as to plan the delivery routes from the open facilities to the customers (as in VRPs and MDVRPs). Hereby, the demand of all customers has to be satisfied, and the capacity limits per route have to be met. An example of an LRP solution is given in Figure 7.1, and a concise overview about the recent progress in the domain of LRPs can be found in Prodhon and Prins (2014) and in Schneider and Drexl (2017). For the purpose of a uniform terminology, we will use the word *depot* (as in VRPs) to denote facilities in the following.

In the last years a number of effective heuristics have been proposed to tackle LRPs. Almost all successful heuristics consists of different stages and combine heuristic components with integer linear program formulations (ILPs). The improvement of the routing is usually carried out by heuristics, whereas the decisions which depots to open is taken by ILPs. One of the most efficient algorithms with such a design has been proposed by Escobar et al. (2013) and improved in Escobar et al. (2014). Initially, the customers are divided into clusters and an ILP assigns customer clusters to depots such that the routing from the open depots is minimized. Afterwards, the routing is improved with various heuristic techniques. Prins et al. (2007) proposed an algorithm which exchanges information between these two stages. The FLP is solved by a Lagrangean relaxation technique, followed by a granular tabu search to solve the corresponding MDVRP. A similar design is proposed in Harks et al. (2013) to solve very large scale LRP instances. In a first stage a solution for the corresponding FLP is approximated using minimal spanning trees, and in a second stage the delivery routes are constructed and improved. In contrast to that, Tuzun and Burke (1999) iterate between the improvement of the location and routing decisions, rather than solving them one after another. Contardo et al. (2014) suggest yet another heuristic design by optimising locations and routing simultaneously. A GRASP technique combined with local search is used to construct a set of promising solutions, which are then iteratively improved with a mix of ILP-based techniques and destroy-and-repair techniques. The most successful of these heuristics are usually complex, both in terms of their design and in the number of parameters. Even though they are able to find good solutions in a relatively short time, simplicity and flexibility are also valuable properties of a heuristic (Cordeau et al., 2002), especially when it comes to practical usability.

In this chapter, we demonstrate that it is possible to design an LRP heuristic with a relatively simple design that is able to compete with the best results in literature. More concretely we provide the following contributions to the research on LRPs:

- We estimate an upper bound for the number of open depots, which drastically reduces the search space.

- We show how an LRP can entirely be solved by a routing heuristic. If we can solve an MDVRP, we can solve an LRP.

- We show that the heuristic is the first which is able to solve LRPs of different magnitudes, and can even tackle instances with 10,000 customers and 1,000 potential depots.

We demonstrate how LRPs can be efficiently decomposed into a FLP and a routing component in Section 2. In Section 3 we use these insights to design a simple filtering heuristic which can also be extended to other problem variants. The performance of the heuristic is validated in Section 4 on popular benchmark instances. We conclude with a brief summary of our findings in Section 5.

## 7.2 Decomposition of location and routing decisions

The objective of the LRP is to open a certain number of depots at different possible locations from which customers are delivered, such that the joint cost of opening the depots and distribution from the depots to customers are minimized. More formally, a candidate set $C = \{f_1, f_2, \ldots, f_F\}$ of potential depot locations is given, and each depot is annotated with a cost $c_o(f_i)$ that is realized if the respective depot is opened. From all open depots a set of $N$ customers with pre-defined locations and demand has to be delivered on routes with vehicles. Each customer has to be visited once, and each vehicle has a maximum capacity so that usually multiple routes have to be planned. In the standard problem formulation, the capacity of the depots is unconstrained and all customers could be supplied from a single depot. Below, we also consider a problem variant in which the capacity of depots is limited. For a more elaborate overview of the LRP and its variants we refer to Prodhon and Prins (2014).

The optimization task is to determine a set $O \subseteq C$ of open depots, as well as a routing solution from the open depots to satisfy the demand of the customers, that result in minimal overall costs. In the following, we will denote a set of open depots $O \subseteq C$ as *location option*. Let $R_O$ be the routing costs (in most benchmark instances $R_O$ is defined as the sum of the distance traveled on the delivery routes and a fixed costs per route), then the costs of a location option $O$ can be expressed by $c_{LRP}(O) = \sum_{f_i \in O} c_o(f_i) + R_O$.

If the set of open depots is given, then the first term in the objective function can be neglected, and the problem reduces to computing delivery routes with minimal costs. The computation of optimal delivery routes, or Vehicle Routing Problem (VRP), is one of the most-studied problems in the field of Operations Research. The standard variant of the VRP only considers the delivery from one depot, and could solve the routing in the case of $|O| = 1$. For $|O| > 1$ we have to solve a Multi-Depot Vehicle Routing Problem (MDVRP).

Since the VRP and MDVRP are well-studied problems, there is a strong intuition to decompose an LRP into its two subproblems, an FLP and an MDVRP. This intuition has been picked up in many LRP heuristics as described in the introduction (Prins et al., 2007; Escobar et al., 2013, 2014; Tuzun and Burke, 1999). Instead of computing only one FLP solution, one could also compute a number of promising options. More concretely, an LRP could be solved by (1) the determination of promising location options $O_1, O_2, \ldots$, and (2) the computation of delivery routes for each option. This decomposition is visualized in Figure 7.1. Since the quality of a certain location option can only be determined after a routing solution has been computed, one could also say that the routing solution constitutes an *evaluation* of a location option. If we can execute this evaluation in a short time, we could evaluate and compare a large number of location options.

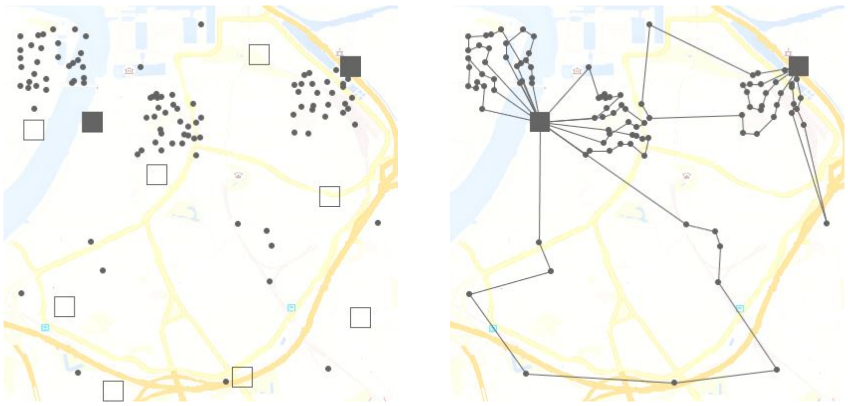


Figure 7.1: Decomposition of an LRP. We first generate location options (left), and evaluate each location option by solving the respective MDVRP (right). The example is taken from instance '112112' by Tuzun and Burke (1999).

In the last decades, remarkable research efforts have been spent to tackle this problem, and efficiently compute high-quality routing solutions for VRPs and MDVRPs. State-of-the-art routing heuristics can solve MDVRPs with 100 customers almost optimally in a few seconds (Vidal et al., 2012). In view of this progress, an algorithm that completely decomposes an LRP and purely relies on an effective computation of routing solutions does seem feasible, especially since the evaluation of a location option $O_1$ does not necessarily require to solve the respective routing problem to optimality. In order to make a statement about whether $O_1$ represents a good location option (e.g., it is better than $O_2$), it might be sufficient to solve the corresponding MDVRP up to a certain quality, or in other words, to approximate the routing. The feasibility of this approach relies on the assumption that the number of location options that needs to be evaluated is limited or can be reduced.

### 7.2.1 *Finding promising location options*

The number of candidates for depot locations $F = |C|$ is rather limited in popular benchmark instances. On most instances we have up to 10 possible locations for opening depots, and on some we have up to 20 (Prodhon and Prins, 2014). The resulting number of possible location options seems manageable in comparison to other combinatorial problems. Since we have $\binom{F}{f}$ possibilities to open exactly $f$ depots on $F$ possible locations, the number of location options amounts to $\sum_{f=1}^{F} \binom{F}{f} = 2^F - 1$. For $F = 10$ this adds up to 1023 options. Where usually heuristics have to ignore large parts of the search space, this limited number of options allows us to conduct a complete search through the search space. In fact, we could evaluate all location decisions for $F = 10$ possible locations and $N = 100$ customers with a fast construction heuristic like the one by Clarke and Wright (1964) in a few seconds.

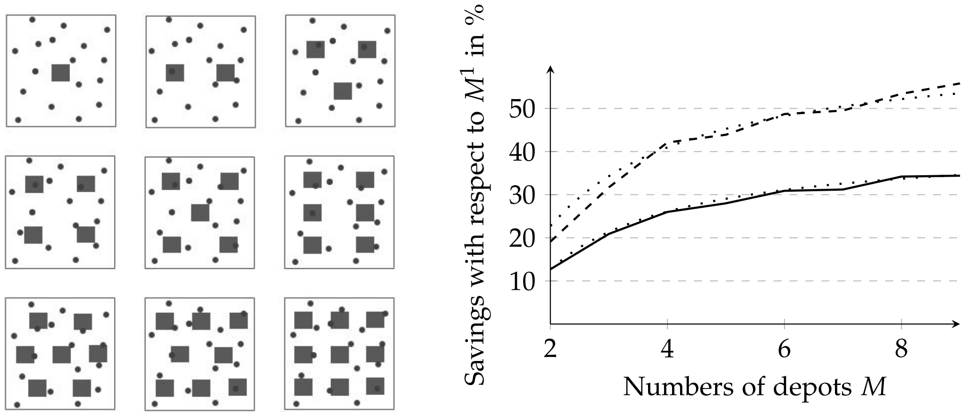


Figure 7.2: (right) Average reduction in routing costs when delivering from an increasing number of depots, with respect to delivering from a single depot: solid line $N = 100$, dashed line $N = 1000$, the respective dotted lines present the approximated functions. (left) The depots are distributed uniformly in a grid-pattern.

However, we could significantly increase the efficiency of a decomposition-based LRP heuristic, and also generalize the heuristic to instances of larger size, if we can exclude unpromising location options before their routing evaluation.

Opening depots usually comes at a cost, but it generally can be expected that more open depots enable a more efficient routing. With more open depots, the average distance between customers and depots decreases, assuming the depots are opened at 'good' locations. Therefore, there should be a trade-off between opening costs and routing costs.

We investigate this trade-off in a set of experiments in which we generate and solve MDVRP instances. The instances are generated in such a way that 100 customers are randomly placed in a square and visited from an increasing number of depots on delivery routes that can visit 10 customers on average. To capture the idea that depots are opened in good positions, we locate the depots evenly on the squared plane. Since the customers are located according to a uniform distribution, an even distribution of the depots should minimize the average distance between a customer and its nearest depot, and thus, this setup should represent a decent location option. To obtain an even distribution we use a grid pattern, as visualized in Figure 7.2. The algorithm of how to generate this grid pattern is illustrated in the appendix. Starting with a single central depot, we generate 100 instances and compute solutions with the routing heuristic outlined below, allowing 10 seconds of computation time, and obtain the average routing costs $R^1$ per instance. We repeat this step with an increasing number of depots $M$, and compute the reduction in routing costs with respect to a single depot $r(M) = \frac{R^1 - R^M}{R^1}$.

The results in Figure 7.2 confirm the hypothesis that more open depots result in lower routing costs. Even though the data does not result in a smooth curve, the marginal benefit of another open depot generally decreases with more depots. The observed reduction in routing costs $r(M)$ for $M$ depots with respect to a single open depot can be approximated by the function $r(M) = 1/2^{\frac{1}{M}} - 0.58$. This function is derived on the basis of empirical observations on a specific instance setup and thus presents an approximation, rather than an analytically-proven relationship. For instance, when we repeated the experiments for larger instances with 1,000 customers, we observed that more open depots have greater benefits.

Consequently, the approximation of $r(M)$ is certainly not precise enough to compute the optimal number of open depots for any LRP instance. However, it can be beneficial in the estimation of some upper bound for the number of open depots. For simplicity, assume that each potential depot has the same opening costs $c_o$, in the case that the opening costs are not constant let $c_o$ be the average opening costs. Then the sum of opening costs increases linearly in the number of open depots, while the marginal reduction in routing costs diminishes with more open depots, as shown in Figure 7.2. Thus, for some number of open depots, the costs of opening another depot will outweigh the benefits in terms of routing. We can derive an estimate for this upper bound $M^U$ by determining the smallest $M$ for which we have

$$R^1 \cdot (r(M) - r(M-1)) < c_o, \tag{7.1}$$

where $R^1$ is an approximation of the routing costs from a central depot. This equation holds, if the estimated savings in terms of routing are lower than the costs of opening a depot, and it is thus not worthwhile to open the depot.

Given a particular LRP instance, we compute $M^U$ by determining the most central depot and by using the simple construction heuristic by Clarke and Wright to estimate $R^1$. The most central depot is defined as the depot with the smallest average distance to all customers. We can then derive $M^U$ with the above formula and exclude all location options with more than $M^U$ open depots from the search. This cut-off can drastically reduce the number of location options which need to be evaluated. For some benchmark instances with $F = 20$ we obtain $M^U = 3$ (e.g., for instance 112222). Therefore, instead of looking at all possible $2^{20} - 1 = 1,048,575$ location options, we only consider all $\sum_{f=1}^{3} \binom{20}{f} = 6,195$ options with less than 4 depots, a reduction by 99.4%.

If the capacities in the depots are constrained, $M^U$ might constitute too low an upper bound to obtain a feasible solution. This occurs on the instances by Prins et al. (2006), in which opening costs are relatively high and depots have limited capacities, so that a minimum number of depots have to be opened to supply all customers. In this case, we simply choose the upper bound as the maximum of $M^U$ and the minimal number of open depots with which we can satisfy all demand. If the capacity constraints are relatively loose, the corresponding upper bound is likely to be $M^U$, but if they are tight, this approach ensures that we do not open more depots than we have to. In

Table 7.3 and in Table 7.4 in the appendix we list the computed upper bounds for the 36 instances by Tuzun and Burke (1999) and the 30 instances by Prins et al. (2006), together with the number of open depots that we observe in the best solutions in the experiments in Section 4. On most instances without capacitated depots, we observe that the derived upper bounds slightly overestimate the number of open depots in the best solutions, while on most instances with constrained capacities in the depots, as few depots as the constraints allow are opened. We cannot validate that the derived upper bounds hold for all instances, since we do not always obtain the best known solution, and for some instances the optimal solutions are not yet known. However, we obtain high-quality solutions with these upper bounds so that they are likely to include the best location options, while drastically reducing the search space. The remaining location options need to be evaluated with a routing heuristic.

### 7.2.2 *Efficient approximation of routing solutions*

We evaluate promising location options with the heuristic defined in Chapter 3. The heuristic has been shown to be effective for many VRP and MDVRP benchmark instances, and computes near-optimal solutions for MDVRP instances with up to 300 customers in a few seconds. Even though the quality of the computed solutions (also called accuracy) is an important metric, in the context of LRPs we have to evaluate many location options as fast as possible and as accurate as necessary. The evaluation should provide a good indication of whether a location option is better or worse than other options, rather than being perfectly accurate. Since longer computation times typically allow to compute routing solutions of higher quality, the question arising is 'how long do we need to evaluate a location option?'.

To obtain an intuition about the performance behaviour of the heuristic over time, we generated and solved 100 randomly-sampled MDVRP instances with 100 and 200 customers being delivered from two depots. These instances are produced with the same mechanism as outlined in the previous section. The results in Figure 7.3 present the average performance of the heuristic over time, with respect to the final solutions computed after 1 minute of computation time. We observe that the CW algorithm computes reasonable solutions with a gap of about 8% almost instantaneously (0.002 seconds for $N = 100$ and 0.004 seconds for $N = 200$). Within the first second of computation time, the heuristic improves the quality of the initial solution significantly, and after two seconds the solutions are usually within a 1% range of high-quality solutions. Hereby, the computational effort to obtain marginal improvements increases with better solutions. In general, for MDVRPs with more customers it requires more time to obtain the same accuracy.

Even though the performance curves might be different for other instance types where customers are, for example, more clustered, these results provide a good

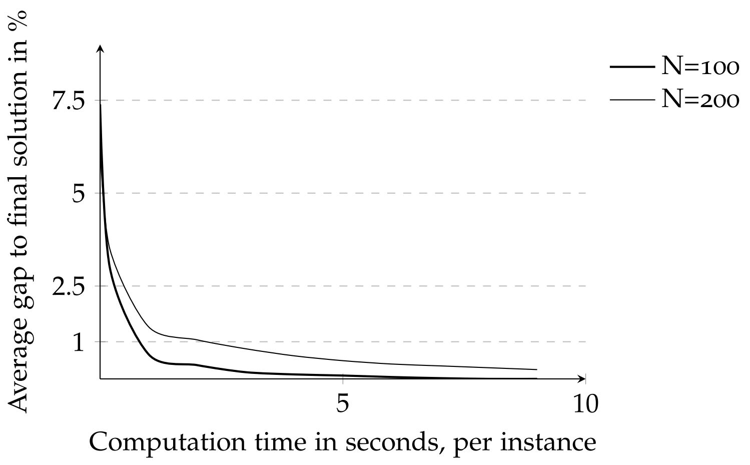


Figure 7.3: Average performance of the routing heuristic on 100 random MDVRP instances over time, depending on instance size.

intuition about how fast the heuristic is able to obtain a certain solution quality, and thus, how fast and how accurate it can evaluate a certain location option.

## 7.3 **Solving location–routing problems through iterative filtering**

We have outlined how to use problem knowledge to reduce the number of promising location options in LRPs to a manageable size, and how a state-of-the-art routing heuristic can evaluate these options quite accurately within a few seconds. Together, these findings lead to a simple heuristic design to solve LRPs: evaluate promising location options with a routing heuristic, and choose the best location option, together with the respective routing solution, as the solution for the LRP. However, an accurate evaluation of all promising location options might be time-consuming. As an example, if we allow 10 seconds to compute routes for each of 6,195 location options in the example above, the heuristic would require more than 17 hours of computation time. On the other hand, an evaluation with CW can be executed in a fraction of a second, but it might be too imprecise to identify the best location option. The time-quality trade-off observed in Figure 7.3 suggests that an iterative approach might be an efficient compromise. We evaluate a location option until we are sufficiently confident that there exists a better option.

More formally, given a set of location options, we approximate the routing solution for each option. This allows us to approximate $c_{LRP}$ for each option, as well as to determine the best option with costs $c^*_{LRP}$. We then remove all location options that are not within a $t$% range of the best one, i.e., $\frac{c_{LRP}}{c^*_{LRP}} > 1 + t$. The set of location options is, in a manner of speaking, filtered. Less options remain and we can evaluate them more accurately. Iteratively, we reduce the number of remaining location options and

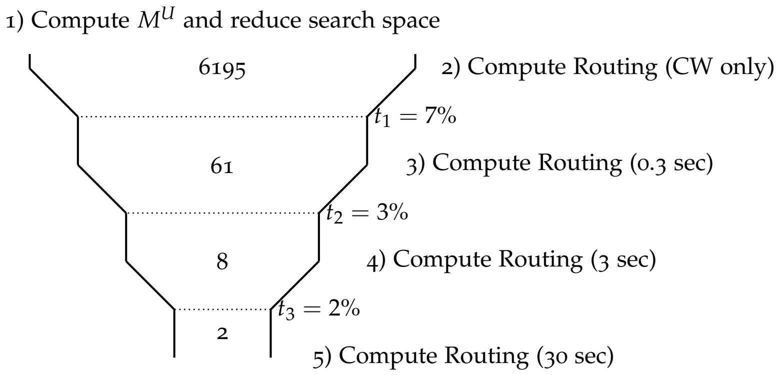

1) Compute $M^U$ and reduce search space
6195
2) Compute Routing (CW only)
$t_1 = 7\%$
61
3) Compute Routing (0.3 sec)
$t_2 = 3\%$
8
4) Compute Routing (3 sec)
$t_3 = 2\%$
2
5) Compute Routing (30 sec)

Figure 7.4: Illustration of the iterative filtering. At each stage, the routing heuristic is given more time, while less location options are accepted. The number of remaining location options as well as the computation times are taken from instance '111122'.

increase the accuracy of the routing evaluation. We continue this iterative filtering until only few solutions remain and we can run the routing heuristic for a maximum runtime. More concretely, given an instance with $N$ customers, the iterative filtering involves the following steps.

1) Determine the most central depot and approximate $R^1$ with CW. Compute $M^U$ with equation 7.1, obtain an upper bound, and remove all location options in which the number of open depots exceeds the upper bound.

2) For each remaining location option, compute a routing solution with CW (about 0.002 sec for $N = 100$). Remove all location options with an objective value greater than $t_1 = 7\%$ of the best one (or keep the 100 best).

3) For each remaining location option, compute a routing solution, applying the initial optimisation step of the routing heuristic. Remove all location options with an objective value greater than $t_2 = 3\%$ of the best one (or keep the 10 best).

4) For each remaining location option, compute a routing solution, applying the initial optimisation step of the routing heuristic and additionally 100 iterations of perturbation and optimisation. Remove all location options with an objective value greater than $t_3 = 2\%$ of the best one (or keep the 3 best).

5) For each remaining location option, compute a routing solution, running the heuristic for $3\frac{N}{10}$ sec. The location option (together with the routing solution) with the best objective value constitutes the final solution for the LRP.

An overview of this filtering framework is illustrated in Figure 7.4. This heuristic design is simple, scalable and readily extendible to other problem variants, since the routing heuristic can be treated as a black box. It is up to the developer how much effort he wants to invest into the routing heuristic. The first stages of the heuristic only requires the implementation of the CW algorithm. The solution quality will improve with the implementation of additional local search operators, and if one implements a fully-fledged routing heuristic, one can obtain high-quality LRP solutions as demonstrated below.

The performance of this framework hinges on the conditions that (1) no stage filters out the best (or one of the best) location option, and that (2) the last stage computes a high-quality routing solution for the remaining options. If both conditions are met, it is likely that the best location option is identified and an near-optional routing solution is computed. In practice, however, it is possible that the best location options only reveal themselves after a certain amount of routing effort and might not pass one of the filters. We therefore also experimented with different thresholds and time limits per filter, however, we observed only little changes in solution quality. The above setup appears to work well for many instances, and we keep the same parameter configuration for all instance types.

### 7.3.1 *Capacitated location–routing problems*

One of the most popular variants of LRPs imposes a limit on the demand that can be fulfilled from each depot. More formally, the aggregated demand that is fulfilled on tours originating from a depot must not exceed the capacity limit of the depot. Solving instances of this kind with standard heuristics for MDVRPs could result in infeasible solutions. In Chapter 5 we developed an extension for MDVRPs that considers inventory constraints in the depots, and can also distinguish between different products. This problem variant can be reduced to solve MDVRPs with depot capacities, if we consider a single product and the available inventory in the depots corresponds to the capacity of the depots.

The routing heuristic above can be readily adapted to solve instances of this type. The initial solution is created in a greedy fashion, where each customer is delivered by the nearest depot that still has sufficient capacity, and the subsequent local search only considers moves that maintain feasibility with respect to all constraints. Location options, in which too few depots are open to deliver all customers, can be excluded from the search.

### 7.3.2 *Large location–routing problems*

The iterative filtering framework relies on the two assumptions that the number of initial location options is manageable, and that the corresponding routing problems

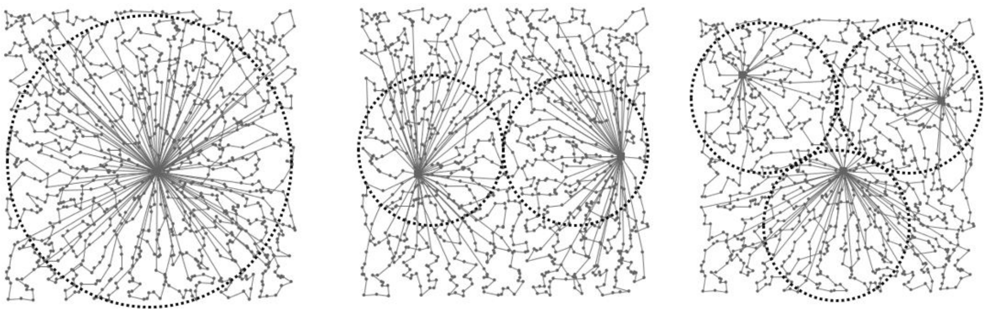


Figure 7.5: Location options for large scale LRPs are chosen in a grid-like pattern. For increasing $M$, we place $M$ circular regions in the plane, and in each region we open one depot, before computing a routing solution with CW. The routing of the best location option generated in this way is improved further.

can be solved efficiently. In LRPs in which the number of potential depot locations $F$ or the number of customers $N$ is exceedingly large (or both), these assumptions might no longer hold, and the framework might no longer be computationally feasible.

Such very large instances were introduced by Harks et al. (2013), with up to 10,000 customers being delivered from up to 1,000 possible depot locations. To overcome this complexity, we need to reduce the number of initial location options even more drastically, and limit the routing evaluation. Drawing on the intuition from Section 2, if customers are distributed more or less uniformly in the considered plane, then the open depots should be located uniformly as well. Two open depots in close proximity are likely to be less beneficial, than two open depots that are spaced more moderately (given that they have similar opening costs), and in extreme cases all open depots are spaced equidistantly. In other words, we assume that in good location options the open depots are located in a grid-pattern as outlined in Figure 7.2. The respective algorithm is given in the appendix.

We identify such location options, by firstly distributing $M$ points on the plane such that they form a grid. Each point is the center of a circular region, the radius of which equals half the distance to the neighboring point. In each region we try to open one depot. The depot should be close to the center of the region and have low opening costs. Thus, for each depot in the region we compute the sum of the distance to the center and the opening costs, and open the depot with the lowest value. For larger $M$, the regions become smaller, and it can occur that there is no potential depot in a region. In this case, we simply ignore the region and open less depots in total. After all regions have been investigated, we have generated a location option.

This location option is then evaluated with the heuristic of Clarke and Wright (1964). The time complexity of CW grows quadratically in the number of customers, and thus, for large $N \geq 1000$ we might observe long runtimes. In Chapter 4 we proposed a simple idea to linearize this complexity for very large VRPs, by only considering edges between a customer and its 100 nearest customers. We observed that this approach yields solutions of similar quality while significantly reducing computation time. In summary, for each $M \in \{1, 2, \ldots, 200\}$ we generate a location option, and evaluate this location option with the fast version of CW. This process is illustrated in Figure 7.5. For simplicity, we choose an upper bound for $M$ of 200 for each instance. We identify the best location option and apply the routing heuristic for $\frac{N}{100}$ seconds. Since a routing evaluation requires more computation time for very large $N$, we decide not to use any more filters.

This design requires only a small modification of the heuristic above, and, even though it appears simplistic, it has the potential to obtain good LRP solutions in a very short time. It can be extended to obtain better solution, e.g., by taking more location options into consideration or by allowing more computation time for the improvement of the routes.

## 7.4 Computational experiments

We test the performance of the heuristic on the basis of the most popular benchmark sets by Tuzun and Burke (1999) ($\mathbb{T}$) and Prins et al. (2006)($\mathbb{P}$). The $\mathbb{T}$-instances have between 100 and 200 customers located on a squared plane that have to be delivered from up to 20 potential depot locations with uniform opening costs. The $\mathbb{P}$-instances constitute a complementary benchmark set with capacitated depots and varying opening costs, while the instances are of similar size with 20-200 customers and up to 10 potential depot locations. Additionally, we test the scaling of the heuristic on the recently introduced benchmark set by Harks et al. (2013). These instances comprise between 1,000 and 10,000 customers that can be delivered from up to 1000 potential locations with varying opening costs.

The performance is compared to the most effective LRP heuristics in literature: the GRASP+ILP metaheuristic by Contardo et al. (2014), the granular variable neighborhood search (GVTNS) by Escobar et al. (2014), and the tree-based search algorithm (TBSA) by Schneider and Löffler (2017). These heuristics have only been applied to the first two benchmark sets. The large instances have so far only been tackled by Harks et al. (2013) (Approx+TSP) and Guemri et al. (2016) (2-SH), and we compare with both methods. The above heuristic has been implemented in Java and all tests have been performed within the Eclipse development environment on an AMD Ryzen 3 1300X CPU working at 3.5GHz on Windows 10, using a single thread. To allow a fair comparisson of computation times, we normalise the CPU times with respect to the processor (PassMark Software, 2018), as in Schneider and Drexl (2017)

Table 7.1: Results on the 𝕋-instances and the ℙ-instances, based on Schneider and Drexl (2017).

| | GRASP+ILP | | GVTNS | | TBSA$_{speed}$ | | TBSA$_{quality}$ | | A&S | |
|---|---|---|---|---|---|---|---|---|---|---|
| | Gap | Time* | Gap | Time* | Gap | Time* | Gap | Time* | Gap | Time* |
| 𝕋 | 0.66 | 1379 | 0.86 | 68 | 0.57 | 48 | 0.15 | 725 | 0.30 | 141 |
| ℙ | 0.38 | 619 | 0.43 | 31 | 0.20 | 23 | 0.02 | 452 | 0.13 | 55 |

* normalized with PassMark Software (2018).

Table 7.2: Results on the large ℍ-instances.

| Approx+TSP | | 2-SH | | A&S (only CW) | | A&S (improved routes) | |
|---|---|---|---|---|---|---|---|
| Gap | Time | Gap | Time | Gap | Time | Gap | Time |
| 12.42 | 221 | 11.01 | 66 | 5.37 | 93 | 0.00 | 146 |

(for the comparison on the larger instance set the computation time is less important and we do not normalize here).

The average performance of the above heuristic (denoted A&S) on the LRP benchmark sets are presented in Table 7.3 and Table 7.4. For the 𝕋-instances and ℙ-instances we present the average gaps with respect to the best known solutions (BKS) as defined in Schneider and Drexl (2017), and for the ℍ-instances the gaps are expressed with respect to the results of this heuristic. All times are given in seconds. The detailed results per instance can be found in the appendix. GRASP+ILP and TBSA include stochastic components and, thus, we compare with the average performance of these heuristics, whereas all other heuristics are deterministic and obtain the same solution after every run.

For the smaller benchmark sets, we observe that the filtering heuristic computes high-quality solutions that are very close to the best known solutions for almost all instances. Overall, the heuristic achieves a better accuracy than GVTNS and GRASP+ILP and a similar performance than TBSA on both benchmark sets. All of these solutions are computed in short computation times, which place the heuristic among the most efficient ones in literature. The corresponding quality–time tradeoff is highlighted in Figure 7.6.

The heuristic can also tackle large problems successfully, as demonstrated by the results on the ℍ-instances. It improves the best known results on almost all instances by more than 11% on average in a comparable computation time. Notably, the relatively simple approach of opening depots in a grid-like pattern and constructing routes with CW already improves the previous solutions by more than 6%.

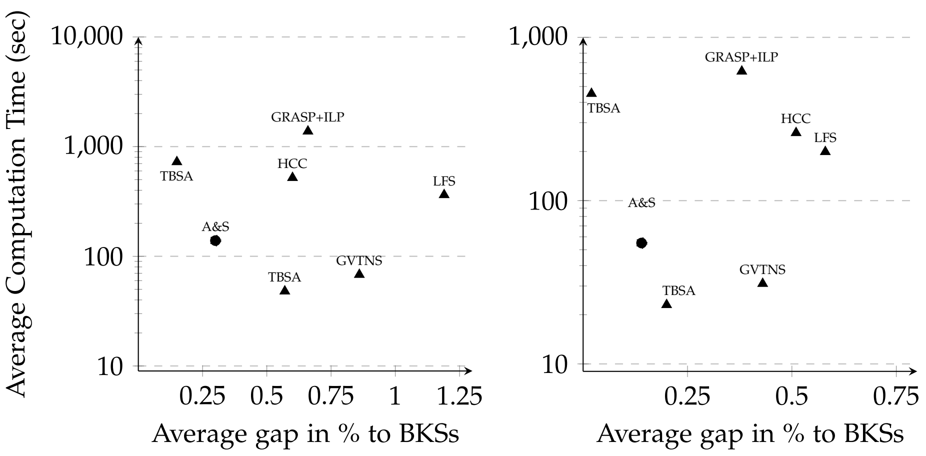


Figure 7.6: Performance versus computation time in comparison to state-of-the-art heuristics on the $\mathbb{T}$-instances (left) and the $\mathbb{P}$-instances (right), based on Schneider and Drexl (2017).

In summary, not only is the heuristic the first which can be successfully applied to LRPs of different problem sizes, but it also constitutes one of the most efficient heuristics for most LRP benchmark instances in literature. At the same time, its simple design allows for a scalable and flexible implementation. The heuristic can also be readily parallelized, since the evaluation of location options at each filtering stage can be executed simultaneously. This parallelization would increase efficiency further.

## 7.5 Conclusion

In this paper we have designed a heuristic for LRPs that is almost entirely based on a routing heuristic. We estimate an upper bound for the number of open depots to reduce the number of promising location options, which are then evaluated with a routing heuristic. The improvements found by the routing heuristic decline over time, and thus, we use a filtering framework to iteratively decrease the number of promising options in different stages. In each stage, the accuracy of the computed routing solutions increases. The same heuristic, with some minor modifications, can also be used to solve LRPs with capacitated depots and large LRPs. Computational tests on benchmark sets show that the resulting heuristic can effectively solve a wide range of LRP instances within short computation times.

The heuristic design is flexible and scalable, and therefore useful in practical cases where implementation time is limited. Already a simple CW implementation can result in decent results, while a state-of-the art routing heuristic is able to compete with the best results in literature. In more general terms, these findings suggest that the reduction of an LRP to a routing problem constitutes a successful heuristic approach. It decomposes the problem and results in a straight-forward heuristic design. The same approach could be used to solve other LRP variants, e.g., multi-echolon LRPs, or other problems that revolve around routing, e.g., the multi-trip VRP.

## 7.6 Appendix: Placement algorithm and detailed benchmark results

**Algorithm 4** Placing $M$ points in a square with length $L$

```
1: $cols \leftarrow \lceil \sqrt{M} \rceil$
2: $rows \leftarrow \lceil \frac{M}{cols} \rceil$
3: $shortRows \leftarrow cols \cdot rows - M$
4: $d_x \leftarrow \frac{L}{cols}$
5: $d_y \leftarrow \frac{L}{rows}$
6: $makeShort \leftarrow true$
7: if $shortRows < \frac{rows}{2}$ then
8:     $makeShort \leftarrow false$
9: end if
10: $cur_y \leftarrow \frac{d_y}{2}$
11: $cur_{row} \leftarrow 0$
12: while $cur_{row} < rows$ do
13:     if $shortRows > 0$ AND $makeShort$ then
14:         $points \leftarrow cols - 1$
15:         $cur_x \leftarrow d_x$
16:         $makeShort \leftarrow false$
17:         $shortRows \leftarrow shortRows - 1$
18:     else
19:         $points \leftarrow cols$
20:         $cur_x \leftarrow \frac{d_x}{2}$
21:         $makeShort \leftarrow true$
22:     end if
23:     $cur_{col} \leftarrow 0$
24:     while $cur_{col} < points$ do
25:         $placePoint(cur_x, cur_y)$
26:         $cur_x \leftarrow cur_x + d_x$
27:         $cur_{col} \leftarrow cur_{col} + 1$
28:     end while
29:     $cur_y \leftarrow cur_y + d_y$
30:     $cur_{row} \leftarrow cur_{row} + 1$
31: end while
```

Table 7.3: Results on the $\mathbb{T}$-instances by Tuzun and Burke (1999). Gap in % to the BKS, time in seconds, estimated upper bounds of open depots $U$ and obtained number of open depots $M$.

| Instance | BKS | GRASP+ILP | | | GVTNS | | | A&S | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | Value | Gap | Time | Value | Gap | Time | Value | Gap | Time | M | U |
| 111112 | 1467.68 | 1475.50 | 0.53 | 198 | 1479.21 | 0.79 | 84 | 1468.29 | 0.04 | 101 | 3 | 4 |
| 111122 | 1448,37 | 1452.00 | 0.25 | 580 | 1485.28 | 2.55 | 126 | 1449.20 | 0.06 | 104 | 2 | 4 |
| 111212 | 1394.80 | 1405.80 | 0.79 | 220 | 1402.59 | 0.56 | 74 | 1394.80 | 0.00 | 105 | 2 | 4 |
| 111222 | 1432.29 | 1440.60 | 0.58 | 755 | 1463.23 | 2.16 | 99 | 1432.29 | 0.00 | 104 | 2 | 4 |
| 112112 | 1167.16 | 1176.20 | 0.77 | 278 | 1167.16 | 0.00 | 83 | 1167.16 | 0.00 | 35 | 2 | 4 |
| 112122 | 1102.24 | 1103.60 | 0.12 | 634 | 1102.24 | 0.00 | 105 | 1102.24 | 0.00 | 67 | 2 | 4 |
| 112212 | 791.66 | 795.80 | 0.52 | 227 | 791.66 | 0.00 | 96 | 791.66 | 0.00 | 35 | 2 | 3 |
| 112222 | 728.30 | 728.50 | 0.03 | 550 | 728.30 | 0.00 | 126 | 728.30 | 0.00 | 66 | 2 | 3 |
| 113112 | 1238.44 | 1239.60 | 0.11 | 286 | 1238.49 | 0.02 | 82 | 1238.49 | 0.02 | 97 | 3 | 4 |
| 113122 | 1245.30 | 1246.30 | 0.09 | 646 | 1247.27 | 0.17 | 127 | 1245.31 | 0.00 | 104 | 3 | 4 |
| 113212 | 902.26 | 902.80 | 0.06 | 231 | 902.26 | 0.00 | 71 | 902.26 | 0.00 | 65 | 3 | 4 |
| 113222 | 1018.29 | 1018.29 | 0.00 | 749 | 1018.29 | 0.00 | 85 | 1018.29 | 0.00 | 104 | 3 | 4 |
| 131112 | 1892.17 | 1924.10 | 1.68 | 1640 | 1933.67 | 2.19 | 179 | 1892.17 | 0.00 | 149 | 3 | 5 |
| 131122 | 1819.68 | 1831.00 | 0.62 | 3612 | 1852.14 | 1.78 | 173 | 1826.69 | 0.39 | 162 | 4 | 4 |
| 131212 | 1960.02 | 1969.30 | 0.47 | 1275 | 1983.09 | 1.18 | 184 | 1960.02 | 0.00 | 110 | 3 | 4 |
| 131222 | 1792.77 | 1800.30 | 0.43 | 3099 | 1803.01 | 0.58 | 175 | 1792.77 | 0.00 | 161 | 3 | 5 |
| 132112 | 1443.32 | 1450.40 | 0.49 | 871 | 1443.43 | 0.20 | 186 | 1446.53 | 0.22 | 146 | 2 | 4 |
| 132122 | 1429.30 | 1447.20 | 1.25 | 2738 | 1441.43 | 0.84 | 210 | 1444.66 | 1.07 | 118 | 2 | 4 |
| 132212 | 1204.42 | 1205.90 | 0.12 | 2082 | 1204.42 | 0.00 | 128 | 1204.85 | 0.04 | 145 | 3 | 4 |
| 132222 | 924.68 | 931.90 | 0.78 | 3734 | 931.28 | 0.71 | 177 | 931.43 | 0.73 | 52 | 3 | 4 |
| 133112 | 1694.18 | 1703.80 | 0.57 | 938 | 1701.34 | 0.42 | 182 | 1709.26 | 0.89 | 148 | 3 | 4 |
| 133122 | 1392.01 | 1401.50 | 0.68 | 2751 | 1416.74 | 1.78 | 175 | 1400.50 | 0.61 | 170 | 3 | 4 |
| 133212 | 1197.95 | 1199.60 | 0.13 | 1010 | 1213.87 | 1.32 | 207 | 1200.24 | 0.19 | 96 | 3 | 4 |
| 133222 | 1151.37 | 1158.70 | 0.64 | 3560 | 1151.80 | 0.04 | 208 | 1156.61 | 0.46 | 159 | 3 | 4 |
| 121112 | 2237.73 | 2251.30 | 0.61 | 2805 | 2258.02 | 0.91 | 315 | 2255.26 | 0.78 | 205 | 3 | 5 |
| 121122 | 2137.45 | 2154.90 | 0.82 | 5680 | 2166.20 | 1.35 | 300 | 2142.73 | 0.25 | 286 | 4 | 5 |
| 121212 | 2195.17 | 2226.10 | 1.41 | 3004 | 2239.65 | 2.03 | 287 | 2201.94 | 0.31 | 202 | 4 | 5 |
| 121222 | 2214.86 | 2241.70 | 1.21 | 6143 | 2236.73 | 0.99 | 351 | 2218.88 | 0.18 | 281 | 4 | 5 |
| 122112 | 2070.43 | 2093.80 | 1.13 | 3462 | 2103.82 | 1.61 | 278 | 2071.42 | 0.05 | 214 | 3 | 5 |
| 122122 | 1685.52 | 1704.40 | 1.12 | 8547 | 1717.92 | 1.92 | 433 | 1694.66 | 0.54 | 326 | 3 | 5 |
| 122212 | 1449.93 | 1467.80 | 1.23 | 3471 | 1469.45 | 1.35 | 318 | 1449.92 | 0.00 | 73 | 2 | 4 |
| 122222 | 1082.46 | 1086.70 | 0.39 | 5292 | 1082.46 | 0.00 | 349 | 1082.87 | 0.04 | 171 | 3 | 5 |
| 123112 | 1942.23 | 1986.70 | 2.29 | 3865 | 1969.38 | 1.40 | 261 | 1965.70 | 1.21 | 155 | 4 | 5 |
| 123122 | 1910.08 | 1936.20 | 1.37 | 9367 | 1935.74 | 1.34 | 344 | 1915.30 | 0.27 | 288 | 4 | 5 |
| 123212 | 1761.11 | 1766.20 | 0.29 | 3766 | 1776.90 | 0.90 | 349 | 1800.44 | 2.23 | 132 | 3 | 4 |
| 123222 | 1390.86 | 1392.70 | 0.13 | 5157 | 1391.50 | 0.05 | 317 | 1391.71 | 0.06 | 152 | 5 | 5 |

Table 7.4: Results on the $\mathbb{P}$-instances by Prins et al. (2006). Gap in % to the BKS, time in seconds, estimated upper bounds of open depots $U$ and obtained number of open depots $M$.

| Instance | BKS | GRASP+ILP | | | GVTNS | | | A&S | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | Value | Gap | Time | Value | Gap | Time | Value | Gap | Time | M | U |
| 20-5-1a | 54793 | 54793 | 0.00 | 2 | 54793 | 0.00 | 2 | 54793 | 0.00 | 6 | 3 | 3 |
| 20-5-1b | 39104 | 39104 | 0.00 | 3 | 39104 | 0.00 | 3 | 39104 | 0.00 | 6 | 2 | 2 |
| 20-5-2a | 48908 | 48908 | 0.00 | 1 | 48945 | 0.08 | 2 | 48908 | 0.00 | 6 | 3 | 3 |
| 20-5-2b | 37542 | 37542 | 0.00 | 3 | 37542 | 0.00 | 3 | 37542 | 0.00 | 6 | 2 | 2 |
| 50-5-1a | 90111 | 90111 | 0.00 | 15 | 90111 | 0.00 | 13 | 90111 | 0.00 | 17 | 3 | 3 |
| 50-5-1b | 63242 | 63281 | 0.06 | 18 | 63242 | 0.00 | 9 | 63242 | 0.00 | 18 | 2 | 3 |
| 50-5-2a | 88293 | 88333 | 0.05 | 18 | 89342 | 1.19 | 12 | 88298 | 0.01 | 32 | 3 | 3 |
| 50-5-2b | 67308 | 67436 | 0.19 | 22 | 67951 | 0.96 | 10 | 67308 | 0.00 | 47 | 3 | 3 |
| 50-5-2bis | 84055 | 84055 | 0.00 | 21 | 84126 | 0.08 | 8 | 84055 | 0.00 | 33 | 3 | 4 |
| 50-5-2bbis | 51822 | 51898 | 0.15 | 27 | 52213 | 0.75 | 9 | 51822 | 0.00 | 32 | 3 | 3 |
| 50-5-3a | 86203 | 86203 | 0.00 | 17 | 86203 | 0.00 | 18 | 86203 | 0.00 | 17 | 2 | 3 |
| 50-5-3b | 61830 | 61853 | 0.04 | 23 | 61885 | 0.09 | 20 | 61830 | 0.00 | 48 | 2 | 3 |
| 100-5-1a | 274814 | 275628 | 0.30 | 220 | 276137 | 0.48 | 75 | 275450 | 0.23 | 35 | 3 | 3 |
| 100-5-1b | 213568 | 214785 | 0.57 | 230 | 216154 | 1.21 | 59 | 213967 | 0.19 | 35 | 3 | 3 |
| 100-5-2a | 193671 | 194054 | 0.20 | 122 | 193896 | 0.12 | 76 | 193671 | 0.00 | 32 | 2 | 2 |
| 100-5-2b | 157095 | 157311 | 0.14 | 100 | 157180 | 0.05 | 82 | 157150 | 0.04 | 32 | 2 | 2 |
| 100-5-3a | 200079 | 200394 | 0.16 | 97 | 200777 | 0.35 | 69 | 200127 | 0.02 | 32 | 2 | 2 |
| 100-5-3b | 152441 | 152814 | 0.24 | 100 | 153435 | 0.65 | 68 | 152441 | 0.00 | 33 | 2 | 2 |
| 100-10-1a | 287661 | 292657 | 1.74 | 2622 | 287864 | 0.07 | 203 | 289449 | 0.62 | 112 | 3 | 3 |
| 100-10-1b | 230989 | 236026 | 2.18 | 1067 | 232599 | 0.70 | 117 | 230989 | 0.00 | 68 | 3 | 3 |
| 100-10-2a | 243590 | 243851 | 0.11 | 236 | 245484 | 0.78 | 52 | 243590 | 0.00 | 64 | 3 | 3 |
| 100-10-2b | 203988 | 204253 | 0.13 | 259 | 204252 | 0.13 | 42 | 203988 | 0.00 | 65 | 3 | 3 |
| 100-10-3a | 250882 | 253610 | 1.09 | 723 | 254558 | 1.47 | 82 | 252890 | 0.80 | 102 | 3 | 3 |
| 100-10-3b | 203114 | 205110 | 0.98 | 584 | 205824 | 1.33 | 78 | 204567 | 0.72 | 37 | 3 | 3 |
| 200-10-1a | 474850 | 477656 | 0.59 | 3960 | 477009 | 0.45 | 320 | 475225 | 0.08 | 132 | 3 | 3 |
| 200-10-1b | 375177 | 378656 | 0.93 | 4006 | 377716 | 0.68 | 239 | 376884 | 0.45 | 195 | 3 | 3 |
| 200-10-2a | 448077 | 449797 | 0.38 | 4943 | 449006 | 0.21 | 231 | 449124 | 0.23 | 64 | 3 | 3 |
| 200-10-2b | 373696 | 374996 | 0.35 | 3486 | 374717 | 0.27 | 290 | 374192 | 0.13 | 68 | 3 | 3 |
| 200-10-3a | 469433 | 471272 | 0.39 | 4075 | 471978 | 0.54 | 330 | 471183 | 0.37 | 132 | 3 | 3 |
| 200-10-3b | 362320 | 363581 | 0.35 | 7888 | 362827 | 0.14 | 214 | 362748 | 0.12 | 133 | 3 | 3 |

Table 7.5: Results on the large scale instances by Harks et al. (2013). Gap in % to the computed results, time in seconds and obtained number of open depots $M$.

| Instance | Approx+TSP | | | 2-SH | | | A&S | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | Value | Gap | Time | Value | Gap | Time | Value | Gap | Time | M |
| M 1,1 | 13,478.9 | 22.13 | 1 | 12,014.2 | 8.87 | 1 | 11,035.6 | 0.00 | 17 | 50 |
| M 1,2 | 3,499.1 | 10.30 | 1 | 3,470.3 | 9.38 | 1 | 3,172.3 | 0.00 | 16 | 16 |
| M 1,3 | 2,478.9 | 1.28 | 2 | 2,814.5 | 15.01 | 11 | 2,446.8 | 0.00 | 16 | 4 |
| M 2,1 | 17,997 | 20.01 | 1 | 15,760.5 | 5.10 | 1 | 14,995.7 | 0.00 | 15 | 33 |
| M 2,2 | 4,468.7 | 9.08 | 1 | 4,494.1 | 9.71 | 1 | 4,096.2 | 0.00 | 15 | 6 |
| M 2,3 | 2,620.8 | 1.78 | 2 | 3,010.6 | 16.93 | 11 | 2,754.1 | 0.00 | 15 | 1 |
| M 3,1 | 22,926.8 | 23.49 | 1 | 19,839.5 | 6.86 | 1 | 18,565.6 | 0.00 | 15 | 18 |
| M 3,2 | 5,345.9 | 11.05 | 1 | 5,207.4 | 8.18 | 1 | 4,813.2 | 0.00 | 15 | 6 |
| M 3,3 | 2,779.3 | 3.64 | 1 | 3,156.6 | 17.70 | 11 | 2,681.4 | 0.00 | 15 | 1 |
| L 1,1 | 32,325.9 | 15.89 | 36 | 29,538.1 | 5.90 | 65 | 27,893.1 | 0.00 | 98 | 177 |
| L 1,2 | 8,106.1 | 6.50 | 59 | 8,176.0 | 7.42 | 20 | 7,611.0 | 0.00 | 96 | 52 |
| L 1,3 | 5,463.7 | 0.84 | 122 | 6,228.4 | 14.96 | 67 | 5,417.7 | 0.00 | 98 | 14 |
| L 2,1 | 50,229.7 | 24.00 | 41 | 43,496.5 | 24.00 | 64 | 40,508.1 | 0.00 | 103 | 89 |
| L 2,2 | 12,059.5 | 13.13 | 74 | 11,551.6 | 8.36 | 19 | 10,659.6 | 0.00 | 104 | 19 |
| L 2,3 | 6,624.8 | 10.60 | 165 | 7,058.2 | 17.85 | 69 | 5,989.2 | 0.00 | 100 | 4 |
| L 3,1 | 63,905.1 | 26.55 | 51 | 54,976.6 | 8.87 | 64 | 50,495.9 | 0.00 | 133 | 55 |
| L 3,2 | 14,372.4 | 15.63 | 98 | 13,426.2 | 8.02 | 20 | 12,429.4 | 0.00 | 100 | 15 |
| L 3,3 | 6,966.5 | 8.60 | 213 | 7,467.2 | 16.41 | 66 | 6,414.4 | 0.00 | 117 | 4 |
| XL 1,1 | 48,677.1 | 7.99 | 214 | 43,939.8 | -2.52 | 179 | 45,074.3 | 0.00 | 265 | 193 |
| XL 1,2 | 11,872 | 8.19 | 369 | 11,867.3 | 8.14 | 89 | 10,973.3 | 0.00 | 225 | 95 |
| XL 1,3 | 7,754.7 | 1.21 | 750 | 8,824.0 | 15.16 | 161 | 7,661,6 | 0.00 | 218 | 25 |
| XL 2,1 | 77,580.6 | 22.51 | 243 | 68,118.7 | 7.57 | 171 | 63,325.5 | 0.00 | 324 | 168 |
| XL 2,2 | 18,159.1 | 10.76 | 435 | 17,638.8 | 7.58 | 89 | 16,395,0 | 0.00 | 234 | 25 |
| XL 2,3 | 9,296.1 | 6.42 | 887 | 10,172.32 | 16.45 | 163 | 8,734.9 | 0.00 | 354 | 5 |
| XL 3,1 | 101,454 | 29.03 | 307 | 85,509.5 | 8.75 | 178 | 78,628.1 | 0.00 | 372 | 108 |
| XL 3,2 | 21,341.5 | 12.57 | 570 | 20,659.7 | 8.97 | 89 | 18,958.5 | 0.00 | 280 | 18 |
| XL 3,3 | 10,389.9 | 12.21 | 1323 | 10,897.5 | 17.68 | 167 | 9,259.5 | 0.00 | 390 | 5 |

# 8 Simulation of B2C e-commerce distribution in Antwerp

## 8.1 Introduction

The distribution of goods plays a major role in enabling economic and social activities in cities. Especially with the rise of e-commerce shopping, an increasing number of people order products online and have them delivered at home. Nowadays, this B2C distribution of parcels accounts for 56% of all shipments in e-commerce (Copenhagen Economics, 2013) and, thus, B2C e-commerce has been identified as a major challenge in the urban logistics literature (Taniguchi and Kakimoto, 2004; Visser et al., 2014; Edwards et al., 2010; Van Duin et al., 2016; Weltevreden, 2007). The growth in parcel transportation is accompanied by an increase in externalities like emissions, which affect the quality of urban life in a negative way. This trade-off between the need to distribute goods and the liveability of cities can be analysed from the perspective of different stakeholders.

From the perspective of a logistic service provider (LSP), there is a growing pressure from the e-commerce sector to keep prices for shipping and handling as low as possible. This competition for lower prices in the last-mile delivery has pushed LSPs to cut their operational costs to the minimum. In other words, the last mile delivery of parcels is a purely cost-driven business which discourages the development of more sustainable distribution solutions (Ducret, 2014). Therefore, standard deliveries are still vastly based on traditional distribution networks, using vehicles such as diesel vans instead of eco-friendly alternatives.

In contrast, local authorities and inhabitants strive for cities with a high quality of life, including efficient transportation and traffic systems without too much congestion, noise and emissions. These negative effects of distribution in urban areas are expressed by external cost metrics. In increasingly complex cities, external costs can only be minimised by promoting distribution systems that are sustainable and efficient.

Thus, there is a clash of interests between different stakeholders when it comes to today's parcel distribution systems. In order to compromise and put sustainable and efficient delivery solutions into practice, stakeholders need to be able to compare possible options. However, there are usually no numbers available to compare the state-of-the-art with other 'what-if' scenarios, and if so, they are rough estimations at best. This situation makes it difficult to argue in favour of one delivery solution over another.

In this chapter, we demonstrate how this problem can be addressed with the help of a simulation approach. This approach allows us a realistic assessment of the current situation and possible alternatives. Using the city of Antwerp as a case study, we analyse the cost structure of 'what-if' scenarios for B2C parcel distribution and compare them with the current situation. In the first alternative scenario, customers can choose to pick-up their parcels from delivery points (DP) instead of being delivered at home. In the second alternative scenario, an LSP implement a delivery system via cargo-carrying capable bikes. Parcels are delivered by vans to DPs in the city centre, from where they are distributed to the customers on bike routes. With this study, we aim to answer the following research question: How do different designs in urban parcel distribution affect the operational and external costs, and is there a way to minimise both and, thus, satisfy all stakeholders?

This work is structured as follows. In Section 2 we introduce the basic concepts and state-of-the-art research in urban parcel distribution. In Section 3 and Section 4 we motivate and explain the design of our simulation study. The results are described in Section 5, followed by a discussion in Section 6.

## 8.2 **Urban logistics and e-commerce deliveries**

Most products that are bought remotely are shipped as parcels in trucks or vans and brought to people's doorstep, a concept which we call in the following 'traditional home deliveries'. The advancing development of e-commerce has changed the landscape of home deliveries profoundly. Instead of going to physical stores, more and more people purchase products online. These changes in shopping behaviour have an effect on the mobility in cities, with some shopping trips being substituted by parcel transportation. However, the precise impact of this substitution on the overall traffic volume is not clear (Cullinane, 2009; Weltevreden, 2007). For instance, in a survey-based study, the author found that the e-commerce-related increase in freight transport was higher than the corresponding decrease in customer trips (Weltevreden and Rotem-Mindali, 2009). In general, the effects of e-commerce on transport are still uncertain and have been the focus of research during the last years (Brown and Guiffrida, 2014; Visser et al., 2014; Edwards et al., 2010). Browne (2001) argues that the traffic volume due to home deliveries is affected by several factors, such as the customer behaviour, the consolidation of deliveries and the number of returned goods. Mokhtarian (2004) agrees that the impact of e-commerce on transport depends on both, changes in shopping behaviour as well as changes in the distribution system.

These findings lead to the question 'What kind of distribution system is an adequate response to changes in shopping behaviour?'. Possible alternatives to traditional home deliveries have been widely studied recently, specifically for the e-groceries market (Lin and Mahmassani, 2002) and in the context of so-called urban distribution centres (Van Der Helm, 2015).

The concept of self-pick-up involves the customer in the delivery process. Instead of delivering parcels to the customer, the parcels are delivered to delivery points (DP), from where the customer collects their order. DPs are spreading rapidly across Europe and have been the focus of recent research. Early contributions focused on the accessibility of delivery point networks (Weltevreden, 2008; Morganti et al., 2014a,b). Durand and Gonzalez-Feliu (2012) compared self-pick-up to traditional home deliveries and found that an 'all delivery point' scenario would be the most beneficial in terms of total kilometres driven with vans and trucks. Accordingly, several studies agree that delivery points have the potential to reduce the travel time of freight vehicles as well as that of customers (Song et al., 2009; Edwards et al., 2010; Brown and Guiffrida, 2014).

The success of DPs can also be attributed to the possibility of failed deliveries. A home delivery can fail, if the customer or neighbours are not at home at the time of delivery. In this case, the parcel needs to be shipped to a nearby service point or DP, which leads to a substantial extra delivery effort. For instance, in the UK the additional costs due to failed deliveries amount to more than one billion dollars per year (Deloitte, 2015).

Cargo bikes present a more recently-developed distribution solution, which is especially focused on the reduction of environmental impacts. The idea of cargo bikes is to avoid the dense car traffic in urban areas, and instead deliver parcels on bike routes, which are more flexible and cause less externalities. EY (2015) found that home delivery via cargo bikes causes significantly fewer external effects than conventional shopping, traditional home delivery via vans and deliveries via delivery points. Results of a pilot study in London confirmed that last-mile delivery operations can be cheaper without adding relevant costs to the distribution by combining urban distribution centres and bike deliveries (Conway et al., 2012). Similarly, Maes and Vanelslander (2012) concluded that delivery costs of vans and bikes are almost comparable. The authors identified the higher speed on highways and the relatively low load capacity of cargo bikes as major barriers to the implementation of a B2C bike distribution system. In contrast to that, a cost calculation based on data from Belgian companies showed a decrease in overall costs by up to 45% (Gevaers et al., 2014).

In conclusion, solutions for e-commerce transport have received wide attention. However, their precise effects on operational and external costs are not always clear and results are usually based on analytical estimations or pilot results. Moreover, most studies are limited to one or two of the concepts described above. The goal of this chapter is to conduct a comprehensive quantitative simulation study which analyses the benefits and shortcomings of all those different concepts in the context of B2C distribution in the city of Antwerp.

## 8.3 Simulation

In this chapter, we explore the potential benefits and shortcomings of different urban distribution strategies in the B2C delivery sector. Our methodology is hereby based on the concept of simulation. The main reason for this choice is that conducting a real-life case study is intractable in this case due to its prohibitive costs (e.g., in order to study the impact of bike-deliveries, we would need to use and acquire delivery bikes). Moreover, the use of a simulation allows us to (1) generate a multitude of virtual case-and (2) collect sufficient data for an analysis. The most important benefits of this approach are its feasibility, scalability and flexibility. Experiments can be set up rather quickly and in a short amount of time, even though they require a careful planning of the design. Once the implementation of the experimental design is completed, any amount of data can be generated for any size and layout of the simulated entity, e.g., for a neighbourhood, for a city, or for a whole country. Finally, input parameters e.g., cost values or locations, can be changed, and the sensitivity of these changes can be incorporated in the analysis. Simulation has successfully been used before in urban logistics (Nuzzolo et al., 2014; Van Duin et al., 2014; Teo et al., 2012; Russo and Comi, 2011; Anderluh et al., 2017).

There are some limitations to simulation studies, which have to be considered carefully. Most importantly, the constructed simulation model is an abstraction of reality, and care must be taken that no important features or attributes are lost in this abstraction process. In other words, the practical implications of the results are only as meaningful as the simulation model correctly reflects reality. Secondly, a simulation requires accurate input data to model the considered processes precisely, e.g., travel times of distribution routes or distances between two locations. We will take care to explain and motivate our model assumptions in Section 4.

Finally, the evaluation of a simulation study is based on a statistical analysis. A simulation usually captures the dynamics of complex systems. In the context of logistics, the delivery locations will change day-by-day, and so will the delivery routes. To account for these dynamics and still derive a general idea of how the system behaves, different simulation runs have to be executed. In this context, it is important to choose a sufficient number of simulation runs and a sufficient length of each run. The target metrics will then be computed as the average over all simulation runs.

## 8.4 A simulation study for B2C e-commerce distribution in Antwerp

The goal of this study is to analyse the cost structure of state-of-the-art B2C distribution in Antwerp and compare it to alternative scenarios. On the basis of the presented findings in the literature, our hypothesis is that the implementation of delivery systems based on DPs and cargo bikes can present a reasonable alternative in urban logistics. Moreover, we will analyse the effect of three parameters on the cost structure of B2C distribution: (1) the demand density, (2) the percentage of self-pick-up customers and the (3) congestion within the city. In the following, we first describe the current delivery process in Antwerp, and then we present how we transform this real-world activities into a simulation framework.

### 8.4.1 *The distribution process in Antwerp*

The simulation study is based upon the daily distribution activities of a B2C logistic service provider (LSP) in Belgium. The current situation was studied by interviews with drivers and managers as well as by a field study where one of the authors accompanied a driver on a typical delivery day. We further obtained two datasets from the LSP, one including the delivery destinations over a period of three months, and a second one comprising the aggregated travel times and distances per driver per day.

The LSP's delivery operation is executed via medium-sized diesel vans with an assumed maximum capacity of about 300 parcels. A typical delivery day of a driver starts around 6:00 am at the distribution centre at the fringe of the city of Antwerp. He loads the parcels and plans the route, before driving into the city and starting the distribution. Each driver performs a distribution tour alone and visits 99 customers on average. For each customer on the tour, the driver gets at close proximity, parks, fetches the parcel from the van and delivers it at the customer's door. If the customer or any neighbours are not at home, the respective parcel is delivered to a nearby DP, from where the customer can pick it up. We computed from the dataset that the average duration per stop, including parking, fetching and delivering the parcel, amounts to 2.5 minutes. The delivery routes are planned by the driver, without any computer assistance. After all parcels have been delivered, the driver returns to the distribution centre for a debriefing. We estimated from the dataset that a driver typically spends 6 hours for delivering activities in the city, and two hours for the remaining activities before and after each tour (loading, preparation, driving into the city and returning to the distribution centre, debriefing). Note that we cannot derive the actual delivery routes nor the specific durations of tours from the data, and we will derive those values by simulating the distribution process on the basis of the above observations.

### 8.4.2 *Simulation of the distribution process*

The simulation of delivery routes is done in two steps. Firstly, we generate demand, by defining the location of customers. Secondly, we compute routes to deliver the parcels to customers.

#### 8.4.2.1 *Generation of demand*

We generate customer demand on the basis of the real-world dataset. The dataset contains the locations of all deliveries in Antwerp over a period of three months. We use this dataset to compute the spatial distribution of parcel demand.

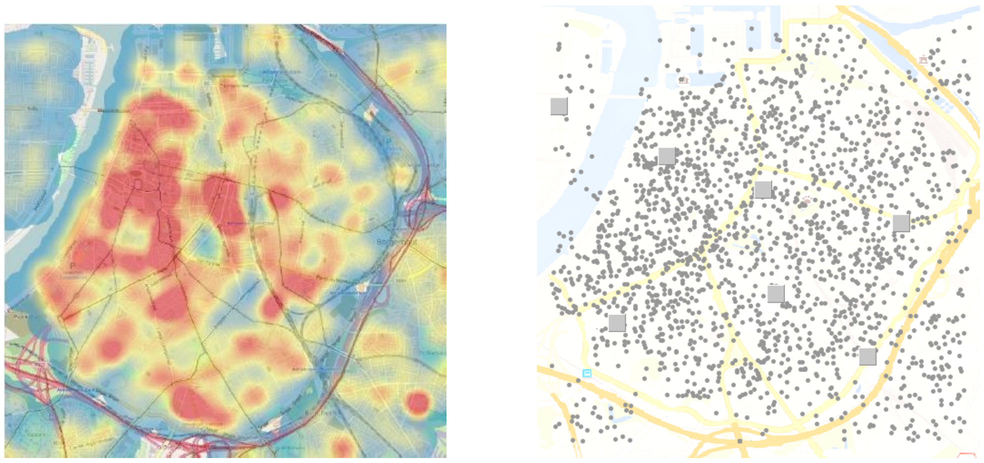


Figure 8.1: Generation of customer locations and DPs (right) on the basis of actual demand (left, Source: Cardenas et al. (2016a)).

First, we divide the urban area of Antwerp (about 4 $km^2$) into a grid of 100 smaller districts (400 $m^2$ each) and compute for each district the average number of demands per day. This resolution is a compromise between the accuracy of a demand location (size of a district) and the accuracy of the estimated demand quantity (data points per district). Figure 8.1 visualises the resulting distribution where the demand is especially high in the residential areas in the centre and in the southwest of the city. In each simulation run we define the total demand in the city, e.g., 100 parcels per day, and distribute this total demand among the districts (e.g., if 2% of the demand in the dataset falls in district A, the probability of assigning one particular demand to A is also 2%). The precise demand location within each district is chosen randomly.

An example of this process from a spatial demand distribution to specific customer locations is presented in Figure 8.1. Since the final location is determined randomly, it might not represent a valid address, e.g., the location might be a point in the river. These invalid customer locations are reassigned to the nearest valid address when generating the routes in the next step. Finally, we locate seven DPs in our simulation model at the actual locations of service points of a large LSP in Antwerp.

#### 8.4.2.2 *Computation of delivery routes*

After the generation of the customer locations, we compute the travel time between each pair of customers and DPs with Open Street Maps. Open Streets Maps is a freely available web service to obtain trip durations between two locations based on the real street network. Also, it assigns our randomly chosen locations to the nearest available address. The distances between each pair of destinations is computed with the Manhattan distance, which is one of the most accurate estimators of road distances in inner cities (see for instance Shahid et al. (2009)). On the basis of these values, we compute delivery routes. The length of one delivery route is hereby constrained by the working hours of the driver and the capacity of the vehicle. We computed from the dataset that a driver spends on average 6 hours delivering parcels in the city. We use this time horizon as a constraint. Before every simulation run we conduct a pre-test and determine how many parcels can be delivered within 6 hours with the current parameters. The length and number of required delivery routes then follow from the results. For instance, if the customer density is higher, it takes less time to drive from customer to customer, and therefore, more customers can be visited on one route. Each route also visits a near DP from time to time, to return failed deliveries.

The planning of delivery routes resembles the popular Vehicle Routing Problem (VRP) in the field of combinatorial optimisation. Thus, we can compute cost-efficient delivery routes by utilising one of the many heuristics developed to solve the VRP (Demir et al., 2014; Cattaruzza et al., 2017). Because of its low implementation complexity and fast processing time, we use the Clarke-Wright Savings algorithm. Even though it does not compute optimal routes, i.e., routes that have minimal travel time, the gap to the optimal solution is usually relatively small. Given that routes, in reality, are usually not optimal either, e.g., because drivers rely on intuition or companies do not use a planning instrument, this should represent a good estimation of actual delivery routes. In the case studies below we need to compute routes for several thousand customers in a feasible time and we, thus, speed up the route computation with the following idea. Since customers in the same neighbourhood are usually delivered on the same delivery route, we assign customers to spatial clusters. These clusters are computed for each delivery day in such a way that they do not exceed the maximum number of customers per route (i.e., the number of customers that can be visited within 6 hours). Each cluster of customers as well

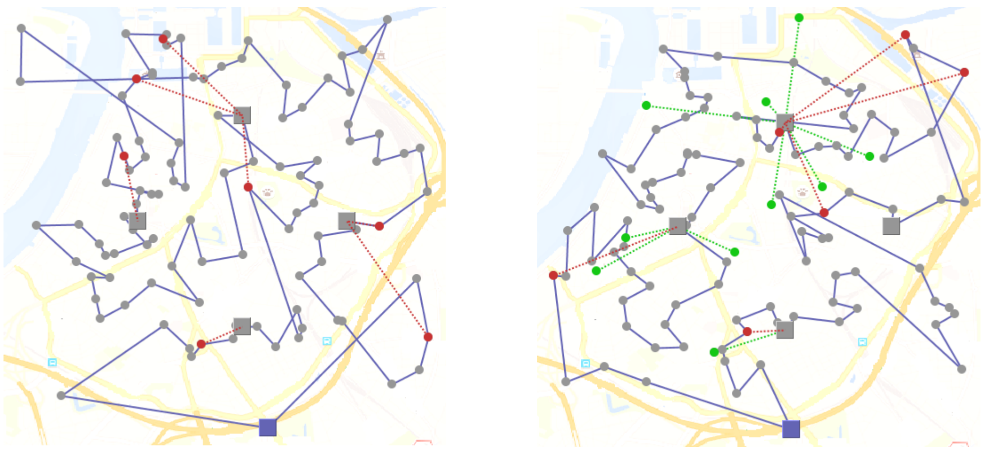


Figure 8.2: Simulated delivery route. Starting from the highway exit, all customers (dots) and DPs (squares) are visited once. Red dots denote failed deliveries. (right) Green dots indicate self-pick-up customers.

as the nearest DP is then delivered by a separate tour which is computed with the heuristic by Clarke and Wright. An example of this clustering approach is visualised in Figure 8.3.

We make the following assumptions when computing the routes:

1. Open Street Maps computes the duration of a trip under the assumption of free flow. Thereby, it ignores traffic-related delays (e.g., traffic jams during rush hours). To account for traffic-related delays, we need to apply a factor for congestion. Since the choice of this factor might change the results, we conduct a sensitivity analysis to investigate the impact of congestion on delivery costs. More concretely, we consider free-flow (about 26 km/h on average), minor congestion (about 17 km/h), and heavy congestion (about 13 km/h). Since, for instance, in French cities the average car speed was estimated to be around 16 m/h (Prud'homme and Lee, 1999), we assume that this value represents a good estimate of the actual traffic situation in Antwerp (even though in big cities such as London the average speed can drop as low as 8 km/h (Browne et al., 2011)). In reality, congestion is also dependent on the specific time and the specific road, but since such accurate data is not available, we assume the same congestion factor for the whole day.

2. The simulation focuses on the distribution of parcels within the city, and models the activities before and after as fixed events. Therefore, the computed delivery routes start and end at motorway exits at the city border. We assume that the routing from the distribution centre to the city and back is the same

for every delivery route, and model this stem mileage as a fixed cost per route, as explained in Section 4.3.

3. All parcels have the same priority, i.e., it is not necessary to fulfil certain demands before others.

4. Finally, we derived from the dataset that about 11% of the deliveries fail, i.e., both the customer and their neighbours are not at home at the time of delivery. These customers are chosen randomly, and their parcels are delivered to the nearest DPs from where they need to pick it up. Likewise, a pre-defined percentage of customers are self-pick-up customers. Those customers chose not to be delivered at home, and are therefore not included in the delivery routes. Instead, they have to go to the nearest DP. These model assumptions are visualised in Figure 8.2.

### 8.4.3 *Analysis of delivery costs*

The computed delivery routes reflect the B2C delivery activities of one day, and we are interested in the resulting operational and external costs. Since the location of demand is stochastic, the distribution routes of every simulated day are slightly different. To account for this variability, we simulate 100 individual days and average the costs over all days. The results between different simulated days are relatively stable with a low variance. With 100 datapoints, the 95% confidence interval for estimated external costs and operational costs is in all experiments less than 1% around the average (i.e., if the average is 100, then the 95% confidence interval is at most [99,101]).

As operational costs, we consider all variable costs related to the distribution activities in the city. All in all, we can distinguish between the labour costs for the carriers, the costs for using the vehicles, and the costs for using DPs. The labour costs are computed on the basis of the required working hours, assuming one driver per vehicle. Loading the vehicle in the morning, preparing the tour, driving from the distribution centre to the city, returning to the distribution centre and debriefing in the evening requires 2 hours per route, as estimated from the dataset. Within the city, the travel time in the city is obtained from the results of the simulation and the average time for a stop at a customer amounts to 2.5 minutes, including parking and handing over the parcel.

For the vehicles we consider variable costs of €0.18 / km, and neglect fixed costs since we assume that sufficient vehicles are available. The total number of kilometres driven is determined by the delivery routes in the city, as well as 10 km for trips from and back to the distribution centre. More precisely, let D denote the length of all delivery routes within the city in km, T the respective travel time in minutes, R the number of routes, and S the number of deliveries. Then the operational costs O are

Table 8.1: Overview of parameters to determine operational and external costs. Source: Cardenas et al. (2016b).

| | |
|---|---|
| General parameters | |
| Labour costs for drivers | €0.30 / minute |
| Average time per delivery | 2.5 minutes |
| Probability that a delivery fails | 11 % |
| Parameters for delivery tours by van | |
| Driving speed in the city | 17 km / h |
| Capacity limit of a van | 300 parcels |
| Operational costs of delivery van | €0.18 / km |
| Stem mileage per delivery tour | 10 km |
| Time limit for a delivery tour within the city | 6 hours |
| Time required for activities before and after a delivery tour | 2 hours |
| Parameter for delivery tours by cargo bikes | |
| Driving speed in the city | 12 km / h |
| Capacity limit | 10 parcels |
| Parameters to compute external costs | |
| Emissions | €0.11 / km |
| Noise | €0.05 / km |
| Congestion | €0.49 / km |

computed as O=0.30(T+2.5S+120R)+0.18(D+10R). Hereby, T, D and R are obtained as results from the simulation. A complete overview of all parameters used in simulation is given in Table 8.1.

As external costs, we consider the externalities caused either by delivery vans or by customer trips to a DP. Since the exact quantification of corresponding costs is still under discussion in the literature, we chose the externalities that have received the most attention, namely emission, noise and congestion. We chose corresponding cost values on the basis of calculations in MOVE (2014). Hereby, we need to consider the modal choice of customers when picking up their parcel. If a customer uses their car, their trip contributes to the delivery-related external costs. On the other hand, walking or biking does not result in externalities. Intuitively, the greater the distance between a customer and the nearest DP, the more likely it is that he will use a car. Findings of modal choices in Belgium confirm this intuition and we extract the following estimates for our study: If the distance between customer and DP is smaller than 200 meters, the customer will use their car in 10% of the cases.

If it is between 200 and 500 meters, the likelihood of car usage increases to 30%, for 500 to 1000 meters to 50%, and for distances of more than 1000 meters, the customer will take the car in 70% of the cases. Let P denote the distance that customers travel to DPs with their car to pick-up parcels. Then the external costs E are computed as E=(0,11+0,05+0,49)(P+D+10R) (Cardenas et al., 2016b).

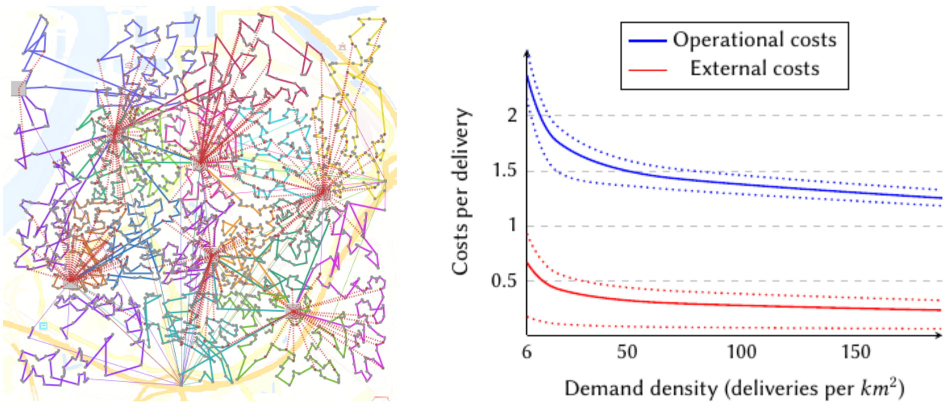


Figure 8.3: (left) Routes for high demand are computed by clustering the customers, and serving each cluster by a separate tour, as indicated by the different colours. (right) Costs per delivery as a function of demand, the dotted lines below and above present free-flow (26km/h) and heavy congestion (13 km/h), respectively.

## 8.5 Results

We analyse the cost structure of four B2C distribution scenarios in Antwerp. In the first experiment, we analyse the state-of-the-practice of the distribution system of e-commerce in Antwerp (home deliveries by vans) as a function of the demand density. The results of this analysis constitute the baseline, which we will compare to the other hypothetical alternatives. In the first alternative, we investigate the effect of customer self-pick-up from DPs. In the second alternative, we study the possible implementation of a bike delivery system. Finally, we combine the ideas of bike delivery and self-pick up in a hybrid system. For each of these experiments, we compute the operational and external costs per delivery, and conduct a sensitivity analysis for the most impactful parameters.

### 8.5.1 *Simulation of home deliveries by vans*

The B2C parcel distribution market in Belgium is composed of one large carrier and several smaller ones. The large LSP is estimated to deliver about 2000 parcels per day in the centre of Antwerp, whereas the smaller ones deliver about 100 parcels. Depending on this demand level, LSP have different cost structures. If the routes are planned well, we should observe an economy-of-scales effect. We compute the costs per delivery for varying demand and also conduct a sensitivity analysis to determine the impact of congestion on costs. The setup and the results are visualized in Figure 8.3.

Table 8.2: Results from the simulation of van deliveries for different demand densities.

| | 100 deliveries (6.25 del/km$^2$) | 500 deliveries (31.25 del/km$^2$) | 2000 deliveries (125 del/km$^2$) |
|---|---|---|---|
| Routing of vans | | | |
| Number of routes | 2 | 5 | 17.1 |
| Deliveries per route | 50 | 100 | 117 |
| Distance driven between two deliveries (m) | 683 | 325 | 173 |
| Time driven between two deliveries (min) | 2.2 | 1.1 | 0.6 |
| Time (as % of time spend in the city) | | | |
| Time driving in the city | 47 | 31 | 20 |
| Time delivering in the city | 53 | 69 | 80 |
| External Costs | | | |
| Distance driven by delivery vans (km) | 68 | 162 | 346 |
| Distance driven by customers to DPs by car (km) | 12 | 64 | 262 |

All in all, the congestion factor seems to have only a minor impact on the operational costs, the results for free-flow and heavy congestion are within a 10% margin around the results for slight congestion. This relatively low sensitivity can be explained by the observation that carriers spend the majority of their time with non-driving activities (parking, fetching and delivering the parcel), since distances in the city centre are rather short. In contrast to that, congestion has a much stronger effect on external costs. Even though the distances remain similar, according to MOVE (2014) the external costs related to congestion drop to about €0.01/km for free flow while reaching about €0.76/km for heavy congestion.

In line with our expectations, we observe a decrease in operational and external costs with a growing number of deliveries. While the operational costs per delivery drop from €2.37 for 6 deliveries per $km^2$ to €1.25 for 190 del/$km^2$, the external costs decrease from €0.66 to €0.23 per delivery, assuming slight congestion. This cost decrease can be attributed to a more efficient routing. With a higher demand, the distance and travel time between two successive customers on a route becomes lower, as shown in Table 8.2, and thus, more customers can be visited on a route. With a density of 125 del/$km^2$, the routes are so efficient that driving from customer to customer accounts for only 20% of the time in the city. External costs account for about 28% (6 del/$km^2$) to 18% (190 del/$km^2$) of the operational costs.

In these experiments, we assume that each LSP has its own DPs in the city, independent of the demand. However, in reality, only larger LSPs have this infrastructure, whereas smaller LSPs cannot afford to maintain their own service points. They usually collaborate with shops from which customers can pick up nondelivered parcels in exchange for a service fee paid by the LSP. Thus, the B2C delivery market is biased towards size. Not only do large LSPs have the advantage of smaller variable costs per delivery, and can, therefore, offer more competitive prices, they also own the infrastructure to offer better services.

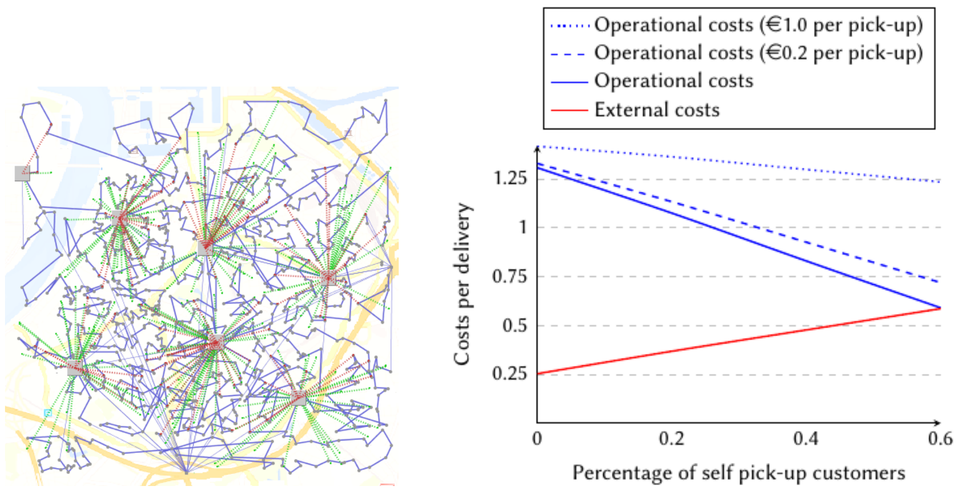


Figure 8.4: (left) Delivery routes for a large LSP with self-pick-up. (right) Costs per delivery as a function of the percentage of self-pick-up customers.

### 8.5.2 *Simulation of self-pick-up*

As discussed in Section 2, there is an increasing interest of LSPs to explore the opportunity of customer self-pick-up, i.e., customers can choose to pick up parcels themselves from a nearby DP instead of being delivered at home. This concept has two advantages for LSPs. Most importantly, it reduces the number of deliveries since fewer customers need to be visited. Also, it can reduce the number of failed deliveries, e.g., when households with working people choose the self-pick-up option since they will not be at home anyway. On the downside, self-pick-up results in additional traffic and external costs when customers travel to the DPs by car to pick up their parcel. In the following, we investigate these opposing effects from the perspective of a large LSP with 2000 deliveries per day.

As expected, the variable operational costs per delivery decrease linearly with the number of self-pick-up customers. More precisely, it drops by about €0.01 for each additional one percent of self-pick-up customers, as shown in Figure 8.4. The magnitude of these savings changes if we consider a variable cost for each parcel that is picked-up at a DP, e.g. for service and storage. For larger LSPs that have their own service points this variable costs is likely to be small, whereas smaller LSPs have to collaborate with external shops and pay about €1 per picked-up parcel. We investigated the effect of these costs in a sensitivity analysis, and found that even with high pick-up costs, self-pick up is still favourable for LSPs. However, the decrease in operational costs comes at the expense of higher external costs, which grow by half a Cent for each additional one percent self-pick-up since more customers need to travel to the DPs.

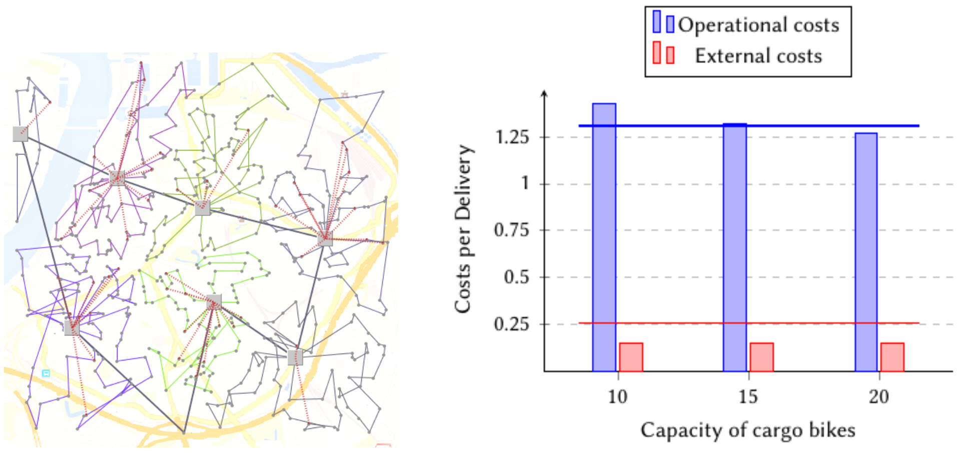


Figure 8.5: (left) Parcels are brought to the DPs (grey lines) and delivered from there to the customers on bike routes. (right) Costs per delivery for different capacities of the cargo bikes, compared to traditional delivery (lines).

These results highlight that customer self-pick-up is highly cost-efficient for LSPs. However, the promotion of self-pick-up might be difficult, since most customers are used to the high comfort of home deliveries. One idea to promote self-pick-up is the offering of price reductions. In our case, the delivery price could be reduced by up to about €1 (the operational variable cost per delivery) for those customers that choose to pick-up the parcel themselves, without touching the LSP's profit. On the other hand, there is no reason for public authorities to promote a self-pick-up based delivery concept, since externalities increase.

### 8.5.3 *Simulation of bike deliveries*

With self-pick-up we have identified a distribution concept that benefits LSPs, but does not enhance the quality of life in cities. Reversely, with cargo bikes, we now analyse a distribution concept that is expected to decrease the externalities of parcel distribution. We model this scenario as follows. The parcels are brought from the distribution centre to the DPs in the city by vans. At the DPs the parcels are unloaded and buffered, and then distributed by cargo bikes to the customers. In other words, the DPs act as transshipment points between vans and cargo bikes. This process is visualized in Figure 8.5. A similar system has already been implemented and tested in London (Browne et al., 2011).

We compute two sets of routes, the routing for the vans, and the routing for the cargo bikes. The routing for the vans from the city border to the DPs is computed in the same fashion as above, assuming a fixed stem mileage and preparation time per tour and an average speed of about 17 km/h in the city. Since the vans only visit a

Table 8.3: Comparison of simulation results for van and bike delivery for an LSP with 2000 deliveries (125 del/km$^2$).

| | Delivery by vans | Delivery by bikes |
|---|---|---|
| Number of routes by van | 17.1 | 7 |
| Distance driven by van (km) | 527 | 211 |
| Distance driven by van in the city (km) | 356 | 141 |
| Distance driven by cargo bike (km) | 0 | 502 |
| Time spend driving in the city, with bikes and vans (hours) | 20.8 | 48.6 |
| Time spend driving in the city (as % of time spend in the city) | 20 | 35 |

few points, the maximum duration of 6 hours per tour is never reached, and we set the number of parcels per van to an assumed maximum van capacity of 300. The unloading of parcels at a DP is defined to take 20 minutes.

Secondly, we compute the bike routes from the DPs to the customers. Each customer is assigned to its closest DP, and then for each DP we solve the resulting VRP. We assume that the cargo bikes can carry at most 10 parcels at a time, so that drivers have to return to the DP for a refill several times. Each refill is assumed to take 5 minutes. Further, we assume that the bikes drive at an average speed of about 12 km/h. The service time per delivery remains 2.5 minutes as above. Unlike the van routes, the bike routes are not associated with external costs.

Consistent with previous studies, e.g., Browne et al. (2011), we observe that bike deliveries can yield a drastic decrease in external costs by 40% from €0.25 to €0.15 per delivery, compared to traditional home delivery via vans. These results are presented in Figure 8.5. The reason for this cost reduction is a decrease in the distance that is travelled with vans in the city, as shown in Table 8.3. Despite these findings, one of the main argument against bike deliveries in practise is an expected increase in travel time and, thus, working hours. Our results confirm that the driving time in the city would increase by almost 134%. However, the stem mileage drops significantly, and the travel time accounts for a relatively low percentage of the total time spend on delivering activities and, thus, the operational costs for the LSP increase by only about 9% from €1.31 to €1.43 per delivery. There are two main reasons for these longer travel times: (1) The limited capacity of the cargo bikes renders the routing more inefficient. In fact, if we assume that the bikes can carry up to 20 parcels, we observe a decrease in operational costs, as demonstrated by the sensitivity analysis in Figure 8.5 (Electrically-assisted cargo tricycles can even carry more than 30 parcels at a time (Browne et al., 2011)). (2) We assume that the trips between two customers by bike takes longer than by van. Especially in dense city areas this assumption might no longer be valid, since some areas are easier accessible by bike and shortcuts can be used. Also, the average service time might be lower, since parking and fetching the parcel should be simpler compared to using a van.

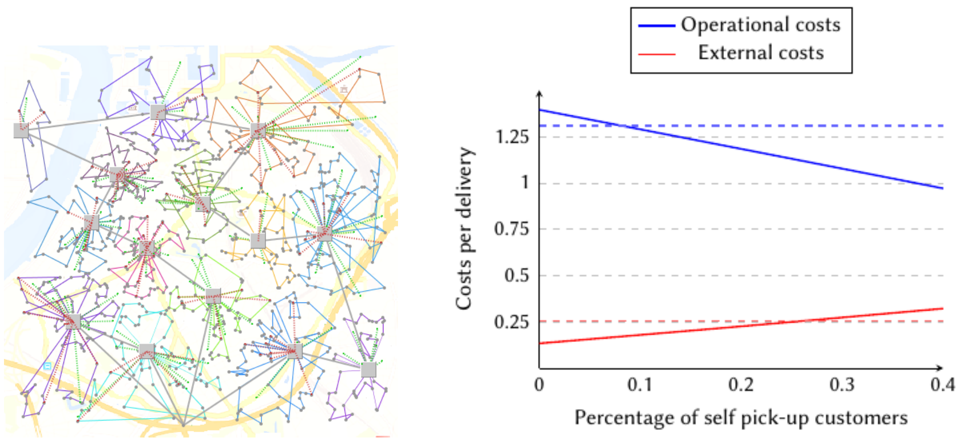


Figure 8.6: (left) Parcel distribution via cargo bikes with additional DPs and self-pick-up. (right) Costs per delivery for a varying number of self-pick-up customers, compared to traditional home delivery (dashed lines).

Consequently, our parameter choice is rather on the low side (low bike capacity, slow biking times, long service times) so that the operational cost increase of 9% can be interpreted as estimate, and the real increase in variable costs is likely to be smaller. Additionally, the two scenarios have other structural differences that might influence the decision-making process in reality. In the case of joint bike deliveries the LSP need to acquire cargo-carrying capable bikes, and can reduce the number of vans. Moreover, this scenario might enable other potential benefits, such as night deliveries of the DPs.

### 8.5.4 *A compromise between low externalities and low operational costs*

With the previous experiments, we have demonstrated that the concepts self-pick-up and bike deliveries can only decrease either operational or external costs at the expense of the other. Thus, both concepts cannot satisfy all stakeholders and will, therefore, be difficult to implement in practice. These findings lead to the question, whether a feasible compromise can be found that benefits LSP as well as enhances the quality of life in cities.

The starting point for a compromise is the observation that the external costs of failed deliveries and self-pick-ups depend crucially on the distance between customers and DPs. If the distance is rather low and the next DP is only a few streets away, fewer people will use a car to pick up their parcel. At the same time, customers will feel more inclined to accept the self-pick-up option. The distance between customers and DPs can be decreased by opening more DPs in the city, which requires a significant investment that might discourage LSPs. However, public authorities could provide those additional DPs, under the condition that LSPs lower external costs by changing the distribution system to bike deliveries. In this way, public authorities directly incentivize bike deliveries and greener cities. In the following, we analyse whether these ideas would be beneficial for all stakeholders.

We choose the same setup as in the previous section, and consider a large LSP with 2000 deliveries per day that implement a bike delivery systems originating from DPs. This time, public authorities provide another 6 DPs. We choose the location of these additional virtual DPs in such a way that the city is roughly covered uniformly with DPs, and the average distance between customers and DPs decreases from 290 to 220 meters As a possible consequence, more customers choose to use the DPs for self-pick-up, and we investigate the resulting effects on costs. The results are presented in Figure 8.6.

With an increasing number of self-pick-ups we again observe growing external and shrinking operational costs. The break-even point for the LSP is reached at about 10% of self-pick-ups. If at least 200 customers choose to not be delivered at home, this bike delivery scenario becomes profitable for the LSP, compared to the state-of-the-practise. On the other hand, external costs are lower than those of traditional home delivery, if less than about 25% of customers choose self-pick-up. Thus, there is a margin of between 10% and 25% of self-pick-ups, in which bike deliveries with additional DPs are beneficial for both, public authorities and LSP.

These findings suggest that a delivery system based on cargo bikes can be beneficial for all the stakeholders, if it is correctly implemented and incentivized. It requires a sufficient density of DPs in the city, and a possibility for customers to pick up parcels themselves. At the same time, the percentage of self-pick-ups should either be not too high, or customers should be encouraged to not use their car for pick-up trips. The latter could be achieved by further increasing the number of DPs and thereby improving customer access.

### 8.5.5 *Discussion of limitations*

In the simulation studies above we tried to annotate the delivery activities with as realistic cost and time values as possible, mostly on the basis of observations from a real-world dataset. For those parameters that we needed to estimate and that showed to have a significant effect on the overall result, such as the average congestion in the city or the capacity of a cargo bike, we conducted a sensitivity analysis to investigate how changes in these parameters would affect the outcome. A cost that we only considered in a sensitivity analysis is the cost for those parcels that are stored and picked-up from a DP. The extent of this cost depends on the infrastructure of the considered LSP. Smaller LSPs usually collaborate with shops from where customers can pick up their parcel, while larger LSP have their own service points which also offer other services. While in the first case the cost per picked-up parcel amounts to about €1, in the latter case the costs are almost negligible, since the infrastructure is there anyway (and which we assumed in the analysis).

The collaboration with shops presents an interesting option to readily extend the coverage of DPs without large capital investments, even though shops might not be suitable transhipment points for bike deliveries. A thorough analysis of different infrastructure options, however, is beyond the scope of this chapter. Such an analysis would require investigating such topics as the implementation of DPs, the upgrading to transhipment points of DPs, the payments to shops, and the purchasing and selling of cargo bikes and delivery vans, all of which present a considerable research challenge. In this chapter, we have focused on the cost of various transportation options, given a certain infrastructure set-up.

Finally, we implicitly assumed that all parcels are sufficiently small and light to be transported by a cargo bike. Even though this assumption probably holds for a vast majority of parcels, some parcels might have to be distributed by van. The investigation of such a hybrid system of bike and van deliveries is also beyond the scope of this work.

## 8.6 Conclusion

In this chapter, we have investigated the cost structure of different scenarios for urban B2C distribution in Antwerp. We generated demand on the basis of a real-world dataset and computed delivery routes with realistic cost values. By the comparison of different scenarios, we found that external costs, related to the transportation with delivery vans, account for 18%-28% of the operational costs. Also, we showed that the parcel delivery market is unbalanced in the sense that small LSPs have higher operational costs per delivery than a large LSP. Those operational costs can be reduced by stimulating self-pick-up, at the expense of rising external costs. Reversely, a bike delivery system can significantly reduce external costs but slightly increases the costs of LSPs. Consequently, neither self-pick-up nor bike deliveries alone seem to be beneficial for all stakeholders. However, a combination of both concepts, fueled by the implementation of additional DPs, represents a B2C delivery system that improves the quality of life in Antwerp and is also appealing to LSPs. The efficiency of such a delivery system could be further enhanced if, for instance, multiple LSPs collaborate and execute and plan the last-mile delivery jointly to make use of the economy-of-scales effect that we observed.

These results highlight the importance of looking at urban B2C distribution from a global perspective. Several stakeholders are involved that follow different goals and strategies. Public authorities have no incentive to support the introduction of self-pick-up and, likewise, an LSP will be rather unwilling to consider a bike delivery system when facing higher variable costs. However, these arguments arise from an isolated perspective, and they change if stakeholders look for alternatives jointly.

Furthermore, a fruitful dialogue between stakeholders requires a realistic assessment of possible 'what-if' scenarios. We demonstrated how such an assessment is possible with simulation studies. By means of simulation, we could model and evaluate different delivery strategies, which allowed us to extract reasonable cost values. Overall, our simulation model is relatively simple and easy to use. However, an accurate simulation requires accurate input data, and the availability of good data might present the biggest hindrance for simulation studies in practice. In our case, we used a real-world dataset of deliveries, the number of demands, information about the B2C delivery market in Belgium, cost values, and information about travel times and distances. Additionally, we observed that a slight change in parameter settings already impacts the results and following conclusions.

Finally, our study focused on the urban B2C parcel distribution in the city of Antwerp. Therefore, care should be taken in generalising our findings to other cities. Every city has a different size, infrastructure, demand density and market distribution among LSPs, and we have shown that all of these parameters affect the cost structure of B2C delivery services, and therefore also the practical relevance of the considered scenarios. Furthermore, we did not consider the time-dependency of travel behaviour and congestion. Especially during rush hour travel times and therefore routing choices might be different than during other times. However, the consideration of time-dependency requires detailed data that is available for only few cities. Another interesting extension could be an analysis of the precise effect of the location and number of DPs on costs. We showed that a higher density of DPs in a city can be beneficial for all stakeholders, and this effect could be further explored.

# 9 General conclusions

Routing problems are omnipresent in reality and will play an important role in the near future. Be it public transportation, private trips, delivery of goods or movements in supply chains, mobility is an inherent aspect of our interconnected world. In this thesis, I have developed models and algorithms to solve such routing problems. In each step I took great care to follow a thorough methodology and obtain findings that can hopefully be used beyond the scope of this work.

Combinatorial optimisation problems like routing are usually difficult to solve, and we found that data mining can be a promising tool to obtain more insights into the respective problem structure. These insights can help to design more efficient heuristics, as we have successfully demonstrated with the help of the classical VRP. A straight-forward next step would be the application of the same approach to rich vehicle routing problems, such as the vehicle routing problem with time windows (VRPTW). The VRPTW constitutes a highly popular problem due to its additional constraints and complexity, and a more thorough investigation of this problem could help to design efficient and simple solution frameworks. Indeed, the question "Can this data mining framework also be applied to the VRPTW?" has probably been the one most frequently asked after my talks, and confirms the interest in this research direction.

I used the obtained insights to design a heuristic for the VRP which is driven by local search. Many good heuristics for the VRP involve multiple components and parameters, often without offering sufficient motivation for the respective design choices. This situation has improved drastically in the last decade, with the classification of heuristics into meta-heuristics and the rise of standards for testing and tuning. In this thesis I tried to go a step further, and attempted to show that a good implementation and combination of existing ideas is sufficient to obtain a high-quality routing heuristic that can compete with the state-of-the-art. The resulting VRP heuristic can also be applied to tackle the largest instances in literature, and we showed that the pruning of the search space is an effective tool when facing very large problems. Of course, I did not start with these ideas in mind. As most enthusiastic newcomers to the field, I attempted to build a novel heuristic, to implement new components and frameworks. One reason for this 'trial and error' approach is that there are still some gaps in the understanding of heuristic components for the VRP. Such research is not always incentivised, since a heuristic for the VRP is expected to perform well rather than to be perfectly understood. From my experience in the context of this thesis, I would like to encourage researchers to conduct more studies in this direction, especially in the direction of local search.

Routing problems in practice are mostly governed by additional constrains and components. In this thesis, I had a particular look on how to integrate routing problems with inventory management. Such integrated problems arise in complex supply chain environments in which goods have to be stored and delivered. We showed that constrains from the inventory side can significantly impact the routing side. In more detailed experiments we investigated the influence of different parameters. We used these insights as the motivation to further study the integration of inventory and routing decisions. With a simulation framework, we showed that a tight collaboration between different depots on the routing and the inventory level can elevate the performance of the overall system. In the case that the exact locations of these depots have to be determined, we proposed a simple heuristic to solve the corresponding location routing problem. Generally, we found that an LRP can be solved by reducing it to a classical routing problem. On the basis of the successful results, I assume that the same idea can be applied to a wider range of problems with a routing component, since routing usually constitutes the most difficult part in the solution process.

Finally, I demonstrated the usability of routing algorithms in a practical context. In the design of urban transportation systems, it is crucial to study and compare different 'what-if' scenarios. We proposed a simulation framework to obtain a realistic assessment of possible scenarios. In the context of this study, it was less important to obtain high-quality routing solutions, and it was more important that routing solutions can be derived quickly and within a realistic model. Abstract models like the VRP rely on the assumption that all data is available and precise, whereas in reality, much data is based on rough estimations and is dynamic rather than static (e.g., traffic jams change the cost of a certain edge). Therefore, in practice it is often important to have flexible and fast algorithms at hand that can derive a good approximation of the routing, and the statement whether a certain heuristic performs 0.1% better or worse is less relevant. I do think that a thorough comparison between different methods on benchmark instances is beneficial to test and improve heuristics (and I also used it myself a lot in the course of this thesis). It clearly motivates researchers to further challenge the state-of-the-art. However, in terms of usability and research findings, it should not be the sole determinant of the quality of research outcomes. There are less take-aways from a paper that is filled with tables of benchmark results, than a paper that applies a thorough methodology and extracts a clear finding that can be generalized. Such research is done more and more, and allows to apply the research beyond the field of vehicle routing and heuristics, for instance, in practical case studies and interdisciplinary research. Such an application validates and strengthens the importance of the research field, now and in the future.

# Bibliography

Aickelin, U. and Dowsland, K. (2000). Exploiting problem structure in a genetic algorithm approach to a nurse rostering problem. *Journal of Scheduling*, 3(3):139–153.

Anderluh, A., Hemmelmayr, V. C., and Nolz, P. C. (2017). Synchronizing vans and cargo bikes in a city distribution network. *Central European Journal of Operations Research*, 25(2):345–376.

Andersson, H., Hoff, A., Christiansen, M., Hasle, G., and Løkketangen, A. (2010). Industrial aspects and literature survey: Combined inventory management and routing. *Computers & Operations Research*, 37(9):1515–1536.

Applegate, D., Cook, W., and Rohe, A. (2003). Chained Lin-Kernighan for large traveling salesman problems. *INFORMS Journal on Computing*, 15(1):82–92.

Arnold, F., Belinfante, A., Van der Berg, F., Guck, D., and Stoelinga, M. (2013a). DFTCalc: A tool for efficient fault tree analysis. In *International Conference on Computer Safety, Reliability, and Security*, pages 293–301. Springer.

Arnold, F., Cardenas, I., Sörensen, K., and Dewulf, W. (2017a). Simulation of B2C e-commerce distribution in Antwerp using cargo bikes and delivery points. *European Transport Research Review*, 10(1):2. doi:10.1007/s12544-017-0272-6.

Arnold, F., Gebler, D., Guck, D., and Hatefi, H. (2014a). A tutorial on interactive markov chains. In *Stochastic Model Checking. Rigorous Dependability Analysis Using Model Checking Techniques for Stochastic Systems*, pages 26–66. Springer.

Arnold, F., Gendreau, M., and Sörensen, K. (2017b). Efficiently solving very large scale routing problems. Working paper 75, Polytechnique Montreal, CIRRELT. Submitted to Computers & Operations Research.

Arnold, F., Guck, D., Kumar, R., and Stoelinga, M. (2014b). Sequential and parallel attack tree modelling. In *International Conference on Computer Safety, Reliability, and Security*, pages 291–299. Springer.

Arnold, F., Hermanns, H., Pulungan, R., and Stoelinga, M. (2014c). Time-dependent analysis of attacks. In *International Conference on Principles of Security and Trust*, pages 285–305. Springer.

Arnold, F., Pieters, W., and Stoelinga, M. (2013b). Quantitative penetration testing with item response theory. In *Information Assurance and Security (IAS), 2013 9th International Conference on*, pages 49–54. IEEE.

Arnold, F. and Sörensen, K. (2017). Knowledge-guided local search for the Vehicle Routing Problem. Working paper 12, University of Antwerp, Faculty of Applied Economics. Submitted to Computers & Operations Research.

Arnold, F. and Sörensen, K. (2018a). Efficiently solving location routing problems with routing heuristics. Working paper, University of Antwerp, Faculty of Applied Economics. Submitted to European Journal of Operations Research.

Arnold, F. and Sörensen, K. (2018b). What makes a VRP solution good? The generation of problem-specific knowledge for heuristics. *Computers & Operations Research*. Advance online publication. doi:10.1016/j.cor.2018.02.007.

Arnold, F., Sörensen, K., and Palhazi Cuervo, D. (2017c). From storage to shipment - The effect of ignoring inventory when planning routes. Working paper 2, University of Antwerp, Faculty of Applied Economics. Submitted to European Journal of Operations Research.

Arnold, F., Sörensen, K., and Springael, J. (2018). Vertical integration of distribution and inventory management with pooling. Working paper, University of Antwerp, Faculty of Applied Economics. Submitted to Omega.

Augerat, P., Belenguer, J. M., Benavent, E., Corberán, A., Naddef, D., and Rinaldi, G. (1995). Computational results with a branch-and-cut code for the capacitated vehicle routing problem. *Technical report 949-M. University Joseph Fourier, Grenoble, France*.

Baldacci, R., Toth, P., and Vigo, D. (2007). Recent advances in vehicle routing exact algorithms. *4OR*, 5(4):269–298.

Berman, O., Krass, D., and Mahdi Tajbakhsh, M. (2011). On the benefits of risk pooling in inventory management. *Production and operations management*, 20(1):57–71.

Beullens, P., Muyldermans, L., Cattrysse, D., and Van Oudheusden, D. (2003). A guided local search heuristic for the capacitated arc routing problem. *European Journal of Operational Research*, 147(3):629–643.

Bimpikis, K. and Markakis, M. G. (2015). Inventory pooling under heavy-tailed demand. *Management Science*, 62(6):1800–1813.

Braekers, K., Ramaekers, K., and Van Nieuwenhuyse, I. (2016). The vehicle routing problem: State of the art classification and review. *Computers & Industrial Engineering*, 99:300–313.

Breiman, L. (2001). Random forests. *Machine learning*, 45(1):5–32.

Brown, J. R. and Guiffrida, A. L. (2014). Carbon emissions comparison of last mile delivery versus customer pickup. *International Journal of Logistics Research and Applications*, 17(6):503–521.

Browne, M. (2001). E-commerce and urban transport. In *joint OECD/ECMT-Seminar 'The Impact of E-commerce on Transport', Session*, volume 4.

Browne, M., Allen, J., and Leonardi, J. (2011). Evaluating the use of an urban consolidation centre and electric vehicles in central london. *IATSS Research*, 35(1):1–6.

Burke, E. K., Hyde, M., Kendall, G., Ochoa, G., Özcan, E., and Woodward, J. R. (2010). A classification of hyper-heuristic approaches. In *Handbook of Metaheuristics*, pages 449–468. Springer.

Cardenas, I., Beckers, J., Vanelslander, T., Verhetsel, A., and Dewulf, W. (2016a). Spatial characteristics of failed and successful e-commerce deliveries in belgian cities. In *In Information Systems, Logistic and Supply Chain Conf.*

Cardenas, I. D., Dewulf, W., and Vanelslander, T. (2016b). The e-commerce parcel delivery market: developing a model for comparing home B2C deliveries vs pick-up points. In *World Conference on Transport Research: WCTR 2016, Shanghai, 10-15 July 2016*, pages 1–13.

Cattaruzza, D., Absi, N., Feillet, D., and González-Feliu, J. (2017). Vehicle routing problems for city logistics. *EURO Journal on Transportation and Logistics*, 6(1):51–79.

Chandra, P. and Fisher, M. L. (1994). Coordination of production and distribution planning. *European Journal of Operational Research*, 72(3):503–517.

Christopher, M. (2016). *Logistics & supply chain management*. Pearson UK.

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.

Coelho, L. C., Cordeau, J.-F., and Laporte, G. (2013). Thirty years of inventory routing. *Transportation Science*, 48(1):1–19.

Contardo, C., Cordeau, J.-F., and Gendron, B. (2014). A GRASP+ ILP-based metaheuristic for the capacitated location-routing problem. *Journal of Heuristics*, 20(1):1–38.

Conway, A., Fatisson, P.-E., Eickemeyer, P., Cheng, J., and Peters, D. (2012). Urban micro-consolidation and last mile goods delivery by freight-tricycle in manhattan: Opportunities and challenges. In *Proceedings of the 91st Transportation Research Board Annual Meeting, Washington, DC, USA*, pages 22–26.

Copenhagen Economics (2013). E-commerce and delivery. `https://www.copenhageneconomics.com/`. Accessed: 2018-02-05.

Cordeau, J.-F., Gendreau, M., and Laporte, G. (1997). A tabu search heuristic for periodic and multi-depot vehicle routing problems. *Networks*, 30(2):105–119.

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.

Corstjens, J., Caris, A., Depaire, B., and Sörensen, K. (2016). A multilevel methodology for analysing metaheuristic algorithms for the VRPTW. Working paper, Hasselt University.

Cullinane, S. (2009). From bricks to clicks: the impact of online retailing on transport and the environment. *Transport Reviews*, 29(6):759–776.

Dantzig, G. B. and Ramser, J. H. (1959). The truck dispatching problem. *Management Science*, 6(1):80–91.

Delfmann, W., Albers, S., and Gehring, M. (2002). The impact of electronic commerce on logistics service providers. *International Journal of Physical Distribution & Logistics Management*, 32(3):203–222.

Deloitte (2015). Click and collect booms in europe. `https://www2.deloitte.com/`. Accessed: 2018-02-05.

Demir, E., Bektaş, T., and Laporte, G. (2014). A review of recent research on green road freight transportation. *European Journal of Operational Research*, 237(3):775–793.

Drake, J. H., Kililis, N., and Özcan, E. (2013). Generation of vns components with grammatical evolution for vehicle routing. In *European Conference on Genetic Programming*, pages 25–36. Springer.

Ducret, R. (2014). Parcel deliveries and urban logistics: Changes and challenges in the courier express and parcel sector in europe—the french case. *Research in Transportation Business & Management*, 11:15–22.

Durand, B. and Gonzalez-Feliu, J. (2012). Urban logistics and e-grocery: have proximity delivery services a positive impact on shopping trips? *Procedia-Social and Behavioral Sciences*, 39:510–520.

Edwards, J., McKinnon, A., Cherrett, T., McLeod, F., and Song, L. (2010). Carbon dioxide benefits of using collection-delivery points for failed home deliveries in the united kingdom. *Transportation Research Record: Journal of the Transportation Research Board*, (2191):136–143.

Ergun, Ö., Orlin, J. B., and Steele-Feldman, A. (2006). Creating very large scale neighborhoods out of smaller ones by compounding moves. *Journal of Heuristics*, 12(1):115–140.

Escobar, J. W., Linfati, R., Baldoquin, M. G., and Toth, P. (2014). A granular variable tabu neighborhood search for the capacitated location-routing problem. *Transportation Research Part B: Methodological*, 67:344–356.

Escobar, J. W., Linfati, R., and Toth, P. (2013). A two-phase hybrid heuristic algorithm for the capacitated location-routing problem. *Computers & Operations Research*, 40(1):70–79.

EY (2015). The green mile? `http://www.ey.com/`. Accessed: 2018-02-05.

Federgruen, A. and Zipkin, P. (1984). A combined vehicle routing and inventory allocation problem. *Operations Research*, 32(5):1019–1037.

Gendreau, M., Hertz, A., and Laporte, G. (1992). New insertion and postoptimization procedures for the traveling salesman problem. *Operations Research*, 40(6):1086–1094.

Gerchak, Y. and He, Q.-M. (2003). On the relation between the benefits of risk pooling and the variability of demand. *IIE Transactions*, 35(11):1027–1031.

Gevaers, R., Van de Voorde, E., and Vanelslander, T. (2014). Cost modelling and simulation of last-mile characteristics in an innovative B2C supply chain environment with implications on urban areas and cities. *Procedia-Social and Behavioral Sciences*, 125:398–411.

Glover, F. (1996). Ejection chains, reference structures and alternating path methods for traveling salesman problems. *Discrete Applied Mathematics*, 65(1):223–253.

Glover, F. (2007). Tabu search—uncharted domains. *Annals of Operations Research*, 149(1):89–98.

Golden, B. L., Assad, A. A., and Wasil, E. A. (2002). Routing vehicles in the real world: applications in the solid waste, beverage, food, dairy, and newspaper industries. In *The vehicle routing problem*, pages 245–286. SIAM.

Golden, B. L., Wasil, E. A., Kelly, J. P., and Chao, I.-M. (1998). The impact of metaheuristics on solving the vehicle routing problem: algorithms, problem sets, and computational results. In *Fleet management and logistics*, pages 33–56. Springer.

Groër, C., Golden, B., and Wasil, E. (2010). A library of local search heuristics for the vehicle routing problem. *Mathematical Programming Computation*, 2(2):79–101.

Guemri, O., Beldjilali, B., Bekrar, A., and Belalem, G. (2016). Two-stage heuristic algorithm for the large-scale capacitated location routing problem. *International Journal of Mathematical Modelling and Numerical Optimisation*, 7(1):97–119.

Gutjahr, W. (2010). Stochastic search in metaheuristics. In Gendreau, M. and Potvin, J.-Y., editors, *Handbook of Metaheuristics (2nd ed.)*, volume 146 of *International Series in Operations Research & Management Science*, pages 573–597. Springer, New York.

Harks, T., König, F. G., and Matuschke, J. (2013). Approximation algorithms for capacitated location routing. *Transportation Science*, 47(1):3–22.

Helsgaun, K. (2000). An effective implementation of the lin–kernighan traveling salesman heuristic. *European Journal of Operational Research*, 126(1):106–130.

Helsgaun, K. (2017). An extension of the lin-kernighan-helsgaun TSP solver for constrained traveling salesman and vehicle routing problems. Technical report, Roskilde Universitet.

Herer, Y. T., Tzur, M., and Yücesan, E. (2006). The multilocation transshipment problem. *IIE Transactions*, 38(3):185–200.

Hiassat, A., Diabat, A., and Rahwan, I. (2017). A genetic algorithm approach for location-inventory-routing problem with perishable products. *Journal of Manufacturing Systems*, 42:93 – 103.

Hsu, C.-W., Chang, C.-C., Lin, C.-J., et al. (2003). *A practical guide to support vector classification*. Taipei, Taiwan.

Irnich, S., Funke, B., and Grünert, T. (2006). Sequential search and its application to vehicle-routing problems. *Computers & Operations Research*, 33(8):2405–2429.

Jäger, G. and Arnold, F. (2015). SAT and IP based algorithms for magic labeling including a complete search for total magic labelings. *Journal of Discrete Algorithms*, 31:87–103.

Johnson, D. S., Papadimitriou, C. H., and Yannakakis, M. (1988). How easy is local search? *Journal of Computer and System Sciences*, 37(1):79–100.

Juan, A. A., Faulin, J., Grasman, S. E., Rabe, M., and Figueira, G. (2015). A review of simheuristics: Extending metaheuristics to deal with stochastic combinatorial optimization problems. *Operations Research Perspectives*, 2:62–72.

Kilby, P., Prosser, P., and Shaw, P. (1999). Guided local search for the vehicle routing problem with time windows. In *Meta-heuristics*, pages 473–486. Springer.

Kytöjoki, J., Nuortio, T., Bräysy, O., and Gendreau, M. (2007). An efficient variable neighborhood search heuristic for very large scale vehicle routing problems. *Computers & Operations Research*, 34(9):2743–2757.

Lahyani, R., Khemakhem, M., and Semet, F. (2015). Rich vehicle routing problems: From a taxonomy to a definition. *European Journal of Operational Research*, 241(1):1–14.

Laporte, G. (2007). What you should know about the vehicle routing problem. *Naval Research Logistics (NRL)*, 54(8):811–819.

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

Laporte, G., Ropke, S., and Vidal, T. (2014). Chapter 4: Heuristics for the vehicle routing problem. In *Vehicle Routing: Problems, Methods, and Applications, Second Edition*, pages 87–116. SIAM.

Lin, I. and Mahmassani, H. (2002). Can online grocers deliver?: Some logistics considerations. *Transportation Research Record: Journal of the Transportation Research Board*, (1817):17–24.

Lin, S. and Kernighan, B. W. (1973). An effective heuristic algorithm for the traveling-salesman problem. *Operations Research*, 21(2):498–516.

Liu, S. C. and Lee, S. B. (2003). A two-phase heuristic method for the multi-depot location routing problem taking inventory control decisions into consideration. *The International Journal of Advanced Manufacturing Technology*, 22(11):941–950.

Maes, J. and Vanelslander, T. (2012). The use of bicycle messengers in the logistics chain, concepts further revised. *Procedia-Social and Behavioral Sciences*, 39:409–423.

Marshall, R. J., Johnston, M., and Zhang, M. (2014). Hyper-heuristics, grammatical evolution and the capacitated vehicle routing problem. In *Proceedings of the Companion Publication of the 2014 Annual Conference on Genetic and Evolutionary Computation*, pages 71–72. ACM.

Martens, D., Baesens, B., Van Gestel, T., and Vanthienen, J. (2007). Comprehensible credit scoring models using rule extraction from support vector machines. *European Journal of Operational Research*, 183(3):1466–1476.

McNabb, M. E., Weir, J. D., Hill, R. R., and Hall, S. N. (2015). Testing local search move operators on the vehicle routing problem with split deliveries and time windows. *Computers & Operations Research*, 56:93–109.

Mendoza, J. E., Guéret, C., Hoskins, M., Lobit, H., Pillac, V., Vidal, T., and Vigo, D. (2014). VRP-REP: the vehicle routing community repository. In *Third Meeting of the EURO Working Group on Vehicle Routing and Logistics Optimization (VeRoLog). Oslo, Norway*.

Mester, D. and Bräysy, O. (2007). Active-guided evolution strategies for large-scale capacitated vehicle routing problems. *Computers & Operations Research*, 34(10):2964–2975.

Min, H. and Zhou, G. (2002). Supply chain modeling: past, present and future. *Computers & Industrial Engineering*, 43(1):231–249.

Mladenović, N. and Hansen, P. (1997). Variable neighborhood search. *Computers & Operations Research*, 24(11):1097–1100.

Moin, N. and Salhi, S. (2007). Inventory routing problems: a logistical overview. *Journal of the Operational Research Society*, 58(9):1185–1194.

Mokhtarian, P. L. (2004). A conceptual analysis of the transportation impacts of B2C e-commerce. *Transportation*, 31(3):257–284.

Montoya-Torres, J., Franco, J., Isaza, S., Jiménez, H., and Herazo-Padilla, N. (2015). A literature review on the vehicle routing problem with multiple depots. *Computers & Industrial Engineering*, 79:115–129.

Morganti, E., Dablanc, L., and Fortin, F. (2014a). Final deliveries for online shopping: The deployment of pickup point networks in urban and suburban areas. *Research in Transportation Business & Management*, 11:23–31.

Morganti, E., Seidel, S., Blanquart, C., Dablanc, L., and Lenz, B. (2014b). The impact of e-commerce on final deliveries: alternative parcel delivery services in France and Germany. *Transportation Research Procedia*, 4:178–190.

MOVE, D. (2014). Update of the handbook on external costs of transport. *DG MOVE*.

Nagata, Y. and Bräysy, O. (2009). Edge assembly-based memetic algorithm for the capacitated vehicle routing problem. *Networks*, 54(4):205.

Nuzzolo, A., Comi, A., and Rosati, L. (2014). City logistics long-term planning: simulation of shopping mobility and goods restocking and related support systems. *International Journal of Urban Sciences*, 18(2):201–217.

PassMark Software (2018). CPU benchmarks. `https://www.cpubenchmark.net/`. Accessed: 2018-02-05.

Paterson, C., Kiesmüller, G., Teunter, R., and Glazebrook, K. (2011). Inventory models with lateral transshipments: A review. *European Journal of Operational Research*, 210(2):125–136.

Pecin, D., Pessoa, A., Poggi, M., and Uchoa, E. (2017). Improved branch-cut-and-price for capacitated vehicle routing. *Mathematical Programming Computation*, 9(1):61–100.

Pichpibul, T. and Kawtummachai, R. (2012). An improved Clarke and Wright savings algorithm for the capacitated vehicle routing problem. *ScienceAsia*, 38(3):307–318.

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

Pisinger, D. and Ropke, S. (2010). Large neighborhood search. In *Handbook of Metaheuristics*, pages 399–419. Springer.

Prins, C., Prodhon, C., and Calvo, R. W. (2006). Solving the capacitated location-routing problem by a grasp complemented by a learning process and a path relinking. *4OR: A Quarterly Journal of Operations Research*, 4(3):221–238.

Prins, C., Prodhon, C., Ruiz, A., Soriano, P., and Wolfler Calvo, R. (2007). Solving the capacitated location-routing problem by a cooperative lagrangean relaxation-granular tabu search heuristic. *Transportation Science*, 41(4):470–483.

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

Prud'homme, R. and Lee, C.-W. (1999). Size, sprawl, speed and the efficiency of cities. *Urban Studies*, 36(11):1849–1858.

Rego, C. (2001). Node-ejection chains for the vehicle routing problem: Sequential and parallel algorithms. *Parallel Computing*, 27(3):201–222.

Reimann, M., Tavares Neto, R., and Bogendorfer, E. (2014). Joint optimization of production planning and vehicle routing problems: a review of existing strategies. *Pesquisa Operacional*, 34(2):189–214.

Reinelt, G. (1991). TSPLIB–A traveling salesman problem library. *ORSA Journal on Computing*, 3(4):376–384.

Russo, F. and Comi, A. (2011). A model system for the ex-ante assessment of city logistics measures. *Research in Transportation Economics*, 31(1):81–87.

Salhi, S. (2014). Handbook of metaheuristics. *Journal of the Operational Research Society*, 65(2):320–320.

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

Sarmiento, A. M. and Nagi, R. (1999). A review of integrated analysis of production–distribution systems. *IIE Transactions*, 31(11):1061–1074.

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

Schneider, M. and Löffler, M. (2017). Large composite neighborhoods for the capacitated location-routing problem. *Transportation Science*.

Shahid, R., Bertazzon, S., Knudtson, M. L., and Ghali, W. A. (2009). Comparison of distance measures in spatial analytical modeling for health service planning. *BMC Health Services Research*, 9(1):200.

Shmueli, G. et al. (2010). To explain or to predict? *Statistical Science*, 25(3):289–310.

Song, L., Cherrett, T., McLeod, F., and Guan, W. (2009). Addressing the last mile problem–the transport impacts of collection/delivery points, paper given to the 88th annual meeting of the transportation research board.

Sörensen, K. (2015). Metaheuristics—the metaphor exposed. *International Transactions in Operational Research*, 22(1):3–18.

Sörensen, K., Arnold, F., and Palhazi Cuervo, D. (2017). A critical analysis of the "improved Clarke and Wright savings algorithm". *International Transactions in Operational Research*. Advance online publication. doi:10.1111/itor.12443.

Sörensen, K., Sevaux, M., and Glover, F. (2018). A history of metaheuristics. pages 1–18. Springer.

Sörensen, K., Sevaux, M., and Schittekat, P. (2008). Multiple neighbourhood search in commercial VRP packages: Evolving towards self-adaptive methods. In *Adaptive and Multilevel Metaheuristics*, pages 239–253. Springer.

Subramanian, A., Uchoa, E., and Ochi, L. S. (2013). A hybrid algorithm for a class of vehicle routing problems. *Computers & Operations Research*, 40(10):2519–2531.

Tagaras, G. (1999). Pooling in multi-location periodic inventory distribution systems. *Omega*, 27(1):39–59.

Taillard, É., Badeau, P., Gendreau, M., Guertin, F., and Potvin, J.-Y. (1997). A tabu search heuristic for the vehicle routing problem with soft time windows. *Transportation Science*, 31(2):170–186.

Taniguchi, E. and Kakimoto, Y. (2004). Modelling effects of e-commerce on urban freight transport. *Logistics Systems for Sustainable Cities*, 10:135–146.

Teo, J. S., Taniguchi, E., and Qureshi, A. G. (2012). Evaluating city logistics measure in e-commerce with multiagent systems. *Procedia-Social and Behavioral Sciences*, 39:349–359.

Toth, P. and Vigo, D. (2002). *The vehicle routing problem*. SIAM.

Toth, P. and Vigo, D. (2003). The granular tabu search and its application to the vehicle-routing problem. *Informs Journal on Computing*, 15(4):333–346.

Tuzun, D. and Burke, L. I. (1999). A two-phase tabu search approach to the location routing problem. *European Journal of Operational Research*, 116(1):87–99.

Tversky, A. and Kahneman, D. (1974). Judgment under uncertainty: Heuristics and biases. *Science*, 185(4157):1124–1131.

Uchoa, E., Pecin, D., Pessoa, A., Poggi, M., Vidal, T., and Subramanian, A. (2017). New benchmark instances for the capacitated vehicle routing problem. *European Journal of Operational Research*, 257(3):845–858.

Van Der Helm, P. (2015). Competitiveness of logistics service centers in the high volume parcel delivery market. *Technical University of Eindhoven, Eindhoven*.

Van Duin, J., De Goffau, W., Wiegmans, B., Tavasszy, L., and Saes, M. (2016). Improving home delivery efficiency by using principles of address intelligence for B2C deliveries. *Transportation Research Procedia*, 12:14–25.

Van Duin, J., Kortmann, R., and Van den Boogaard, S. (2014). City logistics through the canals? A simulation study on freight waterborne transport in the inner-city of Amsterdam. *International Journal of Urban Sciences*, 18(2):186–200.

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.

Vidal, T., Crainic, T. G., Gendreau, M., and Prins, C. (2013a). Heuristics for multi-attribute vehicle routing problems: A survey and synthesis. *European Journal of Operational Research*, 231(1):1–21.

Vidal, T., Crainic, T. G., Gendreau, M., and Prins, C. (2013b). A hybrid genetic algorithm with adaptive diversity management for a large class of vehicle routing problems with time-windows. *Computers & Operations Research*, 40(1):475–489.

Visser, J., Nemoto, T., and Browne, M. (2014). Home delivery and the impacts on urban freight transport: A review. *Procedia-Social and Behavioral Sciences*, 125:15–27.

Voudouris, C. and Tsang, E. P. (2003). *Guided local search*. Springer.

Weltevreden, J. W. (2007). Substitution or complementarity? How the internet changes city centre shopping. *Journal of Retailing and Consumer Services*, 14(3):192–207.

Weltevreden, J. W. (2008). B2C e-commerce logistics: the rise of collection-and-delivery points in the netherlands. *International Journal of Retail & Distribution Management*, 36(8):638–660.

Weltevreden, J. W. and Rotem-Mindali, O. (2009). Mobility effects of B2C and C2C e-commerce in the netherlands: a quantitative assessment. *Journal of Transport Geography*, 17(2):83–92.

Wilson, R. (2006). 17th annual state of logistics report. *Council of Supply Chain Management Professionals*, 17(1).

Xavier, Ian (2018). CVRPLIB. `http://vrp.atd-lab.inf.puc-rio.br/index.php/en/`. Accessed: 2018-02-05.