EMERGING TRENDS IN LOGISTICS:
NEW MODELS AND ALGORITHMS FOR VEHICLE ROUTING

Vom Fachbereich Wirtschaftswissenschaften der Technischen Universität Kaiserslautern zur Verleihung des akademischen Grades Doctor rerum politicarum (Dr. rer. pol.) genehmigte

DISSERTATION

vorgelegt von

Dipl.-Wirtsch.-Ing. Dominik Janosch Goeke

Tag der mündlichen Prüfung: 23. Mai 2018
Dekan und Vorsitzender der Prüfungskommission: Prof. Dr. Jan Wenzelburger
Erster Berichterstatter: Prof. Dr. Oliver Wendt, Technische Universität Kaiserslautern
Zweiter Berichterstatter: Prof. Dr. Stefan Irnich, Johannes Gutenberg-Universität Mainz

D 386
(2018)

Contents

List of Figures

List of Tables

List of Abbreviations

2E-CVRP two-echelon capacitated vehicle-routing problem

2PLs second-party logistics providers

3PLs third-party logistics providers

4PLs fourth-party logistics providers

ALNS adaptive large neighborhood search

ATC arrival time consistency

BKS best-known solution

BKV best-known value

BPC branch-price-and-cut

CCG cluster column generation

CG column generation

CNV cumulative number of vehicles

ConVRP consistent vehicle-routing problem

CTD cumulative traveled distance

CVRP capacitated vehicle-routing problem

DC driver consistency

DP dynamic programming

E-VRPTW electric vehicle-routing problem with time windows

E-VRPTWMF electric vehicle-routing problem with mixed fleet

ECV electric commercial vehicle

ESPPRC elementary shortest-path problem with resource constraints

EV electric vehicle

GRASP greedy randomized adaptive search

GTS granular tabu search

ICCV internal combustion commercial vehicle

ICV internal combustion vehicle

LB lower bound

LDS limited discrepancy search

lmSR limited memory subset row

LNS large neighborhood search

LNS-SP large neighborhood search with set partitioning

LP linear programming

LS local search

MILP mixed-integer linear programming

NFL no-free-lunch

OR operations research

PDPTW pickup and delivery problem with time windows

PDPTW-EV electric pickup and delivery problem with time windows and electric vehicles

PRP pollution-routing problem

REF resource extension function

SA simulated annealing

SEC generalized subtour elimination constraint

SEC subtour elimination constraint

SP set partitioning

SR subset row

TS tabu search

TSP traveling-salesman problem

UB upper bound

VND variable neighborhood descent

VNS variable neighborhood search

VRP vehicle-routing problem

VRPPC vehicle-routing problem with private fleet and common carrier

VRPTW vehicle-routing problem with time windows

Chapter 1

Thesis Overview

Section 1.1 highlights recent trends in logistics and provides the context for the four optimization problems discussed in this thesis. Section 1.2 gives a brief introduction to optimization. Finally, Section 1.3 motivates the relevance of each of the problems studied and explains how this thesis contributes to them in terms of new models and algorithms.

1.1. Background

The logistics sector is one of the main drivers of economical growth and prosperity. In the European Union, the transport and storage services sector accounted for 5.1% of Gross Value Added in 2014[1] (European Commission, 2016). However, this figure only hints at the importance of logistics as many other economic activities rely on it. Disruptive social and technological transitions begin to change the way in which logistics services are provided. Besides changing the work processes related to logistics, this transformation can affect everyday life, i.e., how people shop, how they live in urban and rural area, how they are taken care when they age, and ultimately how human health is affected by transportation related emissions.

Sophisticated algorithms enable this transition because simple strategies are not sufficient to cope with these complex and interrelated planning tasks: "Logistics management is that part of supply chain management that plans, implements, and controls the efficient, effective forward and reverse flow and storage of goods, services, and related information between the point of origin and the point of consumption in order to meet customers' requirements. Logistics management activities typically include inbound and outbound transportation management, fleet management, warehousing, materials handling, order fulfillment, logistics network design, inventory management, supply/demand planning, and management of third party logistics services providers." (Council of Supply Chain Management Professionals, 2013) The archetype of many planning tasks in logistics is the vehicle-routing problem (VRP). The VRP is about planning routes that start and end at a depot in order to visit a set of customers with minimum cost. This thesis aims to develop and study algorithms for VRPs that emerge from current real-world trends. In the following, trends are mapped out that potentially have a high impact on logistics. First, trends that are mainly motivated by organizational changes on the supply side are described. Then, trends motivated by changes in consumer behavior are discussed. Finally, advances in technology are presented, putting a focus on electric vehicles (EVs).

[1] Excluding self-provided transportation services by non-logistics companies.

Logistics processes change from an organizational perspective. A traditional view on supply chains is that the flow of material and information through the participants is organized linearly. These unidirectional relationships are undergoing a transition to non-linear networks and will eventually evolve into collaborative supply chain clusters (Stevens and Johnson, 2016). The management of these complex supply networks is often transferred to fourth-party logistics providers (4PLs) that do not possess own transportation capabilities but the information technology and know-how to supervise the network. Integrating different businesses, 4PLs can encourage the sharing of existing networks that so far only coexist. Groceries and medicine can be transported using the same cold-chain truck. Vehicles for passenger transportation can also be used for the delivery of parcels. Examples for such co-utilization are taxi cabs that perform home-deliveries[2], coaches that transport parcels between stops[3], and the well-tried practice of transporting parcels with passenger trains between stations[4].

Prediction of customer behavior can be used to send out shipments to geographical areas before customer orders arrive from there. The actual delivery address is then specified while the package is in transit. Anticipatory shipping—patented by Amazon in the US in 2013 (Spiegel et al., 2013)—increases delivery reliability and helps to realize same-day deliveries. Demand peaks can be countered that otherwise would cause high costs because additional transportation capacity must be acquired. Dynamic pricing is another instrument that can increase robustness against demand peaks and exogenous disruptions. Prices that reflect cost can be a means to dynamically balance demand and supply. Online marketplaces that aim at price-conscious customers are established. For example, the marketplace Jet[5] reduces prices when (i) larger quantities are bought, (ii) purchase orders are issued close to distribution centers, and (iii) bundles of goods are purchased that are stocked together.

Logistics marketplaces are online platforms that allow customers to place transportation requests from a pickup to a destination location. Transporters then bid for these contracts. Since 2015, several marketplaces[6] have been started that for the first time allow to book cargo electronically. Besides reduced transaction times, the number of empty tours can be reduced, excess capacity can be sold, and customers have a convenient way to purchase transportation capacity for less-than-truckload cargo.

Focusing on the consumers' side, online commerce is growing, and online and offline commerce begin to coalesce (Bernon, Cullen, and Gorst, 2016). In this new environment, a distinguishing competitive factor for retail operations is the quality of last-mile deliveries. The organization of inventory is crucial for high-quality delivery , i.e., the proximity to the demand and the quantity of items kept in stock. As it is expensive to provide an ubiquitous infrastructure, new alternative concepts are introduced: (i) the entire fulfillment service, including storage, can be rented from specialized companies[7], (ii) warehouse space can be shared via online market-places that act as intermediary[8], (iii) conventional stores can act as micro-warehouses (as practiced by IKEA), and (iv) inventory can be laterally shifted between stores (Paterson et al., 2011). There are also ideas to improve customer convenience by new modes of delivery

[2]Lineup (Australia): `http://www.taxilineup.com`, accessed 12/11/2017
[3]Greyhound Package Express (USA): `http://www.shipgreyhound.com`, accessed 12/11/2017
[4]time:matters (Germany): `https://www.time-matters.com/de/transportloesungen/ickurier`, accessed 12/11/2017
[5]Jet (USA): `https://jet.com`, accessed 12/11/2017
[6]For example: (i) FreightNet: `http://www.freightnet.com`, (ii) SimpliShip: `https://www.simpliship.com`, and (iii) Freightos: `https://www.freightos.com`, all accessed 12/11/2017
[7]Shipwire: `https://www.shipwire.com`, accessed 12/11/2017
[8]Flexe: `https:/https://www.flexe.com`, accessed 12/11/2017

and return, i.e., to deliver orders to personal parcel lockers, car trunks, or stores, and to make all ways of delivery also available for the simple return of products.

Two other trends go hand in hand: end consumers—turning into prosumers—desire highly-customized products and want to be involved in the design process (Wulfsberg, Redlich, and Bruhns, 2011); and industries aim to produce locally and just-in-time, e.g., rarely used parts on-demand to reduce storage cost (Lanz and Tuokko, 2017). Both trends facilitate a transition towards micro-factories that are located close to the point-of-need. This leads to a highly decentralized production, ideally provided at costs close to that of modern mass-production. For logistics, micro-factories vastly increase the complexity. Services must be rendered in shorter times, pickups and deliveries become more frequent, and the size of the network increases. In addition, the quantity of cargo loaded in one operation decreases, so it becomes more important to already consider at the level of route planning how the items are later arranged within the trucks so that they can be easily retrieved. When requests can arrive at any time, it might be necessary to implement dynamic re-routing strategies.

From a technological perspective, the installation of micro-depots in delivery areas is a main opportunity to decrease traffic caused by last-mile package shipping. Electrified load tricycles can pick up small packages and deliver them local-emission-free in the proximity of these micro-depots. Typically, these bikes can carry a load between 1 m^2 and 2 m^2 and they have an electrically supported range that is about 30 km (Bogdanski, 2017). Regular depots are not suited as pickup points as they are usually located in the city's outskirts and to reposition them would claim considerable space in the city center. For example, UPS is conducting a model-project in Hamburg, Germany since 2012, where swap-bodies (exchangeable containers with four up-folding legs) are placed by trucks at four locations. In this way, 13 delivery men using bikes proved to be sufficient to replace seven diesel-powered trucks with 7.5 ton, driving in total 800 km (Reichel, 2016).

Drone-based deliveries have the advantage of quicker deliveries, less inner-city traffic, and independence from road networks. For example, the airborne delivery drone M2[9] produced by Matternet can travel up to 20 km carrying a payload of 1 kg with one battery charge and carry a maximum payload of 2 kg. However, one obstacle is the limitation of the drones' range when used as a stand-alone system. In 2016, Amazon proposed one possible remedy (Berg, Isaacs, and Blodgett, 2016): Drone platforms (airships) are positioned above a delivery area. They dispatch drones which exploit the force of gravity to descend to their target destination, and then they continue without payload and therefore lower energy consumption to ground-based collection points. Another approach was recently introduced by Daimler[10]. They presented prototypes of delivery vans serving as road-based drone platforms. In urban centers, smaller autonomous transport vehicles (droids) are used to drive into pedestrian areas. Another van prototype operates airborne drones for faster deliveries in rural regions.

The deployment of EVs, also referred to as zero-emission vehicles, is becoming a major topic of public and political interest. Incentives for EVs are set in motion, e.g., direct payments, tax credits, parking benefits, exemptions from driving bans for certain types of vehicles or kilometer taxes, lifting of legal constraints or funding of research. For an overview of current policies and measures, see, e.g., Ajanovic

[9] `https://www.post.ch/companycars/-/media/post/ueber-uns/medienmitteilungen/2017/drohnen/spezifikationen-matternet-m2.pdf`, accessed 12/11/2017

[10] `https://www.daimler.com/innovation/specials/future-transportation-vans/vans-drones.html`, accessed 12/11/2017

and Haas (2016). The European Union set a binding target for the year 2030 to reduce greenhouse gas emissions by at least 40% compared to the level of 1990. Sectors not participating in the Emission Trading System, e.g., non-aviation transportation, have a lower but still considerable goal to reduce emissions by 30% compared to the level of 2005 (European Commission, 2014). As the transportation sector caused 25.8% of all greenhouse gas emissions in the EU in 2015 (European Environment Agency, 2017b), it is of particular importance. But, especially the transportation sector is encountering difficulties to meet these targets. Between 1990 and 2015, emissions of green house gases increased in the transportation sector by 15.9% opposed to total emissions over all sectors that decreased by 21.1% (European Environment Agency, 2017a). Especially urban areas are affected negatively by transportation, with freight transportation accounting for a significant share. For example, in London 17% percent of all road traffic is related to freight transportation, thus causing 24% of all CO_2 emissions related to road traffic (Wainwright, 2015). Using electric commercial vehicles (ECVs) presents an opportunity to address these problems.

Because ECVs have never been widely used, transport operators have little knowledge about the actual specifications of currently available vehicles and there exist many concerns about using them. The most prominent ones being the limited range of ECVs and failing batteries. If ECVs are to be used, these anxieties need to be dispelled. Especially so, as typically many businesses in the field of city logistics are small and operate only few vehicles (Lebeau, Macharis, and van Mierlo, 2016). To this end, the Commission of the European Union conducted a demonstration project: *Freight Electric Vehicles in Urban Europe*, in which 127 ECVs were used in different applications in eight large European cities. They were found to have a satisfying technical performance that is mostly according to the manufacturer's specification and to be suitable for most tasks in urban logistics with few changes necessary to the companies processes (Quak, Nesterova, and Rooijen, 2016).

However, there are also concerns raised concerning ECVs. Many argue that they are not competitive from a cost point of view under various scenarios (Davis and Figliozzi, 2013; Lebeau, Macharis, van Mierlo, and Lebeau, 2015). This can mainly be attributed to the high cost of the battery and the cost of the power electronics. As heavy trucks need larger batteries, this is, in particular, a setback for trucks above the 3.5 ton category. In addition, there is evidence that the production of EVs has a higher environmental impact than the production of internal combustion vehicles (ICVs) and is accompanied by increased levels of (human) toxicity (Hawkins, Gausen, and Strømman, 2012; Nordelöf et al., 2014; van Mierlo, Messagie, and Rangaraju, 2017), that can be mainly attributed to mining processes. In Lithium-ion batteries, cobalt (possible blended with other metals) is the currently most used and attractive cathode material but also the major cost factor (Nitta et al., 2015). About half of the world production of cobalt originates from the Katanga Province, Congo; mainly from artisanal cobalt mining, i.e., independent miners working with primitive hand tools. The working conditions are often poor and the risk of accidents, exposure to harmful substances, and child labor is high (Tsurukawa, Prakash, and Manhart, 2011). However, despite these negative factors, the authors stress the importance of artisanal mining to alleviate poverty because it requires little skill and capital.

Another severe issue is that external costs—costs that are not paid for by the users of the specific transportation service but that are, e.g., related to air pollution, noise and climate change—were expected to be lower for EVs (Juan et al., 2016). But, there is evidence that this is only true if certain premises are met. Jochem, Doll, and Fichtner (2016) find that neither under present conditions, nor under projections

for the year 2030, significant differences over the entire lifetime of a regular passenger car in Germany exist. This can be mainly attributed to the heavy reliance of the German power generation on fossil fuel power plants and the ratio between driving in urban and rural areas for standard driving profiles. Nevertheless, they conclude that reducing tail-pipe emissions in (mega-)cities might still be advantageous; especially, if life quality is significantly compromised by air pollution.

1.2. Fundamentals

This work focuses on combinatorial optimization problems that can be stated as: Find the element $x^* \in X$ that is associated with the extremum of an objective function f defined over the elements $x \in X$, where X denotes a *finite* set of possible solutions to the investigated problem, i.e., look for $x^* = \arg\min_{x \in X} f(x)$ with $|X| < \infty$. However, typically the *search space* X, shaped by the problem constraints, is not known explicitly and too large to be entirely enumerated. Therefore, one resorts to algorithms that implicitly search the space X. In this work, the primary emphasis is on heuristics that sample step-by-step from X by defining operators that generate at iteration t the neighborhood $\mathcal{N}(x_t) \subset X$ of solution x_t. Then, the next solution $x_{t+1} \in \mathcal{N}(x_t)$ is selected according to some criterion. The way in which this sampling is performed often follows a specific set of (stochastic) rules that are independent from the problem investigated. These universal frameworks are generally referred to as metaheuristics since the term was coined in Glover (1986); for an excellent introduction to metaheuristics, see, e.g., Gendreau and Potvin (2010). As second algorithmic contribution, two variants of column generation (CG) are studied. CG is an exact solution approach that is suitable if each solution $x \in X$ is composed of several meaningful variables (denoted as columns), e.g., the solution to a VRP consists of several routes, each route serving a subset of customers. Consequently, it is sufficient to select the correct routes if the set of all possible routes has been listed. As this is typically prohibitive, CG resorts to iteratively generating new columns in a pricing-subproblem, e.g., adding promising routes until no improving columns can be found. Recommendable introductions to CG can be found in, e.g., Desrosiers and Lübbecke (2005) and Feillet (2010).

The remainder of this section is dedicated to a discussion of the limitations of metaheuristics and why it is important to devise algorithms that are tailored to the problem that needs to be solved. Wolpert and Macready (1997) present no-free-lunch (NFL) theorems for optimization problems, and show that no black-box algorithm—no information about the problem solved is used—can be expected to perform better than any other *on all problems*. An interesting and surprising implication is that if one aims to maximize an objective function, a hill-climbing strategy does on average (with the same number of steps) not work better than random search, or even a hill-descending strategy. Certainly, if hill-climbing is confronted with only monotonic increasing objective functions, i.e., $f(x_{t+1}) > f(x_t)\ \forall t$, it performs better, and when confronted with flat ones $f(x_{t+1}) = f(x_t)\ \forall t$ the performance of all three strategies would be the same. Yet, for objective functions shaped in every conceivable profile, no strategy, assume that the strategy randomly continues the search from a different point whenever it encounters a local optimum, dominates the other (Ho and Pepyne, 2002).[11] Therefore, if one desired to debate the relevance

[11] Donald Duck reached this conclusion already 50 years earlier when he joined the Great Society of Flippists—"Life is but a gamble! Let Flipism chart your ramble"—and decided to let coin flips govern all his decisions from then on (Barks, 1953).

of endeavors to approach optimization problems with black-box algorithms, i.e., metaheuristics, one could ask the following question:

> *"'What is exactly the contribution of a given metaheuristic, since NFL theorem proves that all heuristics are globally equivalent'."* (Weinberg and Talbi, 2004)

For many classes of problems—here the interest is in those for which no polynomial run-time algorithm is known—there seems to be strong empirical evidence that considerable improvements on random search can be achieved. However, at the same time, there is also evidence that algorithms can drastically fail. For example, the symmetric traveling-salesman problem (TSP)—find one shortest loop that visits all given n-cities—has $(n-1)!/2$ possible tours. Nevertheless, the state-of-the-art Concorde solver can solve instances with more than 80,000 cities to optimality using branch-and-cut (Applegate et al., 2007). But, Ahammed and Moscato (2011) demonstrate that by only slightly perturbing the cities' locations, the run-time of the Concorde solver can be increased up to 30,000-fold compared to the run-time required for the original instances. The branching rule at the heart of exact branching algorithms can also be considered a metaheuristic. An often used procedure for choosing the next variable to branch on is to select the variable that is desired to be integer and whose fractional part is closest to 0.5. Computational experiments on a library of instances from different real-world mixed-integer problems show that this heuristic does in general not perform better than randomly selecting the next variable (Achterberg, Koch, and Martin, 2005).

Turning away from finding black-box algorithms that serve all purposes, a more promising goal can be pursed, i.e., to focus on the problems of interest and to inject problem-specific information into algorithms.[12] As stated in Dembski and Marks II (2009): "Problem-specific information is almost always embedded in search algorithms. Yet, because this information can be so familiar, we can fail to notice its presence". For example, it is well-known that the optimal solution to the TSP with an euclidean metric does not have any intersecting edges. If a clever way could be devised to only sample from tours without intersecting edges, the chance of finding a good solution would increase. This might be the reason that the two-opt operator, that removes two edges and replaces them with two different edges, works so well for many problems that are structurally similar to the TSP. On the other hand, if the information is wrong, the algorithm can even perform worse than random search: Considering the TSP with time windows, it is easy to construct instances that do not have a feasible solution without intersecting edges.

> *"Finally, we can conclude that if there is no dominance of heuristics, we can still believe in the efficiency of some strategies for a certain class of problems. Indeed the search space is usually defined with a specific (sometimes non-trivial) neighborhood operator."* (Weinberg and Talbi, 2004)

This is the motivation to approach the problems studied in this thesis in a practical-oriented way. This work does not aim to suggest new metaheuristic paradigms. Instead, figuratively speaking, it follows the "practitioners in the earlier days of computing [that] sometimes referred to themselves as 'penalty function artists'" (Dembski and Marks II, 2009). The focus is on the development of problem-specific and interacting components that exploit the structure of the problems investigated. For some problems,

[12] However, when Donald is soon after charged with a heavy fine for "letting a dime do your thinking", he abandons Flipism (Barks, 1953).

observations can be made that help with the design decision. For other problems, a trial-and-error approach can be used to detect rules that work at least for the subset of studied instances, and hopefully for other instances from the same distribution. Further, this thesis aims to develop techniques to increase the speed of the heuristics used, as this certainly helps to sample more solutions. This is not meant to implicate that there is no merit in the development of metaheuristics. As experience teaches, guiding principles are needed to control the problem-specific heuristics.

1.3. Contribution and Organization

In the following, the VRPs studied in this work and the solution methods applied to them are presented. The selection of VRPs is motivated by the transformation of traditional supply chains into networks of actors, challenges in improving customer satisfaction, and the emergence of EVs in logistics. As they exhibit rich constraints, they elude simple methods. Note the use of "we" to indicate that the first three of the four chapters of this thesis originate from collaborating with other researchers in the field of vehicle routing. However, the author was responsible for a large portion of the work related to these projects. Because the problems studied are too divers, no entirely unified notation is used. Self-contained notation is introduced in every chapter, and every chapter can be read on its own.

Chapter 2 is concerned with an important strategic decision when designing the logistics activities of a company, i.e., the extent to which deliveries are performed by vehicles privately owned by the company and to which extent they are subcontracted to second-party logistics providers (2PLs). While there are benefits of using 2PLs—low capital commitment, economies of scope and scale, contracting away of risk, and avoiding complex regulations—providing own transport capacities has the advantage of staying in control of a central business function with a high level of customer contact. However, if the decision is in favor of providing logistics services in-house, procuring sufficient capacity to meet fluctuating demands in every scenario is likely to result in excess capacity on many days and to cause high cost. A possible remedy is to define a sufficient level of service and to outsource customers on high-demand days to 2PLs. The planning problem where one simultaneously needs to determine customers to subcontract and routes respecting capacity for the remaining customers is called the VRP with private fleet and common carrier (VRPPC). An example is depicted in Figure 1.1.a: The private fleet carries out three tours, and four customers are subcontracted. Note that customers might also be subcontracted in case there is sufficient capacity to serve all of them if this beneficial from a cost point of view, i.e., because the customers require long detours or have a low subcontracting cost. In addition, the VRPPC can also be used for scenarios where customers are not subcontracted to 2PLs but one desires to distinguish between low and high priority customers. This can be accomplished by assigning different costs to the customers. Then, customers not served by the private fleet on one day, can be postponed to another day (and their priority can be increased to ensure that they are eventually served). We propose the first exact solution method for the VRPPC by developing a branch-and-price-and-cut algorithm that provides high-quality lower bounds. In addition, we develop a large neighborhood search (LNS) to solve the VRPPC heuristically, that shows a very convincing performance on instances from the literature. This paper was submitted to the special issue Combinatorial Optimization: between Practice and Theory (Discrete Applied Mathematics) as *"Upper and lower bounds for the vehicle-routing problem with private fleet and common carrier"* by Dominik Goeke, Timo Gschwind, and Michael Schneider.

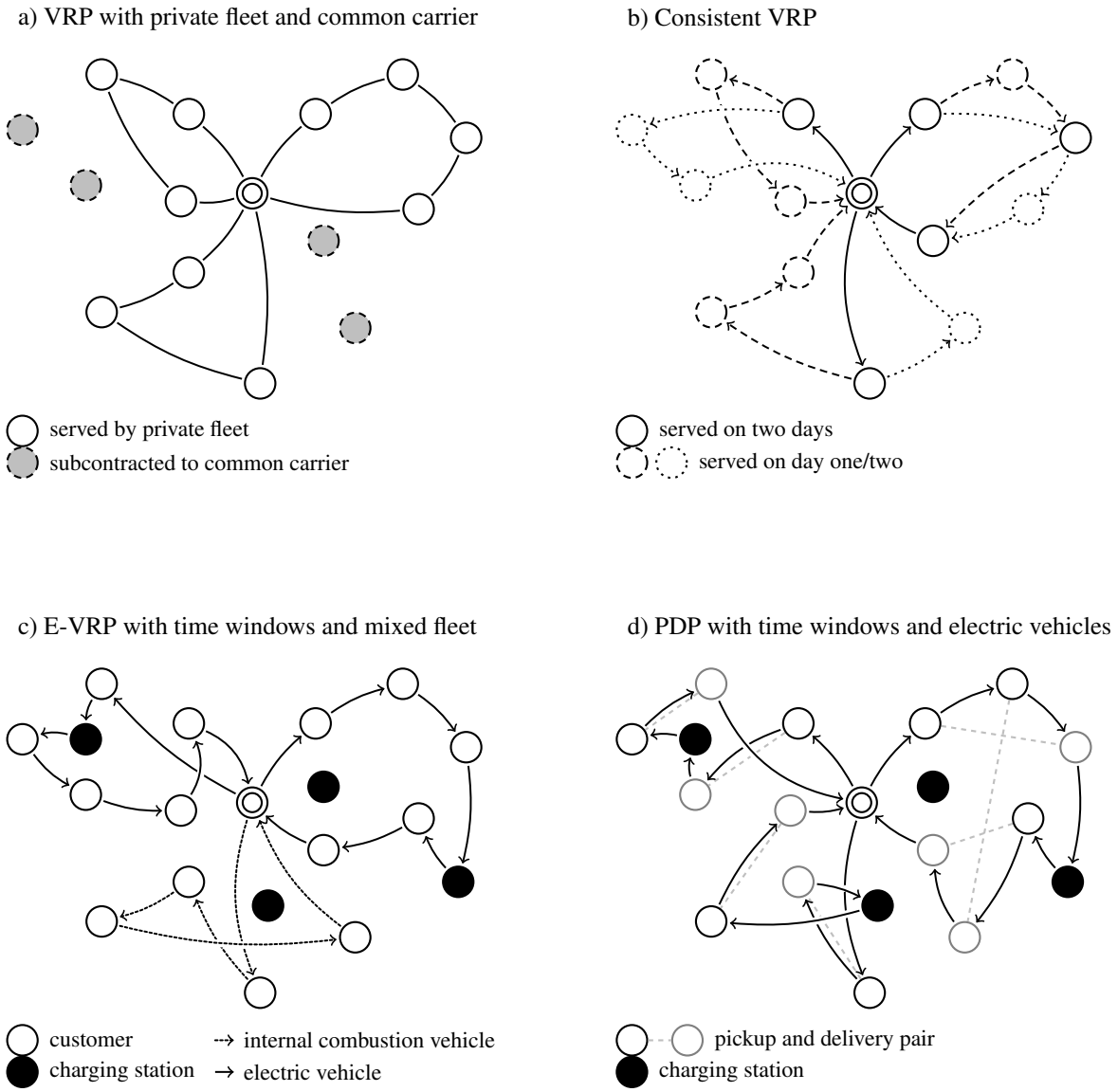

Figure 1.1.: Example instances for the four investigated problem variants.

Chapter 3 is about providing consistent service by satisfying customer demands with the same driver (driver consistency) at approximately the same time (arrival time consistency). This allows companies in last-mile distribution to stand out among competitors. The consistent VRP (ConVRP) is a multi-day problem addressing such consistency requirements along with traditional constraints on vehicle capacity and route duration. A simplified example with two days and three drivers is depicted in Figure 1.1.b. The literature offers several heuristics, but no exact method for this problem. The state-of-the-art exact technique to solve VRPs—CG applied to route-based formulations where columns are generated via dynamic programming (DP)—cannot be successfully extended to the ConVRP because the linear relaxation of route-based formulations is weak. We propose the first exact method for the ConVRP, which can solve medium-sized instances with five days and 30 customers. The method solves, via CG, a formulation where each variable represents the set of routes assigned to a vehicle over the planning horizon. As upper bounding procedure, we develop a LNS featuring a repair procedure specifically designed to improve the arrival time consistency of solutions. Used as stand-alone heuristic, the LNS is able to significantly improve the solution quality on benchmark instances from the literature compared to state-of-the-art heuristics. This chapter was submitted to Transportation Science as *"Exact and Heuristic Solution of the Consistent Vehicle-Routing Problem"* by Dominik Goeke, Roberto Roberti, and Michael Schneider.

Chapter 4 discusses how rising energy costs and new regulations concerning the emission of greenhouse gases render ECVs an alternative to internal combustion commercial vehicles (ICCVs). Several companies are gradually introducing ECVs into their fleet, thus performing their last-mile deliveries with a mixed vehicle fleet. This motivates the development of models and algorithms to plan delivery routes that take the characteristics of ECVs into account. We propose the VRP with mixed fleet (E-VRPTWMF) to optimize the routing of a mixed fleet of ECVs and ICCVs. Contrary to existing routing models for ECVs, which assume energy consumption to be a linear function of traveled distance, we utilize a realistic energy consumption model that incorporates speed, gradient and cargo load distribution. This is highly relevant in the context of ECVs because energy consumption determines the maximal driving range of ECVs and the recharging times at stations. In Figure 1.1.c, we present an example with two ECVs and one ICCV. Each of the ECVs is recharged once en route. Two of the three routes are not in the blossom shape that is typical for many VRPs to clarify that time windows at customers can impose routes that drastically deviate from what is often perceived as reasonable (pleasing to the eye). To address the problem, we develop an adaptive LNS (ALNS) algorithm that is enhanced by a local search (LS) for intensification. In numerical studies on newly designed E-VRPTWMF test instances, we investigate the effect of considering the actual load distribution on the structure and quality of the generated solutions. Moreover, we study the influence of different objective functions on solution attributes and on the contribution of ECVs to the overall routing costs. Finally, we demonstrate the performance of the developed algorithm on benchmark instances of the related VRP with time windows (VRPTW) and electric VRPTW (E-VRPTW). This chapter was published in Goeke and Schneider (2015).

Chapter 5 examines the electric pickup and delivery problem with time windows and electric vehicles (PDPTW-EV) that generalizes the classical VRP. In the literature, problems have been studied that consider the distribution of goods with ECVs from a central depot. However, many real-world applications have a pickup and delivery structure. In these applications, one needs to plan routes in order to satisfy requests; each request requires that a commodity is transported from a pickup to a delivery location. The

PDPTW-EV is about planning routes to satisfy these requests. In addition, we consider that access to locations is restricted by a time window. ECVs, which are constrained by battery capacity and a capacity for commodities, serve the routes. The vehicles start with a fully charged battery at the depot, and they can recharge the battery en route at dedicated recharging stations. Figure 1.1.d provides an example with three ECVs. The dotted connections indicate which pairs must be served together in a route, i.e., pickup before delivery. To address the problem, we develop a granular tabu search (GTS) algorithm that features a policy to determine the amount of energy recharged at visits to stations. Further, we introduce a set of small-sized instances and another set of larger-sized instances for the PDPTW-EV. In numerical studies, we validate our approach on the small-sized instances. To this end, we compare the results obtained with a commercial solver to the results of GTS. In addition, we demonstrate on the second set of instances that GTS can handle the partial recharging aspect by comparing it to mandatory full recharging. Finally, we show that our algorithm is also competitive on benchmark instances of the related pickup and delivery problem with time windows (PDPTW). This chapter was submitted to the European Journal of Operational Research as *"Granular Tabu Search for the Pickup and Delivery Problem with Time Windows and Electric Vehicles"* by Dominik Goeke.

Finally, Chapter 6 concludes the thesis and gives an outlook on future directions of research.

Bibliography

Achterberg, T., T. Koch, and A. Martin (2005). Branching rules revisited. In: *Operations Research Letters* 33 (1), pp. 42–54.

Ahammed, F. and P. Moscato (2011). Evolving L-systems as an intelligent design approach to find classes of difficult-to-solve traveling salesman problem instances. In: *Applications of Evolutionary Computation. EvoApplications 2011*. Ed. by C. Di Chio, S. Cagnoni, C. Cotta, M. Ebner, A. Ekárt, A. I. Esparcia-Alcázar, J. J. Merelo, F. Neri, et al. Vol. 6624. Lecture Notes in Computer Science. Berlin, Heidelberg: Springer, pp. 1–11.

Ajanovic, A. and R. Haas (2016). Dissemination of electric vehicles in urban areas: Major factors for success. In: *Energy* 115 (Part 2), pp. 1451–1458.

Applegate, D. L., R. E. Bixby, V. Chvatal, and W. J. Cook (2007). The Traveling Salesman Problem: A Computational Study (Princeton Series in Applied Mathematics). Princeton, NJ, USA: Princeton University Press.

Barks, C. (1953). Walt Disney's Comics and Stories. Vol. 149. New York, NY, USA: Dell Comics.

Berg, P., S. Isaacs, and K. Blodgett (2016). *Airborne fulfillment center utilizing unmanned aerial vehicles for item delivery*. US Patent 9,305,280. April 5, 2016.

Bernon, M., J. Cullen, and J. Gorst (2016). Online retail returns management: Integration within an omni-channel distribution context. In: *International Journal of Physical Distribution & Logistics Management* 46 (6/7), pp. 584–605.

Bogdanski, R. (2017). *Innovationen auf der letzten Meile - Bewertung der Chancen für die nachhaltige Stadtlogistik von morgen - Nachhaltigkeitsstudie 2017*. Ed. by Bundesverband Paket & Express Logistik. Accessed 7/24/2017. URL: www.biek.de/index.php/studien.html?file=tl_files/biek/Nachhaltigkeitsstudie%202017/BIEK_Nachhaltigkeitsstudie_2017.pdf.

Council of Supply Chain Management Professionals (2013). *Supply chain management terms and glossary*. Accessed July 20, 2017. URL: http://cscmp.org/CSCMP/Educate/SCM_Definitions_and_Glossary_of_Terms/CSCMP/Educate/SCM_Definitions_and_Glossary_of_Terms.aspx.

Davis, B. A. and M. A. Figliozzi (2013). A methodology to evaluate the competitiveness of electric delivery trucks. In: *Transportation Research Part E: Logistics and Transportation Review* 49 (1), pp. 8–23.

Dembski, W. A. and R. J. Marks II (2009). Conservation of information in search: Measuring the cost of success. In: *IEEE Transactions on Systems, Man, and Cybernetics - Part A: Systems and Humans* 39 (5), pp. 1051–1061.

Desrosiers, J. and M. E. Lübbecke (2005). A primer in column generation. In: *Column Generation*. Ed. by G. Desaulniers, J. Desrosiers, and M. M. Solomon. Boston, MA, USA: Springer, pp. 1–32.

European Commission (2014). *Communication from the Commission to the European Parliament, the Council, the European Economic and Social Committee and the Committee of the regions. A policy framework for climate and energy in the period from 2020 to 2030, COM(2014)15, SWD(2014) 16 final*. Accessed July 25, 2017. URL: http://eur-lex.europa.eu/legal-content/EN/ALL/?uri=CELEX:52014DC0015.

– (2016). EU Transport in figures: Statistical pocketbook. Luxembourg: Publications Office of the European Union.

European Environment Agency (2017a). *Data viewer on greenhouse gas emissions and removals, sent by countries to UNFCCC and the EU Greenhouse Gas Monitoring Mechanism (EU Member States)*. Accessed July 25, 2017. URL: http://www.eea.europa.eu/data-and-maps/data/data-viewers/greenhouse-gases-viewer.

– (2017b). *Greenhouse gas emissions from transport*. Accessed July 25, 2017. URL: https://www.eea.europa.eu/data-and-maps/indicators/transport-emissions-of-greenhouse-gases/transport-emissions-of-greenhouse-gases-10.

Feillet, D. (2010). A tutorial on column generation and branch-and-price for vehicle routing problems. In: *4OR - A Quarterly Journal of Operations Research* 8 (4), pp. 407–424.

Gendreau, M. and J.-Y. Potvin (2010). Tabu search. In: *Handbook of Metaheuristics*. Ed. by M. Gendreau and J.-Y. Potvin. Vol. 146. International Series in Operations Research & Management Science. Springer, pp. 41–59.

Glover, F. (1986). Future paths for integer programming and links to artificial intelligence. In: *Computers & Operations Research* 13 (5), pp. 533–549.

Goeke, D. and M. Schneider (2015). Routing a mixed fleet of electric and conventional vehicles. In: *European Journal of Operational Research* 245 (1), pp. 81–99.

Hawkins, T. R., O. M. Gausen, and A. H. Strømman (2012). Environmental impacts of hybrid and electric vehicles: A review. In: *The International Journal of Life Cycle Assessment* 17 (8), pp. 997–1014.

Ho, Y. and D. Pepyne (2002). Simple explanation of the no-free-lunch theorem and its implications. In: *Journal of Optimization Theory and Applications* 115 (3), pp. 549–570.

Jochem, P., C. Doll, and W. Fichtner (2016). External costs of electric vehicles. In: *Transportation Research Part D: Transport and Environment* 42, pp. 60–76.

Juan, A., C. Méndez, J. Faulin, J. de Armas, and S. Grasman (2016). Electric vehicles in logistics and transportation: A survey on emerging environmental, strategic, and operational challenges. In: *Energies* 9 (2), pp. 86–107.

Lanz, M. and R. Tuokko (2017). Concepts, methods and tools for individualized production. In: *Production Engineering* 11 (2), pp. 205–212.

Lebeau, P., C. Macharis, and J. van Mierlo (2016). Exploring the choice of battery electric vehicles in city logistics: A conjoint-based choice analysis. In: *Transportation Research Part E: Logistics and Transportation Review* 91, pp. 245–258.

Lebeau, P., C. Macharis, J. van Mierlo, and K. Lebeau (2015). Electrifying light commercial vehicles for city logistics? A total cost of ownership analysis. In: *European Journal of Transport and Infrastructure Research* 15 (4), pp. 551–569.

Nitta, N., F. Wu, J. T. Lee, and G. Yushin (2015). Li-ion battery materials: Present and future. In: *Materials Today* 18 (5), pp. 252–264.

Nordelöf, A., M. Messagie, A.-M. Tillman, M. Ljunggren Söderman, and J. van Mierlo (2014). Environmental impacts of hybrid, plug-in hybrid, and battery electric vehicles—What can we learn from life cycle assessment? In: *The International Journal of Life Cycle Assessment* 19 (11), pp. 1866–1890.

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

Quak, H., N. Nesterova, and T. van Rooijen (2016). Possibilities and barriers for using electric-powered vehicles in city logistics practice. In: *Tenth International Conference on City Logistics 17-19 June 2015, Tenerife, Spain*. Ed. by E. Taniguchi and G. Russell. Vol. 12. Transportation Research Procedia. Elsevier, pp. 157–169.

Reichel, J. (2016). Lastenrad-Projekt: UPS gewinnt Nachhaltigkeitspreis. In: *Logistra - News*. Accessed 7/24/2017. URL: http://www.logistra.de/news-nachrichten/nfz-fuhrpark-lagerlogistik-intralogistik/7687/maerkte-amp-trends/lastenrad-projekt-ups-gewinnt-nachhaltigkeitsp.

Spiegel, J., M. McKenna, G. Lakshman, and P. Nordstrom (2013). *Method and system for anticipatory package shipping*. US Patent 8,615,473. December 24, 2013.

Stevens, G. C. and M. Johnson (2016). Integrating the supply chain ... 25 years on. In: *International Journal of Physical Distribution & Logistics Management* 46 (1), pp. 19–42.

Tsurukawa, N., S. Prakash, and A. Manhart (2011). *Social impact of artisanal cobalt mining in Katanga, Democratic Republic of Congo*. Tech. rep. Accessed December 13, 2017. Freiburg, Germany: ÖkoInstitut e.V. URL: https://www.oeko.de/oekodoc/1294/2011-419-en.pdf.

van Mierlo, J., M. Messagie, and S. Rangaraju (2017). Comparative environmental assessment of alternative fueled vehicles using a life cycle assessment. In: *World Conference on Transport Research - WCTR 2016 Shanghai. 10-15 July 2016*. Vol. 25. Transportation Research Procedia. Elsevier, pp. 3435–3445.

Wainwright, I. (2015). *TfL Freight and fleet presentation to UKNMB. Presentation on sustainable delivery and servicing - Lessons from London*. Accessed July 25, 2017. URL: http://www.ukroadsliaisongroup.org/download.cfm/docid/31DA53DC-664B-4164-87603AFECDFAE6C7.

Weinberg, B. and E. G. Talbi (2004). NFL theorem is unusable on structured classes of problems. In: *Proceedings of the 2004 Congress on Evolutionary Computation*. Vol. 1. IEEE, pp. 220–226.

Wolpert, D. H. and W. G. Macready (1997). No free lunch theorems for optimization. In: *IEEE Transactions on Evolutionary Computation* 1 (1), pp. 67–82.

Wulfsberg, J. P., T. Redlich, and F.-L. Bruhns (2011). Open production: Scientific foundation for co-creative product realization. In: *Production Engineering* 5 (2), pp. 127–139.

Chapter 2

Outsourcing and Postponement of Deliveries

2.1. Introduction

The vehicle-routing problem with private fleet and common carrier (VRPPC) is a variant of the vehicle-routing problem (VRP) in which customers can be subcontracted at a customer-dependent cost if the privately-owned capacity is insufficient to serve all customers, or if doing so is beneficial from a cost point of view. Consequently, the subcontracted customers do not need to be served on vehicle routes of the privately-owned fleet, but a cost is paid for outsourcing customers to the so-called common carrier.

The VRPPC has direct applications in manufacturing (Tang and Wang, 2006) and less-than-truckload shipping (Chu, 2005; Stenger, Schneider, and Goeke, 2013). It is also closely related to problems arising in collaborative transportation, in which carriers can pass on requests to other carriers and accept or decline requests offered by their partners (Liu et al., 2010), and to the integrated operational transportation planning problem that considers different subcontracting options (Krajewska and Kopfer, 2009). The VRPPC also has applications in the planning of same-day parcel deliveries. By choosing adequate customer-dependent outsourcing costs, important customers—e.g., subscribers of Amazon Prime or customers that have already been postponed on previous days—can be favored over regular new requests.

Despite its practical relevance, relatively few papers have focused on solution methods for the VRPPC. The following heuristic paradigms have been proposed in the literature: simple construction and improvement heuristics (Chu, 2005; Bolduc, Renaud, and Boctor, 2007), randomized construction–improvement–perturbation (RIP, Bolduc, Renaud, Boctor, and Laporte, 2008), tabu search (TS, Côté and Potvin, 2009; Potvin and Naud, 2011), variable neighborhood search (VNS, Stenger, Schneider, and Goeke, 2013; Stenger, Vigo, et al., 2013), multi-start local search (MS-LS, Vidal, Maculan, et al., 2016), iterated local search (MS-ILS, Vidal, Maculan, et al., 2016), and a memetic algorithm (MA, Vidal, Maculan, et al., 2016). All of the listed methods have been investigated on instances assuming a homogeneous vehicle fleet, and the MA of Vidal, Maculan, et al. (2016) shows the best performance with regards to solution quality. On instances with a heterogeneous fleet composition, only RIP and the two TS algorithms have been tested.

It is notable that early approaches (Chu, 2005; Bolduc, Renaud, and Boctor, 2007) only take the decision which customers to subcontract into account when constructing the initial solution. Later approaches consider the subcontracting decision when generating the neighborhood of a solution (Bolduc, Renaud, Boctor, and Laporte, 2008; Côté and Potvin, 2009; Potvin and Naud, 2011; Stenger, Vigo, et al., 2013;

Stenger, Schneider, and Goeke, 2013). The most recent approach (Vidal, Maculan, et al., 2016) uses an implicit customer selection, i.e., for every move a resource-constrained shortest path problem is solved to evaluate which customers should be subcontracted.

The contribution of this chapter is twofold and concerns the heuristic and exact domain:

- We develop a large neighborhood search (LNS) to solve the VRPPC heuristically. Our LNS features a new decomposition procedure, and we demonstrate the effectiveness of this component by comparing it to an LNS without this component on the VRPPC benchmark instances from the literature. We are able to provide several new best-known solutions on the larger instances from the literature, and we demonstrate that our heuristic is among the best solution methods published for the VRPPC.
- To the best of our knowledge, we are the first to propose an exact solution method for the VRPPC. Our method uses a path-based formulation that is solved by means of a branch-price-and-cut algorithm (BPC). The BPC is able to provide optimal solutions for some small to medium-sized instances. For larger instances, it provides tight lower bounds that can be used to assess the quality of the heuristic solutions.

This chapter is organized as follows: Section 2.2 formally describes the problem. Our two solution methods, BPC and LNS, are explained in Sections 2.3 and 2.4. Section 2.5 details the parameter setting, the test instances and presents the numerical results. Section 2.6 concludes the chapter.

2.2. The Vehicle-Routing Problem with Private Fleet and Common Carrier

To define the VRPPC as a graph-theoretical problem, let $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ be a complete undirected graph with vertices $\mathcal{V} = \{v_0\} \cup \mathcal{N}$ and edges $\mathcal{E} = \mathcal{V} \times \mathcal{V}$. Vertex v_0 denotes the depot, the other vertices represent customers $i \in \mathcal{N}$. Each customer $i \in \mathcal{N}$ is assigned a demand q_i and a cost h_i for subcontracting the customer. Each edge $\{i, j\} \in \mathcal{E}$ is assigned a travel cost c_{ij}. At the depot, a set of vehicles $\mathcal{K}$, which represent the private fleet, is based. The vehicles $k \in \mathcal{K}$ can differ with regard to capacity C_k and fixed cost F_k. The fixed cost F_k is only incurred if a route is assigned to vehicle k. Typically, not all vehicles are different, and we can group the vehicles according to their attributes such that vehicles with identical attributes are in the same group $l \in \mathcal{L}$. We denote the number of vehicles in group l as z_l. The VRPPC now calls for: (i) satisfying the demand of every customer with exactly one visit, either using the common carrier or a vehicle of the private fleet, and (ii) planning at most one route for each vehicle of the private fleet so that every route starts and ends at the depot, and the vehicle capacity is respected. The goal is to minimize the total cost consisting of the sum of fixed cost, the cost of routing the vehicles of the private fleet, and the cost of subcontracting customers.

2.3. Branch-Price-and-Cut Algorithm for the VRPPC

In this section, we give details on our exact approach to the VRPPC that is used to obtain optimal solutions for small to medium-sized instances. For larger instances, we provide lower bounds, which can be used to assess the quality of the heuristic solutions. The approach is based on a path-based formulation that is solved by means of a BPC algorithm.

2.3.1. Path-Based Formulation

Master program Let Ω_l be the set of all feasible routes for vehicle group $l \in \mathcal{L}$. We denote by c_r the routing cost of route $r \in \Omega_l$. Binary decision variables λ_r indicate if the route $r \in \Omega_l$ is selected ($\lambda_r = 1$) or not ($\lambda_r = 0$) while binary decision variables y_i indicate if customer $i \in \mathcal{N}$ is served by the common carrier ($y_i = 1$) or by the private fleet ($y_i = 0$). Finally, let a_{ri} be the number of times route r visits customer i. The VRPPC can then be defined as:

$$\min \quad \sum_{l \in \mathcal{L}} \sum_{r \in \Omega_l} (F_l + c_r)\lambda_r + \sum_{i \in \mathcal{N}} h_i y_i \tag{2.1a}$$

$$\text{s.t.} \quad \sum_{l \in \mathcal{L}} \sum_{r \in \Omega_l} a_{ri}\lambda_r + y_i = 1 \qquad \forall i \in \mathcal{N} \tag{2.1b}$$

$$\sum_{r \in \Omega_l} \lambda_r \leq z_l \qquad \forall l \in \mathcal{L} \tag{2.1c}$$

$$\lambda_r \in \{0, 1\} \qquad \forall l \in \mathcal{L}, r \in \Omega_l \tag{2.1d}$$

$$y_i \in \{0, 1\} \qquad \forall i \in \mathcal{N} \tag{2.1e}$$

The objective function (2.1a) minimizes the total cost comprising vehicle fixed costs, routing costs, and subcontracting costs. Partitioning constraints (2.1b) ensure that each customer is served exactly once either by the private fleet or by the common carrier. Note that constraints (2.1b) can be replaced by their covering counterpart if routing costs satisfy the triangle inequality. Convexity constraints (2.1c) limit the number of vehicles of each type. The variable domains are specified in (2.1d) and (2.1e).

Because of the huge number of feasible routes, model (2.1) cannot be solved directly and one has to resort to column-generation (or Lagrangean-relaxation) based methods: The linear relaxation of model (2.1) is initialized with a proper subset of routes and missing routes (=columns) with negative reduced costs are dynamically identified by calling the pricing subproblems and added to the master program (2.1). Integrality is finally ensured by integrating the column-generation process into a branch-and-bound algorithm (Lübbecke and Desrosiers, 2005).

Pricing subproblem The task of the pricing subproblems is to find feasible routes with negative reduced costs or to prove that no such routes exist. Similar to many other VRP variants, the pricing subproblems of the VRPPC are elementary shortest-path problems with resource constraints (ESPPRCs) on graphs with negative-cost cycles (Irnich and Desaulniers, 2005). In the VRPPC, there are $|\mathcal{L}|$ different pricing problems, one for each vehicle type $l \in \mathcal{L}$. Given a vehicle type $l \in \mathcal{L}$, a feasible VRPPC route starts and ends at the depot v_0, visits some customers $i \in \mathcal{N}$ in between, and respects the vehicle capacity C_l.

Let π_i and μ_l be the dual prices of constraints (2.1b) and (2.1c), respectively. The reduced cost of a route $r \in \Omega_l$ is given by:

$$\tilde{c}_r = c_r - \sum_{i \in \mathcal{N}} \pi_i - \mu_l = \sum_{(i,j) \in \mathcal{E}(r)} \tilde{c}^l_{ij}, \tag{2.2}$$

where $\mathcal{E}(r)$ denotes the sequence of edges traversed by route r and $\tilde{c}^l_{ij} = c_{ij} - 1/2\tilde{\pi}_i - 1/2\tilde{\pi}_j$ with

$\tilde{\pi}_{v_0} = \mu_l$ and $\tilde{\pi}_i = \pi_i$ for all customers $i \in \mathcal{N}$. The pricing subproblem for vehicle type $l \in \mathcal{L}$ can then be formalized as:

$$\min_{r \in \Omega_l} \{\tilde{c}_r\}. \tag{2.3}$$

It is well-known that the ESPPRCs (2.3) are strongly $\mathcal{NP}$ -hard. To obtain better-solvable pricing subproblems, the elementarity condition of routes can be relaxed so that routes containing cycles can be priced out. This comes at the cost of weaker lower bounds of formulation (2.1). A good trade-off between the hardness of the pricing subproblems and the strength of the lower bounds is often achieved by the so-called ng-routes (Baldacci, Mingozzi, and Roberti, 2011) that forbid certain types of cycles. Here, each customer $i \in \mathcal{N}$ is assigned a neighborhood $N_i \subset \mathcal{N}$ with $i \in N_i$. Typically, the cardinalities $|N_i|$ and the neighborhoods N_i themselves are fixed a priori for all customers $i \in \mathcal{N}$. An ng-route now allows multiple visits to a customer i provided that another customer j with $i \notin N_j$ is visited in between, i.e., an ng-route forgets previous visits to those customers that are not in the neighborhoods of the subsequently visited customers. In the following, we redefine the set Ω_l as the set of all ng-feasible routes. Clearly, the cycles that are allowed and, thus, the quality of the lower bounds provided by the ng-route relaxation of (2.1) critically depends on the choices of N_i. Elementarity of all routes of a solution is finally ensured by branching.

2.3.2. Cutting Planes

Path-based models like the extended set-partitioning formulation (2.1) generally provide much stronger bounds compared to edge-based formulations. Still, even with large $|N_i|$ or pure elementary sets Ω_l, the lower bounds provided by (2.1) are not sufficiently tight for the effective solution of even small to medium-sized instances of the VRPPC. To further strengthen the formulation, we add the following additional families of valid inequalities.

Robust cuts The first type of cuts describes inequalities on the aggregated flow on edges $\{i, j\} \in \mathcal{E}$. Such inequalities can be incorporated into the master problem using expressions $x(\delta(S)) \leq rhs$ or $x(\delta(S)) \geq rhs$, where $\delta(S) = \{\{i, j\} \in \mathcal{E} : i \in S, j \in \mathcal{V} \setminus S\}$ denotes the cut-set of $S \subset \mathcal{V}$ and $x(\delta(S)) = \sum_{l \in \mathcal{L}} \sum_{r \in \Omega_l} \sum_{\{i,j\} \in \delta(S)} b_{ijr} \lambda_r$ denotes the cut-set flow. Parameter b_{ijr} gives the number of times edge $\{i, j\}$ is traversed by route r. The dual prices of these inequalities directly transfer to the reduced cost $\tilde{c}^l_{ij}, l \in \mathcal{L}$ of the included edges meaning that they do not change the structure and the complexity of the pricing subproblems, i.e., they are *robust cuts*. In our BPC approach, we use rounded capacity cuts, which are separated with the CVRPSEP package (Lysgaard, 2003).

Non-robust cuts To further strengthen the linear relaxation, we also incorporate *non-robust* cuts into model (2.1). The addition of non-robust cuts has to be done carefully because each cut makes the pricing subproblem harder. Subset-row inequalities (SR) originally introduced by Jepsen et al. (2008) are a family of non-robust cuts that have been successfully used in many exact approaches to VRP variants. Each SR is defined on a subset of customers. As proposed by Jepsen et al. (2008), we restrict ourselves to SR defined on three customers because they can be separated by straightforward enumeration. Let

$U \subset \mathcal{N}$ be a set of three customers. The corresponding inequality is defined as $\sum_{l \in \mathcal{L}} \sum_{r \in \Omega_l} \left\lfloor \frac{g_r}{2} \right\rfloor \lambda_r \leq 1$, where g_r is the number of times route r visits customers in U. Denote by $\sigma \leq 0$ its associated dual price. For every second visit to a customer in U, σ has to be subtracted from the reduced cost of a route. This complicates the solution of the pricing subproblem (see Section 2.3.3).

Recently, Pecin, Contardo, et al. (2017) proposed the use of limited memory SR (lmSR) which are a generalization of SR and whose impact on the solution of the pricing subproblem is typically reduced compared to standard SR. With each lmSR are associated a set of (three) customers U and a memory set M of nodes with $U \subseteq M \subseteq \mathcal{N}$. The basic idea of lmSR is similar to the ng-routes. Roughly speaking, each time a route r visits a node $j \notin M$ not in the memory of the cut, the coefficient g_r 'forgets' a previous visit to one of the customers $i \in U$ if the 'remembered' number of visits up to node j is odd. In the following section, we clarify why this simplifies the solution of the pricing subproblem if $|M| < |\mathcal{N}|$, and we give details on the computation of the coefficient g_r. For a detailed description of lmSR, the computation of the coefficient g_r, and the determination of the smallest-possible memory sets M we refer to Pecin, Contardo, et al. (2017).

Dynamic neighborhood extension As mentioned in Section 2.3.1, the quality of the lower bounds depends on the choices of the neighborhoods N_i of the ng-route relaxation. However, it is not clear a priori what good choices for N_i are. Roberti and Mingozzi (2014) proposed the dynamic extension of these neighborhoods, which can be interpreted as adding valid inequalities to formulation (2.1) forbidding routes with certain cycles. Let r be a route of the current linear programming (LP) solution that contains a cycle $C = (i, \ldots, i)$ with $i \in \mathcal{N}$. Then, we add customer i to the neighborhoods N_j of all nodes $j \in C$, forbidding this cycle in all routes that are priced out. In addition, all routes that are not feasible with respect to the new neighborhoods are removed from the master program (2.1).

2.3.3. Labeling Algorithm

The predominant technique to solve ESPPRCs are dynamic-programming labeling algorithms (Irnich and Desaulniers, 2005). In labeling algorithms, partial paths are gradually extended in a network looking for a minimum-cost path from a given source node to a given sink node. The partial paths are represented by labels storing the accumulated cost and resource consumption along the partial path. To avoid a complete enumeration of all feasible paths, dominance relations between different labels as well as other fathoming rules are typically exploited to eliminate unpromising labels. For a more comprehensive discussion on ESPPRCs and labeling algorithms, we refer to Irnich and Desaulniers (2005).

Forward labeling In the VRPPC, source and sink node of the pricing network are both given by the depot v_0. A partial path $P = (v_0, \ldots, i)$ from the depot v_0 to a vertex $i \in \mathcal{V}$ is represented by a label $L(P) = (\tilde{c}(P), v(P) = i, q(P), \Pi(P), S(P))$ storing its reduced cost $\tilde{c}(P)$, its last vertex $v(P)$, the load $q(P)$ of the vehicle, the set $\Pi(P)$ of visited customer nodes (in the ng-sense), and a binary vector $S(P)$ representing the states of the lmSR. Let Θ be the set of all lmSR with strictly negative dual price in the current pricing iteration. We denote by $S_s(P)$, U_s, σ_s, and M_s the state, customer set, dual price, and memory associated with an lmSR $s \in \Theta$. The initial label at the depot v_0 is given by $(0, v_0, 0, \emptyset, \mathbf{0})$. The extension of a label $L(P)$ to a node $j \in \mathcal{V}$ along edge $\{v(P), j\} \in \mathcal{E}$ is feasible if $q(P) + q_j \leq C_l$ and

$j \notin \Pi(P)$. If the extension is feasible, a new label $L(P') = (\tilde{c}(P'), j, q(P'), \Pi(P'), S(P'))$ is created according to the following resource extension functions (REFs):

$$\tilde{c}(P') = \tilde{c}(P) + \tilde{c}^l_{v(P)j} - \sum_{s \in \Theta : j \in U_s \wedge S_s(P) = 1} \sigma_s \tag{2.4}$$

$$v(P') = j \tag{2.5}$$

$$q(P') = q(P) + q_j \tag{2.6}$$

$$\Pi(P') = (\Pi(P) \cup \{j\}) \cap N_j \tag{2.7}$$

$$S_s(P') = \begin{cases} 0 & \text{if } j \notin M_s \vee (j \in U_s \wedge S_s(P) = 1) \\ 1 & j \in U_s \wedge S_s(P) = 0 \\ S_s(P) & otherwise \end{cases} \qquad \forall s \in \Theta \tag{2.8}$$

To eliminate unpromising labels that cannot lead to an improved complete path compared to another label, the following dominance rule is used. A label $L(P_1)$ dominates another label $L(P_2)$ with the same last vertex i if:

$$\tilde{c}(P_1) - \sum_{s \in \Theta : S_s(P_1) > S_s(P_2)} \sigma_s \leq \tilde{c}(P_2), \tag{2.9}$$

$$q(P_1) \leq q(P_2), \tag{2.10}$$

$$\Pi(P_1) \subseteq \Pi(P_2). \tag{2.11}$$

REFs (2.4) and (2.8) together with the dominance relation (2.9) give the intuition of how the lmSR are handled in the labeling algorithm and why their impact on the solution of the pricing subproblems is reduced compared to the standard SR. The overall handling of the lmSR is analog to the SR. For every second visit to a customer $i \in U_s$ (there can be several visits to the same customer in an ng-route), the dual price σ_s has to be incorporated in the reduced cost. Thus, a binary representation of the state is sufficient for each cut $s \in \Theta$, and the state $S_s(P)$ changes whenever a node $i \in U_s$ is visited. Additionally, the state of cut s is reset in the lmSR case whenever a customer is visited that is not in the memory set M_s. In the dominance rule, two labels $L(P_1)$ and $L(P_2)$ with different states $S_s(P_1)$ and $S_s(P_2)$ for cut s can still be compared by penalizing the dominating label $L(P_1)$ if it is inferior with respect to the state of s (Jepsen et al., 2008). When using lmSR instead of SR, many more labels are directly comparable without penalization because the states for all lmSR $s \in \Theta$ for which $v(P) \notin M_s$ are reset.

In the remainder of this section, we describe several techniques that are used to speed-up the pricing process, namely bidirectional labeling, completion bounds, edge elimination, and heuristic pricing.

Bidirectional labeling In labeling algorithms, the number of generated labels typically increases strongly with the length of the generated partial paths. Bounded bidirectional labeling, originally introduced by Righini and Salani (2006) and successfully used in many state-of-the-art approaches to VRPs variants, can help mitigate this effect and is therefore typically superior to its monodirectional counterpart. In bidirectional labeling, forward and backward partial paths are extended only up to a so-called halfway point

(HWP) defined on one of the resources (that needs to be monotone). After the labeling process, suitable forward and backward partial paths have to be merged to complete feasible paths. Recently, Pecin, Contardo, et al. (2017) and Tilk et al. (2017) proposed the use of a HWP that is dynamically detected during the bidirectional labeling process based on the expected remaining forward and backward work. The idea is to reduce the overall workload by better balancing the necessary forward and backward labeling because they might be unequally complex due to asymmetry in the instance data or in the labeling itself. The VRPPC pricing subproblem instances of this section that use the reduced cost $\tilde{c}^l_{ij}, l \in \mathcal{L}$ from (2.2), however, are completely symmetric so that forward and backward labeling are essentially identical. As a consequence, it is sufficient to perform only the forward labeling up to the HWP. The resulting labels can then be interpreted as both forward and backward partial paths.

The bidirectional labeling algorithm for the VRPPC pricing subproblems works as follows. The HWP is defined on the load resource $q(P)$ of the labels. Forward labeling is then executed, extending a label $L(P)$ only if $q(P) \leq C_l/2$ holds. When the labeling process terminates, suitable labels $L(P)$ and $L(P')$ are merged. To avoid creating the same path from different combinations of labels, the 'first' label $L(P)$ is a candidate for merging only if $q(P) > C_l/2$ or $v(P) = v_0$. The two labels $L(P)$ and $L(P')$ can be merged to a complete feasible route if they end at the same vertex ($v(P) = v(P')$), the capacity of the vehicle is not exceeded ($q(P) + q(P') - q_{v(P)} \leq C_l$), and the sequence of customer visits is feasible in the ng-route sense ($|\Pi(P) \cap \Pi(P')| = 1$). The reduced cost of the resulting route are:

$$\tilde{c}(P) + \tilde{c}(P') - \sum_{s \in \Theta : v(P) \notin U_s \wedge S_s(P) + S_s(P') = 2} \sigma_s + \sum_{s \in \Theta : v(P) \in U_s \wedge S_s(P) + S_s(P') = 0} \sigma_s.$$

Completion bounds The dominance between labels allows the elimination of unpromising labels if there exists a label that is provably superior. Another strategy to discard unpromising labels is the use of completion bounds. The basic idea is to compute lower bounds for the cost of completing a label $L(P)$ to a feasible route. Clearly, if the resulting estimated reduced cost of a complete route is not negative, the corresponding label can be discarded because the real reduced cost of the route cannot be negative. Completion bounds are typically obtained by running the labeling algorithm on a relaxation of the pricing subproblem in the opposite direction, i.e., solving the relaxed problem with backward labeling to obtain completion bounds for the original problem solved with a forward labeling algorithm. As pointed out before, labeling in the VRPPC pricing subproblem is completely symmetric so that completion bounds for the forward labeling of the bidirectional labeling algorithm can be obtained by using the forward labeling on a relaxed version of the problem. In our approach, we relax the pricing subproblem in two different ways. First, we use smaller ng-neighborhoods N_i. Second, we consider only a fraction of the lmSR cuts (those with smallest dual price) explicitly in the labeling. As proposed by Contardo and Martinelli (2014), the effect of the remaining lmSR is partly incorporated into the completion bounds by subtracting $\sigma_s/2$ from the reduced cost $\tilde{c}^l_{ij}, l \in \mathcal{L}$ of the edges $\{i, j\}$ with $i, j \in U_s$.

Let $\hat{c}(i, q)$ be the minimum reduced cost of a label at vertex i with load q that results from the forward labeling algorithm solving the described relaxation of the pricing subproblem for vehicle type $l \in \mathcal{L}$. Then, in the bidirectional labeling algorithm for the original pricing subproblem of l, all labels $L(P)$ at node $v(P)$ for which $\tilde{c}(P) + \min_{q \leq C_l - q(P)} \hat{c}(i, q) \geq 0$ holds are discarded.

Edge elimination A final acceleration technique that we use in our algorithm is the elimination of edges that cannot be part of any optimal solution as proposed by Irnich, Desaulniers, et al. (2010). Define the edge reduced cost $\hat{c}_{ij}^l$ for edge $\{i,j\} \in \mathcal{E}$ and vehicle type $l \in \mathcal{L}$ as the minimal reduced cost of any route $r \in \Omega_l$ regarding the current dual solution that uses edge $\{i,j\}$. If for some $\{i,j\} \in \mathcal{E}, l \in \mathcal{L}$ the edge reduced cost $\hat{c}_{ij}^l$ is larger than the current integrality gap, then edge $\{i,j\}$ can be removed from the pricing subproblem for vehicle type $l \in \mathcal{L}$. Irnich, Desaulniers, et al. (2010) have shown that the values $\hat{c}_{ij}^l$ for all edges $\{i,j\} \in \mathcal{E}$ can be computed by concatenating forward and backward labels $L(P)$ with $v(P) = i$ and $L(P')$ with $v(P') = j$ resulting from a call to the full forward and backward labeling algorithm for $l \in \mathcal{L}$. Again, due to symmetry reasons, labels from the forward labeling can be interpreted also as backward labels in the VRPPC so that a single call to the forward algorithm is sufficient to compute the $\hat{c}_{ij}^l$ for each $l \in \mathcal{L}$.

Heuristic pricers For the column-generation process, it is not necessary to identify a route with minimal reduced cost in every iteration. Instead, it is sufficient to provide any route with negative reduced cost. Consequently, pricing heuristics can be used to solve the pricing subproblems as long as they find such routes. The exact solution algorithm for the pricing subproblems only has to be invoked if the heuristic pricers fail to identify additional routes. In our BPC approach, we use limited discrepancy search (LDS, Feillet, Gendreau, and Rousseau, 2007) to solve the pricing subproblems heuristically. The basic idea of LDS is to divide the set of edges into good and bad edges and to limit the number of bad edges that are allowed in the routes by discarding labels that exceed the allowed number. We define for each node the five best edges (w.r.t. reduced cost) as good edges, all other edges are bad edges. Furthermore, we consider two different values (zero and one) for the number of allowed bad edges, giving rise to two different heuristic pricers with differing computational effort. The pricing solvers are then executed in the following order: LDS with no bad edges, LDS with one bad edge, the exact labeling algorithm. Within the three pricing algorithms, the pricing subproblems for the different vehicle types $l \in \mathcal{L}$ are solved in the order of increasing capacity C_l. Whenever one or more routes with negative reduced cost are found by a combination of pricing solvers and pricing subproblem, they are returned to the master program, and the remaining solver-subproblem combinations are not invoked in this pricing iteration.

2.3.4. Branching

Denote by $\bar{\lambda}_r$ and $\bar{y}_i$ the values of variables λ_r and y_i in a solution of the LP relaxation of model (2.1). Furthermore, let $\bar{x}_{ij} = \sum_{l \in \mathcal{L}} \sum_{r \in \Omega_l} b_{ijr} \bar{\lambda}_r$ be the aggregated flow over edge $\{i,j\} \in \mathcal{E}$ in this solution. Recall that b_{ijr} gives the number of times edge $\{i,j\}$ is traversed by route r.

We use the following hierarchical branching scheme. First, we branch on the overall number of customers served by the common carrier $\sum_{i \in \mathcal{N}} \bar{y}_i$. Second, we branch on the number of vehicles of type $l \in \mathcal{L}$ given by $\sum_{r \in \Omega_l} \bar{\lambda}_r$. If this is fractional for several vehicle types, we branch on the one that is closest to 0.5. Third, we branch on single y_i variables and the customer with $\bar{y}_i$ closest to 0.65 is chosen first. Finally, we branch on the edges $\mathcal{E}$ of the undirected graph $\mathcal{G}$ giving priority to the edge $\{i,j\}$ for which $\bar{x}_{ij}$ is closest to 0.5. All branching decision can be implemented by adding a single constraint to model (2.1). Moreover, the pricing subproblems remain structurally unchanged, and all branching decision preserve the inherent symmetry of the VRPPC. The node selection strategy is best first.

2.4. Large Neighborhood Search for the VRPPC

This section describes our LNS for solving VRPPC. LNS was introduced by Shaw (1998) as a local search method with larger moves that make distant solutions accessible. The large moves are realized by means of a removal and an insertion step: a possibly large part of the solution, i.e., in the context of VRPs a subset of customers, is removed and then reintegrated into the partial solution. A similar approach was proposed by Schrimpf et al. (2000) as ruin and recreate. Ropke and Pisinger (2006b) introduced adaptive LNS, which allows to use a range of different operators that are selected with a probability depending on their past success. Our LNS is based on the latter approach, but we select each operator with the same fixed probability instead of adapting probabilities during the search. Figure 2.1 shows our solution method in pseudocode.

1: $\mathcal{S}_c \leftarrow$ generateInitialSolution()
2: **for** η iterations **do**
3: $\quad\mathcal{S}_t \leftarrow \mathcal{S}_c$
4: $\quad$**for** two rounds **do**
5: $\quad\quad$**for all** $r \in \mathcal{S}_t$ **do**
$\quad\quad$ {Decompose solution into subproblem $\mathcal{P}_r$ for selected route r.}
$\quad\quad$ {Generate solution of subproblem from the tentative solution of the original problem.}
6: $\quad\quad\mathcal{S}_{\mathcal{P}_r} \leftarrow$ copyPartialSolution($\mathcal{P}_r, \mathcal{S}_t$)
$\quad\quad$ {Randomly select removal operator. Draw δ customers to remove from $\mathcal{S}_{\mathcal{P}_r}$.}
7: $\quad\quad\mathcal{S}_{\mathcal{P}_r} \leftarrow$ applyRemoval($\mathcal{S}_{\mathcal{P}_r}, \delta$)
$\quad\quad$ {Randomly select insertion operator to reinsert customers.}
8: $\quad\quad\mathcal{S}_{\mathcal{P}_r} \leftarrow$ applyInsertion($\mathcal{S}_{\mathcal{P}_r}$)
9: $\quad\quad$**if** acceptSA($\mathcal{S}_{\mathcal{P}_r}, \mathcal{S}_t$) **then**
10: $\quad\quad\quad\mathcal{S}_t \leftarrow$ integrate($\mathcal{S}_{\mathcal{P}_r}, \mathcal{S}_t$)
11: $\quad\quad$**end if**
12: $\quad$**end for**
13: $\quad$**end for**
14: $\quad$**if** $\mathcal{S}_t$ is feasible and with a probability of 0.25 **then**
15: $\quad\quad\mathcal{S}_t \leftarrow$ VND($\mathcal{S}_t$)
16: $\quad$**end if**
17: $\quad$updatePenaltyFactor($\mathcal{S}_t$)
18: $\quad\Omega^{LNS} \leftarrow$ addFeasibleRoutes($\mathcal{S}_t$)
19: $\quad$**if** acceptSA($\mathcal{S}_t, \mathcal{S}_c$) **then**
20: $\quad\quad\mathcal{S}_c \leftarrow \mathcal{S}_t$
21: $\quad$**end if**
22: **end for**
23: $\mathcal{S}_{best} \leftarrow$ postProcessing(Ω^{LNS})

Figure 2.1.: Overview of the LNS algorithm.

First, we generate an initial feasible solution $\mathcal{S}_c$ with a modified savings algorithm (Section 2.4.1). In the following improvement phase, we allow infeasible solutions and penalize violations in the objective function (Section 2.4.2). The improvement phase (Section 2.4.3) works as follows: In every iteration, we decompose the original problem into a sequence of subproblems $\mathcal{P}_r$. Then, we derive a first solution $S_{\mathcal{P}_r}$ to subproblem $\mathcal{P}_r$ from the tentative solution $\mathcal{S}_t$ and apply a randomly selected removal and insertion operator on the solution of the subproblem. Afterwards, the new solution is reintegrated into the current solution of the original problem based on a simulated annealing (SA) acceptance criterion. After all

subproblems are processed, we repeat the procedure for another round. If the resulting complete solution is feasible, we apply a variable neighborhood descent (VND) with a probability of 0.25. The final acceptance decision of the iteration is again based on SA. Finally, we save every feasible route found during the search. In a post-processing step, these routes are recombined into the best possible feasible solution (Section 2.4.4).

2.4.1. Generation of Initial Solution

We use a modified savings algorithm (see Clarke and Wright, 1964) to create an initial feasible solution for the VRPPC. The idea is to first serve each customer with a dedicated route, and then to merge pairs of routes as long as a positive saving can be realized and vehicle capacity is not violated. In each iteration, we merge the pair with the highest saving. To merge two routes r_1 and r_2, we consider only the edges incident to the depot and remove one edge of r_1 and one edge of r_2. Then, we replace them by an edge directly linking the corresponding customer i of r_1 and j of r_2.

In the basic version of the algorithm, a tendency to favor peripheral routes can be observed, i.e., to prefer routes that serve customers that are far from the depot and to eventually isolate customers that are located close to the depot (Gaskell, 1967). We calculate the saving $s(i, j)$ when linking customers i and j as $s(i, j) = c_{0i} + c_{j0} - \lambda \cdot c_{ij}$, where λ is a weight to balance between the distance to the depot and the distance between customers (see, e.g., Gaskell, 1967; Yellow, 1970). We randomly select $\lambda \in [0.6, 1.6]$ using the values proposed in Li, Golden, and Wasil (2005) to define the interval. To ensure that at most $|\mathcal{K}|$ vehicles are used, we implement the following simple procedure. In the beginning, none of the initial single-customer routes are assigned to vehicles. We only merge two single-customer routes if an unused vehicle is available, in this case we assign a vehicle to the resulting two-customer route. Otherwise, we refrain from merging the two routes. When two multiple-customer routes are merged, we release one vehicle. Finally, all the customers of the remaining single-customer routes are assigned to the common carrier.

2.4.2. Generalized Cost Function and Penalty Calculation

During the search, we allow solutions that violate the vehicle capacity constraint and add a penalty to the objective function value to account for the respective violation. Although restoring feasibility of a solution is always possible by assigning certain customers to the common carrier, we allow temporary violations to be able to traverse the solution space more freely. The objective value of a solution $\mathcal{S}$ is given by the generalized cost function $f_{gen}(\mathcal{S})$ (see, e.g., Gendreau, Hertz, and Laporte, 1994):

$$f_{gen}(\mathcal{S}) = f(\mathcal{S}) + \gamma \cdot L(\mathcal{S}) = f_{var}(\mathcal{S}) + f_{fix}(\mathcal{S}) + \gamma \cdot L(\mathcal{S}).$$

The term $f(S)$ comprises two parts: $f_{var}(\mathcal{S})$ denotes the cost for the distance traveled by the private fleet and the cost of outsourcing customers to the common carrier; $f_{fix}(\mathcal{S})$ denotes the cost of using the vehicles of the private fleet. The penalty for capacity violations is calculated as the product of a penalty factor γ and the total capacity violation $L(\mathcal{S})$ of solution S. For all operators that our algorithm uses to modify a solution, $L(\mathcal{S})$ can be calculated in $\mathcal{O}(1)$ time.

We initially set the penalty factor to $\gamma = \bar{h}_{max}$, where $\bar{h}_{max}$ denotes the maximum cost per demand unit to subcontract any customer, i.e., $\bar{h}_{max} = \max_{i \in \mathcal{N}} (h_i / q_i)$. Then, we update γ as follows: If the capacity constraint has been violated for two iterations, the penalty factor is multiplied by ϱ, and vice versa it is divided by ϱ if the capacity constraint is satisfied for two iterations. We restrict the value of the penalty factor to the interval $\gamma \in [0.001, 10.0 \cdot \bar{h}_{max}]$.

2.4.3. Solution Improvement

In every iteration of the improvement phase, we decompose the original problem into a series of subproblems $\mathcal{P}_r$, one for each route $r \in \mathcal{S}_t$, and solve these subproblems one at a time. However, instead of solving each subproblem from scratch, we derive a solution $\mathcal{S}_{\mathcal{P}_r}$ of $\mathcal{P}_r$ from $\mathcal{S}_t$ and improve this solution (Section 2.4.3.1). Improvement is achieved by applying a randomly selected removal and insertion operator to $\mathcal{S}_{\mathcal{P}_r}$ (Section 2.4.3.2). Then, an acceptance criterion based on SA decides whether $\mathcal{S}_{\mathcal{P}_r}$ is integrated into $\mathcal{S}_t$ or discarded (Section 2.4.3.3). If we accept the new solution, we immediately replace the corresponding routes in $\mathcal{S}_t$ with the modified routes and update the customers assigned to the common carrier according to $\mathcal{S}_{\mathcal{P}_r}$. Note that the definition of the next subproblem to be investigated depends on the solution of the current subproblem because we modify $\mathcal{S}_t$ continuously. The entire procedure is repeated for two rounds (in every round one subproblem $\mathcal{P}_r$ originates from each route r).

If the resulting solution is feasible, we further improve the solution by applying a VND with a probability of 0.25 (Section 2.4.3.4). Then, at the end of every iteration, the SA criterion (Section 2.4.3.3) decides whether the search is continued from the new solution. The search terminates after η iterations.

2.4.3.1. Decomposition Strategy

Although problem decomposition is generally used in algorithms developed to tackle large-scale VRP instances (see, e.g., Vidal, Crainic, et al., 2013), we observed that even for medium-sized instances of the VRPPC, we often find better solutions if we focus the search on partial solutions instead of the overall problem. We define subproblems by means of the set of customers to be served and the vehicles available to carry out the service. Our decomposition strategy works as follows: In randomized order, we generate one subproblem $\mathcal{P}_r$ for each route r of the private fleet. Based on the tentative solution $\mathcal{S}_t$, each subproblem $\mathcal{P}_r$ is defined by:

i) *The customers and the vehicle that are currently assigned to route r.*
ii) *All customers and vehicles of a random number α of routes that are closest to route r.* The number of routes α is drawn from the interval $[2, \lfloor |\mathcal{K}|/4 \rfloor]$ if $|\mathcal{K}| \geq 12$ and set to $\alpha = \min(|\mathcal{K}| - 1, 2)$, otherwise. We select the α routes that are closest to route r by measuring the distance between routes as the Euclidean distance between their centers of gravity, where the center of gravity of a route is calculated as the average of the coordinates of the vertices of the route.
iii) *Customers that are closely-located to route r and that are currently assigned to the common carrier.* To select such customers, we create a rectangular box that contains all customers already present in $\mathcal{P}_r$ after steps (i) and (ii), and we add a margin to every side of the bounding box that corresponds to 10% of the maximal distance between any two customers in the instance. Then, we add all

customers to $\mathcal{P}_r$ that are assigned to the common carrier in $\mathcal{S}_t$ and that are positioned within the enlarged box.

2.4.3.2. Removal and Insertion Operators

We remove δ customers from $\mathcal{S}_{\mathcal{P}_r}$. Let $|\mathcal{P}_r|$ denote the number of vertices in the current subproblem of our instance $\mathcal{P}_r$. Then δ is drawn from the interval $[\omega_{min}, \omega_{max}] \cdot \min(|\mathcal{P}_r|, 100)$ with parameters ω_{min} and ω_{max}.

Our LNS uses the following removal operators:

Random removal removes arbitrary customers from the routes of the private fleet and from the common carrier.

Route and common carrier removal selects at random routes from the private fleet or it selects the common carrier and removes the customers assigned until δ customers are removed.

Worst removal was proposed in Ropke and Pisinger (2006b) in order to remove customers whose presence in the solution strongly contributes to the objective function value. We do not use the direct contribution to $f_{gen}(\mathcal{S})$ to identify these customers but introduce problem-specific measures to select customers for removal. Whenever the worst removal operator is selected, we select one of three measures at random that is then used until δ customers are removed. Let S^{-i} denote a solution where customer i is removed and is either replaced by the connection between its predecessor and successor if it was previously assigned to the private fleet or removed without additional modification if it was assigned to the common carrier. For each customer i, let $W_x(i)$ denote the value of one of the following measures $x \in \{1, 2, 3\}$:

(i) change of variable cost: $W_1(i) = f_{var}(\mathcal{S}) - f_{var}(\mathcal{S}^{-i})$,

(ii) change of variable cost compared to subcontracting: $W_2(i) = f_{var}(\mathcal{S}) - f_{var}(\mathcal{S}^{-i}) - h_i$,

(iii) and change of variable cost per unit of demand: $W_3(i) = (f_{var}(\mathcal{S}) - f_{var}(\mathcal{S}^{-i}))/q_i$.

Measure W_1 is implemented in two variants, one variant including customers assigned to the common carrier and one variant ignoring these customers, and measure W_2 does not consider customers assigned to the common carrier. In the next step, the customers are sorted in descending order of $W_x(i)$, and the vertex at position $\lfloor D \cdot b^{\chi_{rem}} \rfloor$ is chosen, where D is the size of the list, b is a uniform random number $\in [0, 1]$, and χ_{rem} is a parameter to control the amount of diversification. After a customer is removed, the values are updated.

Shaw removal was introduced in Shaw (1997) in order to select customers for removal that are similar to each other. We define the similarity $R(i, j)$ between two customers i and j by their geographical distance d_{ij}, the difference in demand $|q_i - q_j|$, and the difference in subcontracting cost $|h_i - h_j|$. Each term is weighted with a parameter χ and normalized using the maximum value in the instance:

$$R(i,j) = \chi_d \frac{d_{ij}}{\max\limits_{i,j \in V}(d_{ij})} + \chi_q \frac{|q_i - q_j|}{\max\limits_{i \in \mathcal{N}}(q_i) - \min\limits_{i \in \mathcal{N}}(q_i)} + \chi_h \frac{|h_i - h_j|}{\max\limits_{i \in \mathcal{N}}(h_i) - \min\limits_{i \in \mathcal{N}}(h_i)}.$$

The first customer is randomly selected, and we sort all remaining customers $j \in \mathcal{S}$ in ascending order of their $R(i, j)$-value. From this list, the customer at position $\lfloor D \cdot b^{\chi_{rem}} \rfloor$ is chosen as

described above. The next iteration starts from an already removed customer that is selected at random.

Historical node-pair removal was introduced in Ropke and Pisinger (2006a). The idea is to remove edges from the current solution that have so far been only present in solutions with poor quality. To this end, an auxiliary graph $\tilde{\mathcal{G}} = (\tilde{\mathcal{V}}, \tilde{\mathcal{E}})$ is created and initially a large weight $w_{\tilde{e}}$ is assigned to each edge $\tilde{e} \in \tilde{\mathcal{E}}$. In the following, for each edge e that is present in the current solution S, the edge weight in the auxiliary graph is replaced if the current objective function value is smaller than the previous weight, i.e., $w_{\tilde{e}} := f(S)$ if $f(S) < w_{\tilde{e}}$.

To select customers, we assign to every customer served by the private fleet the sum of the weights $w_{\tilde{e}}$ of the two edges incident to the customer and then sort the customers in descending order according to this value. Now, the customers with the highest value are removed. We adapt this procedure to also account for customers that are assigned to the common carrier (for which there are no edges) as follows: (i) in addition to the edge weights, we store for each vertex $\tilde{v}$ a weight $w_{\tilde{v}}$ that contains the best objective function value of any solution encountered so far in which the customer was subcontracted and include them in the list, (ii) because each customer served by the private fleet is assigned the sum of two edge weights, but customers served by the common carrier are only assigned one vertex weight, we divide each edge-related value by 2 before we add it to the list to make the measures comparable.

The following insertion operators are used in our LNS:

Greedy insertion basic determines for each unassigned customer i the minimal increase of the generalized cost function $\Delta f_{gen}(S^{+i})$ when i is inserted into the routes of the private fleet. Then, the customer with the smallest value of $\min(\Delta f_{gen}(S^{+i}), h_i)$ is assigned to either the best route of the private fleet at its best position (if the first term is smaller) or to the common carrier (if the second term is smaller). This is repeated until all customers are assigned.

Greedy insertion priority works similar to the basic version, but we modify the selection criterion for the next customer to insert. The idea is to prefer customers in the beginning that are expensive to subcontract and that have a low demand. Therefore, instead of selecting the customer i based on the minimum cost increase $\Delta f_{gen}(S^{+i})$, we calculate for each customer (i) the difference between the minimum cost change for the assignment to the private fleet and the cost of subcontracting the customer in relation to its demand, i.e., $(\Delta f_{gen}(S^{+i}) - h_i)/q_i$, and (ii) the direct subcontracting cost per unit, i.e., h_i/q_i. Then, we select the customer for insertion where the minimum of these two values is smallest. If the first value is smaller, we insert the customer at the cost-minimal position in the corresponding route of the private fleet. If the second value is smaller, we assign it to the common carrier.

Regret-2 insertion aims at finding a customer insertion order that tries to avoid negative future consequences, i.e., we insert a customer now because otherwise we might regret it. A description of the regret-k insertion is given in Ropke and Pisinger (2006a), we only implement the case $k = 2$ because larger values of k did not improve the solution quality in preliminary studies. The regret-2 value is calculated as the difference between the second best assignment to a route of the private fleet or to the common carrier and the best assignment. In every step, we choose the customer with the currently highest regret-2 value and insert this customer into the best route or assign it to the common carrier, whichever is cheaper. If the best assignment corresponds to the common carrier,

we assume a regret-2 value of zero because the common carrier has unlimited capacity.

Insertion diversification adapts the three previously introduced insertion operators by adding an additional term to the change of the objective function value. On the one hand, we set the cost h_i of subcontracting customer i to $h_i := h_i + \iota$ where ι is uniformly chosen from the interval $\iota \in [-\zeta \cdot h_{max}, \zeta \cdot h_{max}]$ with $h_{max} = \max_{i \in \mathcal{N}}(h_i)$ and parameter ζ in order to try different configurations of subcontracted customers. On the other hand, for assignments to the private fleet, we use a principle known as continuous diversification (Cordeau, Laporte, and Mercier, 2001) that originated in the context of tabu search but is less restrictive than using a tabu list. Assignments to routes of the private fleet are changed based on the history of the solution process in order to encourage customer-route combinations that have not occurred very frequently. To this end, we measure the frequency u_{ij} of assigning customer i to route j and derive a penalty term $\kappa \cdot |\mathcal{K}| \cdot \sqrt{u_{ij} \cdot f(\mathcal{S}_{best})/|\mathcal{N}|}$ that grows sublinearly with this frequency. The penalty depends on a parameter $\kappa = 0.1$ that controls the extent of diversification, the number of vehicles $|\mathcal{K}|$, and the currently best objective function value per customer $f(\mathcal{S}_{best})/|\mathcal{N}|$.

Probabilistic insertion randomizes the order of customer insertions and the decision whether to assign a customer to one of the vehicle routes or to the common carrier. The next customer i is selected randomly, and then we calculate the changes of the objective function value $\Delta f^x_{gen}(S^{+i})$ with $x = r$ for assigning i to route r at its cost-minimal position and with $x = c$ for assigning i to the common carrier. Now, we use roulette wheel selection with probabilities inversely proportional to the weighted changes in the objective value $(\Delta f^x_{gen}(S^{+i}))^{\chi_{ins}}$ to decide whether to assign customer i to route r at the best position on the route or to the common carrier, i.e., the higher the increase when assigned to x, the lower the probability to select it. The parameter χ_{ins} controls the amount of diversification.

2.4.3.3. Acceptance Criterion

We use an SA based acceptance criterion (Kirkpatrick, Gelatt, and Vecchi, 1983) to decide (i) whether to replace the solution to each subproblem with the newly generated solution for the subproblem, and (ii) whether to continue the search from the tentative solution $\mathcal{S}_t$ after all subproblems have been solved or to continue from the current solution $\mathcal{S}_c$. Based on the temperature T and the difference between the objective function value of a new solution $\mathcal{S}_{new}$ and the previous solution $\mathcal{S}_{old}$, SA decides whether to accept deteriorating solutions (improving ones are always accepted) with probability $p(\mathcal{S}_{new}, \mathcal{S}_{old}, T) = e^{(f_{gen}(\mathcal{S}_{old}) - f_{gen}(\mathcal{S}_{new}))/T}$.

We set start and end temperature such that a new solution that deteriorates the initial solution by $\tau_{max}\%$ and $\tau_{min}\%$, respectively, is accepted with a probability of 50%. After each iteration, T decreases by a constant factor, which is determined such that the end temperature is reached after 50% of the total iterations. From there, we keep T constant instead of further decreasing it to allow more diversification in the search. Pretests have shown that this has strong positive effects on the solution quality. To evaluate the acceptance of a new solution $\mathcal{S}_{\mathcal{P}_r}$, we determine $f_{gen}(\mathcal{S}_{new})$ as the objective value of the complete solution that would be obtained if $\mathcal{S}_{\mathcal{P}_r}$ replaced the previous solution to $\mathcal{P}_r$. This is necessary because the temperature T is scaled to the value of complete solutions of the original problem.

2.4.3.4. Variable Neighborhood Descent

After two rounds of problem decomposition and improvement using LNS, feasible solutions are improved by a VND with a probability of 0.25. The VND follows a first-improvement strategy, and the list of neighborhoods contains the following operators in the given order: relocate (Waters, 1987), exchange (Savelsbergh, 1992), and a restricted version of 2-add-drop (Bolduc, Renaud, Boctor, and Laporte, 2008). Relocate and exchange are implemented in inter- and intra-route fashion. The 2-add-drop operator is specific to the VRPPC and was originally introduced as combined operator that (i) transfers up to two customers from the common carrier to the private fleet or vice versa, and (ii) transfers a single customer from the private fleet to the common carrier and at the same time inserts another customer currently assigned to the common carrier at the best possible position within the routes of the private fleet. We use only the second variant.

To limit the computational effort, we restrict the search to promising moves as follows: For each customer i, we store the closest $0.3 \cdot \min(|\mathcal{V}|, 150)$ vertices in an immutable neighbor list. We generate only those relocate moves of which either the new successor or predecessor of i is contained in the neighbor list of i, but we always evaluate the move where i is assigned to the common carrier. For exchange moves, only vertices in the neighbor list of i are considered as exchange partners of i. For 2-add-drop moves, we do not restrict the search to closely located vertices because good insertion positions can be far from the removal position.

2.4.4. Set Covering with Fleet Constraints

Finally, we apply a post-processing step that aims at improving on the best found solution by solving a set-covering problem with constraints on the fleet composition. Similar techniques have been successfully used in, e.g., Rochat and Taillard (1995) and Groër, Golden, and Wasil (2011), and Subramanian, Uchoa, and Ochi (2013). For each vehicle group $l \in \mathcal{L}$, we collect all feasible routes encountered during the search in a pool of routes Ω_l^{LNS}. Whenever we find a route of a vehicle of group l that serves the same customers as a route already present in the respective pool but with lower cost, we replace the corresponding route. Then, we solve formulation (2.1) using a commercial solver with a time limit of $\min(300, |\bigcup_{l \in \mathcal{L}} \Omega_l^{LNS}|/100)$ seconds. We initialize the solver with $\mathcal{S}_{best}$ in order to decrease the computing time. If we obtain a solution in which customers are contained in more than one route, we simply remove the redundant occurrences.

2.5. Numerical Studies

This section details the experiments to assess the performance of our BPC and our LNS. Section 2.5.1 introduces the benchmark instances on which both algorithms are evaluated. Section 2.5.2 discusses the parameter tuning, the value of individual components, and the comparison to the state-of-the-art for our LNS. Results of our BPC algorithm are presented in Section 2.5.3.

2.5.1. Benchmark Instances

In our computational studies, we use the benchmark sets available from the literature. Bolduc, Renaud, Boctor, and Laporte (2008) introduce two sets that assume a homogeneous vehicle fleet. These sets are based on the well-known capacitated vehicle-routing problem (CVRP) instances of Christofides, Mingozzi, and Toth (1979) (14 instances with 50–199 customers) and Golden et al. (1998) (20 instances with 200–483 customers). The names of the instances of these sets start with CE and G, respectively. To obtain VRPPC instances, the original instances are adapted as follows: (i) the number of vehicles is reduced such that only 80% of the total demand of the customers can be satisfied by the private fleet, (ii) the vehicle fixed costs are set based on the average route length of the best known solution to the original instance, and (iii) the cost of subcontracting a customer is based on the best known objective value of the original instance, the distance of the customer to the depot, and its demand. To obtain instances with a heterogeneous fleet, both sets are further modified in Bolduc, Renaud, Boctor, and Laporte (2008). The vehicles are divided into two or three different vehicle types, with 80%, 100% and 120% of the capacity and of the fixed cost of the vehicles of the homogeneous fleet. The instance names of these sets start with CE-H and G-H, respectively.

2.5.2. Performance of the LNS

Experimental environment and parameter setting For our LNS, we perform all numerical experiments with a single core of a desktop computer equipped with an Intel I7 processor at 2.8 GHz with 8 GB of RAM and running Windows 7 Enterprise. The algorithm is implemented in Java, and the commercial solver used to solve the set-covering problem (Section 2.4.4) is Gurobi at version 7.0.1. Ten runs per instance are performed.

We set the total number of search iterations to $\eta = 20,000$ because this value offers a good trade-off between run-time and solution quality. We tune the other parameters of the algorithm as follows: We use only three randomly selected instances from the set CE and three instances from the set G to keep the computational tuning effort low and to avoid overfitting the algorithm to the benchmark set. We begin with a reasonable base setting of the parameters that we have determined during the development of our algorithm. Then, we iteratively modify the base value of each parameter to a reasonable lower and higher value. We keep the best value for each parameter (based on the average quality of 10 runs) and continue with the next parameter. Parameters that are closely related are grouped and changed simultaneously to keep the testing effort moderate. We examine the following parameters in the given order: The external cost noise factor (ζ), the weight factors for the Shaw removal operator (χ_d, χ_h, χ_q), the minimal and maximal factors for the number of customers to remove $(\omega_{min}, \omega_{max})$, the removal diversification factor (χ_{rem}), the continuous diversification factor (κ), the probabilistic insertion factor χ_{ins}, the minimal and maximal SA deterioration percentage (τ_{min}, τ_{max}), and finally the penalty update factor ϱ. Table 2.1 summarizes the results of our parameter study. The base value is given in the middle and the best value is marked in bold and used as final setting. For each setting, we report the deviation in percent of the objective value Δ_f to the value of the best setting. We conclude that our solution method is quite robust against parameter variations as the deviation from the best setting is always below 0.2%.

Table 2.1.: Results of our parameter study on a subset of the VRPPC instances. We mark the best setting for each parameter in bold and use it as final setting. For each setting, we report the deviation (Δ_f) to the best setting of the respective parameter.

LNS			
ζ	0.25	**0.5**	0.75
$\Delta_f(\%)$	0.02	0.00	0.1
(χ_d, χ_h, χ_q)	$(\mathbf{4}, \mathbf{5}, \mathbf{6})$	$(6, 5, 4)$	$(6, 4, 5)$
$\Delta_f(\%)$	0.00	0.03	0.19
$(\omega_{min}, \omega_{max})$	$(0.05, 0.4)$	$(\mathbf{0.1}, \mathbf{0.4})$	$(0.1, 0.6)$
$\Delta_f(\%)$	0.09	0.00	0.02
χ_{rem}	26	**36**	46
$\Delta_f(\%)$	0.16	0.00	0.03
κ	**0.02**	0.1	0.5
$\Delta_f(\%)$	0.00	0.04	0.17
χ_{ins}	2	**4**	6
$\Delta_f(\%)$	0.07	0.00	0.09
SA & Penalties			
τ_{min}	0.01	**0.05**	0.1
$\Delta_f(\%)$	0.17	0.00	0.03
τ_{max}	0.1	**0.2**	0.4
$\Delta_f(\%)$	0.07	0.00	0.02
ϱ	1.02	**1.1**	1.5
$\Delta_f(\%)$	0.14	0.00	0.12

Influence of algorithmic components To assess the effect that different components of the algorithm have on the quality and speed of our LNS, we compare the following variants of the LNS on the benchmark sets with homogeneous (Table 2.2) and heterogeneous fleet (Table 2.3):

LNS Our LNS with all components as described above

LNS–noSC In this setting, we omit the set covering as post-processing step. To be able to observe the undistorted effect of the post processing, the results of LNS–noSC and LNS presented in the following are based on the same runs: LNS–noSC is the result of the LNS before the post-processing phase starts.

LNS–noDec This setting does not use the problem decomposition technique but instead applies the removal and insertion operators on complete solutions. To make the comparison fair, we double the number of total search iterations because we always do two rounds of destroy and repair in the variants featuring decomposition.

For each instance, column $|\mathcal{N}|$ reports the number of customers, column BKS the previously best known solution value from the literature (in the case of Table 2.2 taken from Vidal, Maculan, et al. (2016) and in the case of Table 2.3 taken from Bolduc, Renaud, Boctor, and Laporte (2008) and Potvin and Naud (2011) and from the updated results for Côté and Potvin (2009) reported in Appendix A). For each variant of the LNS, column Δ_{best} reports the percentage gap between the best solution obtained in the 10 runs and the previous best known solution, and column t gives the run-time in seconds. Finally, columns

$\overline{\text{LNS}}$ list the best objective value f that we found during the overall testing and the respective gap to the BKS Δ_f. Note that the best solution value for each instance is marked in bold, and average values are provided in the last row.

The solution behavior is rather similar on both types of instances: The two variants using decomposition show a clearly superior solution quality while having higher run-times. Of these two, LNS shows a better solution quality than LNS–noSC, but the average run-time increases by roughly 25%. The results show that both algorithmic components—decomposition and post-processing—have a positive impact on solution quality and a negative one on run-time. In real-world applications, the decision for one of the variants depends on the desired tradeoff between solution quality and run-time that the planner wants to achieve. For the following comparison to the state-of-the-art, we put the major emphasis on solution quality and only investigate variant LNS.

Comparison to the state-of-the-art As Table 2.2 shows, LNS is able to match 10 and improve 6 previous BKS out of the 34 instances with a homogeneous vehicle fleet. During the overall testing, $\overline{\text{LNS}}$ matches 12 and improves 12 BKS of these instances. On the instances with a heterogeneous fleet (see Table 2.3), LNS matches 1 and improves 26 out of 34 instances, and $\overline{\text{LNS}}$ matches 3 and improves 29 instances.

Table 2.4 gives an aggregate comparison of LNS to the state-of-the-art methods from the literature: RIP (Bolduc, Renaud, Boctor, and Laporte, 2008), TS (Côté and Potvin, 2009), TS+ (Potvin and Naud, 2011), AVNS (Stenger, Vigo, et al., 2013), AVNS-RN (Stenger, Schneider, and Goeke, 2013), UGHS (Vidal, Maculan, et al., 2016), MS-ILS (Vidal, Maculan, et al., 2016), and MS-LS (Vidal, Maculan, et al., 2016). The upper part of the table is devoted to the comparison on the instances with a homogeneous fleet, the lower part to those with a heterogeneous fleet. The first three rows in each part report the average percentage gap to the previous best known solution in percent achieved by each method in the best of the runs on the respective instance set. The solutions reported for RIP are obtained by solving each instance only once, all other heuristics report the best solution found in 10 runs. The fourth row gives the average run-time per instance in minutes. Finally, in row CPU@GHz, we list the processor and clock rate of the computers on which the respective methods were tested. We indicate with † results that are not directly comparable because they are obtained using truncated customer coordinates. Please note that the results reported for TS are based on non-truncated coordinates because the authors repeated the testing of their algorithm and provided us with this data. We report their updated results in Appendix A.

On the homogeneous-fleet instances, LNS provides the best solution quality on set CE and the second-best quality after UHGS on set G within very competitive run-times. On the heterogeneous-fleet instances, LNS is able to significantly improve the average solution quality by more than 1% compared to all competitors. However, run-times compared to TS are also clearly higher.

Table 2.2.: Detailed results and comparison of algorithmic components on the instance sets with a homogeneous fleet.

			LNS		**LNS–noSC**		**LNS–noDec**		$\overline{\textbf{LNS}}$	
Inst.	$\lvert\mathcal{N}\rvert$	BKS	$\Delta_{best}(\%)$	t(sec)	$\Delta_{best}(\%)$	t(sec)	$\Delta_{best}(\%)$	t(sec)	f	$\Delta_f(\%)$
CE										
CE-01	50	**1119.47**	**0.00**	27	**0.00**	27	**0.00**	13	**1119.47**	**0.00**
CE-02	75	**1814.52**	**0.00**	53	**0.00**	53	**0.00**	118	**1814.52**	**0.00**
CE-03	100	**1919.05**	**0.00**	73	**0.00**	73	0.11	67	**1919.05**	**0.00**
CE-04	150	**2505.39**	0.08	133	0.08	133	**0.00**	161	**2505.39**	**0.00**
CE-05	199	**3081.59**	0.23	298	0.25	272	0.35	374	3086.75	0.17
CE-06	50	**1207.47**	**0.00**	27	**0.00**	27	**0.00**	13	**1207.47**	**0.00**
CE-07	75	**2004.53**	**0.00**	53	**0.00**	53	**0.00**	109	**2004.53**	**0.00**
CE-08	100	**2052.05**	**0.00**	73	0.07	73	0.09	94	**2052.05**	**0.00**
CE-09	150	2422.74	**-0.12**	130	**-0.12**	128	-0.08	190	**2419.84**	**-0.12**
CE-10	199	3381.67	-0.12	288	-0.09	269	0.08	404	**3376.80**	**-0.14**
CE-11	120	**2330.94**	**0.00**	104	**0.00**	104	**0.00**	58	**2330.94**	**0.00**
CE-12	100	**1952.86**	0.04	73	0.04	73	**0.00**	82	**1952.86**	**0.00**
CE-13	120	**2858.83**	**0.00**	104	**0.00**	104	**0.00**	60	**2858.83**	**0.00**
CE-14	100	**2213.02**	**0.00**	72	**0.00**	72	**0.00**	65	**2213.02**	**0.00**
G										
G-01	240	**14131.18**	0.21	349	0.25	305	0.58	352	14134.20	0.02
G-02	320	**19142.75**	0.06	471	0.20	400	1.35	463	19145.60	0.01
G-03	400	**24409.02**	0.45	604	0.63	465	1.90	526	24563.34	0.63
G-04	480	34362.8	**-0.52**	731	-0.45	554	1.29	623	**34183.06**	**-0.52**
G-05	200	**14223.63**	1.07	193	1.19	189	0.70	132	14246.68	0.16
G-06	280	**21382.16**	0.92	279	1.01	264	0.54	216	21496.93	0.54
G-07	360	**23373.38**	0.50	550	0.51	399	1.21	464	23398.93	0.11
G-08	440	29797.62	**-0.34**	683	-0.10	510	1.59	587	**29697.75**	**-0.34**
G-09	255	1326.2637	0.35	460	0.35	323	0.29	437	**1325.03**	**-0.09**
G-10	323	1593.79492	0.13	697	0.13	498	0.43	474	**1586.50**	**-0.46**
G-11	399	2173.82151	0.30	917	0.30	627	0.63	510	**2163.72**	**-0.46**
G-12	483	2494.56071	0.27	1036	0.27	757	0.49	543	**2490.23**	**-0.17**
G-13	252	**2258.02**	0.08	869	0.08	661	0.99	536	2259.75	0.08
G-14	320	2683.73	0.00	1209	0.29	914	1.27	579	**2682.90**	**-0.03**
G-15	396	**3145.11**	0.07	1672	0.07	1370	0.74	637	3147.21	0.07
G-16	480	3620.71	0.11	2107	0.11	1803	0.82	695	**3614.79**	**-0.16**
G-17	240	**1666.31**	**0.00**	446	**0.00**	385	0.04	325	**1666.31**	**0.00**
G-18	300	**2730.55**	0.09	1015	0.17	712	0.39	560	2731.98	0.05
G-19	360	3497.2	-0.02	1476	0.06	1170	0.61	634	**3492.31**	**-0.14**
G-20	420	4312.45	-0.10	2080	-0.01	1771	0.90	734	**4303.56**	**-0.21**
Avg.			0.11	569	0.16	457	0.51	348		-0.03

Table 2.3.: Detailed results and comparison of algorithmic components on the instance sets with a heterogeneous fleet.

			LNS		**LNS–noSC**		**LNS–noDec**		$\overline{\textbf{LNS}}$	
Inst.	$\lvert\mathcal{N}\rvert$	BKS	$\Delta_{best}(\%)$	t(sec)	$\Delta_{best}(\%)$	t(sec)	$\Delta_{best}(\%)$	t(sec)	f	$\Delta_f(\%)$
CE-H										
CE-H-01	50	**1191.70**	0.19	27	0.19	27	0.19	16	**1191.70**	**0.00**
CE-H-02	75	1790.67	0.46	49	0.46	49	**-0.07**	121	**1789.41**	**-0.07**
CE-H-03	100	1917.96	-0.10	70	-0.10	70	**-0.23**	78	**1913.49**	**-0.23**
CE-H-04	150	2475.16	-0.10	131	0.04	129	-0.01	149	**2465.51**	**-0.39**
CE-H-05	199	3143.01	**-0.76**	322	**-0.76**	274	-0.50	416	**3119.10**	**-0.76**
CE-H-06	50	**1204.48**	0.00	26	0.00	26	0.00	13	**1204.48**	**0.00**
CE-H-07	75	**2025.98**	0.34	53	0.34	53	0.04	103	2026.70	0.04
CE-H-08	100	1984.36	**-0.28**	71	**-0.28**	71	0.00	64	**1978.79**	**-0.28**
CE-H-09	150	2438.73	-0.32	130	-0.28	128	-0.42	150	**2424.43**	**-0.59**
CE-H-10	199	3267.85	-0.71	287	-0.58	253	-0.54	387	**3240.00**	**-0.85**
CE-H-11	120	2303.13	**-0.06**	105	**-0.06**	105	0.00	62	**2301.78**	**-0.06**
CE-H-12	100	1908.74	**-0.04**	74	**-0.04**	72	**-0.04**	109	**1908.05**	**-0.04**
CE-H-13	120	2842.18	-0.07	104	-0.07	104	-0.02	59	**2832.88**	**-0.33**
CE-H-14	100	**1907.74**	0.30	71	0.30	71	0.17	68	**1907.75**	**0.00**
G-H										
G-H-01	240	14251.75	-0.81	325	-0.69	303	-0.74	277	**14097.33**	**-0.95**
G-H-02	320	18560.07	-0.57	539	-0.51	400	-0.04	453	**18412.40**	**-0.80**
G-H-03	400	25356.63	**-1.34**	589	-1.19	465	-0.45	503	**25016.72**	**-1.34**
G-H-04	480	34589.11	**-0.75**	774	-0.65	553	0.85	627	**34328.99**	**-0.75**
G-H-05	200	15667.13	-1.10	184	-1.04	182	-0.25	99	**15398.76**	**-1.71**
G-H-06	280	19975.32	-0.74	337	-0.70	301	-0.60	306	**19743.63**	**-1.16**
G-H-07	360	23510.98	-0.04	531	0.08	398	0.73	431	**23293.54**	**-0.92**
G-H-08	440	27420.68	**-0.23**	659	-0.13	496	0.65	554	**27358.69**	**-0.23**
G-H-09	255	1331.83	0.10	468	0.10	331	0.24	448	**1324.99**	**-0.27**
G-H-10	323	1561.52	0.23	596	0.23	425	-0.02	463	**1556.39**	**-0.05**
G-H-11	399	2195.31	**-0.30**	935	-0.21	638	0.37	524	**2185.08**	**-0.30**
G-H-12	483	**2487.38**	0.36	1176	0.36	873	1.03	559	2488.08	0.03
G-H-13	252	2239.18	-0.42	727	-0.29	568	0.05	519	**2218.92**	**-0.85**
G-H-14	320	2682.85	-0.88	1156	-0.88	916	-0.17	625	**2649.32**	**-1.25**
G-H-15	396	3131.89	-0.39	1705	-0.35	1403	0.55	681	**3108.53**	**-0.45**
G-H-16	480	3629.41	**-0.60**	2111	**-0.60**	1807	0.77	738	**3598.41**	**-0.60**
G-H-17	240	1695.75	**-0.58**	571	-0.38	376	-0.29	460	**1685.97**	**-0.58**
G-H-18	300	2740.05	-0.37	1004	-0.27	701	0.46	590	**2729.61**	**-0.38**
G-H-19	360	3464.70	-0.22	1451	-0.16	1145	0.23	675	**3453.41**	**-0.33**
G-H-20	420	4352.35	-0.80	2129	-0.80	1820	0.45	798	**4311.17**	**-0.95**
Avg.			-0.31	573	-0.26	457	0.07	357		-0.51

Table 2.4.: Overview of results obtained with LNS and heuristics from the literature. Note that Xe stand for Intel Xeon and Opt for AMD Opteron.

	RIP	**TS**	**TS+**	**AVNS**	**AVNS-RN**	**UHGS**	**MS-ILS**	**MS-LS**	**LNS**
Avg. $\Delta_{best}(\%)$									
CE	1.063	0.345	0.309	0.196	0.131	0.015	0.065	1.405	0.008
G	1.988	1.807	†0.225	0.636	0.554	0.110	0.469	2.643	0.181
CE & G	1.666	1.258	†0.266	0.473	0.380	0.074	0.317	2.211	0.110
Avg. t**(min)**	1×17.45	10×2.98	10×34.86	10×11.94	10×12.14	10×26.40	10×16.62	10×1.89	10×9.49
Avg. $\Delta_{best}(\%)$									
CE-H	0.646	0.391	0.291						-0.082
G-H	1.324	1.162	†-0.344						-0.473
CE-H & G-H	1.084	0.879	†-0.093						-0.305
Avg. t**(min)**	1×17.50	10×2.86	10×40.56						10×9.55
CPU@GHz	Xe@3.6	I7@3.4	Opt@2.2	I5@2.67	I5@2.67	Xe@3.07	Xe@3.07	Xe@3.07	I7@2.8

2.5.3. Results of the BPC

All algorithmic components of the BPC were coded in C++ and compiled into 64-bit single-thread code with MS Visual Studio 2013. The callable library of CPLEX 12.6.0 was used for re-optimizing the master program. The computational experiments of the BPC were conducted on a standard PC with an Intel I7-5930k 3.5 GHz processor with 64 GB of main memory. The time limit was set to two hours.

Tables 2.5 and 2.6 summarize the results of our BPC on the homogeneous and heterogeneous instance sets, respectively. The columns have the following meaning: IUB is the initial upper bound given to the algorithm (from our LNS or from the literature), LB denotes the lower bound provided by the algorithm when the time limit was reached (values in bold indicate that the instance was solved to proven optimality within the time limit while values marked with an asterisk constitute new best known solutions), $\Delta_{LB}(\%)$ is the optimality gap in percent, t(sec) the run-time in seconds taken by the algorithm to solve an instance to proven optimality (or TL if the time limit was reached), #Nds is the number of solved branch-and-bound nodes, and #Cuts the number of generated cuts.

Our results reveal that the proposed exact approach is able to solve some small to medium-sized instances to optimality in reasonable time. Thereby, new best known solutions are found for two of the instances from benchmark set CE-H. For the larger instances, only lower bounds can be provided within the time limit of two hours. For the two largest instances G-12/G-H-12, we were not able to solve the root node in the computation time of two hours. The average optimality gaps are 0.22%, 0.78%, 0.31%, and 1.10% for the instances sets CE, G, CE-H, and G-H, respectively. Thus, the lower bounds provided by our BPC seem to be rather tight while the upper bounds from our LNS (and from other state-of-the-art heuristics) also seem to be of good quality.

Table 2.5.: Results of the BPC for the homogeneous instances.

Inst.	$\lvert\mathcal{N}\rvert$	IUB	LB	$\Delta_{LB}(\%)$	t(sec)	#Nds	#Cuts
CE							
CE-01	50	1119.47	**1119.47**	0.00	10.7	27	127
CE-02	75	1814.52	**1814.52**	0.00	489.7	224	227
CE-03	100	1919.05	1916.49	0.13	TL	237	240
CE-04	150	2505.39	2495.26	0.41	TL	312	289
CE-05	199	3081.59	3066.67	0.49	TL	265	339
CE-06	50	1207.47	**1207.47**	0.00	13.9	25	147
CE-07	75	2004.53	**2004.53**	0.00	280.8	154	229
CE-08	100	2052.05	2049.11	0.14	TL	219	233
CE-09	150	2419.84	2407.18	0.53	TL	257	267
CE-10	199	3376.80	3355.75	0.63	TL	240	270
CE-11	120	2330.94	2323.89	0.30	TL	5	21
CE-12	100	1952.86	1950.79	0.11	TL	217	225
CE-13	120	2858.83	2850.75	0.28	TL	7	21
CE-14	100	2213.02	2211.63	0.06	TL	149	225
G							
G-01	240	14131.20	14044.00	0.62	TL	150	225
G-02	320	19142.80	18987.20	0.82	TL	56	225
G-03	400	24409.00	24200.60	0.86	TL	1	0
G-04	480	34183.10	33898.40	0.84	TL	1	0
G-05	200	14223.60	14144.50	0.56	TL	33	225
G-06	280	21382.20	21299.60	0.39	TL	29	225
G-07	360	23373.40	23172.40	0.87	TL	2	0
G-08	440	29679.80	29439.60	0.82	TL	1	0
G-09	255	1325.03	1308.85	1.24	TL	11	60
G-10	323	1586.50	1567.65	1.20	TL	1	0
G-11	399	2163.72	2137.78	1.21	TL	1	0
G-12	483	2490.23	—	—	TL	0	0
G-13	252	2258.02	2234.83	1.04	TL	44	225
G-14	320	2682.90	2658.79	0.91	TL	49	225
G-15	396	3145.11	3109.45	1.15	TL	16	60
G-16	480	3614.79	3578.88	1.00	TL	2	0
G-17	240	1666.31	**1666.31**	0.00	43.5	1	0
G-18	300	2730.55	2719.11	0.42	TL	270	225
G-19	360	3492.31	3479.43	0.37	TL	154	225
G-20	420	4303.56	4278.97	0.57	TL	47	225

Table 2.6.: Results of the BPC for the heterogeneous instances.

Inst.	$\|\mathcal{N}\|$	IUB	LB	$\Delta_{LB}(\%)$	t(sec)	#Nds	#Cuts
CE-H							
CE-H-01	50	1191.70	**1191.70**	0.00	598.4	78	226
CE-H-02	75	1790.67	*__1789.41__	0.00	2773.3	604	226
CE-H-03	100	1913.49	1906.78	0.35	TL	91	233
CE-H-04	150	2465.51	2449.38	0.66	TL	140	225
CE-H-05	199	3119.10	3095.40	0.77	TL	129	225
CE-H-06	50	1204.48	**1204.48**	0.00	56.7	36	186
CE-H-07	75	2025.98	*__2025.05__	0.00	5426.8	762	225
CE-H-08	100	1978.79	1973.81	0.25	TL	163	225
CE-H-09	150	2424.43	2414.22	0.42	TL	198	225
CE-H-10	199	3240.00	3218.85	0.66	TL	104	225
CE-H-11	120	2301.78	2286.11	0.69	TL	3	10
CE-H-12	100	1908.05	1903.41	0.24	TL	118	225
CE-H-13	120	2832.88	2824.19	0.31	TL	3	10
CE-H-14	100	1907.74	**1907.74**	0.00	2088.6	78	225
G-H							
G-H-01	240	14097.30	14002.80	0.67	TL	83	225
G-H-02	320	18412.40	18218.10	1.07	TL	15	101
G-H-03	400	25016.70	24729.70	1.16	TL	1	0
G-H-04	480	34329.00	33407.20	2.76	TL	1	0
G-H-05	200	15398.80	15370.10	0.19	TL	69	225
G-H-06	280	19743.60	19595.60	0.76	TL	8	70
G-H-07	360	23293.50	23158.40	0.58	TL	1	0
G-H-08	440	27358.70	26864.30	1.84	TL	1	0
G-H-09	255	1324.99	1308.17	1.29	TL	1	0
G-H-10	323	1556.39	1530.34	1.70	TL	1	0
G-H-11	399	2185.08	2152.47	1.52	TL	1	0
G-H-12	483	2488.08	—	—	TL	0	0
G-H-13	252	2218.92	2194.40	1.12	TL	47	225
G-H-14	320	2649.32	2621.75	1.05	TL	16	50
G-H-15	396	3108.53	3059.82	1.59	TL	9	40
G-H-16	480	3598.41	3547.73	1.43	TL	1	0
G-H-17	240	1685.97	1682.40	0.21	TL	320	225
G-H-18	300	2729.61	2718.39	0.41	TL	165	225
G-H-19	360	3453.41	3430.93	0.66	TL	67	225
G-H-20	420	4311.17	4270.90	0.94	TL	28	151

2.6. Summary and Conclusion

We present the first exact solution method for the VRPPC. Our BPC algorithm solves instances with up to 75 customers to optimality and provides tight lower bounds for instances with up to 480 customers. In addition, we propose an LNS as upper bounding procedure, which is among the best heuristic solution methods for the VRPPC. Our LNS features a decomposition procedure which may also be interesting for the solution of other VRP variants.

As future research, it seems worthwhile to extend the VRPPC to a planning problem spanning multiple days. This is motivated by applications in e-commerce, where often some customers have a desired delivery date, but others are indifferent about the concrete delivery date as long as it lies within a certain time frame. The idea here is to simultaneously solve the VRPPC on multiple days, i.e., one type of customers must be served on specified days, the other type can be scheduled freely, and those that cannot be served economically may be subcontracted.

Bibliography

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

Bolduc, M.-C., J. Renaud, and F. Boctor (2007). A heuristic for the routing and carrier selection problem. In: *European Journal of Operational Research* 183 (2), pp. 926–932.

Bolduc, M.-C., J. Renaud, F. Boctor, and G. Laporte (2008). A perturbation metaheuristic for the vehicle routing problem with private fleet and common carriers. In: *Journal of the Operational Research Society* 59 (6), pp. 776–787.

Christofides, N., A. Mingozzi, and P. Toth (1979). The vehicle routing problem. In: *Combinatorial Optimization*. Ed. by N. Christofides, A. Mingozzi, P. Toth, and C. Sandi. Chichester, UK: Wiley, pp. 315–338.

Chu, C.-W. (2005). A heuristic algorithm for the truckload and less-than-truckload problem. In: *European Journal of Operational Research* 165 (3), pp. 657–667.

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

Contardo, C. and R. Martinelli (2014). A new exact algorithm for the multi-depot vehicle routing problem under capacity and route length constraints. In: *Discrete Optimization* 12, pp. 129–146.

Cordeau, J.-F., G. Laporte, and A. Mercier (2001). A unified tabu search heuristic for vehicle routing problems with time windows. In: *Journal of the Operational Research Society* 52 (8), pp. 928–936.

Côté, J.-F. and J.-Y. Potvin (2009). A tabu search heuristic for the vehicle routing problem with private fleet and common carrier. In: *European Journal of Operational Research* 198 (2), pp. 464–469.

Feillet, D., M. Gendreau, and L.-M. Rousseau (2007). New refinements for the solution of vehicle routing problems with branch and price. In: *Information Systems and Operational Research* 45 (4), pp. 239–256.

Gaskell, T. J. (1967). Bases for vehicle fleet scheduling. In: *Journal of the Operational Research Society* 18 (3), pp. 281–295.

Gendreau, M., A. Hertz, and G. Laporte (1994). A tabu search heuristic for the vehicle routing problem. In: *Management Science* 40 (10), pp. 1276–1290.

Golden, B., E. Wasil, J. Kelly, and I.-M. Chao (1998). Fleet management and logisitics. In: *The impact of metaheuristics on solving the vehicle routing problem: Algorithms, problem sets, and computational results*. Ed. by T. G. Grainic and G. Laporte. Springer, pp. 33–56.

Groër, C., B. Golden, and E. Wasil (2011). A parallel algorithm for the vehicle routing problem. In: *INFORMS Journal on Computing* 23 (2), pp. 315–330.

Irnich, S. and G. Desaulniers (2005). Shortest path problems with resource constraints. In: *Column Generation*. Ed. by G. Desaulniers, J. Desrosiers, and M. Solomon. New York, NY: Springer. Chap. 2, pp. 33–65.

Irnich, S., G. Desaulniers, J. Desrosiers, and A. Hadjar (2010). Path-reduced costs for eliminating arcs in routing and scheduling. In: *INFORMS Journal on Computing* 22 (2), pp. 297–313.

Jepsen, M., B. Petersen, S. Spoorendonk, and D. Pisinger (2008). Subset-row inequalities applied to the vehicle-routing problem with time windows. In: *Operations Research* 56 (2), pp. 497–511.

Kirkpatrick, S., C. D. Gelatt, and M. P. Vecchi (1983). Optimization by simulated annealing. In: *Science* 220 (4598), pp. 671–680.

Krajewska, M. A. and H. Kopfer (2009). Transportation planning in freight forwarding companies: Tabu search algorithm for the integrated operational transportation planning problem. In: *European Journal of Operational Research* 197 (2), pp. 741–751.

Li, F., B. Golden, and E. Wasil (2005). Very large-scale vehicle routing: New test problems, algorithms, and results. In: *European Journal of Operational Research* 32 (5), pp. 1165–1179.

Liu, R., Z. Jiang, X. Liu, and F. Chen (2010). Task selection and routing problems in collaborative truckload transportation. In: *Transportation Research Part E: Logistics and Transportation Review* 46 (6), pp. 1071–1085.

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

Lysgaard, J. (2003). *CVRPSEP: A package of separation routines for the Capacitated Vehicle Routing Problem*. Working Paper 03-04. Aarhus, Denmark: Department of Management Science and Logistics, Aarhus School of Business.

Pecin, D., C. Contardo, G. Desaulniers, and E. Uchoa (2017). New enhancements for the exact solution of the vehicle routing problem with time windows. In: *INFORMS Journal on Computing* 29 (3), pp. 489–502.

Potvin, J.-Y. and M.-A. Naud (2011). Tabu search with ejection chains for the vehicle routing problem with private fleet and common carrier. In: *Journal of the Operational Research Society* 62 (2), pp. 326–336.

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

Roberti, R. and A. Mingozzi (2014). Dynamic ng-path relaxation for the delivery man problem. In: *Transportation Science* 48 (3), pp. 413–424.

Rochat, Y. and É. D. Taillard (1995). Probabilistic diversification and intensification in local search for vehicle routing. In: *Journal of Heuristics* 1 (1), pp. 147–167.

Ropke, S. and D. Pisinger (2006a). A unified heuristic for a large class of vehicle routing problems with backhauls. In: *European Journal of Operational Research* 171 (3), pp. 750–775.

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

Savelsbergh, M. (1992). The vehicle routing problem with time windows: Minimizing route duration. In: *ORSA Journal on Computing* 4 (2), pp. 146–154.

Schrimpf, G., J. Schneider, H. Stamm-Wilbrandt, and G. Dueck (2000). Record breaking optimization results using the ruin and recreate principle. In: *Journal of Computational Physics* 159 (2), pp. 139–171.

Shaw, P. (1997). *A new local search algorithm providing high quality solutions to vehicle routing problems*. Tech. rep. Glasgow, Scotland: Department of Computer Science, University of Strathclyde.

– (1998). Using constraint programming and local search methods to solve vehicle routing problems. In: *Principles and Practice of Constraint Programming – CP98*. Ed. by M. Maher and J.-F. Puget. Vol. 1520. Lecture Notes in Computer Science. London, United Kingdom: Springer, pp. 417–431.

Stenger, A., M. Schneider, and D. Goeke (2013). The prize-collecting vehicle routing problem with single and multiple depots and non-linear cost. In: *EURO Journal on Transportation and Logistics* 2 (1-2), pp. 57–87.

Stenger, A., D. Vigo, S. Enz, and M. Schwind (2013). An adaptive variable neighborhood search algorithm for a vehicle routing problem arising in small package shipping. In: *Transportation Science* 47 (1), pp. 64–80.

Subramanian, A., E. Uchoa, and L. S. Ochi (2013). A hybrid algorithm for a class of vehicle routing problems. In: *Computers & Operations Research* 40 (10), pp. 2519–2531.

Tang, L. and X. Wang (2006). Iterated local search algorithm based on very large-scale neighborhood for prize-collecting vehicle routing problem. In: *The International Journal of Advanced Manufacturing Technology* 29 (11), pp. 1246–1258.

Tilk, C., A.-K. Rothenbächer, T. Gschwind, and S. Irnich (2017). Asymmetry matters: Dynamic half-way points in bidirectional labeling for solving shortest path problems with resource constraints faster. In: *European Journal of Operational Research* 261 (2), pp. 530–539.

Vidal, T., T. G. Crainic, M. Gendreau, and C. Prins (2013). A hybrid genetic algorithm with adaptive diversity management for a large class of vehicle routing problems with time-windows. In: *Computers & Operations Research* 40, pp. 475–489.

Vidal, T., N. Maculan, L. S. Ochi, and P. H. V. Penna (2016). Large neighborhoods with implicit customer selection for vehicle routing problems with profits. In: *Transportation Science* 50 (2), pp. 720–734.

Waters, C. D. J. (1987). A solution procedure for the vehicle-scheduling problem based on iterative route improvement. In: *Journal of the Operational Research Society* 38 (9), pp. 833–839.

Yellow, P. (1970). A computational modification to the savings method of vehicle scheduling. In: *Operational Research Quarterly* 21, pp. 281–283.

Chapter 3

Respecting Consistency Requirements in Delivery

3.1. Introduction

Vehicle-routing problems (VRPs) with consistency considerations have received substantial interest in recent years because of the practical importance of providing consistent service in many industries, like, e.g., small package shipping, health care, or vendor-managed inventory systems (for a survey, see Kovacs, Hartl, et al., 2014). To boost customer satisfaction, customers should be served at roughly the same time (arrival time consistency, ATC) by the same driver (driver consistency, DC), or at least a small set of familiar drivers, each time they require service. Taking the driver's perspective, serving the same customers repeatedly makes the driver familiar with the geographic region and the characteristics of the customer, and thus more efficient in fulfilling his tasks.

The most prominent variant of the class of VRPs with consistency considerations is the consistent VRP (ConVRP), introduced by Groër, Golden, and Wasil (2009). The ConVRP is a multi-day VRP requiring that, in addition to the traditional constraints on vehicle capacity and route duration, the same driver serves the same customers at approximately the same time on each day that these customers require service, given by a maximum allowed difference between the arrival times on the different days. Originally, the problem is motivated from the delivery and collection operations at United Parcel Services, where strong emphasis is put on customer and employee satisfaction.

In the academic literature, the ConVRP has received adequate attention from the heuristic side. Groër, Golden, and Wasil (2009) develop a two-phase algorithm based on record-to-record travel, which first constructs template routes and then uses them to generate the daily routes by removing non-occurring customers and inserting new ones. The template routes are based on a simple precedence principle, which states that if two customers a and b are served by the same driver on a specific day, then the driver that serves them and the order in which they are served must be the same on all days on which they both require service. Since the publication of this article, four algorithms have been proposed that are able to solve ConVRP instances, three of which use the idea of a template that is adjusted to the individual days (Sungur et al., 2010; Tarantilis, Stavropoulou, and Repoussis, 2012; Kovacs, Parragh, and Hartl, 2014) and one approach that applies search over all routes of all days (Kovacs, Golden, et al., 2015).

Sungur et al. (2010) actually solve a different problem called courier delivery problem, which is modeled as a multi-day VRP with soft time windows, using robust optimization and scenario-based stochastic programming to represent uncertainty in service time and probabilistic customers. With slight adaptations,

their tabu search (TS) approach can provide solutions that adhere to the precedence principle of Groër, Golden, and Wasil (2009) and thus solve the ConVRP. Tarantilis, Stavropoulou, and Repoussis (2012) use a TS to improve the template routes and the resulting daily routes in a sequential manner. Kovacs, Parragh, and Hartl (2014) present an adaptive large neighborhood search (ALNS) that is solely applied to the template routes; the daily routes are improved using a truncated 2-opt operator. In addition, the paper proposes a relaxed version of the problem called ConVRP with shiftable starting times, in which it is possible to delay the departure at the depot to better meet the ATC requirements. Finally, Kovacs, Golden, et al. (2015) introduce the generalized ConVRP, which (i) allows each customer to be served by a set of drivers (instead of a single one), (ii) features shiftable starting times and AM/PM time windows, and (iii) does not integrate the maximum time differences between arrivals on different days as hard constraints but penalizes them in the objective function. The proposed large neighborhood search (LNS) works on entire solutions instead of a template and currently represents the state-of-the-art heuristic also for the standard ConVRP. Multi-objective variants of the ConVRP are investigated by Kovacs, Parragh, and Hartl (2015), and Lian, Milburn, and Rardin (2016) and are addressed by means of multi-directional LNS and local search (LS), respectively. Feillet, Garaix, et al. (2014) present an ALNS to tackle a VRP with ATC in the context of transporting people with disabilities.

To the best of our knowledge, no exact approach to the ConVRP has been proposed yet. The only two papers addressing consistency considerations in an exact fashion are owed to Subramanyam and Gounaris (2016) and Subramanyam and Gounaris (2017) who study the consistent traveling-salesman problem (TSP), i.e., only one route per day is planned and routes must adhere to the ATC requirements. Subramanyam and Gounaris (2016) present three mixed-integer linear programming (MILP) formulations and several classes of valid inequalities that are embedded in a branch-and-cut framework; they are able to solve all instances with up to five planning periods and 25 customers to guaranteed optimality, but also some instances with up to 50 customers. Subramanyam and Gounaris (2017) decompose the problem into a sequence of single-period TSP with time windows and solve the consistent TSP via branch-and-bound; they are able to solve instances with up to five planning periods and 100 customers, outperforming the results of Subramanyam and Gounaris (2016), but a few instances with 33 customers remain open.

The major contribution of this work is twofold:

- We propose the first exact method for the ConVRP, which is able to solve small and medium-sized instances with up to five planning periods and 30 customers. Most of the state-of-the-art exact methods to solve VRPs are based on column generation (CG) applied to formulations where each variable represents a feasible route, and the pricing problem is solved via dynamic programming (DP). However, these methods cannot be directly extended to solve the ConVRP because the linear relaxation of route-based formulations provides weak lower bounds due to the interdependency between the daily routes, which is caused by the required ATC at customers. Therefore, we propose an exact method based on CG applied to a formulation in which each variable represents the set of routes assigned to a vehicle over the planning horizon. The exact method initially takes into account DC only, and addresses ATC at a later stage.
- As upper bounding procedure, we develop a LNS featuring suitable penalty mechanisms to deal with infeasible solutions and a repair procedure specifically designed to improve the ATC of solutions. Used as stand-alone heuristic, the LNS is able to significantly improve the solution quality

on benchmark instances from the literature compared to state-of-the-art heuristics. This is especially true for instances that assume small service frequencies, i.e., the probability that a customer requires service on a given day is relatively small, and that are therefore more difficult for template-based methods.

A minor contribution of this work is to provide a new compact formulation for the ConVRP that improves upon the compact formulation presented by Groër, Golden, and Wasil (2009). The new compact formulation contains fewer variables and constraints and is able to provide optimal solutions of small-sized instances in significantly shorter computing times.

The organization of this chapter is as follows. In Section 3.2, we formally define the ConVRP and the notation used throughout the chapter. In Section 3.3, we introduce the new compact formulation. Section 3.4 describes the proposed exact method, Section 3.5 the LNS. Section 3.6 is devoted to the computational results. Finally, conclusions and future research directions are summarized in Section 3.7.

3.2. Problem Definition

In the ConVRP, a set of customers $\mathcal{N}$ require delivery of a single commodity over a set $\mathcal{D}$ of days. The demand of customer $i \in \mathcal{N}$ on day $d \in \mathcal{D}$ is denoted by q_{id} (we assume that $q_{id} = 0$ if customer i does not require service on day d); $\mathcal{D}_i \subseteq \mathcal{D}$ indicates the subset of days on which customer i must be served (i.e., $\mathcal{D}_i = \{d \in \mathcal{D} \,|\, q_{id} > 0\}$), and $\mathcal{N}_d \subseteq \mathcal{N}$ is the subset of customers that must be served on day $d \in \mathcal{D}$ (i.e., $\mathcal{N}_d = \{i \in \mathcal{N} \,|\, q_{id} > 0\}$).

A homogeneous fleet $\mathcal{K}$ of capacitated vehicles based at a single depot, denoted by 0, is available to satisfy all customer requests. The capacity of the vehicles is given by Q. We indicate with $\mathcal{V}$ the set of customers plus the depot (i.e., $\mathcal{V} = \mathcal{N} \cup \{0\}$). The travel time t_{ij} between two locations $i, j \in \mathcal{V}$ is assumed to be deterministic and symmetric (i.e., $t_{ij} = t_{ji}$). The service time at customer $i \in \mathcal{N}$ on day $d \in \mathcal{D}$ is denoted by s_{id}. The maximum route duration of a vehicle on each day is T time units.

To respect DC, each customer must be served by the same driver/vehicle on every day of the planning horizon on which service is required. ATC is expressed by requiring that the service must take place roughly at the same time, so the difference between the latest and the earliest arrival time at each customer over the planning horizon cannot exceed the maximum allowed time difference L. As defined in Groër, Golden, and Wasil (2009), we assume that vehicles are not allowed to wait at a customer nor at the depot to meet ATC. The objective of the ConVRP is to find a set of routes for the vehicle fleet that minimizes the total vehicle operating time z, defined as the total travel and service time over the planning horizon.

3.3. A New Compact Formulation for the Consistent Vehicle-Routing Problem

To the best of our knowledge, the only formulation proposed in the literature for the ConVRP is owed to Groër, Golden, and Wasil (2009), who introduced a compact formulation for the problem. In the following, we present a new compact formulation that uses fewer variables than the one of Groër, Golden, and Wasil (2009), namely at most $|\mathcal{V}^2| \cdot |\mathcal{K}| \cdot |\mathcal{D}| + |\mathcal{V}| \cdot |\mathcal{K}| + |\mathcal{V}| \cdot |\mathcal{D}|$ instead of at most $|\mathcal{V}^2| \cdot |\mathcal{K}| \cdot$

$|\mathcal{D}| + |\mathcal{V}| \cdot |\mathcal{K}| \cdot |\mathcal{D}| + |\mathcal{V}| \cdot |\mathcal{D}|$. The computational efficiency of the two formulations is compared in Section 3.6.2, where we show that the new formulation is always better both in terms of the quality of the lower bound provided by its linear relaxation and in terms of the computing time to find the optimal solutions of small-sized instances.

We represent the ConVRP on a directed multi-graph $\mathcal{G} = (\mathcal{V}, \mathcal{A})$. The arc set $\mathcal{A}$ is defined as $\mathcal{A} = \cup_{d\in\mathcal{D}}\mathcal{A}_d$, where $\mathcal{A}_d = \{(0,j) \,|\, j \in \mathcal{N}_d\} \cup \{(i,0) \,|\, i \in \mathcal{N}_d\} \cup \{(i,j) \,|\, i,j \in \mathcal{N}_d : i \neq j\}$. Let $\hat{t}_{ijd}$ be the modified travel time associated with arc $(i,j) \in \mathcal{A}_d$, $d \in \mathcal{D}$, defined as $\hat{t}_{ijd} = t_{ij}$ if $i = 0$, and $\hat{t}_{ijd} = t_{ij} + s_{id}$ otherwise. By defining the following three sets of variables

- $x_{ijkd} \in \{0,1\}$: binary variable equal to 1 if arc $(i,j) \in \mathcal{A}_d$ is traversed by vehicle $k \in \mathcal{K}$ on day $d \in \mathcal{D}$ (0 otherwise),
- $y_{ik} \in \{0,1\}$: binary variable equal to 1 if customer $i \in \mathcal{N}$ is served by vehicle $k \in \mathcal{K}$ (0 otherwise),
- $b_{id} \in \mathbb{R}_+$: continuous variable indicating the arrival time at customer $i \in \mathcal{N}$ on day $d \in \mathcal{D}_i$,

the ConVRP can be formulated as follows:

$$z = \min \sum_{d\in\mathcal{D}} \sum_{k\in\mathcal{K}} \sum_{(i,j)\in\mathcal{A}_d} \hat{t}_{ijd} x_{ijkd} \tag{3.1}$$

$$\text{s.t.} \sum_{(0,j)\in\mathcal{A}_d} x_{0jkd} \leq 1 \qquad k \in \mathcal{K} \quad d \in \mathcal{D} \tag{3.2}$$

$$\sum_{k\in\mathcal{K}} y_{ik} = 1 \qquad i \in \mathcal{N} \tag{3.3}$$

$$\sum_{i\in\mathcal{N}_d} q_{id} y_{ik} \leq Q \qquad k \in \mathcal{K} \quad d \in \mathcal{D} \tag{3.4}$$

$$\sum_{(i,j)\in\mathcal{A}_d} x_{ijkd} = \sum_{(j,i)\in\mathcal{A}_d} x_{jikd} \qquad j \in \mathcal{N} \quad k \in \mathcal{K} \quad d \in \mathcal{D}_j \tag{3.5}$$

$$\sum_{(i,j)\in\mathcal{A}_d} x_{ijkd} = y_{jk} \qquad j \in \mathcal{N} \quad k \in \mathcal{K} \quad d \in \mathcal{D}_j \tag{3.6}$$

$$b_{id} \leq T - (T - \hat{t}_{0id}) \sum_{k\in\mathcal{K}} x_{0ikd} \qquad d \in \mathcal{D} \quad i \in \mathcal{N}_d \tag{3.7}$$

$$b_{id} + (\hat{t}_{ijd} + T) \sum_{k\in\mathcal{K}} x_{ijkd} + (T - \hat{t}_{jid}) \sum_{k\in\mathcal{K}} x_{jikd} \leq b_{jd} + T \qquad d \in \mathcal{D} \quad i,j \in \mathcal{N}_d : i \neq j \tag{3.8}$$

$$b_{id} - b_{id'} \leq L \qquad i \in \mathcal{N} \quad d,d' \in \mathcal{D}_i : d \neq d' \tag{3.9}$$

$$x_{ijkd} \in \{0,1\} \qquad k \in \mathcal{K} \quad d \in \mathcal{D} \quad (i,j) \in \mathcal{A}_d \tag{3.10}$$

$$y_{ik} \in \{0,1\} \qquad i \in \mathcal{N} \quad k \in \mathcal{K} \tag{3.11}$$

$$\hat{t}_{0id} \leq b_{id} \leq T - \hat{t}_{i0d} \qquad i \in \mathcal{N} \quad d \in \mathcal{D}_i \tag{3.12}$$

The objective function aims at minimizing the total operating time. Constraints (3.2) guarantee that each vehicle performs at most one route on each day. Constraints (3.3) assign each customer to exactly one vehicle. Constraints (3.4) guarantee that the capacity of the vehicles is not exceeded. Constraints (3.5) are flow conservation constraints. Constraints (3.6) link $\boldsymbol{x}$ and $\boldsymbol{y}$ variables to ensure DC. Constraints (3.7) ensure that the arrival time at a customer i is not greater than the travel time from the depot to i if arc $(0,i)$ is traversed. Constraints (3.8) link $\boldsymbol{x}$ and $\boldsymbol{b}$ variables to set the arrival times based on the traversed

arcs and prevent subtours. Constraints (3.9) model ATC requirements. Constraints (3.10)–(3.12) define the range of the decision variables and ensure that maximum route duration is respected.

Adding subtour elimination constraints (SECs) to formulation (3.1)–(3.12) allows to significantly decrease the computing time to find an optimal solution of small ConVRP instances. Therefore, in the computational experiments in Section 3.6.2, the following set of generalized SECs (GSECs) are added to our formulation:

$$\sum_{(i,j)\in\mathcal{A}_d\,:\,i\in\mathcal{V}\setminus S,\,j\in S} Qx_{ijkd} \geq \sum_{i\in S} q_{id}y_{id} \qquad k\in\mathcal{K} \quad d\in\mathcal{D} \quad S\subset\mathcal{N}_d : |S|\geq 2. \tag{3.13}$$

Constraints (3.13) are clearly redundant when integrality constraints (3.10) and (3.11) are present, but they strengthen the linear relaxation of formulation (3.1)–(3.12).

3.4. An Exact Method for the ConVRP

In this section, we introduce a formulation of the ConVRP with exponentially many variables. Two lower bounds based on this formulation are presented in Section 3.4.1. An outline of the proposed exact method is provided in Section 3.4.2, and the different steps of the algorithm are detailed in Sections 3.4.3–3.4.6.

Let Ω be the set of all possible subsets of customers (hereafter called *clusters*) that can be served by a single vehicle over the planning horizon without violating capacity, route duration, DC, and ATC constraints. Moreover, let g_C be the minimum cost to serve cluster $C\in\Omega$ with a single vehicle, i.e., g_C is the sum of the costs of the least-cost routes satisfying the listed constraints performed by the vehicle on each day $d\in\mathcal{D}$ to serve the customers $C\cap\mathcal{N}_d$. By introducing a binary decision variable ξ_C that is equal to 1 if cluster $C\in\Omega$ is assigned to a vehicle (0 otherwise), the ConVRP can be formulated as the following set-partitioning (SP) problem:

$$z(\text{SP}) = \min \sum_{C\in\Omega} g_C\xi_C \tag{3.14}$$

$$\sum_{C\in\Omega\,:\,i\in C} \xi_C = 1 \qquad i\in\mathcal{N} \tag{3.15}$$

$$\sum_{C\in\Omega} \xi_C \leq |\mathcal{K}| \tag{3.16}$$

$$\xi_C\in\{0,1\} \qquad C\in\Omega \tag{3.17}$$

The objective function (3.14) asks for minimizing the cost of the selected clusters. Constraints (3.15) ensure that each customer belongs to exactly one selected cluster. Constraint (3.16) ensures that at most $|\mathcal{K}|$ clusters are selected. Constraints (3.17) define variables $\boldsymbol{\xi}$ as binary. Each feasible solution of problem (3.14)–(3.17) is a subset of clusters of Ω. The ConVRP solution corresponding to that subset of clusters consists of the least-cost routes associated with costs g_C.

Note that SP contains exponentially many variables, so a CG approach must be applied to find an optimal ConVRP solution when solving problem SP. For many variants of the VRP, the state-of-the-art exact

methods are based on CG (see, e.g., Jepsen et al., 2008; Baldacci, Mingozzi, and Roberti, 2011; Dabia et al., 2013; Contardo and Martinelli, 2014; Pecin, Contardo, et al., 2017; Pecin, Pessoa, et al., 2017). In all of these methods, the pricing problem is solved via DP. Unfortunately, DP cannot directly be applied to price out clusters in a CG approach based on formulation SP because of the number and the range of the state variables that are needed, and the weakness of the dominance rules that can be applied.

However, it is possible to derive some tight lower bounds from formulation SP (see Section 3.4.1), which are used by our exact method to find an optimal ConVRP solution. In the remainder of the chapter, we refer to our exact method as cluster column generation (CCG).

3.4.1. Lower Bounds based on Formulation SP

Recall that, for a given cluster $C \in \Omega$, the minimum cost g_C to serve all customers in C with a single vehicle throughout the planning horizon is given by the sum of the costs of the routes that on each day $d \in \mathcal{D}$ serve all customers $\mathcal{N}_d \cap C$ without violating capacity, route duration, and ATC constraints. Because of the ATC requirement, the routes performed by a vehicle on the individual days do not necessarily correspond to the least-cost routes (i.e., the TSPs) to serve all customers in the cluster.

Let $\hat{\Omega} \supseteq \Omega$ be the set of all possible clusters that can be served by a single vehicle over the planning horizon without violating capacity and route duration constraints. Let $\hat{g}_C$ be the cost to serve all customers of the cluster $C \in \hat{\Omega}$ with a single vehicle so that capacity and route duration constraints are respected, but ATC constraints can be violated. It is easy to observe that cost $\hat{g}_C$ of cluster $C \in \hat{\Omega}$ is given by the sum of the cost of the TSPs to serve customers $C \cap \mathcal{N}_d$ on each day $d \in \mathcal{D}$.

A valid lower bound to the ConVRP is therefore given by the optimal value, $z(\text{LP}_0)$, of the following linear problem, hereafter called LP_0:

$$z(\text{LP}_0) = \min \sum_{C \in \hat{\Omega}} \hat{g}_C \xi_C \tag{3.18}$$

$$\text{s.t.} \sum_{C \in \hat{\Omega} : i \in C} \xi_C = 1 \qquad i \in \mathcal{N} \tag{3.19}$$

$$\sum_{C \in \hat{\Omega}} \xi_C \leq |\mathcal{K}| \tag{3.20}$$

$$\xi_C \geq 0 \qquad C \in \hat{\Omega} \tag{3.21}$$

The lower bound $z(\text{LP}_0)$ can be improved by adding the following valid inequalities:

- *Minimum number of vehicles:*

$$\sum_{C \in \hat{\Omega}} \xi_C \geq \mathcal{K}_{\min}, \tag{3.22}$$

  where $\mathcal{K}_{\min}$ is a lower bound on the minimum number of vehicles used in any optimal solution of the ConVRP.

- *Subset-row (SR) inequalities* that state that, for each triplet of customers $\{i, j, h\} \subseteq \mathcal{N}$, no more than one of the clusters serving at least two of the three customers $\{i, j, h\}$ can be selected:

$$\sum_{C \in \hat{\Omega} : |C \cap \{i,j,h\}| \geq 2} \xi_C \leq 1 \qquad \{i, j, h\} \subseteq \mathcal{N} : i \neq j \neq h. \tag{3.23}$$

  Inequalities (3.23) are a special case of the well-known SR inequalities introduced by Jepsen et al. (2008) and can be separated by complete enumeration.

In the following, we denote by $z(\text{LP}_1)$ the optimal value of problem LP_0 plus inequalities (3.22), and by $z(\text{LP}_2)$ the optimal value of problem LP_0 plus inequalities (3.22) and (3.23). Moreover, let $\alpha_i \in \mathbb{R}$ be the dual variable associated with constraint (3.19) of customer $i \in \mathcal{N}$, $\alpha_0 \in \mathbb{R}_-$ the dual variable associated with constraint (3.20), $\beta \in \mathbb{R}_+$ the dual variable associated with constraint (3.22), and $\gamma_{ijh} \in \mathbb{R}_-$ the dual variable associated with constraint (3.23) of the triplet of customers $\{i, j, h\} \subseteq \mathcal{N}$.

3.4.2. Overview of CCG

CCG consists of four main steps that can be outlined as follows:

Step 1: Initialization. An upper bound UB to the ConVRP and a lower bound $\mathcal{K}_{\min}$ to the number of vehicles in any optimal ConVRP solution are computed. The upper bound UB is computed by running 10 times the LNS described in Section 3.5.3, each time with a limit of 25000 iterations (i.e., $\eta_{total} = 25000$). The lower bound $\mathcal{K}_{\min}$ on the minimum number of vehicles is computed by using a MILP (see Section 3.4.3).

Step 2: Generate the set of clusters $\hat{\Omega}$. The goal of this step is to generate the whole set of clusters $\hat{\Omega}$ (i.e., clusters that can be served by a single vehicle over the planning horizon without violating capacity and route duration constraints). As described in Section 3.4.4, this can be done via DP. If it is not possible to generate the whole set $\hat{\Omega}$, then CCG stops without providing a proven optimal solution to the ConVRP.

Step 3: Remove non-optimal clusters from $\hat{\Omega}$. This step aims at removing clusters that cannot belong to an optimal ConVRP solution from the set of clusters $\hat{\Omega}$ by iteratively computing optimal dual solutions of problems LP_1 and LP_2.

First, we use CG as described in Section 3.4.5 to compute an optimal LP_1 dual solution $(\boldsymbol{\alpha}^*, \boldsymbol{\beta}^*)$ of cost $z(\text{LP}_1)$. Any cluster having reduced cost, w.r.t. $(\boldsymbol{\alpha}^*, \boldsymbol{\beta}^*)$, greater than the corresponding gap (i.e., $\text{UB} - z(\text{LP}_1)$) is then removed from the set $\hat{\Omega}$ because it cannot belong to an optimal ConVRP solution. Second, using CG as described in Section 3.4.5, an optimal LP_2 dual solution $(\boldsymbol{\alpha}^*, \boldsymbol{\beta}^*, \boldsymbol{\gamma}^*)$ of cost $z(\text{LP}_2)$ is computed. Any cluster having reduced cost, w.r.t. $(\boldsymbol{\alpha}^*, \boldsymbol{\beta}^*, \boldsymbol{\gamma}^*)$, greater than the corresponding gap (i.e., $\text{UB} - z(\text{LP}_2)$) is then removed from the set $\hat{\Omega}$.

Step 3 is iterated as long as the set of clusters $\hat{\Omega}$ is reduced by using the optimal dual solutions $(\boldsymbol{\alpha}^*, \boldsymbol{\beta}^*)$ and $(\boldsymbol{\alpha}^*, \boldsymbol{\beta}^*, \boldsymbol{\gamma}^*)$.

Step 4: Find an optimal ConVRP solution. Let $\overline{\Omega} \subseteq \Omega$ be a subset of clusters C for which cost g_C is known, such that $\overline{\Omega} \cap \hat{\Omega} = \varnothing$. The optimal value of the following problem $\overline{\text{SP}}$ provides a valid lower bound to the ConVRP:

$$z(\overline{\text{SP}}) = \min \sum_{C \in \overline{\Omega}} g_C \xi_C + \sum_{C \in \hat{\Omega}} \hat{g}_C \xi_C \tag{3.24}$$

$$\sum_{C \in \overline{\Omega} \cup \hat{\Omega} \,:\, i \in C} \xi_C = 1 \qquad i \in \mathcal{N} \tag{3.25}$$

$$\sum_{C \in \overline{\Omega} \cup \hat{\Omega}} \xi_C \leq |\mathcal{K}| \tag{3.26}$$

$$\xi_C \in \{0,1\} \qquad C \in \overline{\Omega} \cup \hat{\Omega} \tag{3.27}$$

The objective function (3.24) aims at minimizing the total cost of the clusters selected from the two sets $\overline{\Omega}$ and $\hat{\Omega}$. Constraints (3.25) ensure that each customer belongs to exactly one of the selected clusters. Constraint (3.26) guarantees that at most $|\mathcal{K}|$ clusters are selected. Constraints (3.27) are integrality constraints.

Let $\Omega^* \subseteq \overline{\Omega} \cup \hat{\Omega}$ be the set of clusters in an optimal solution of $\overline{\text{SP}}$. We can observe that whenever $\Omega^* \subseteq \overline{\Omega}$, then the clusters of the set Ω^* represent an optimal ConVRP solution because they take into account ATC and by definition DC.

To find an optimal ConVRP solution, the last step of CCG consists of iteratively solving $\overline{\text{SP}}$ with a general purpose MILP solver until an optimal ConVRP is found while changing the sets of clusters $\overline{\Omega}$ and $\hat{\Omega}$. At each iteration, $\overline{\text{SP}}$ is solved, the cost g_{C^*} of one of the clusters $C^* \in \Omega^* \cap \hat{\Omega}$ is computed (see Section 3.4.6), and cluster C^* is removed from $\hat{\Omega}$. If $z(\overline{\text{SP}}) + g_{C^*} - \hat{g}_{C^*} \leq \text{UB}$, then cluster C^* is also added to $\overline{\Omega}$ because it can be part of an optimal ConVRP solution of cost between $z(\overline{\text{SP}})$ and UB. At the first iteration, the set $\hat{\Omega}$ is inherited from Step 3, and the set $\overline{\Omega}$ is empty. Note that it may not be possible to serve a cluster C^* with a single vehicle while adhering to ATC; if so, cluster C^* is obviously not added to $\overline{\Omega}$.

Because the complexity of computing cost g_{C^*} for a given cluster C^* also depends on the number of customers in the cluster C^*, at each iteration the selected cluster $C^* \in \Omega^* \cap \hat{\Omega}$ is the one with the smallest number of customers.

3.4.3. Computing $\mathcal{K}_{\text{min}}$ in Step 1

The lower bound $\mathcal{K}_{\text{min}}$ on the minimum number of vehicles in any ConVRP solution is computed by solving the following MILP. Let $\varphi_{ik} \in \{0,1\}$ be a binary variable equal to 1 if customer $i \in \mathcal{N}$ is assigned to vehicle $k \in \mathcal{K}$ (0 otherwise), and let $\vartheta_k \in \{0,1\}$ be a binary variable equal to 1 if vehicle $k \in \mathcal{K}$ is used (0 otherwise). Then, $\mathcal{K}_{\text{min}}$ can be computed as:

$$\mathcal{K}_{\text{min}} = \min \sum_{k \in \mathcal{K}} \vartheta_k \tag{3.28}$$

$$\text{s.t.} \sum_{k \in \mathcal{K}} \varphi_{ik} = 1 \qquad i \in \mathcal{N} \tag{3.29}$$

$$\sum_{i \in \mathcal{N}_d} q_{id} \varphi_{ik} \leq Q \vartheta_k \qquad d \in \mathcal{D} \quad k \in \mathcal{K} \tag{3.30}$$

$$\varphi_{ik} \in \{0,1\} \qquad i \in \mathcal{N} \quad k \in \mathcal{K} \tag{3.31}$$

$$\vartheta_k \in \{0,1\} \qquad k \in \mathcal{K} \tag{3.32}$$

The objective function (3.28) aims at minimizing the number of vehicles used. Constraints (3.29) ensure that each customer $i \in \mathcal{N}$ is assigned to exactly one vehicle. Constraints (3.30) guarantee that the capacity of each vehicle $k \in \mathcal{K}$ is respected on each day $d \in \mathcal{D}$. The range of the decision variables is defined by constraints (3.31) and (3.32).

For small and medium-sized ConVRP instances, problem (3.28)–(3.32) can be solved to optimality with a general purpose MILP solver in less than a second of computing time.

3.4.4. Generating the Set $\hat{\Omega}$ in Step 2

To generate the set $\hat{\Omega}$, we use a simple DP recursion that enumerates all feasible routes Φ_d for each day $d \in \mathcal{D}$ of the planning horizon, and then clusters are generated by combining the routes of the sets Φ_d. A route is feasible if the vehicle capacity Q is not exceeded and its duration does not exceed the maximum route duration T.

Let $f_d(S,i)$ be the cost of the min-cost path starting from the depot, visiting all customers of the set $S \subseteq \mathcal{N}_d$, and ending at customer $i \in S$ on day $d \in \mathcal{D}$. Functions $f_d(S,i)$ for each day $d \in \mathcal{D}$ can be computed via DP as follows. We initialize $f_d(\{i\},i) = \hat{t}_{0id}$ for each $i \in \mathcal{N}_d$. The recursion for computing functions $f_d(S,i)$ for each subset of customers $S \subseteq \mathcal{N}_d$ and each customer $i \in S$ is:

$$f_d(S,i) = \min_{j \in S \setminus \{i\}} \{f_d(S \setminus \{i\}, j) + \hat{t}_{jid}\}.$$

Because routes have to respect the vehicle capacity Q and the maximum route duration T, there is no need to propagate functions $f_d(S,i)$ such that either $f_d(S,i) + \hat{t}_{i0d} > T$ or $\sum_{j \in S} q_{jd} > Q$. For the sake of simplicity, we assume, in the remainder of the section, that $f_d(S,i) = \infty$ if function $f_d(S,i)$ is not computed because of constraint violations.

The cost of the least-cost route to serve the subset of customers $S \subseteq \mathcal{N}_d$ on day $d \in \mathcal{D}$ is given by $\min_{i \in S}\{f_d(S,i) + \hat{t}_{i0d}\}$. From functions $f_d(S,i)$, it is possible to generate the set $\hat{\Omega}$. In particular, cluster $C \subseteq \mathcal{N}$ belongs to the set $\hat{\Omega}$ if $\min_{i \in C \cap \mathcal{N}_d} \left\{f_d(C \cap \mathcal{N}_d, i) + \hat{t}_{i0d}\right\} \leq T$ for each day $d \in \mathcal{D}$; otherwise, cluster C does not belong to the set $\hat{\Omega}$.

The cost $\hat{g}_C$ of cluster $C \in \hat{\Omega}$ is given by:

$$\hat{g}_C = \sum_{d \in \mathcal{D}} \left(\min_{i \in C \cap \mathcal{N}_d} \left\{ f_d(C \cap \mathcal{N}_d, i) + \hat{t}_{i0d} \right\} \right).$$

We can observe that $\hat{g}_C$ is the cost to serve all customers of the set C with a single vehicle over the planning horizon without necessarily satisfying the ATC constraints.

3.4.5. Computing Lower Bounds $z(\mathrm{LP}_1)$ and $z(\mathrm{LP}_2)$ in Step 3

In principle, because all variables are generated a-priori, we could simply solve LP_1 as it is. However, it is computationally convenient to apply a simple CG algorithm that solves LP_1 by starting from a small set of clusters (we use a dummy cluster that contains all customers and has cost equal to UB), and then iteratively adding the 100 clusters with the most negative reduced cost at a time until all clusters of the set $\hat{\Omega}$ have non-negative reduced cost w.r.t. the dual solution $(\boldsymbol{\alpha}, \boldsymbol{\beta})$ of problem LP_1. The reduced cost $\hat{g}_C(\boldsymbol{\alpha}, \boldsymbol{\beta})$ of cluster $C \in \hat{\Omega}$ is computed as $\hat{g}_C(\boldsymbol{\alpha}, \boldsymbol{\beta}) = \hat{g}_C - \alpha_0 - \sum_{i \in C} \alpha_i - \beta$.

Once an optimal dual solution $(\boldsymbol{\alpha}^*, \boldsymbol{\beta}^*)$ of cost $z(\mathrm{LP}_1)$ is found, all clusters $C \in \hat{\Omega}$ having reduced cost $\hat{g}_C(\boldsymbol{\alpha}^*, \boldsymbol{\beta}^*)$ greater than the gap left (i.e., UB $- z(\mathrm{LP}_1)$) can be removed from the set $\hat{\Omega}$ because they cannot belong to an optimal ConVRP solution.

A similar CG procedure is applied to compute $z(\mathrm{LP}_2)$. At the beginning, the master problem contains no SR inequalities (3.23) and just a dummy cluster; then, at each iteration, the 100 clusters having the most negative reduced cost w.r.t. the dual solution $(\boldsymbol{\alpha}, \boldsymbol{\beta}, \boldsymbol{\gamma})$ of LP_2 are added along with the most violated SR inequality (3.23).

Once an optimal dual solution $(\boldsymbol{\alpha}^*, \boldsymbol{\beta}^*, \boldsymbol{\gamma}^*)$ of cost $z(\mathrm{LP}_2)$ is found, all clusters $C \in \hat{\Omega}$ having reduced cost $\hat{g}_C(\boldsymbol{\alpha}^*, \boldsymbol{\beta}^*, \boldsymbol{\gamma}^*)$ greater than the gap left (i.e., UB $- z(\mathrm{LP}_2)$) can be removed from the set $\hat{\Omega}$ because they cannot belong to an optimal ConVRP solution. The reduced cost $\hat{g}_C(\boldsymbol{\alpha}^*, \boldsymbol{\beta}^*, \boldsymbol{\gamma}^*)$ of cluster $C \in \hat{\Omega}$ is computed as:

$$\hat{g}_C(\boldsymbol{\alpha}^*, \boldsymbol{\beta}^*, \boldsymbol{\gamma}^*) = \hat{g}_C - \alpha_0^* - \sum_{i \in C} \alpha_i^* - \beta^* - \sum_{\substack{\{i,j,h\} \in \mathcal{N}: \\ |\{i,j,h\} \cap C| \geq 2}} \gamma_{ijh}^*.$$

3.4.6. Computing Cost $\mathrm{g_C}$ in Step 4

The problem of computing cost g_C for a given cluster $C \in \Omega^* \cap \hat{\Omega}$ can be represented on a directed multi-graph $\mathcal{G}(C) = (\mathcal{V}(C), \mathcal{A}(C))$. The vertex set is defined as $\mathcal{V}(C) = C \cup \{0\}$, and the arc set $\mathcal{A}(C)$ is defined as $\mathcal{A}(C) = \cup_{d \in \mathcal{D}} \mathcal{A}_d(C)$, where $\mathcal{A}_d(C) = \{(0, j) \,|\, j \in \mathcal{N}_d \cap C\} \cup \{(i, 0) \,|\, i \in \mathcal{N}_d \cap C\} \cup \{(i, j) \,|\, i, j \in \mathcal{N}_d \cap C : i \neq j\}$. Let us define the following two sets of variables:

- $x_{ijd} \in \{0, 1\}$: binary variable equal to 1 if arc $(i, j) \in \mathcal{A}_d(C)$ is used on day $d \in \mathcal{D}$ (0 otherwise);
- $b_{id} \in \mathbb{R}_+$: continuous variable indicating the arrival time at vertex $i \in C$ on day $d \in \mathcal{D}_i$.

Then, the cost g_C of cluster C corresponds to the optimal value of the following MILP:

$$g_C = \min \sum_{d \in \mathcal{D}} \sum_{(i,j) \in \mathcal{A}_d(C)} \hat{t}_{ijd} x_{ijd} \tag{3.33}$$

$$\text{s.t.} \sum_{(0,j) \in \mathcal{A}_d(C)} x_{0jd} \leq 1 \qquad d \in \mathcal{D} \tag{3.34}$$

$$\sum_{(i,j) \in \mathcal{A}_d(C)} x_{ijd} = 1 \qquad j \in C \quad d \in \mathcal{D}_j \tag{3.35}$$

$$\sum_{(j,i)\in\mathcal{A}_d(C)} x_{jid} = 1 \qquad j \in C \quad d \in \mathcal{D}_j \tag{3.36}$$

$$b_{id} \leq T - (T - \hat{t}_{0id})x_{0id} \qquad d \in \mathcal{D} \quad i \in \mathcal{N}_d \cap C \tag{3.37}$$

$$b_{id} + (\hat{t}_{ijd} + T)x_{ijd} + (T - \hat{t}_{jid})x_{jid} \leq b_{jd} + T \qquad d \in \mathcal{D} \quad i,j \in \mathcal{N}_d \cap C : i \neq j \tag{3.38}$$

$$b_{id} - b_{id'} \leq L \qquad i \in C \quad d,d' \in \mathcal{D}_i : d \neq d' \tag{3.39}$$

$$x_{ijd} \in \{0,1\} \qquad d \in \mathcal{D} \quad (i,j) \in \mathcal{A}_d(C) \tag{3.40}$$

$$\hat{t}_{0id} \leq b_{id} \leq T - \hat{t}_{i0d} \qquad i \in C \quad d \in \mathcal{D}_i \tag{3.41}$$

The objective function (3.33) aims at minimizing the total operating time to visit all customers of the set C. Constraints (3.34) ensure that the vehicle performs at most one route on each day of the planning horizon. Constraints (3.35) and (3.36) are in-degree and out-degree constraints, respectively. Constraints (3.37) along with constraints (3.41) properly set the arrival time at the first customer of each route of each day. Constraints (3.38) link variables $\boldsymbol{x}$ and $\boldsymbol{z}$ to update the arrival times at the customers depending on the traversed arcs and prevent subtours. Constraints (3.39) guarantee the ATC of the routes. Constraints (3.40) and (3.41) define the range of the decision variables.

We solve problem (3.33)–(3.41) by using a general purpose MILP solver. We also add, in a cutting-plane fashion, the well-known SECs defined as:

$$\sum_{\substack{(i,j)\in\mathcal{A}_d(C):\\ i\in C, j\in\mathcal{V}(C)\setminus C}} x_{ijd} \geq 1 \qquad d \in \mathcal{D} \quad S \subseteq \mathcal{N}_d \cap C \,:\, |S| \geq 2. \tag{3.42}$$

Because the number of customers in the clusters is usually limited to 10–15 customers, it is possible to enumerate all SECs (3.42) a-priori and let the MILP solver add them in a cutting-plane fashion.

As mentioned in Section 3.4.2, problem (3.33)–(3.41) does not necessarily have feasible solutions.

3.5. Large Neighborhood Search for the ConVRP

We propose a LNS for the ConVRP that is used to obtain upper bounds within our exact method, but that can also be used as stand-alone metaheuristic approach. The LNS is enhanced by several components: (i) suitable penalty mechanisms to deal with infeasible solutions, (ii) a repair procedure that is applied to improve the ATC, and (iii) regularly solving a set-partitioning problem using the clusters previously found by the search to improve the solution quality.

In the following description, we represent a solution $\mathcal{S}$ as a set of routes $\{r_{kd} \mid k \in \mathcal{K}, d \in \mathcal{D}\}$. A route $r_{kd} = \langle v_0 = 0, v_1, \ldots, v_{n_{kd}}, v_{n_{kd}+1} = 0\rangle$ is given as a sequence of vertices that starts and ends at the depot vertex 0 and visits a set $\mathcal{N}(r_{kd})$ of n_{kd} customer vertices in between.

In the description of the algorithm, we directly report the utilized values of the algorithm parameters. The latter were determined in experimental fashion during the development of our algorithm, and no systematic fine-tuning was carried out. We found that our algorithm is quite stable with regards to

changes in the parameter values as long as the new values stay within the magnitude of the values of the current setting.

An overview of the algorithm, which we call LNS with set partitioning (LNS-SP), is given in Figure 3.1. First, LNS-SP generates a feasible initial solution $\mathcal{S}_c$ with a savings algorithm that respects the consistency requirements of the ConVRP (see Section 3.5.1). Then, the initial solution is improved in 25000 iterations of LNS, including our specialized component for improving the ATC (Section 3.5.3). Here, infeasible solutions are allowed and are evaluated with a generalized objective function (Section 3.5.2). The set-partitioning problem for feasible clusters is described in Section 3.5.4. Finally, every 250 iterations without improvement of $\mathcal{S}_{best}$, we reset $\mathcal{S}_c$ to $\mathcal{S}_{best}$.

1: $\eta \leftarrow 1$ {Set iteration counter.}
2: $\mathcal{S}_c \leftarrow$ generateInitialSolution
3: **while** $\eta \leq 25000$ **do**
   {Perform large neighborhood search.}
4: $\delta \leftarrow$ drawNumberOfCustomersToRemove
5: $\mathcal{S}_t \leftarrow$ insertCustomers(removeCustomers($\mathcal{S}_c$, δ))
6: $\mathcal{S}_t \leftarrow$ applyATCImprovement ($\mathcal{S}_t$)
7: updatePenalties($\mathcal{S}_t$)
8: **if** acceptSA($\mathcal{S}_c, \mathcal{S}_t$) **then**
9: $\mathcal{S}_c \leftarrow \mathcal{S}_t$
10: **end if**
11: **if** $\mathcal{S}_t$ improves $\mathcal{S}_{best}$ **then**
12: $\mathcal{S}_{best} \leftarrow \mathcal{S}_t$
13: **end if**
   {Perform set partitioning.}
14: $\Omega^{LNS} \leftarrow$ addClusters($\mathcal{S}_t$)
15: **if** 5000 iterations have passed since last set partitioning **then**
16: $\mathcal{S}_{best} \leftarrow$ solveSetPartitioning($\mathcal{S}_{best}, \Omega^{LNS}$)
17: **end if**
18: **if** solution has not improved for 250 iterations **then**
19: $\mathcal{S}_c \leftarrow \mathcal{S}_{best}$
20: **end if**
21: $\eta \leftarrow \eta + 1$
22: **end while**
23: **return** $\mathcal{S}_{best}$

Figure 3.1.: Overview of the LNS-SP algorithm.

3.5.1. Modified Savings Algorithm

To generate an initial solution, we adapt the savings algorithm of Clarke and Wright (1964) to handle the multi-day horizon and the DC and ATC constraints of the ConVRP. Before the merge step of the savings algorithm is applied, the routes to be merged are assigned to two different vehicles. Consequently, for the ConVRP, merging two routes on a single day entails that the routes of the two respective vehicles are also merged on all other days.

In detail, our procedure works as follows: At the beginning, each request is served by a dedicated route, and if a customer requests service on multiple days of the planning horizon, the corresponding routes are all assigned to the same vehicle. In the next step, we evaluate for each pair of vehicles, how the solution

changes if the routes of the two vehicles are merged on all days of the planning horizon. To this end, we sum up the individual savings of merging both routes on each day on which both vehicles provide service. We limit the evaluation to the two cases where either (i) all routes of the first vehicle are served before all routes of the second vehicle or (ii) vice versa, but we do not allow combinations hereof. This makes sense from a practical viewpoint because it entails that requests of a customer are served at about the same time on each day. The cases that we do not evaluate are likely to result in routes on which a customer is served early on one day and late on another day. Finally, we perform the merge that results in the largest total saving, but only if it leads to a feasible solution, i.e., no resulting route exceeds the vehicle capacity or maximum route duration, and the vehicle does not violate the ATC constraint. After each step, we remove one of the two vehicles whose routes were merged.

Figure 3.2 shows an example of four iterations of the modified savings algorithm for a two-day problem. In the beginning, each request is served by a dedicated route. Customer requests occurring on both days are depicted in black, and single-day requests are depicted in gray. In the first step, the routes serving requests of customers A and B are merged on both days, and the vehicle serving customer B is removed. Arcs to be added are depicted as dashed lines, arcs to be removed as dotted lines. In the second step, the routes serving requests of customers G and D are merged on the second day, and we keep the route serving D on the first day (because G does not request service on the first day). The third merge operation affects both days, whereas the fourth merge operation is limited to the first day. The final solution uses three vehicles to serve all customer requests.

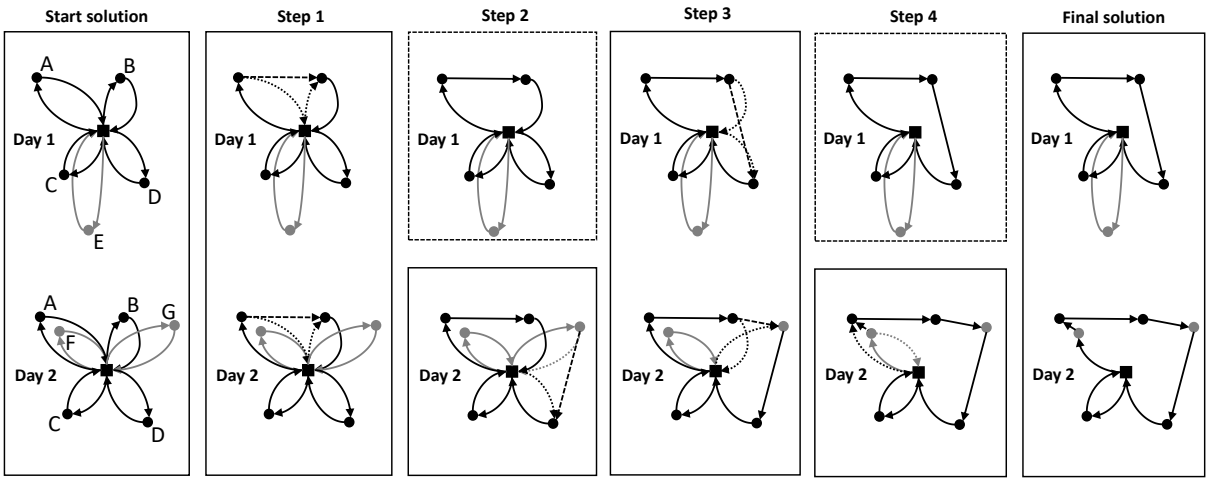

Figure 3.2.: Four steps of our modified savings algorithm on a two-day example problem.

3.5.2. Generalized Objective Function and Penalty Calculation

We allow infeasible solutions during the LNS and evaluate a solution $\mathcal{S}$ using the following generalized objective function that penalizes constraint violations using an adaptive mechanism:

$$z_{gen}(\mathcal{S}) = z(\mathcal{S}) + \sigma_{cap} \cdot G_{cap}(\mathcal{S}) + \sigma_{dur} \cdot G_{dur}(\mathcal{S}) + \sigma_{atc} \cdot G_{atc}(\mathcal{S}),$$

where $z(\mathcal{S})$ denotes the objective value as defined in Equation (3.1), $G_{cap}(\mathcal{S})$ the capacity violation, $G_{dur}(\mathcal{S})$ the route duration violation, and $G_{atc}(\mathcal{S})$ the ATC violation of solution $\mathcal{S}$, and σ_{cap}, σ_{dur}, and σ_{atc} are the respective penalty factors.

The constraint violations are determined as follows:

- *Vehicle capacity violation:* $G_{cap}(\mathcal{S}) = \sum_{k\in\mathcal{K}} \sum_{d\in\mathcal{D}} \max(0, \sum_{i\in\mathcal{N}(r_{kd})} q_{id} - Q)$,
- *Route duration violation:* $G_{dur}(\mathcal{S}) = \sum_{k\in\mathcal{K}} \sum_{d\in\mathcal{D}} \max(0, \max_{i\in\mathcal{N}(r_{kd})}(b_{id} + \hat{t}_{i0d}) - T)$,
- *ATC violation:* $G_{atc}(\mathcal{S}) = \sum_{i\in\mathcal{N}} \sum_{d\in\mathcal{D}_i} \sum_{d'\in\mathcal{D}_i} \max(0, |b_{id} - b_{id'}| - L)$.

All penalty factors are initialized to a value of 10 and are restricted to the interval $[0.01, 1000]$. In every iteration of LNS-SP, the penalty factors are multiplied or divided by a factor of 1.05 based on the following rules:

- Factor σ_{cap} is increased if $G_{cap}(\mathcal{S}) > 0$ and decreased otherwise.
- We link the behavior of the penalty factors σ_{dur} and σ_{atc} because $G_{atc}(\mathcal{S})$ and $G_{dur}(\mathcal{S})$ are strongly interdependent in our algorithm. This is due to our procedure for improving the ATC (Section 3.5.3.2), which often reduces violations of the ATC at the expense of generating longer routes that are likely to violate the route duration constraint. Therefore, σ_{dur} is increased if $G_{dur}(\mathcal{S}) > 0 \wedge G_{atc}(\mathcal{S}) = 0$, decreased if $G_{dur}(\mathcal{S}) = 0 \wedge G_{atc}(\mathcal{S}) = 0$, and kept at its current value otherwise. Analogously, σ_{atc} is increased if $G_{atc}(\mathcal{S}) > 0 \wedge G_{dur}(\mathcal{S}) = 0$, decreased if $G_{atc}(\mathcal{S}) = 0 \wedge G_{dur}(\mathcal{S}) = 0$, and kept fixed otherwise.

3.5.3. Large Neighborhood Search Component

LNS, originally introduced by Shaw (1998), is a metaheuristic principle that aims at iteratively improving an initial solution by first removing a larger part of the solution (using a set of so-called removal operators) and then reinserting the removed solution components (using so-called insertion operators). In recent years, LNS has successfully been applied to many variants of the VRP (see, e.g., Ropke and Pisinger, 2006b; Masson, Lehuédé, and Péton, 2013; Adulyasak, Cordeau, and Jans, 2014).

In each iteration of LNS-SP, the number of customers to be removed is randomly drawn from the interval $\delta = rand([0.05, 0.2]) \cdot \min(150, |\mathcal{N}|)$. Removal, insertion and subsequent ATC improvement (see Sections 3.5.3.1 and 3.5.3.2) create a tentative solution $\mathcal{S}_t$, which may be infeasible because LNS-SP always generates a complete solution and does not leave customer requests in a so-called request bank as often done in LNS. The decision whether to accept $\mathcal{S}_t$ or to keep the current solution $\mathcal{S}_c$ is based on a simulated annealing (SA) criterion (Section 3.5.3.3).

3.5.3.1. Removal and Insertion Operators

In each iteration, LNS-SP randomly selects one of the removal and one of the insertion operators with uniform probability. Removal/insertion of a customer implies the removal/insertion of all service requests of this customer on all days of the planning horizon. The following removal operators are used:

Random removal removes δ arbitrarily selected customers.

Worst removal was introduced by Ropke and Pisinger (2006b) to remove vertices that are served at undesirable positions in the routes. We propose a modified version of the operator that is (i) not randomized, and (ii) adapted to the ConVRP. Let $\mathcal{S}^{-i}$ denote a solution where customer i is removed on all days. We define the following measure κ_i to determine which customers should be removed from the solution:

$$\kappa_i = (z(\mathcal{S}^{-i}) - z(\mathcal{S}))/|\mathcal{D}_i| + \sigma_{cap} \cdot (G_{cap}(\mathcal{S}^{-i}) - G_{cap}(\mathcal{S})) + \sigma_{dur} \cdot ((G_{dur}(\mathcal{S}^{-i}) - G_{dur}(\mathcal{S})).$$

Note that we divide the reduction in total operating time by the number of days on which customer i requires service, i.e., we base the decision on the average operating time reduction per request. Otherwise, the selection would be biased towards customers with a higher number of service requests. Finally, all customers are sorted in ascending order of κ_i, and we select the first δ customers for removal.

Proximity removal removes close customers. Let $t_{max} = \max_{i \in \mathcal{N}, j \in \mathcal{N}} t_{ij}$ be the maximum travel time between any pair of customers. The first customer i to remove is randomly selected. It serves as center point for the subsequent removals of customers j that are randomly selected if $t_{ij} \leq 0.2 \cdot t_{max}$ until δ customers are removed. If the number of customers within $0.2 \cdot t_{max}$ is less than the number of customers to be removed, we randomly select a customer that is served by the same vehicle as the last removed customer to be the next center point.

Vehicle removal selects customers for removal that are served by the same vehicle. We start with a randomly selected vehicle, and remove all customers served by the vehicle. If at least δ customers have been removed, we terminate the procedure. Otherwise, each remaining vehicle k is selected as the next vehicle for removal with a probability $p_k = \chi_{kk'} / \sum_{k'' \in \mathcal{K}} \chi_{k'k''}$ that is proportional to the inverse distance $\chi_{kk'}$ between vehicle k and the previously selected vehicle k'. The distance between vehicles is the Euclidean distance between their centers of gravity, which is determined as the mean of the weighted coordinates of the customers served by the vehicle. The coordinates of a customer i are weighted with the factor $|\mathcal{D}_i|/|\mathcal{D}|$.

We use the following insertion operators:

Greedy insertion iteratively performs the best possible insertion in myopic manner. Computational experience shows that the ATC violations caused by the insertion of customers into partial solutions are not representative for the ATC violations of the final completed solution. In addition, calculating these violations is computationally expensive. Therefore, we do not consider the direct effect of an insertion on the ATC violation by means of G_{atc}, but instead use a learning-based penalty component P_{atc} that aims at indirectly improving the ATC. Not directly considering ATC violations allows to determine the best insertion position separately on each individual day because violations of route duration and capacity are not linked over the days.

Let $r_{kd}^{+(i,p)}$ be the current route of vehicle k on day d with customer i inserted after position p. Then, for each still unassigned customer i and each vehicle k, we compute the cost increase

$$\Delta\hat{z}_{ik} = \sum_{d\in\mathcal{D}_i} \min_{p=0,\ldots,n_{kd}} \big((z(r_{kd}^{+(i,p)}) - z(r_{kd})) + \sigma_{cap}\cdot(G_{cap}(r_{kd}^{+(i,p)}) - G_{cap}(r_{kd}))$$
$$+ \sigma_{dur}\cdot((G_{dur}(r_{kd}^{+(i,p)}) - G_{dur}(r_{kd})) + P_{atc}(r_{kd}^{+(i,p)})\big),$$

and perform the cheapest insertion according to $\Delta\hat{z}_{ik}$. The procedure is iterated until all customers are inserted.

The aim of the penalty P_{atc} is to identify solution components that are critical with regards to the ATC constraints. For every arc $(h,j) \in \mathcal{A}_d$ and for every day $d \in \mathcal{D}$, we store a penalty value μ_{hjd} that is initially set to zero. After each iteration, the penalty values of a subset of the arcs contained in the newly generated solution $\mathcal{S}$ are updated based on the ATC of the vehicle k traveling the arc:

- If vehicle k does not violate the ATC, we set $\mu_{hjd} := \max(0, \mu_{hjd} - \Delta\mu_{hjd})$ with $\Delta\mu_{hjd} = 0.25 \cdot t_{hj}$ for all arcs traveled by vehicle k.
- If vehicle k violates the ATC, we first draw a randomly selected subset $\overline{\mathcal{D}}$ of the days of the planning horizon (each day is drawn with a probability of 0.5) on which the arcs contained in the solution shall be penalized; penalizing on all of the days on which the vehicle is used does not help our algorithm to explore new solution components. Then, we increase on every day $d \in \overline{\mathcal{D}}$ the penalty values μ_{hjd} of all arcs $(h,j) \in \mathcal{A}_d$ that are traveled by the vehicle by $\Delta\mu_{hjd}$.

Thus, we determine the penalty $P_{atc}(r_{kd}^{+(i,p)})$ for inserting customer i after position p into route r_{kd}, i.e., between vertices $v = r_{kd}(p)$ and $w = r_{kd}(p+1)$, as $P_{atc}(r_{kd}^{+(i,p)}) = \mu_{vid} + \mu_{iwd}$. A large penalty μ_{hjd} might prevent the corresponding arc from being included in a solution and consequently, with the rules described above, this penalty value would never be reduced again. To counteract this undesired behavior, we discount in every iteration all penalty values by a constant factor as $\mu_{hjd} := \mu_{hjd}/1.5$.

Regret insertion tries to anticipate and avoid the negative future consequences of greedy insertion (Ropke and Pisinger, 2006b). We calculate the 2-regret value of each customer i as the difference between the insertion cost $\Delta\hat{z}_{ik}$ of assigning customer i to the best vehicle k and the cost $\Delta\hat{z}_{ik'}$ of assigning to the second-best vehicle k'. The customer with the largest absolute 2-regret value is selected for insertion and the procedure is iterated until all customers are inserted.

We implement two additional variants of Greedy and Regret insertion that add a continuous diversification penalty $P_{div}(i,k)$ for assigning customer i to vehicle k:

$$P_{div}(i,k) = rand([0.5,1.0]) \cdot \frac{\sqrt{z(\mathcal{S})\cdot\zeta_{i,k}}}{\sum_{i\in\mathcal{N}}|\mathcal{D}_i|},$$

where $\zeta_{i,k}$ is the frequency with which customer i was assigned to vehicle k, and the randomization is introduced to prevent cycling of the algorithm. The goal is to encourage the experimental exploration of different solutions (see, e.g., Cordeau, Laporte, and Mercier, 2001).

3.5.3.2. ATC Improvement

We find that the generated solutions that violate ATC constraints often contain routes that serve the same set of customers in almost reversed order on different days, i.e, they strongly disregard the precedence principle of Groër, Golden, and Wasil (2009) described in Section 3.1. For each vehicle violating ATC constraints, we try to improve the ATC with the following two-stage procedure:

1. Inversion of a subset of the routes to generate similar orders of the customer visits on all days,
2. Customer relocation to reduce ATC violations.

To determine which routes should be selected for inversion in Step 1, we require (i) a measure for the difference between routes with regard to the order of customer visits, and (ii) a mechanism to decide which subset of day routes should be inverted based on the pairwise difference of the routes with respect to the measure defined in (i).

As difference measure between two routes, we use the number of customer pairs that occur in reverse order in the two routes. More precisely, we first define a function $p(r_{kd}, v)$ that returns the position of vertex v in route r_{kd}. Then, we determine $\mathcal{R}^{<}_{kd} = \{(v, w) \mid v, w \in \mathcal{N}(r_{kd}),\ p(r_{kd}, v) < p(r_{kd}, w)\}$ as the set of all pairs of customers (v, w) where customer v is served before customer w in route r_{kd}. Further, we define a function $\gamma(v, w)$ that returns 1 if $v = w$ and 0 otherwise. With this, we measure the difference $\rho(r_{kd}, r_{kd'})$ between routes r_{kd} and $r_{kd'}$ on two days d and d' as:

$$\rho(r_{kd}, r_{kd'}) = \sum_{(v,w)\in\mathcal{R}^{<}_{kd}} \sum_{(v',w')\in\mathcal{R}^{<}_{kd'}} \gamma(v, w') \cdot \gamma(w, v').$$

We illustrate the calculation with this example: routes $r_{kd} = \langle 0, 4, 1, 2, 0\rangle$ and $r_{kd'} = \langle 0, 1, 2, 4, 0\rangle$ have the corresponding sets $\mathcal{R}^{<}_{kd} = \{(4, 1), (4, 2), (1, 2)\}$ and $\mathcal{R}^{<}_{kd'} = \{(1, 2), (1, 4), (2, 4)\}$, respectively. The difference $\rho(r_{kd}, r_{kd'})$ is equal to two because two pairs occur in reverse order in both routes, namely $(4, 1)$ and $(1, 4)$, and $(4, 2)$ and $(2, 4)$.

To determine the subset of routes to be inverted, we separate the day routes of each vehicle into two groups using average linkage clustering based on the distance measure ρ (see, e.g. Sarstedt and Mooi, 2014). We first generate a solution by inverting all routes of the first group and then a second solution by inverting all routes of the second group. We calculate the number of customers that violate the ATC constraint in each of the two resulting solutions and in the original solution, and we hand the solution with the lower number over to the second stage. To save computational effort spent on evaluating unpromising steps, we skip the clustering and the subsequent route inversion if the routes of a vehicle k are too similar, i.e., if $\sum_{d\in\mathcal{D}} \left(\sum_{d'\in\mathcal{D}, d\neq d'} \rho(r_{kd}, r_{kd'})/|\mathcal{N}(r_{kd})| \right)$ is below a threshold value that we set to 0.01.

The idea of the second stage is to use customer relocates to improve the ATC. We first determine the customer with the largest ATC violation and inspect all requests of this customer. We call every request late (early) if it causes a violation of the maximum allowed time difference assuming that the request served earliest (latest) is fixed. Then, we investigate two options for improving ATC of customer i: (i) to serve customer i later on the days with early service, and (ii) to serve customer i earlier on the days with

late service. For both cases, we determine a time window within which the customer should be served on the violating days.

In case (i), we separate the planning horizon into a set of early days $\Psi_i = \{d | d \in \mathcal{D}_i \wedge b_{id} < (\max_{d \in \mathcal{D}_i} b_{id}) - L\}$ and a set of feasible days $\Theta_i = \mathcal{D}_i \setminus \Psi_i$. Now, we determine a time window $[\max_{d \in \Theta_i} b_{id} - L, \min_{d \in \Theta_i} b_{id} + L]$. Then, for each violating day $d \in \Psi_i$, we relocate customer i such that the new arrival time lies within the time window, and the increase in operating time is minimum. The procedure for case (ii) works analogously. Note that the determined time window does not guarantee ATC of the resulting solution but aims to guide the algorithm towards better ATC.

We tentatively perform the relocates for the selected customer and both cases. Because the relocation of requests may lead to violations of the maximum allowed time difference at other customers, we either keep the original solution or the solution related to case (i) or case (ii) depending on which solution has the lowest number of violating customers. Then, we continue with the next customer until all customers are served consistently, or each customer has been tried once.

Figure 3.3 shows an example application of our ATC improvement procedure. Vertices depicted in black violate the ATC constraint. In the left part, we illustrate the clustering algorithm and the inversion of routes. First, the difference between every pair of days is calculated: days 1 and 3 are identical and have a distance of zero, days 1 and 2 and days 2 and 3 both have a distance of eight. The clustering assigns days 1 and 3 to the first cluster and day 2 to the second cluster. The best solution is obtained by inverting the route on the second day (the inversion of the routes of the first cluster is not shown). In the center part of the figure, the resulting solution is shown: only customer C exhibits an ATC violation. Now, we evaluate both options: to serve customer C later on day 2 or earlier on days 1 and 3 (only the first case is shown). Based on the fixed arrival times on days 1 and 3, we determine the time window within which customer C has to be served on day 2. Customer C is relocated between customers B and D because the increase in operating time is minimum. The final solution, which respects the ATC constraints, is shown in the right part of the figure.

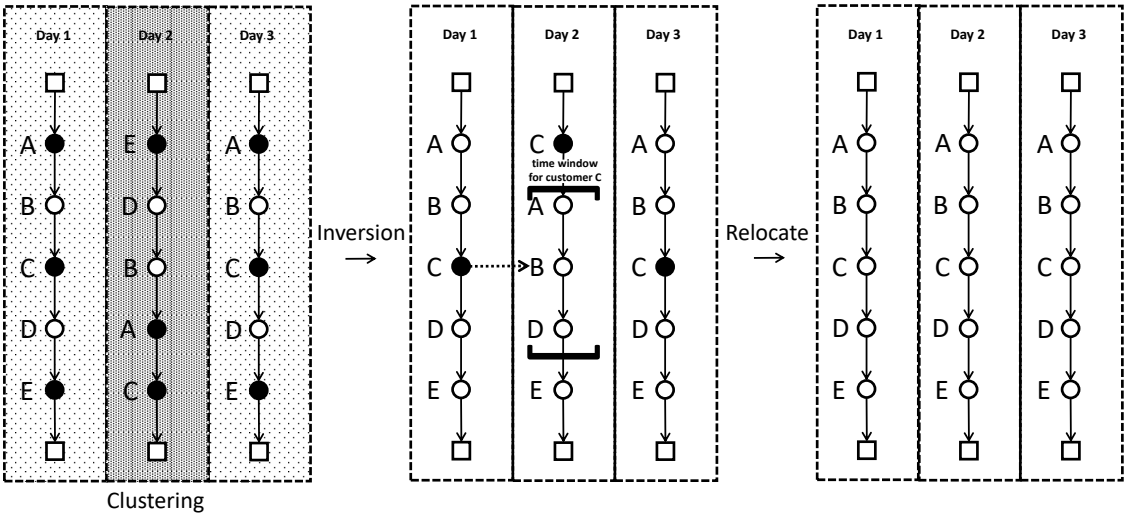

Figure 3.3.: Example application of the ATC improvement procedure.

3.5.3.3. Simulated Annealing Based Acceptance

Our LNS-SP always accepts improving solutions, and a deteriorating solution $\mathcal{S}_t$ is accepted with a probability that depends on the difference between the objective function values Δz_{rel} and a temperature θ (see, e.g., Kirkpatrick, Gelatt, and Vecchi, 1983):

$$p(\mathcal{S}_t, \mathcal{S}_c, \theta) = e^{\frac{-\Delta z_{rel}(\mathcal{S}_t, \mathcal{S}_c)}{\theta}}.$$

To avoid the undesired effect that differences between objective function values also depend on the values of the penalty factors, we use the relative difference between objective function values to calculate the acceptance probabilities (see Goeke and Schneider, 2015):

$$\Delta z_{rel}(\mathcal{S}_t, \mathcal{S}_c) = \frac{z_{gen}(\mathcal{S}_t) - z_{gen}(\mathcal{S}_c)}{z_{gen}(\mathcal{S}_c)}.$$

The temperature follows a predefined cooling schedule defined by an initial temperature and a cooling rate. The initial temperature is such that a solution that deteriorates the initial solution by 0.5% is accepted with a probability of 50%. We decrease the temperature in every iteration by multiplying it with the cooling rate, and we set the cooling rate such that the temperature is below $\theta_{min} = 0.0001$ in the last 20% of iterations.

3.5.4. Set Partitioning

Every 5000 iterations, we try to improve the best solution found so far by solving the set-partitioning formulation (3.14)–(3.17) for a pool Ω^{LNS} of heuristically determined clusters with a commercial solver. If we find a new best solution, we replace the previous best solution $\mathcal{S}_{best}$. To speed up the solution process, we use the current best solution as initial solution. To generate Ω^{LNS}, we add all feasible clusters that we find during the search, and we store the associated objective value g_C and the routing solutions of the individual days. Whenever we encounter a cluster that is already present in Ω^{LNS}, we update the objective value and the routing if the new objective value is better.

Because we never remove any cluster from Ω^{LNS}, the problem size is strictly increasing, and we use the following three approaches to reduce run-time:

Restrict the number of routes. Subramanian, Uchoa, and Ochi (2013) found that restricting the number of available routes can speed up the solution of the set-partitioning problem for a wide range of VRPs. We add the following constraint to restrict the number of selected clusters:

$$|\mathcal{K}'| - 1 \leq \sum_{C \in \Omega^{LNS}} \xi_C \leq |\mathcal{K}'| + 1,$$

with $\mathcal{K}'$ the set of vehicles in the current $\mathcal{S}_{best}$ that serve at least one customer.

Only solve promising problems. We observed that it is unlikely to find a new best solution if the lower bound LB given by the linear relaxation of the current set-partitioning does not improve compared to the LB of the last SP solved. Therefore, before we solve the set-partition problem including the integrality constraints (3.17), we relax the latter to quickly obtain a LB. Now, we only add the integrality constraints if this LB improves the previous LB by more than 0.4%.

Limit the run-time of the solver. We adjust the time limit of the commercial solver depending on the initial optimality gap, i.e., the difference between the objective value $\mathcal{S}_{best}$ and LB. In detail, we calculate the run-time in seconds as $10 + \min(20, 20 \cdot (z(\mathcal{S}_{best}) - \text{LB})/0.04)$. Consequently, the maximum run-time of 30 seconds is used if the optimality gap is at least 4%. Furthermore, the last problem solved during LNS-SP is always performed with the maximum time limit of 30 seconds.

3.6. Numerical Studies

In this section, we present our numerical studies to assess the performance of CCG and LNS-SP. In Section 3.6.1, we describe the benchmark instances available from the literature and the generation of new medium-sized instances. In the first experiment (Section 3.6.2), we solve the compact formulation of Groër, Golden, and Wasil (2009) and our improved formulation using CPLEX, and compare it to CCG on small-sized instances from the literature. In the second experiment (Section 3.6.3), we study the performance of CCG on the newly generated medium-sized instances and investigate how the performance is related to parameters of the problem. Finally, we compare LNS-SP as stand-alone metaheuristic to other state-of-the-art metaheuristics on benchmark instances from the literature (Section 3.6.4).

We performed all tests on a desktop computer with an AMD FX-6300 processor at 3.5 GHz with 8 GB of RAM and running Windows 10 Pro. We used CPLEX 12.6.3 as a MILP solver to solve problem (3.28)–(3.32) in Step 1, problem (3.18)–(3.21) in Steps 2 and 3, problem (3.33)–(3.41) in Step 4 and the set-partitioning problem detailed in Section 3.5.4. CCG is implemented in C and LNS-SP in Java. Both codes and CPLEX were executed using a single core. For CCG, we set a time limit of two hours for all tests. All computing times are reported in seconds.

3.6.1. Benchmark Instances

Several benchmark sets for the ConVRP are available in the literature. Two sets are introduced in Groër, Golden, and Wasil (2009) and differ with regard to instance size (small and large) and the way the instances are generated: *Dataset A* contains five instances with 10 customers and five instances with 12 customers; the planning horizon spans three days; customers have a 70% service frequency, i.e., the probability that a customer requires service on a given day is 0.7. *Dataset B* contains 12 instances with 50 to 199 customers. The instances are derived from instances of the well-known benchmark set for the distance-constrained capacitated VRP presented in Christofides, Mingozzi, and Toth (1979). Five of these instances have route duration constraints; the planning horizon spans five days; customers have a 70% service frequency. Note that Groër, Golden, and Wasil (2009) do not restrict the maximum allowed time difference L on these instances but report the maximum value that they obtain for each instance.

Because later works (Tarantilis, Stavropoulou, and Repoussis, 2012; Kovacs, Parragh, and Hartl, 2014; Kovacs, Golden, et al., 2015) reported this maximum value as a limit on L, we do the same in order to have comparable results.

Dataset C contains 144 instances and was introduced by Kovacs, Parragh, and Hartl (2014) to investigate the impact of varying the maximum allowed time difference and the service frequency. In addition to the instances from Dataset B with 70% service frequency, the authors generate instances with 50% and 90% service frequency. Then, for each service frequency, they vary the maximum allowed time difference and generate four instances: the first has an unbounded value of L (these instances are referred to as L_∞ in the following), while the other three are obtained by setting $L_x = x \cdot L_{max}$, where L_{max} is the maximum arrival time difference obtained by solving the instance L_∞, and $x = 0.4, 0.6, 0.8$ (these instances are referred to as $L_{0.4}$, $L_{0.6}$, $L_{0.8}$).

Datasets A and B, and C are used to assess the performance of LNS-SP, but only Dataset A is suitable to assess the performance of CCG because most of the instances from the other sets are too large. Therefore, we create an additional *Dataset D* with 144 instances with 20 to 30 customers by adapting the instances from Dataset C. We leave the planning horizon of five days unchanged because a weekly plan seems a reasonable setting. From every subset representing one combination of service frequency and maximum allowed time difference, we select the instances labeled 6, 7 and 8 from instances 1–12 because they have both route duration and capacity constraints and create two new instances: the first containing the first 20 customers, and the second the first 30 customers. In addition, we duplicate every new instance by removing the route duration to study the influence that this parameter has on the performance of our approach. We round all distances to the second decimal place to make our results practically independent of the internal precision of the hardware and software used. We name the instances according to the following exemplary format: 6_19_0.5_0.4 means that the instance is based on instance 6 from Dataset C, contains 19 customers that require service (some instances contain customers that do not require service on any day, and for clarity we remove these customers from the instance), has a service frequency of 50% and a maximum allowed time difference of 40% of the maximum arrival time difference obtained for the unbounded instance in Kovacs, Parragh, and Hartl (2014).

3.6.2. Comparison between Compact Formulations and CCG on Dataset A

In this section, we compare the computational performance of the original compact formulation of Groër, Golden, and Wasil (2009) (hereafter called GGW), our improved compact formulation (3.1)–(3.12) (called GRS), and CCG on Dataset A.

Table 3.1 reports the instance name (Inst.) and the best known upper bound (UB), indicated in bold if it corresponds to the optimal solution cost. For both GGW and GRS, we provide the lower bound (LB) obtained by the corresponding linear relaxation in percent of the UB (Δ_{LB}) and the total computing time (t). For CCG, we give the number of routes ($|\Phi|$) and the initial number of clusters generated in Step 2 (ω'), lower bounds $x \in \{z(\text{LP}_1),\ z(\text{LP}_2),\ z(\text{SP})\}$ in percent of the UB (Δ_x), the number of clusters left after executing Steps 1–3 (ω''), the number of times Step 4 is executed ($|\overline{\Omega}|$), and the total computing time of CCG without considering the time to compute UB (t_{noUB}) and including the time to compute UB (t_{tot}). Detailed results for CCG can be found in Table B.1 of the appendix of this chapter.

Table 3.1.: Computational performance of GGW, GRS, and CCG on Dataset A.

		GGW		GRS		CCG								
Inst.	UB	Δ_{LB}	t	Δ_{LB}	t	$\lvert\Phi\rvert$	ω'	$\Delta_{z(\text{LP}_1)}$	$\Delta_{z(\text{LP}_2)}$	ω''	$\Delta_{z(\overline{\text{SP}})}$	$\lvert\overline{\Omega}\rvert$	t_{noUB}	t_{tot}
1_10	**142.03**	61.0	9.4	66.5	5.2	305	744	100.0	100.0	5	100.0	2	0.1	5.1
2_10	**121.07**	57.3	2.0	69.4	1.1	760	951	97.6	100.0	9	100.0	2	0.1	2.6
3_10	**149.41**	53.5	10.4	61.8	9.4	1221	774	100.0	100.0	10	100.0	2	0.1	2.7
4_10	**150.89**	58.5	13.0	62.7	10.1	753	801	100.0	100.0	9	100.0	2	0.1	2.5
5_10	**132.31**	63.9	636.3	71.2	11.4	718	810	95.9	98.8	14	100.0	4	0.2	3.6
1_12	**171.02**	57.2	1524.1	66.4	122.3	852	2150	98.9	99.9	19	100.0	2	0.1	3.3
2_12	**111.54**	64.5	4.5	72.1	3.4	657	3743	99.5	99.5	11	100.0	2	0.1	3.2
3_12	**145.69**	51.4	179.1	59.2	54.4	1573	3666	99.7	99.7	10	100.0	4	0.2	3.0
4_12	**166.37**	51.7	3286.3	60.1	60.6	928	2317	97.1	98.6	47	100.0	13	0.5	3.9
5_12	**140.42**	52.8	66.0	60.5	31.6	1188	3412	98.5	99.5	11	100.0	5	0.2	3.3
Avg.		57.2	573.1	65.0	30.9	896	1937	98.7	99.6	15	100.0	4	0.2	3.3

As the computing times show, all three formulations could solve all instances to optimality. However, GRS outperforms GGW both in terms of lower bound provided by the linear relaxation, which is on average 7.8% higher, and in terms of the total computing time (30.9 vs. 573.1 seconds). CCG is significantly faster on average than GRS, even when taking into account the time to compute UB. We also observe that the lower bounds computed by CCG are of very good quality. In particular, note that the average optimality gaps of LP_1 and LP_2, which do not consider ATC, are quite small (1.3% and 0.4%, respectively). Moreover, the number of clusters for which ATC had to be included a-posteriori is very low (see column $|\overline{\Omega}|$). This suggests that DC in many instances already implies ATC.

Finally, we assessed the impact of using GSECs by removing them from GGW and GRS (not reported in the table): without GSECs, we are still able to solve all instances to optimality within the time limit, but the average computing time increases significantly, i.e., for GGW from 573.1 to 1378.3 seconds and for GRS, from 30.9 to 382.4 seconds.

3.6.3. Computational Results of CCG on Dataset D

In this section, we investigate the computational performance of CCG on the 144 new instances of Dataset D. In Tables 3.2 and 3.3, the results are aggregated according to the number of customers ($|\mathcal{N}|$) and the presence/absence of route duration constraints (T = yes/no). Table 3.2 reports results based on different values of maximum allowed time difference ($L_{0.4}$, $L_{0.6}$, $L_{0.8}$, L_{∞}), and Table 3.3 based on different values of service frequency ($\mathcal{D}_{0.5}$, $\mathcal{D}_{0.7}$, $\mathcal{D}_{0.9}$). The values reported in each row of Tables 3.2 and 3.3 are averages over the corresponding nine and 12 instances, respectively. Column Opt reports the number of instances solved to optimality out of the total number of instances in the group. Detailed results can be found in Tables B.2–B.5 of the appendix of this chapter.

The discussion of the results follows the order of the columns in Tables 3.2 and 3.3. We observe that CCG solves 128 of the 144 instances to optimality. Instances with 30 customers (60 solved) are obviously more difficult than instances with 20 customers (68 solved). The remaining 16 instances could not be solved because either the time limit was reached (eight instances) or CCG ran out of memory

Table 3.2.: Overview of results for different values of maximum allowed time difference (L_x) on newly generated medium-sized instances of Dataset D.

$\|\mathcal{N}\|$	T	Opt	$\|\Phi\|$	ω'	$\Delta_{z(\mathrm{LP}_1)}$	$\Delta_{z(\mathrm{LP}_2)}$	ω''	$\Delta_{z(\overline{\mathrm{SP}})}$	$\|\overline{\Omega}\|$	t_{noUB}	t_{tot}
$\boldsymbol{L_{0.4}}$											
20	yes	9/9	51 143	183 386	97.6	97.8	379	100.0	30	25.0	164.0
20	no	5/9	211 817	561 468	95.8	96.1	39 780	97.8	41	3253.1	3410.8
30	yes	8/9	870 272	17 395 266	97.6	98.0	5575	99.6	30	905.1	1159.0
30	no	4/9	10 924 570	202 276 014	97.6	97.9	6322	99.2	67	3241.5	3482.1
$\boldsymbol{L_{0.6}}$											
20	yes	9/9	51 143	183 386	99.4	99.5	47	100.0	6	0.7	105.8
20	no	9/9	211 817	561 468	98.6	98.7	100	100.0	10	485.0	633.5
30	yes	9/9	870 272	17 395 266	99.3	99.5	200	100.0	6	55.2	290.7
30	no	7/9	10 924 570	202 276 014	99.1	99.2	104	100.0	15	324.8	541.0
$\boldsymbol{L_{0.8}}$											
20	yes	9/9	51 143	183 386	99.6	99.8	34	100.0	4	0.5	86.0
20	no	9/9	211 817	561 468	99.2	99.3	37	100.0	4	59.2	194.8
30	yes	9/9	870 272	17 395 266	99.4	99.6	148	100.0	5	53.9	266.0
30	no	7/9	10 924 570	202 276 014	99.9	99.9	16	100.0	4	280.0	466.0
$\boldsymbol{L_{\infty}}$											
20	yes	9/9	51 143	183 386	99.9	99.9	22	100.0	3	0.4	59.4
20	no	9/9	211 817	561 468	100.0	100.0	10	100.0	2	6.9	81.9
30	yes	9/9	870 272	17 395 266	99.5	99.7	132	100.0	4	53.6	187.5
30	no	7/9	10 924 570	202 276 014	100.0	100.0	12	100.0	3	280.2	406.0

Table 3.3.: Overview of results for different service frequencies ($\mathcal{D}_x$) on newly generated medium-sized instances of Dataset D.

$\|\mathcal{N}\|$	T	Opt	$\|\Phi\|$	ω'	$\Delta_{z(\mathrm{LP}_1)}$	$\Delta_{z(\mathrm{LP}_2)}$	ω''	$\Delta_{z(\overline{\mathrm{SP}})}$	$\|\overline{\Omega}\|$	t_{noUB}	t_{tot}
$\boldsymbol{\mathcal{D}_{0.5}}$											
20	yes	12/12	3831	372 678	99.3	99.4	47	100.0	12	2.0	76.3
20	no	11/12	6239	699 534	97.8	97.8	27 938	99.4	29	882.5	970.5
30	yes	12/12	43 382	47 459 139	99.5	99.6	141	100.0	15	105.4	260.0
30	no	7/12	93 653	205 949 404	98.9	98.9	2059	99.9	49	1077.5	1208.1
$\boldsymbol{\mathcal{D}_{0.7}}$											
20	yes	12/12	30 451	112 843	98.6	98.7	210	100.0	13	12.0	108.1
20	no	11/12	85 691	513 249	98.7	98.8	989	99.5	10	654.2	793.4
30	yes	12/12	519 274	3 438 466	98.7	99.3	782	100.0	11	45.5	272.5
30	no	11/12	9 137 334	316 518 019	99.3	99.4	1624	99.8	16	994.0	1221.0
$\boldsymbol{\mathcal{D}_{0.9}}$											
20	yes	12/12	119 146	64 637	99.4	99.6	104	100.0	7	5.8	127.0
20	no	10/12	543 521	471 622	98.7	98.9	1019	99.4	4	1316.4	1476.9
30	yes	11/12	2 048 161	1 288 192	98.5	98.7	3619	99.7	8	650.0	894.9
30	no	7/12	24 436 342	27 239 617	99.2	99.4	1152	99.7	6	1042.3	1243.6

in Step 2 while generating the set of all routes Φ or the initial set of all clusters $\hat{\Omega}$ (eight instances). Instances without route duration constraint are harder to solve: 15 out of the 16 unsolved instances have unlimited route duration, and all cases of insufficient memory occur for this type of instances. As Table 3.2 shows, instances that have a lower maximum allowed time difference are more difficult: 10 of the 16 open instances belong to the group $L_{0.4}$, and the average computing times increase significantly if the maximum allowed time difference decreases. The latter effect is due to the lower quality of the bounds provided by CCG. Table 3.3 shows that no similar effect can be observed for the service frequency: six of the open instances are in group $\mathcal{D}_{0.5}$, two in $\mathcal{D}_{0.7}$, and eight in $\mathcal{D}_{0.9}$.

The number of routes $|\Phi|$ and the initial number of clusters ω' increases with the number of customers and with unlimited route duration. As Table 3.2 shows, both values are independent of the group L_x because the maximum allowed time difference is not considered in this step. From Table 3.3, we observe that the number of routes increases drastically with the service frequency because more customers request service on any of the days. The number of clusters ω' decreases with rising service frequency because there are fewer ways to combine the routes to clusters that respect the DC. Note that the values ω' provided for $|\mathcal{N}| = 30$ and $T =$ no of groups $\mathcal{D}_{0.5}$ and $\mathcal{D}_{0.9}$ can be compared with each other because for both groups we could not generate the initial routes for the same set of instances due to insufficient memory. This does not hold for the comparison with $\mathcal{D}_{0.7}$.

CCG provides high-quality lower bounds: $\Delta_{z(\mathrm{LP}_1)}$ is 98.9% averaged over all instances of Dataset D (computed from the results reported in Tables B.2–B.5 of the appendix), $\Delta_{z(\mathrm{LP}_2)}$ is 99.0%; the lowest value for any instance is 91.3% for both bounds, and for none of the groups considered in Tables 3.2 and 3.3 the bounds lie below 96.1%. The quality of the bounds decreases for lower values of L_x, but there is no clear relationship between their quality and the service frequency $\mathcal{D}_x$, the number of customers, or the existence of a limit on the route duration. Analogous effects can be observed for the number of clusters ω''. It is noteworthy how strongly the number of clusters can be reduced because of Step 3 when bounds are tight.

The final lower bounds $\Delta_{z(\mathrm{SP})}$ show that for groups with a maximum allowed time difference larger than $L_{0.4}$, we are able to solve all instances to optimality whenever we have enough memory to generate the initial clusters. For a maximum allowed time difference of $L_{0.4}$, we obtain aggregated final bounds of at least 97.8%. In addition, we find that for large values of L_x or $\mathcal{D}_x$, the number of clusters for which ATC had to be included a-posteriori (see column $|\overline{\Omega}|$) is typically very low, i.e., the SP often quickly identifies the optimal clusters in Step 4. As already discussed above, removing the route duration or decreasing the maximum allowed time difference makes the instances more difficult, and thus the total computing time increases. On the other hand, there is no clear relationship between the service frequency and the total computing time.

3.6.4. Computational Results of LNS-SP

In this section, we investigate the performance of LNS-SP as stand-alone method. First, we compare LNS-SP to the approaches from the literature on Dataset B; for the sake of conciseness, we limit the comparison to the two best-performing approaches, i.e., the template-based ALNS of Kovacs, Parragh, and Hartl (2014) (denoted as KPH) and the LNS method of Kovacs, Golden, Hartl, and Parragh (2015) (denoted as KGHP). Second, we study the performance of LNS-SP in comparison to KPH on Dataset

C (no results are available for KGHP) and assess the influence of different service frequencies $\mathcal{D}_x$ and maximum allowed time differences L_x on the comparison.

Table 3.4 shows the results for Dataset B. Two different versions of LNS-SP are studied: LNS-SP–25k using a total of $\eta_{total} = 25000$ iterations, and LNS-SP–5k using a reduced number of iterations $\eta_{total} = 5000$. For each instance, we report the name and the previous best-known solution (BKS). For each solution method, we report the percentage gap of the best solution found in 10 runs to the BKS (Δz_b), the gap of the average solution value of the 10 runs to the BKS (Δz_a), and the average computation time in seconds (t). In addition, the best solution that we found during the overall testing of our method and its gap to the BKS are reported in columns $\overline{\text{LNS-SP}}$. For each instance, the best solution found by any of the tested methods is marked in bold.

Table 3.4.: Comparison of LNS-SP to the best-performing approaches from the literature: KPH (Kovacs, Parragh, and Hartl, 2014) and KGHP (Kovacs, Golden, Hartl, and Parragh, 2015) on Dataset B.

		KPH			KGHP			LNS-SP–25k			LNS-SP–5k			$\overline{\text{LNS-SP}}$	
Inst.	BKS	Δz_b	Δz_a	t	Δz_b	Δz_a	t	Δz_b	Δz_a	t	Δz_b	Δz_a	t	z	Δz
1_50_0.7	2124.21	0.0	3.3	5.5	0.0	0.4	15.1	0.0	0.2	40.5	0.0	0.9	9.2	**2121.84**	**-0.1**
2_75_0.7	3540.80	1.7	1.8	14.7	0.0	1.4	18.8	-1.3	-0.9	86.9	-1.0	0.2	16.9	**3481.72**	**-1.7**
3_100_0.7	3280.47	1.4	1.8	25.6	0.0	0.9	40.2	**-0.1**	0.5	195.3	0.1	1.5	33.6	**3278.36**	**-0.1**
4_149_0,7	4473.31	1.9	2.8	84.3	0.0	1.9	62.7	-1.4	-0.2	369.7	-1.4	1.3	80.1	**4355.47**	**-2.6**
5_199_0.7	5632.22	0.6	0.9	122.2	0.0	0.8	87.3	-2.6	-0.9	477.6	-0.3	0.7	97.7	**5480.00**	**-2.7**
6_49_0.7	**4051.48**	**0.0**	0.0	6.6	0.5	0.6	14.6	**0.0**	0.0	31.7	**0.0**	0.1	6.9	**4051.48**	**0.0**
7_75_0.7	6673.61	1.5	2.0	18.3	0.0	0.6	19.7	**-0.4**	-0.4	73.6	**-0.4**	0.5	16.4	**6645.05**	**-0.4**
8_100_0.7	7126.29	0.0	0.9	32.2	0.0	1.0	31.3	**-0.5**	-0.1	145.5	-0.4	0.4	30.1	**7094.05**	**-0.5**
9_150_0.7	10 381.90	0.0	0.7	97.4	0.1	0.6	50.2	-0.5	-0.1	367.6	-0.6	0.1	66.8	**10 318.99**	**-0.6**
10_198_0.7	12 955.10	1.1	2.2	146.3	0.0	0.7	78.7	**-0.9**	0.1	467.0	-0.2	0.3	83.9	**12 839.78**	**-0.9**
11_119_0.7	4471.22	0.3	0.3	36.0	0.0	2.6	83.6	-0.5	0.5	227.1	-0.3	3.7	119.1	**4447.45**	**-0.5**
12_100_0.7	3497.93	0.0	0.0	25.6	0.7	2.5	27.4	**-2.3**	-2.0	125.9	-2.1	0.3	27.2	**3416.08**	**-2.3**
Avg.		0.7	1.4	51.2	0.1	1.2	44.1	-0.9	-0.3	217.3	-0.6	0.8	49.0		-1.0

The performance of LNS-SP on Dataset B is very convincing. LNS-SP–25k improves the previous BKS on 10 out of the 12 instances (for two of the instances, the improvement is above 2%) and matches it on the remaining two. The average improvement based on the best run is nearly 1%, and even the average of the runs shows a negative gap of -0.3% to the previous BKS. Concerning the comparison of the run-times of the different solution methods, we think that a relatively fair comparison is possible because all algorithms were tested on modern desktop computers with processors of similar speed (Intel Xeon X5550 at 2.67 GHz for KPH and KGHP). The run-time of LNS-SP–25k is approximately five times the run-times of the comparison methods, however, the run-time stays below eight minutes for all instances, which we deem very reasonable for a multi-period problem with up to 199 customers and five periods from a practical perspective.

The fast variant of our algorithm, LNS-SP–5k, has roughly the same run-times as the comparison methods but is able to improve the previous BKS for nine of the 12 instances, matches it on two, and yields a gap of 0.1% on one instance. On average, LNS-SP–5k still shows a negative gap to the previous BKS of -0.6%. Finally, during the overall testing, we find new BKS for all instances with an average gap to the previous BKS of -1.0%. On three instances, we obtain significant improvements of the solution quality with gaps above -2.3%.

Table 3.5 shows the results for Dataset C. Only averages over 10 runs are reported by KGHP, so we

perform the same number of runs and conduct the comparison based on averages: Δz_a reports the percentage gap between the average objective value of LNS-SP–25k and that of KPH, i.e., $\Delta z_a = (z_a(\text{LNS-SP–25k}) - z_a(\text{KPH}))/z_a(\text{KPH})$. Moreover, we report the percentage gap of the average maximum arrival time difference between any two visits to a customer in column $\Delta b_a^{\max}$, i.e., for every run we memorize the maximum arrival time difference that occurs for any of the customers in the best solution obtained during that run, then we average these values over all runs and calculate the gap in percent to the corresponding value reported for KPH. To provide comparison values for future researchers, we additionally provide the best objective function value obtained during 10 runs in column z_b.

We find that the solution quality of LNS-SP–25k is clearly superior to that of KPH: the average gap is negative for 133 of 144 instances with an average improvement of -12.4%. The largest gaps are obtained for small values of L_x; depending on the service frequency, the gaps lie between -44.2% and -32.2% for $L_{0.4}$. With regards to the maximum arrival time difference $b_a^{\max}$, we observe a nonnegative gap between LNS-SP and KPH for all groups. This suggests that our method is able to better utilize the maximum allowed time difference to find high-quality solutions.

Summarizing, the results indicate that, contrary to the template-based approach of KPH, LNS-SP is also suitable for low values of L. As can be expected, the difference between the two methods becomes smaller for a high service frequency of $\mathcal{D}_x$ because instances where days resemble each other with regards to the customers that have to be served are beneficial for the template concept.

3.7. Summary and Conclusion

In this work, we address the ConVRP and present the first exact solution method and a heuristic that represent the new state-of-the-art solution methods to solve the problem.

Unlike most of the state-of-the-art exact methods for VRPs that rely on route-based formulations, the proposed exact method is based on a formulation in which variables represent a set of customers (called cluster) assigned to the same vehicle over the planning horizon. We first generate the entire set of clusters and then eliminate those clusters that cannot belong to any optimal ConVRP solution by computing gradually stronger lower bounds to the problem. The main idea of our algorithm is that the DC consistency is implied by the definition of the clusters, and the ATC is iteratively imposed on a small number of clusters only when necessary. The computational experiments show that, because of the strength of the computed lower bounds, the proposed exact method can solve, within reasonable amounts of computing times, instances with up to 30 customers and a five-day planning horizon. The performance of the method is not affected by the service frequency in the instances, and the method is particularly effective if ATC constraints are not extremely tight.

In addition, we present a LNS that is used in combination with the exact method to find optimal ConVRP solutions and as a stand-alone heuristic to find high-quality solutions in short run-times. The method embeds (i) a suitable penalty mechanism to deal with infeasible solutions, (ii) a repair procedure to improve the ATC, and (iii) the solution of a SP problem to enhance solution quality. The computational experiments show that our LNS is able to clearly improve the solution quality compared to previously published heuristics on benchmark instances from the literature, especially if ATC constraints are tight.

Table 3.5.: Comparison of LNS-SP–25k to KPH (Kovacs, Parragh, and Hartl, 2014) on Dataset C.

	L_∞			$L_{0.8}$			$L_{0.6}$			$L_{0.4}$		
Inst.	z_b	Δz_a	$\Delta b_a^{\max}$	z_b	Δz_a	$\Delta b_a^{\max}$	z_b	Δz_a	$\Delta b_a^{\max}$	z_b	Δz_a	$\Delta b_a^{\max}$
$\mathcal{D}_{0.5}$												
1_50_0.5	1616.37	-2.0	56.7	1628.76	-2.4	-4.0	1641.51	-6.0	4.0	1696.33	-7.6	1.5
2_75_0.5	2554.83	-0.7	66.7	2554.94	-1.2	9.4	2563.74	-1.1	-2.2	2590.12	-27.5	6.1
3_100_0.5	2632.43	-1.7	158.6	2632.96	-1.8	6.4	2658.19	-1.5	0.9	2716.13	-36.7	2.1
4_149_0.5	3317.49	-1.7	103.2	3333.52	-1.3	2.1	3337.12	-9.5	-3.3	3366.54	-75.7	7.6
5_199_0.5	3986.56	-0.2	164.0	3988.21	-1.5	1.9	3994.09	-2.8	-0.6	4098.44	-79.7	27.4
6_49_0.5	2863.55	-0.2	65.8	2872.94	-0.4	3.5	2889.42	-1.8	2.3	2943.77	-22.8	5.5
7_75_0.5	4632.31	-1.3	42.8	4637.52	-1.2	1.9	4642.50	-3.4	-3.1	4662.84	-44.1	16.2
8_100_0.5	5332.55	-0.3	109.0	5335.32	-0.2	0.8	5342.04	-0.5	-1.1	5384.56	-45.4	3.4
9_150_0.5	7347.4	-1.1	70.5	7352.24	-1.5	2.8	7354.43	-4.5	0.7	7402.12	-52.7	4.9
10_198_0.5	9267.06	-0.5	99.7	9238.96	-0.8	-0.3	9363.47	-52.9	1.8	9576.16	-60.2	2.3
11_119_0.5	3245.08	-1.3	208.2	3253.59	-3.5	36.2	3256.72	-3.6	-1.7	3258.40	-7.0	-1.4
12_100_0.5	2835.65	-1.4	339.4	2845.27	-2.0	8.4	2847.79	-2.0	2.8	2895.90	-71.5	27.6
Avg.		-1.0	123.7		-1.5	5.8		-7.5	0.0		-44.2	8.6
$\mathcal{D}_{0.7}$												
1_50_0.7	2105.39	-0.5	183.2	2110.59	-0.5	-9.2	2118.97	-1.0	2.2	2137.65	-29.0	6.1
2_75_0.7	3481.82	-2.4	113.9	3481.72	-1.8	-1.3	3513.81	-3.4	-0.3	3543.00	-55.2	2.0
3_100_0.7	3266.77	-1.0	220.7	3272.50	-1.1	2.0	3283.23	-0.6	0.9	3325.92	-66.8	7.1
4_149_0,7	4346.38	-2.5	249.8	4408.77	-4.1	0.1	4479.40	-26.1	1.8	4640.36	-81.9	-22.0
5_199_0.7	5464.52	-2.8	217.0	5488.70	-2.2	9.9	5525.68	-2.7	5.6	5571.35	-13.9	0.6
6_49_0.7	4048.96	-0.1	129.0	4051.48	0.0	-2.7	4062.70	0.0	0.1	4102.95	-5.9	-1.5
7_75_0.7	6645.05	-2.1	77.3	6645.95	-2.6	5.1	6658.68	-3.7	-4.1	6676.41	-35.4	2.1
8_100_0.7	7092.22	-1.4	195.4	7097.27	-1.4	3.6	7125.73	-7.6	0.4	7321.12	-55.3	2.5
9_150_0.7	10 316.71	-1.5	58.4	10 327.97	-1.4	0.4	10 339.81	-3.0	-1.3	10 373.95	-47.7	1.4
10_198_0.7	12 827.08	-2.1	117.4	12 909.74	-2.0	-1.8	12 912.85	-17.8	-0.4	13 200.08	-60.3	4.0
11_119_0.7	4443.76	-0.7	843.5	4450.56	3.3	-2.2	4458.83	-10.6	0.9	4950.48	-86.9	39.2
12_100_0.7	3408.55	-3.0	325.5	3416.08	-2.9	2.6	3418.03	-3.5	0.4	3489.39	-80.7	10.4
Avg.		-1.7	227.6		-1.4	0.6		-6.6	0.5		-51.6	4.3
$\mathcal{D}_{0.9}$												
1_50_0.9	2478.84	-0.3	128.9	2488.27	-0.1	-8.8	2493.14	-0.1	4.1	2507.37	-21.1	-4.5
2_75_0.9	4001.08	-0.9	232.2	4003.68	-0.6	-2.4	4007.29	-1.8	10.2	4044.71	-8.9	7.7
3_100_0.9	3974.74	-0.4	282.9	3988.08	0.1	-0.4	4001.64	-0.3	-3.1	4039.43	-5.8	2.6
4_149_0,7	4942.23	-0.8	373.6	4971.81	-0.1	1.7	4929.50	-0.2	3.8	5108.54	-79.8	20.1
5_199_0.9	6376.09	-0.6	371.0	6399.05	-3.5	10.1	6397.04	-7.0	-7.3	6453.19	-76.1	45.0
6_49_0.9	4751.79	-0.2	111.8	4761.17	0.0	7.2	4768.31	-1.0	0.6	4877.14	-18.9	-0.8
7_75_0.9	7705.73	-0.5	53.5	7706.18	-0.4	5.2	7706.18	-0.3	-6.2	7718.94	-2.8	1.3
8_100_0.9	8733.72	0.0	196.3	8673.73	0.0	-3.2	8776.89	-0.6	2.1	8835.97	-29.3	3.9
9_150_0.9	12 377.6	-0.5	203.5	12 391.43	-0.4	-1.1	12 442.23	-7.4	0.2	12 618.92	-58.6	5.5
10_198_0.9	15 820.63	-0.4	321.9	15 824.39	-0.4	0.9	15 828.35	-11.0	0.8	16 212.92	-65.6	2.5
11_119_0.9	4986.96	7.7	866.3	4975.01	9.0	-7.2	5452.57	9.9	-6.0	5496.98	-16.3	-1.3
12_100_0.9	4011.73	0.9	524.4	4013.50	0.8	-2.3	4014.73	0.3	35.8	4024.41	-3.4	3.5
Avg.		0.3	305.5		0.4	0.0		-1.6	2.9		-32.2	7.1

Bibliography

Adulyasak, Y., J.-F. Cordeau, and R. Jans (2014). Optimization-based adaptive large neighborhood search for the production routing problem. In: *Transportation Science* 48 (1), pp. 20–45.

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

Christofides, N., A. Mingozzi, and P. Toth (1979). The vehicle routing problem. In: *Combinatorial Optimization*. Ed. by N. Christofides, A. Mingozzi, P. Toth, and C. Sandi. Chichester, UK: Wiley, pp. 315–338.

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

Contardo, C. and R. Martinelli (2014). A new exact algorithm for the multi-depot vehicle routing problem under capacity and route length constraints. In: *Discrete Optimization* 12, pp. 129–146.

Cordeau, J.-F., G. Laporte, and A. Mercier (2001). A unified tabu search heuristic for vehicle routing problems with time windows. In: *Journal of the Operational Research Society* 52 (8), pp. 928–936.

Dabia, S., S. Ropke, T. van Woensel, and T. de Kok (2013). Branch and price for the time-dependent vehicle routing problem with time windows. In: *Transportation Science* 47 (3), pp. 380–396.

Feillet, D., T. Garaix, F. Lehuédé, O. Péton, and D. Quadri (2014). A new consistent vehicle routing problem for the transportation of people with disabilities. In: *Networks* 63 (3), pp. 211–224.

Goeke, D. and M. Schneider (2015). Routing a mixed fleet of electric and conventional vehicles. In: *European Journal of Operational Research* 245 (1), pp. 81–99.

Groër, C., B. Golden, and E. Wasil (2009). The consistent vehicle routing problem. In: *Manufacturing & Service Operations Management* 11 (4), pp. 630–643.

Jepsen, M., B. Petersen, S. Spoorendonk, and D. Pisinger (2008). Subset-row inequalities applied to the vehicle-routing problem with time windows. In: *Operations Research* 56 (2), pp. 497–511.

Kirkpatrick, S., C. D. Gelatt, and M. P. Vecchi (1983). Optimization by simulated annealing. In: *Science* 220 (4598), pp. 671–680.

Kovacs, A. A., B. L. Golden, R. F. Hartl, and S. N. Parragh (2015). The generalized consistent vehicle routing problem. In: *Transportation Science* 49 (4), pp. 796–816.

Kovacs, A. A., R. F. Hartl, S. N. Parragh, and B. L. Golden (2014). Vehicle routing problems in which consistency considerations are important: A survey. In: *Networks* 64 (3), pp. 192–213.

Kovacs, A. A., S. N. Parragh, and R. F. Hartl (2014). A template-based adaptive large neighborhood search for the consistent vehicle routing problem. In: *Networks* 63 (1), pp. 60–81.

– (2015). The multi-objective generalized consistent vehicle routing problem. In: *European Journal of Operational Research* 247 (2), pp. 441–458.

Lian, K., A. B. Milburn, and R. L. Rardin (2016). An improved multi-directional local search algorithm for the multi-objective consistent vehicle routing problem. In: *IIE Transactions* 48 (10), pp. 975–992.

Masson, R., F. Lehuédé, and O. Péton (2013). An adaptive large neighborhood search for the pickup and delivery problem with transfers. In: *Transportation Science* 47 (3), pp. 344–355.

Pecin, D., C. Contardo, G. Desaulniers, and E. Uchoa (2017). New enhancements for the exact solution of the vehicle routing problem with time windows. In: *INFORMS Journal on Computing* 29 (3), pp. 489–502.

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

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

Sarstedt, M. and E. Mooi (2014). A concise guide to market research: The process, data, and methods using IBM SPSS statistics. In: Berlin Heidelberg, Germany: Springer. Chap. Cluster Analysis, pp. 273–324.

Shaw, P. (1998). Using constraint programming and local search methods to solve vehicle routing problems. In: *Principles and Practice of Constraint Programming – CP98*. Ed. by M. Maher and J.-F. Puget. Vol. 1520. Lecture Notes in Computer Science. London, United Kingdom: Springer, pp. 417–431.

Subramanian, A., E. Uchoa, and L. S. Ochi (2013). A hybrid algorithm for a class of vehicle routing problems. In: *Computers & Operations Research* 40 (10), pp. 2519–2531.

Subramanyam, A. and C. E. Gounaris (2016). A branch-and-cut framework for the consistent traveling salesman problem. In: *European Journal of Operational Research* 248 (2), pp. 384–395.

– (2017). A decomposition algorithm for the consistent traveling salesman problem with vehicle idling. In: *Transportation Science*. DOI: 10.1287/trsc.2017.0741.

Sungur, I., Y. Ren, F. Ordóñez, M. Dessouky, and H. Zhong (2010). A model and algorithm for the courier delivery problem with uncertainty. In: *Transportation Science* 44 (2), pp. 193–205.

Tarantilis, C., F. Stavropoulou, and P. Repoussis (2012). A template-based tabu search algorithm for the consistent vehicle routing problem. In: *Expert Systems with Applications* 39 (4), pp. 4233–4239.

Chapter 4

Mixed Fleets: The Transition to Electric Commercial Vehicles

4.1. Introduction

In Europe, recent years have seen a steady increase of energy costs while more and more laws are passed to regulate the emission of greenhouse gases in the transportation sector. These external factors and the society's rising environmental and social awareness have triggered numerous green initiatives at companies. In the logistics field, electric commercial vehicles (ECVs) are now considered a serious alternative to conventional internal combustion commercial vehicles (ICCVs). ECVs have no local greenhouse gas emission and produce only minimal noise, however, they are currently hardly competitive with ICCVs from a cost point of view (Davis and Figliozzi, 2013). Nevertheless, several companies have started to employ ECVs in their last-mile delivery operations, e.g., in the field of small-package shipping (Kleindorfer et al., 2012) or the distribution of food (National Renewable Energy Laboratory, 2014) and beverages (Heineken International, 2014). Moreover, governments and private companies are starting to provide the required infrastructure to further boost this electrification trend (Tesla Motors, Inc., 2014; International Energy Agency, 2012; International Energy Agency, 2013).

One important aspect to render ECVs more competitive is to consider their special characteristics—a limited driving range and the potential need to recharge en route—in the planning of the last-mile delivery operations employing ECVs. These distribution tasks are generally represented as vehicle-routing problems (VRPs), whose goal is to find minimum-cost routes to serve a given set of customers from a central depot (Toth and Vigo, 2014). The first VRPs to address ECVs (or alternative fuel vehicles) with a limited driving range and the possibility of recharging (refueling) at dedicated stations have recently been presented in the literature (Conrad and Figliozzi, 2011; Erdoğan and Miller-Hooks, 2012; Schneider, Stenger, and Goeke, 2014). Although two important constraints of last-mile delivery operations, namely vehicle capacity constraints and customer time windows have already been considered (Conrad and Figliozzi, 2011; Schneider, Stenger, and Goeke, 2014), many relevant real-life constraints are not yet covered by routing models for ECVs and dedicated solution methods.

In this work, we consider two important aspects for route planning with ECVs:

Mixed fleet: Most companies do not operate pure ECV fleets but are gradually introducing ECVs into their existing ICCV fleet. Therefore, our route planning method is able to handle a mixed fleet of ECVs and ICCVs. Compared to ICCVs, energy costs for operating ECVs are generally lower while labor costs may increase due to time spent on potentially necessary recharging activities

along the routes. Energy and labor costs are among the main components of total operating costs (see, e.g., Bektaş and Laporte, 2011)), so high-quality route planning with a mixed vehicle fleet has to consider the cost tradeoff between the two vehicle types.

Energy consumption: Real-life energy consumption is not a linear function of traveled distance as assumed in the models of Erdoğan and Miller-Hooks (2012) and Schneider, Stenger, and Goeke (2014). We use realistic energy consumption functions of ECVs and ICCVs that incorporate vehicle speed, gradients and cargo load. In recent years, realistic energy consumption models have started to play an important role in routing models that consider fuel costs and vehicle emissions (Bektaş and Laporte, 2011; Jabali, Woensel, and Kok, 2012). In the context of ECVs, energy consumption determines electricity costs on the one hand, but, more importantly, also the driving range of an ECV and thus the latest possible moment at which a recharge has to take place in order to prevent an ECV from getting stranded.

We propose the electric VRP with mixed fleet (E-VRPTWMF) to determine optimal routes (according to different alternative objective functions considered in this work) for a given mixed fleet of ECVs and ICCVs. E-VRPTWMF incorporates time window and vehicle capacity constraints. We assume that ECVs can be recharged at any of the available stations causing a recharging time that depends on the battery level on arrival at the station. The energy consumption of ICCVs is calculated by means of the model presented in (Demir, Bektaş, and Laporte, 2012) and we extend this model to compute the battery energy consumption of ECVs.

As E-VRPTWMF extends the notoriously hard-to-solve VRP with time windows (VRPTW), exact methods will not be able to solve instances of realistic size within fast computation times. Therefore, we develop a heuristic solution method to address the problem, namely an adaptive large neighborhood search (ALNS) enhanced by a local search (LS) for intensification. Besides new operators considering recharging stations, our ALNS features several new ideas: (i) an adaptive mechanism to choose the number of customers to be removed in each iteration, (ii) the use of surrogate violations in order to handle the complexity of calculating time window and battery capacity violations, and (iii) an acceptance criterion taking into account the different penalty factors that were used when calculating the objective value of the solutions to be compared.

In numerical studies, we assess the performance of ALNS on benchmark instances of related problems: ALNS achieves convincing results on the well-studied VRPTW benchmark of Solomon (1987), and outperforms previous methods on the electric vehicle-routing problem with time windows (E-VRPTW) benchmark set of Schneider, Stenger, and Goeke (2014). In addition, we create a set of new E-VRPTWMF instances based on the pollution-routing problem (PRP) benchmark of Demir, Bektaş, and Laporte (2012). In experiments on these newly designed instances, we find that consideration of the actual load strongly improves the quality of the generated solutions in comparison to solutions that are generated based on load estimates. Moreover, we find that a large number of solutions that are generated with "optimistic" load estimates are actually infeasible due to battery capacity or time window violations. We further show that our ALNS works effectively with all of the investigated cost functions and that the traditional objective of minimizing traveled distance fails to produce high-quality solutions if routing costs including energy, labor and battery depreciation are considered. The choice of objective function additionally has a strong influence on the level of usage of the ECVs in the fleet.

This work is organized as follows: In Section 4.2, we briefly review the literature related to E-VRPTWMF.

Section 4.3 introduces the energy consumption models for ECVs and ICCVs. Section 4.4 presents a mixed-integer program of E-VRPTWMF. The ALNS is detailed in Section 4.5. The parameter setting, the generation of new E-VRPTWMF instances and the numerical studies on the new instances and on the test instances of related problems are presented in Section 4.6. Section 4.7 summarizes and concludes the chapter.

4.2. Literature

In the following, we discuss the literature related to the E-VRPTWMF. First, we review works on routing alternative fuel vehicles. Second, VRP papers that explicitly model energy consumption or account for the impact of load distribution are presented. Third, we discuss other related fields.

Conrad and Figliozzi (2011) study the Recharging VRP, in which vehicles with limited range can recharge at certain customer locations. Time window constraints are considered and a fixed recharging time is assumed. The authors compute bounds to predict average tour lengths and study the impact of driving range, recharging times, and time window existence. Erdoğan and Miller-Hooks (2012) propose two heuristics for the Green VRP. In this problem, alternative fuel is only available at dedicated points that have to be visited en route. Refueling time is assumed fixed and no capacity or time window constraints are included. Other works addressing the Green VRP or extensions of this problem are (Schneider, Stenger, and Hof, 2015; Montoya et al., 2014; Felipe et al., 2014). Schneider, Stenger, and Goeke (2014) develop a hybrid of variable neighborhood search (VNS) and tabu search (TS) to address E-VRPTW, in which ECVs with a limited battery capacity may visit recharging stations en route, and customer time windows and vehicle capacities have to be respected. Recharging time is proportional to the amount of energy required to recharge the battery to full capacity. Desaulniers et al. (2016) present branch-price-and-cut algorithms to address four variants of the E-VRPTW. In a recent working paper, Hiermann, Puchinger, and Hartl (2014) combine the E-VRPTW and the Fleet Size and Mix VRP with Fixed costs (FSMF). In the resulting E-FSMVRPTW an unlimited number of ECVs with different battery capacities and vehicle-independent routing costs are available. Cost-optimal routes are determined by means of an ALNS enhanced by a labeling algorithm. Barco et al. (2013) propose a comprehensive approach for planning the deployment of electric vehicles (EVs) in an airport shuttle service. They first determine a minimal consumption graph, on which routing decisions under consideration of a limited battery capacity are made. The assignment of vehicles to routes and the scheduling of recharges is determined by an evolutionary algorithm. Finally, Preis, Frank, and Nachtigall (2014) investigate an ECV routing model with customer time windows, fixed recharging times, and the goal of minimizing total energy consumption, which depends on gradients and cargo load. A simple TS algorithm based on a relocate operator is presented.

The second strand of relevant literature integrates energy considerations and the resulting fuel consumption and emissions of ICCVs into routing models. Bektaş and Laporte (2011) propose the PRP, in which they estimate the price of pollution and introduce it as part of the objective function, besides costs for driver wages and fuel consumption. They allow the choice between different speed levels for arcs and consider speed, gradients, and load to calculate the fuel consumption and corresponding emissions. Demir, Bektaş, and Laporte (2011) provide a comparison of vehicle emission models. Demir,

Bektaş, and Laporte (2012) propose an ALNS and a speed optimization algorithm for the PRP. Computational experiments show moderate run-times for problem sizes with 200 customers. Demir, Bektaş, and Laporte (2014b) address a bi-objective function for the PRP that models the conflicting targets of minimizing driver time and usage of fuel and thus avoid the problematic representation of emissions in terms of monetary cost. An extensive review of related literature can be found in (Demir, Bektaş, and Laporte, 2014a). Kopfer, Schönberger, and Kopfer (2014) use CPLEX to solve a heterogeneous VRP with the objective of minimizing fuel consumption. To this end, the authors derive linear relationships between fuel consumption and cargo load for the different vehicle types.

Xiao et al. (2012) calculate a fuel consumption rate depending on the mass of the remaining cargo and they find significant fuel saving potential for the capacitated VRP (CVRP). Zhang, Tang, and Fung (2011) propose a multi-depot VRP that includes, in the objective function, a cost term that increases linearly with the amount of cargo and the traveled distance and which depends on an empirical cost factor. Another related problem is the fuel or emission shortest-path problem, which may be used to define the arc set of a VRP. Nie and Li (2013) propose such a model to minimize total operating cost while meeting a limit on emissions.

E-VRPTWMF is further related to VRPs with intermediate replenishment facilities (Crevier, Cordeau, and Laporte, 2007; Hemmelmayr, Doerner, et al., 2013) or with distance constraints (Laporte, Nobert, and Desrochers, 1985) due to the similar structure of the problems, to refueling problems with a dense infrastructure (Bousonville et al., 2011; Suzuki, 2012), to energy shortest path problems (Artmeier et al., 2010), and in a more general sense to the field of green logistics from an operations research (OR) perspective (Sbihi and Eglese, 2010; Dekker, Bloemhof, and Mallidis, 2012). Moreover, the planning of electric infrastructure has attracted increasing research interest (He et al., 2013; Nie and Ghamami, 2013; Wang and Lin, 2013; Mak, Rong, and Shen, 2013). Finally, Davis and Figliozzi (2013) and Feng and Figliozzi (2013) address the competitiveness of ECVs. Davis and Figliozzi (2013) show that ECVs are competitive in comparison to ICCVs for certain application scenarios with large route distances and low vehicle speeds.

4.3. Energy Consumption Models

Our VRP model incorporates two specific characteristics of employing ECVs, namely a reduced operating range and the possibility to recharge at certain stations in order to increase this range. The necessity to visit a recharging station and the recharging time at the station depend on the battery level of the ECV, which itself depends on the energy consumption along the route. Previous works on vehicle routing problems with refueling or recharging (Erdoğan and Miller-Hooks, 2012; Schneider, Stenger, and Goeke, 2014) assume energy consumption to be a linear function of the traveled distance. In this work, we use a more realistic model and consider the following factors that influence the energy consumption of a vehicle:

Vehicle Mass: The mass of the vehicle is composed of its curb mass and the load. Concerning the routing of the vehicle, the order in which customers are visited and cargo is unloaded, i.e., the distribution of the load during the course of a route, can strongly influence the energy consumption on the route.

Speed: Accelerating a vehicle and keeping the vehicle in motion at a certain speed consumes energy due to the aerodynamic and rolling resistance that has to be overcome. We assume travel speeds to be constant on a each arc, i.e., we neglect acceleration phases, but vehicle speed may be different for different arcs. Note that in principle vehicle speed could be handled as a decision variable and could be increased in order to fulfill time window requirements and be reduced to decrease energy consumption (see, e.g., Demir, Bektaş, and Laporte, 2014a). As traffic conditions have a strong impact on vehicle speed, we chose to refrain from this modeling.

Gradient of the terrain: We assume a non-flat terrain with grades. Going uphill requires higher amounts of energy than traveling flat terrain or going downhill. ECVs may be able to recharge their battery when going downhill, a process called recuperation.

In the following, we present the methods for determining the energy consumption of ECVs (Section 4.3.1) and ICCVs (Section 4.3.2).

4.3.1. Energy Consumption of Electric Vehicles

We calculate the energy consumption of an ECV in three steps as shown in Figure 4.1. First, we determine the mechanical power P_M using the model presented in (Bektaş and Laporte, 2011), which determines energy consumption based on the factors described above (mass, speed, gradient) and the physical environment (road surface, vehicle dimensions, engine properties). Second, P_M is translated into the electric power P_E that the electric motor needs to provide the required amount of mechanical power. This amount is determined by the efficiency of the electric motor.

Third, the required electric power is converted into the amount of power that has to be taken from the battery (P_B), which depends on the battery efficiency. In the following, these three steps are described in more detail.

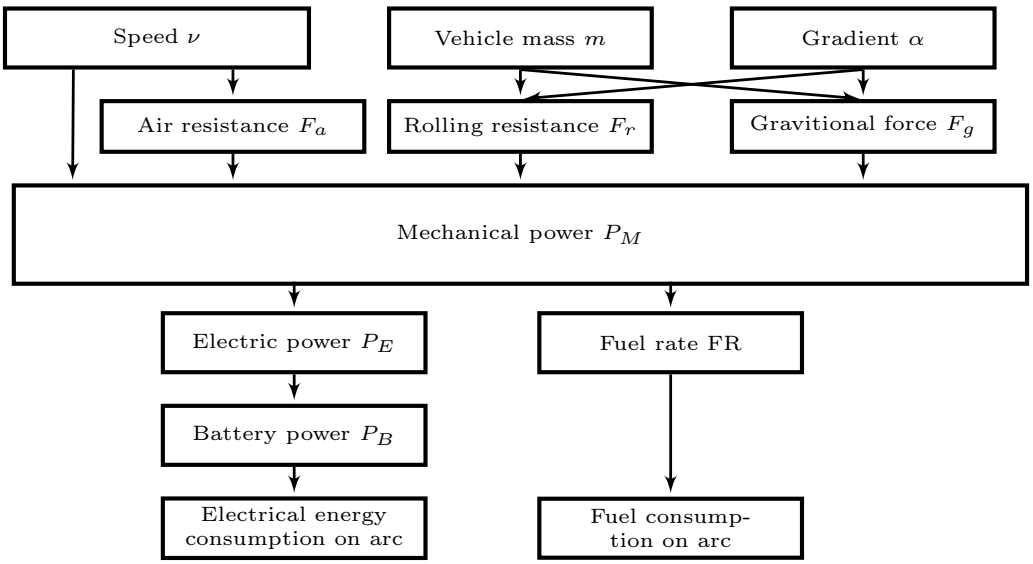

Figure 4.1.: Calculation of required energy and fuel.

The mechanical power P_M is needed to overcome rolling resistance and aerodynamic resistance and is influenced by the gravitational force. Let m denote the total vehicle mass, g the gravitational constant, c_r

the rolling friction coefficient (which depends on tire pressure, road surface conditions and other factors) and α the gradient angle. Then the rolling resistance F_r can be determined as:

$$F_r = c_r \cdot m \cdot g \cdot cos(\alpha).$$

With ν denoting the speed, c_d the aerodynamic drag coefficient, ρ_a the air density and A_f the frontal area of the vehicle, the aerodynamic resistance is:

$$F_a = \frac{1}{2} \cdot \rho_a \cdot A_f \cdot c_d \cdot \nu^2.$$

Adding the gravitational force $F_g = m \cdot g \cdot sin(\alpha)$, the total mechanical power P_M is:

$$P_M = \left(m \cdot a + \frac{1}{2} \cdot c_d \cdot \rho \cdot A \cdot \nu^2 + m \cdot g \cdot sin(\alpha) + c_r \cdot m \cdot g \cdot cos(\alpha) \right) \cdot \nu, \tag{4.1}$$

where a denotes the acceleration. As mentioned above, a is assumed to be zero in the following as we do not consider acceleration and braking processes.

In the second step, the electric power P_E that is needed to achieve the required mechanical power P_M is calculated, taking into account energy losses that occur in the electric engine. To this end, we use the Quantized State System model of an electric engine presented in (Guzzella and Amstutz, 2005). The authors provide values for the engine efficiency at given pairs of torque and rotational velocity. We directly convert torque and rotational velocity to mechanical power. Using homogeneous regression, i.e. a linear regression with a y-axis intercept of zero, on the converted values, we obtain the following relationship between P_M and the discharged electric energy P_E^d (respectively the recuperated electric energy P_E^r) that is described by the regression coefficient ϕ:

$$\begin{aligned} P_E^d &= \phi^d \cdot P_M \text{ for} & 0\ kW \leq & P_M \leq 100\ kW, \\ P_E^r &= \phi^r \cdot P_M \text{ for} & -100\ kW \leq & P_M < 0\ kW. \end{aligned}$$

The determined relationship is adequate for a mechanical power demand of up to 100 kW, which we consider a realistic threshold for commonly employed ECV models.

In the third step, we compute the battery power P_B that is necessary to obtain the required electric energy P_E. For this purpose, we use the relationship for converting electric power P_E into battery power P_B for a lithium-ion accumulator presented in (van Keulen et al., 2010). The presented relationship does not account for external effects (e.g., the ambient temperature) or the influence of the current charge level of the battery. We approximate the given relation by means of a homogeneous regression and obtain as final equations for discharging (P_B^d) and recuperating (P_B^r):

$$\begin{aligned} P_B^d &= \varphi^d \cdot P_E \text{ for} & P_E \geq & 0\ kW, \\ P_B^r &= \varphi^r \cdot P_E \text{ for} & P_E < & 0\ kW, \end{aligned}$$

with the regression coefficient φ that describes the battery efficiency.

To compute energy consumption on a graph, vehicle mass is defined as a function of the currently loaded amount of cargo u, i.e., $m(u) = m_c + m_u \cdot u$, where m_c is the curb mass of the vehicle and m_u the mass

of one unit of cargo. Let u_j denote the amount of cargo when arriving at customer j (before unloading), i.e., the amount required to fulfill the demand at customer j and all following customers. Further, let $P_{ij}(u_j)$ denote the constant mechanical power demand over the course of an arc (i, j) with a load that is equivalent to the remaining customer demand:

$$P_{ij}(u_j) = \Big(\frac{1}{2} \cdot c_d \cdot \rho \cdot A \cdot \nu^2 + m(u_j) \cdot g \cdot \big(sin(\alpha_{ij}) + c_r \cdot cos(\alpha_{ij})\big)\Big) \cdot \nu_{ij}.$$

Then, the battery power demand on this arc is given by $P_B(P_E(P_{ij}(u_j)))$. Multiplying with the associated travel time t_{ij} of the arc yields the electric energy consumption b_{ij} for traveling this particular arc:

$$b_{ij}(u_j) = \begin{cases} \phi^d \cdot \varphi^d \cdot P_{ij}(u_j) \cdot t_{ij} & \text{if } P_{ij}(u_j) \geq 0\ kW, \\ \phi^r \cdot \varphi^r \cdot P_{ij}(u_j) \cdot t_{ij} & \text{if } P_{ij}(u_j) < 0\ kW. \end{cases} \tag{4.2}$$

All variables except the amount of cargo u_j are constant for a given arc. Therefore, it is possible to split $b_{ij}(u_j)$ into a fixed and a load-dependent part and precompute parts of the expression.

4.3.2. Energy Consumption of Combustion Engines

Mechanical power can be directly translated to diesel fuel consumption as shown in (Demir, Bektaş, and Laporte, 2012). The fuel rate is given by $FR = \frac{\xi}{\kappa \cdot \psi}\ (kND\ +\ \frac{P_M}{\eta \cdot \eta_{tf}})$, where P_M is the mechanical power as defined in Equation (4.1), ξ the fuel-to-air mass ratio, κ the heating value of typical diesel fuel, k the engine friction factor, N the engine speed, D the engine displacement, ψ a factor converting the fuel rate from $gram/second$ to $liter/second$, η the efficiency parameter for diesel engines and η_{tf} the drive train efficiency. We assume that driving downhill in gear can reduce the engine consumption down to 0. The constant fuel rate over the course of an arc is given by:

$$FR_{ij}(u_j) = max(\frac{\xi}{\kappa \cdot \psi}\ (kND\ +\ \frac{P_{ij}(u_j)}{\eta \cdot \eta_{tf}}), 0).$$

Thus, the fuel consumption of an ICCV traversing an arc (i, j) with cargo u_j can be calculated as:

$$f_{ij}(u_j) = t_{ij} \cdot FR_{ij}(u_j). \tag{4.3}$$

4.4. The Electric Vehicle Routing Problem with Time Windows and Mixed Fleet

We define E-VRPTWMF on a complete, directed graph $G = (V'_{0,N+1}, A)$. $V = \{1, \ldots, N\}$ denotes the set of customers, F the set of recharging stations, F' the set of visits to vertices in F and V' denotes the union of V and F'. Vertices 0 and $N+1$ denote instances of the depot, and all routes start at 0 and end at $N+1$. Inclusion of one or both depot instances is indicated by subscripting the respective set, e.g., the set $V'_{0,N+1}$ contains both instances of the depot. The set of arcs is given by $A = \{(i, j) \mid i, j \in V'_{0,N+1}, i \neq j\}$.

With each vertex $i \in V'_{0,N+1}$, we associate a nonnegative demand q_i ($q_i = 0, i \notin V$), a nonnegative service time s_i ($s_i = 0, i \notin V$) and a time window $[e_i, l_i]$, within which service at the customer has to start. Time windows are hard, i.e., starting service late is not allowed but waiting in case of early arrival is possible. The time window $[e_0, l_{N+1}]$ at the depot specifies the scheduling horizon of the problem. Each arc is described by the following properties: distance d_{ij}, travel speed ν_{ij}, travel time $t_{ij} = d_{ij}/\nu_{ij}$ and a factor α_{ij} representing the gradient of the terrain.

A mixed vehicle fleet of fixed size consisting of m_E ECVs and m_{IC} ICCVs is positioned at the depot. All vehicles have the same maximal cargo loading capacity Q. On each visit to a recharging station, ECVs are recharged to their maximum battery capacity B. The recharging time depends on the recharging rate r and the difference between B and charge level on arrival at the station. Electricity and fuel consumption of a vehicle traveling an arc are determined by the models introduced in Section 4.3.

Thus, E-VRPTWMF can be formulated as a (nonlinear) mixed-integer program. For every arc $(i, j) \in A$, we define the following binary decision variables: Variables x_{ij}^E are equal to 1 if an arc (i, j) is traveled by an ECV, variables x_{ij}^{IC} if the arc is traveled by an ICCV. Otherwise, the decision variables are 0. Variables τ_i define the arrival time, u_j the remaining cargo (before unloading) and y_i the remaining charge level on arrival at vertex $i \in V'_{0,N+1}$. Table 4.1 summarizes the variables and parameters of our model.

Table 4.1.: Variable and parameter definitions of the E-VRPTWMF model.

$0, N+1$	depot instances
F'	set of visits to recharging stations, dummy vertices of the set of recharging stations F
V	set of customer vertices $V = \{1, \ldots, N\}$
V'	set of customers and recharging visits: $V' = V \cup F'$
d_{ij}	distance between vertices i and j
t_{ij}	travel time between vertices i and j
Q	vehicle capacity
r	recharging rate
B	battery capacity
$b_{ij}(u_j)$	electric energy consumption between between vertices i and j depending on cargo u_j (Equation (4.2))
$f_{ij}(uj)$	fuel consumption between between vertices i and j depending on cargo u_j (Equation (4.3))
q_i	demand of vertex i, 0 if $i \notin V$
e_i	earliest start of service at vertex i
l_i	latest start of service at vertex i
s_i	service time at vertex i ($s_0, s_{N+1} = 0$)
τ_i	decision variable specifying the time of arrival at vertex i
u_i	decision variable specifying the remaining cargo on arrival at vertex i
y_i	decision variable specifying the remaining battery capacity on arrival at vertex i
x_{ij}^E	binary decision variable indicating if arc (i, j) is traveled by an ECV
x_{ij}^{IC}	binary decision variable indicating if arc (i, j) is traveled by an ICCV

In this work, we consider three alternative objective functions for E-VRPTWMF:

1. Minimize traveled distance f_d:

$$f_d = \sum_{i \in V'_0,\, j \in V'_{N+1},\, i \neq j} d_{ij} \, (x_{ij}^E + x_{ij}^{IC}) \rightarrow \min. \tag{4.4}$$

2. Minimize cost f_c, consisting of the cost for vehicle propulsion and labor cost. Cost factors c^E and

c^{IC} denote the cost of electric energy, respective diesel fuel and c^D the driver wage per time unit:

$$f_c = \sum_{i \in V'_0, j \in V'_{N+1}, i \neq j} [c^E \cdot b_{ij}(u_j) \cdot x^E_{ij} + c^{IC} \cdot f_{ij}(u_j) \cdot x^{IC}_{ij}] + \sum_{i \in V'} c^D \, (x^E_{iN+1} + x^{IC}_{iN+1})(\tau_i + s_i + t_{iN+1}) \rightarrow \text{min.} \tag{4.5}$$

3. Minimize cost including cost for battery replacement f_{cb}. The distance traveled by ECVs is multiplied by a cost factor c^B that accounts for the cost of battery replacement. It is expressed as cost per kilometer by dividing acquisition cost by the expected total lifetime mileage:

$$f_{cb} = f_c + \sum_{i \in V'_0, j \in V'_{N+1}, i \neq j} c^B \cdot d_{ij} \cdot x^E_{ij} \rightarrow \text{min.} \tag{4.6}$$

The constraints of E-VRPTWMF are as follows:

$$\sum_{j \in V'} (x^E_{ij} + x^{IC}_{ij}) = 1 \qquad \forall i \in V \tag{4.7}$$

$$\sum_{j \in V'_{N+1}, i \neq j} x^E_{ij} \leq 1 \qquad \forall i \in F' \tag{4.8}$$

$$\sum_{j \in V'_{N+1}, i \neq j} x^E_{ij} - \sum_{j \in V'_0, i \neq j} x^E_{ji} = 0 \qquad \forall i \in V' \tag{4.9}$$

$$\sum_{j \in V_{N+1}, i \neq j} x^{IC}_{ij} - \sum_{j \in V_0, i \neq j} x^{IC}_{ji} = 0 \qquad \forall i \in V \tag{4.10}$$

$$\sum_{j \in V'} x^E_{0j} \leq m_E \tag{4.11}$$

$$\sum_{j \in V} x^{IC}_{0j} \leq m_{IC} \tag{4.12}$$

$$\tau_i + (s_i + t_{ij}) \, (x^E_{ij} + x^{IC}_{ij}) - l_0 \, (1 - x^E_{ij} - x^{IC}_{ij}) \leq \tau_j \qquad \forall i \in V_0, \forall j \in V'_{N+1}, i \neq j \tag{4.13}$$

$$\tau_i + t_{ij} \cdot x^E_{ij} + r \, (B - y_i) \, x^E_{ij} - (l_0 + rB) \, (1 - x^E_{ij}) \leq \tau_j \qquad \forall i \in F', \forall j \in V'_{N+1}, i \neq j \tag{4.14}$$

$$e_i \leq \tau_i \leq l_i \qquad \forall i \in V'_{0,N+1} \tag{4.15}$$

$$u_i - q_i \, (x^E_{ij} + x^{IC}_{ij}) + Q \, (1 - x^E_{ij} - x^{IC}_{ij}) \geq u_j \geq 0 \qquad \forall i \in V'_0, \forall j \in V'_{N+1}, i \neq j \tag{4.16}$$

$$0 \leq u_0 \leq Q \tag{4.17}$$

$$y_i - b_{ij}(u_j) \cdot x^E_{ij} + B \, (1 - x^E_{ij}) \geq y_j \qquad \forall i \in V, \forall j \in V'_{N+1}, i \neq j \tag{4.18}$$

$$B - b_{ij}(u_j) \cdot x^E_{ij} \geq y_j \qquad \forall i \in F'_0, \forall j \in V'_{N+1}, i \neq j \tag{4.19}$$

$$B \geq y_i \geq 0 \qquad \forall i \in V'_{0,N+1} \tag{4.20}$$

$$x^E_{ij}, x^{IC}_{ij} \in \{0, 1\} \qquad \forall i, j \in V'_{0,N+1}, i \neq j \tag{4.21}$$

Constraints (4.7) enforce that each customer visit has exactly one successor. Constraints (4.8) guarantee that each visit to a recharging station is used at most once, i.e., not all recharging visit vertices must be used. Flow conservation constraints (4.9) and (4.10) guarantee for each vertex that the number of incoming arcs is equal to the number of outgoing arcs for each vehicle type. Constraints (4.11) and (4.12) ensure that the maximum number of employed vehicles adheres to the fleet composition. Constraints (4.13) and (4.14) link arrival times at vertices i and j if the arc from i to j is traveled. Constraints (4.14) cover the case with recharging visits: Here, recharging times that depend on the remaining charge level y_i when arriving at station i have to be taken into account. Constraints (4.15) guarantee

arrival within the time window at each vertex. Constraints (4.16) enforce the fulfillment of demand at customer vertices. Constraint (4.17) restricts the initial cargo load level to the maximum capacity of a vehicle.

Constraints (4.18) set the battery level at a vertex succeeding a customer visit in accordance with the energy consumption for discharging or recuperating. For vertices succeeding a visit to a recharging station or the depot, constraints (4.19) set the battery level equal to the maximum battery capacity reduced by the energy required on the respective arc. Here, the battery is either discharged or recuperation takes place. Constraints (4.20) restrict the charge level to the maximum battery capacity in order to prevent recuperation beyond the maximal capacity. Finally, binary decision variables are defined in constraints (4.21).

Although our model covers several real-world aspects of ECVs, we still use some simplifications. Gradient and speed are considered fixed over the course of an arc. However, a more fine-grained topology and acceleration patterns can be integrated, e.g., by considering a path $p(i,j)$ instead of the arc (i,j). A path $p(i,j)$ is defined by a fixed sequence of $n(i,j)$ intermediate vertices $(v_0 = i, v_1, \ldots, v_{n(i,j)-1}, v_{n(i,j)} = j)$ located on arc (i,j), where each of the vertices marks a change in gradient or acceleration. Let $A(p(i,j))$ denote the arcs of path p. Now, we can use the energy consumption function for ECVs in Equation (4.2) to calculate the energy consumption on each arc $a \in A(p(i,j))$ of the path p:

$$b_a(u_j) = P_a(u_j) \cdot t_a \cdot \begin{cases} \phi^d \cdot \varphi^d & \text{if } P_a(u_j) \geq 0\ kW, \\ \phi^r \cdot \varphi^r & \text{if } P_a(u_j) < 0\ kW. \end{cases}$$

Now, given the energy consumptions on each arc of path $p(i,j)$, we can recursively define the energy consumption on arc (i,j) taking into account that (i) recuperation is restricted by the total battery capacity, and (ii) that we are not allowed to have a negative remaining battery level at any point traveling p. To do so, we let y_{v_k} denote the battery level at each intermediate vertex v_k and define:

$$y_{v_0}(u_j) = y_i,$$

$$y_{v_k}(u_j) = \begin{cases} \min(y_{v_{k-1}}(u_j) - b_{v_{k-1}v_k}(u_j), B) & \text{if } y_{v_{k-1}}(u_j) - b_{v_{k-1}v_k}(u_j) \geq 0 \\ -\infty & \text{else} \end{cases} \quad k = 1, \ldots, n(i,j),$$

$$b_{ij}(u_j) = y_{v_0}(u_j) - y_{v_{n(i,j)}}(u_j).$$

The original Equation (4.3) for calculating the fuel consumption of ICCVs can be adapted in analoguous fashion. In this way, the energy consumption on an arc (i,j) is defined by means of the energy consumptions on the path $p(i,j)$, which can be used to represent gradients and accelerations in an arbitrarily fine-grained fashion.

Our model uses additional simplifications. We neglect the influence of outside temperatures on battery capacity. A related issue is the need to power the car heating, which requires additional energy in contrast to ICCVs, for which the waste heat is sufficient to ensure a comfortable temperature in the driver cabin. Further, we neither consider the dependency between charge level and battery efficiency (Hoke et al., 2011), nor the non-linear relationship between charge level and recharging duration (Marra et al., 2012), which are present in real-world recharging processes.

4.5. Adaptive Large Neighborhood Search for Solving E-VRPTWMF

This section details our ALNS for addressing E-VRPTWMF. Large neighborhood search (LNS), originally introduced by Shaw (1998), iteratively destroys and repairs potentially larger parts of an initial solution in order to gradually improve the solution. Ropke and Pisinger (2006b) proposed ALNS, an approach that uses several destroy and repair methods and selects the operators in each iteration based on a probability that depends on the previous success of the respective method.

Our ALNS is inspired by the work of Ropke and Pisinger (2006b) but differs regarding several aspects. First, most ALNS implementations operate on feasible solutions (see, e.g., Ropke and Pisinger, 2006b; Ropke and Pisinger, 2006a; Demir, Bektaş, and Laporte, 2012) and do not reinsert removed customers if their insertion leads to a constraint violation. Contrary to this, we always insert all removed customers in each ALNS step and handle the resulting infeasible solutions by means of a penalty mechanism (cp. Hemmelmayr, Cordeau, and Crainic, 2012; Dayarian et al., 2013). Second, we select the number of customers for removal from dynamically changing intervals. Third, our ALNS features a local search component, which is used for intensification, while the main purpose of the ALNS component is the diversification of search. Finally, we introduce a new mechanism for the acceptance decision of the ALNS, where solutions with objective function values calculated based on different penalty factors are compared.

Figure 4.2 shows a pseudocode overview of our solution method. First, a preprocessing step is applied to determine arcs that cannot be part of a feasible solution (Section 4.5.1). Next, we generate an initial solution S with a given number of ECVs and ICCVs (Section 4.5.3). The initial solution and solutions in the improvement phase may be infeasible and are handled through a penalty mechanism (Section 4.5.2). ALNS and LS are iteratively applied to improve the solution (Section 4.5.4).

```
1:  preprocessArcList()
2:  S ← generateInitialSolution(m_E, m_IC)
3:  while termination criterion not met do
        {Draw number of customers δ to remove from intervals. Probabilities of intervals are given by π^||.}
4:      δ ← drawNumberOfCustomersToRemove(π^||)
        {Apply randomly selected destroy operator. Probabilities of destroy operators are given by π^−.}
5:      S' ← applyDestroy(S, δ, π^−)
        {Apply randomly selected repair operator. Probabilities of repair operators are given by π^+.}
6:      S' ← applyRepair(S', π^+)
        {Apply local search. Penalty factors are updated based on a dynamic mechanism.}
7:      S' ← applyLocalSearch(S')
8:      if acceptSA(S', S) then
9:          S ← S'
10:     end if
11:     σ^||, σ^−, σ^+ ← updateScores(S')
12:     π^||, π^−, π^+ ← updateSelectionProbabilites(σ^||, σ^−, σ^+)
13: end while
```

Figure 4.2.: Overview of the ALNS algorithm.

4.5.1. Preprocessing

In the preprocessing step, we remove all arcs that cannot be part of a feasible solution, i.e., their inclusion leads to a constraint violation. An arc (v, w) is infeasible if one of the following conditions holds:

1. $v, w \in V \wedge q_v + q_w > Q$,
2. $v \in V'_0, w \in V'_{N+1} \wedge e_v + s_v + t_{vw} > l_w$,
3. $v \in V'_0, w \in V' \wedge e_v + s_v + t_{vw} + s_w + t_{wN+1} > l_0$,
4. $v \in V'_0, w \in V' \wedge \alpha_{vw} \geq 0 \wedge b_{vw}(q_w) > B$,
5. $v \in V'_0, w \in V' \wedge \forall j \in F'_0, i \in F'_{N+1}\colon \alpha_{jv}, \alpha_{vw}, \alpha_{wi} \geq 0 \wedge b_{jv}(q_v + q_w) + b_{vw}(q_w) + b_{wi}(0) > B$.

Rules 1–3 eliminate arcs because of capacity and time window violations (cp. Schneider, Stenger, and Goeke, 2014). Rules 4 and 5 are specific to E-VRPTWMF and only hold for ECV routes. Rule 4 determines those arcs that cannot be traveled without violating battery capacities, even if only the demand q_w has to be transported on the arc. Note that a gradient $\alpha_{vw} \geq 0$ is necessary as otherwise recuperation might render the considered arc feasible. Rule 5 additionally considers visits to the best possible recharging station before and after the arc. Note that Rule 4 is not contained in Rule 5 as we have to make the additional assumption that all involved gradients α_{jv}, α_{wi} are greater or equal to zero.

As Rules 4 and 5 are only valid for ECV routes, we keep two separate reduced arc sets: The set A^- contains all arcs that are feasible for ICCVs. The set $A_E^- \subseteq A^-$ excludes arcs that are not feasible in ECV routes.

4.5.2. Generalized Cost Function and Penalty Calculation

During the search, we allow infeasible solutions in order to be more flexible in traversing the solution space. The objective function of a solution S is computed by a generalized cost function $f_{gen}(S)$ (see, e.g., Gendreau, Hertz, and Laporte, 1994):

$$f_{gen}(S) = f(S) + \gamma_{cap} \cdot L_{cap}(S) + \gamma_{tw} \cdot L_{tw}(S) + \gamma_{batt} \cdot L_{batt}(S).$$

$f(S)$ denotes the considered objective function: total traveled distance $f_d(S)$ (Equation 4.4), routing costs without battery deprecation $f_c(S)$ (Equation 4.5), or routing costs including battery deprecation $f_{cb}(S)$ (Equation 4.6). Violations of capacity $L_{cap}(S)$, time windows $L_{tw}(S)$ and battery capacity $L_{batt}(S)$ are scaled by penalty factors γ_{cap}, γ_{tw} and γ_{batt}.

The penalty factors are dynamically adjusted as follows: If a constraint has been violated for β_{pen} iterations, the respective penalty factor is multiplied by a factor ϱ to guide the search towards feasibility. In analogous manner, the respective penalty factor is divided by ϱ if the constraint is met for β_{pen} iterations. The penalty factors are initially set to $\gamma^0_{cap} = \gamma^0_{tw} = \gamma^0_{batt} = 100$ and are restricted to the intervals $[\gamma^{min}_{cap}, \gamma^{max}_{cap}]$, $[\gamma^{min}_{tw}, \gamma^{max}_{tw}]$ and $[\gamma^{min}_{batt}, \gamma^{max}_{batt}]$ respectively. In order to scale the penalty factors to the problem setting and the order of magnitude of the objective function value, we set γ^{max}_{cap},γ^{max}_{tw} and γ^{max}_{batt} to the objective function value of the initial solution. The lower limits γ^{min}_{cap},γ^{min}_{tw} and γ^{min}_{batt} are set to 0.5.

The efficient calculation of changes in constraint violations caused by a LS move is crucial for the quality of our algorithm. Vehicle capacity violations are not affected by the energy consumption model

considered in this work and therefore changes in capacity violations can be calculated in constant time as described in (Kindervater and Savelsbergh, 1997). However, the considered energy consumption model influences the calculation of changes in battery capacity and time window violations for ECV routes. ICCV routes are also affected if the objective functions f_{cb} or f_c are used.

Let a sequence of customers $\langle v_0, v_1, \ldots, v_n, v_{n+1} \rangle$, with v_0 and v_{n+1} representing the depot, define a route r. A solution S is defined as the union of the set of ECV routes $S_E = \{r_k, k = 1, ..., m_E\}$ and the set of ICCV routes $S_{IC} = \{r_k, k = m_E + 1, ..., m_E + m_{IC}\}$, i.e., $S = S_E \bigcup S_{IC}$. Let $Vert(r)$ denote the set of vertices that are part of route r.

We are interested in the constraint violations of a route that is constructed by concatenating two partial routes $\langle v_0, \ldots, u \rangle$ and $\langle w, \ldots, v_{n+1} \rangle$ or by inserting a vertex v between the two partial routes. To calculate battery capacity violations of ECV routes, we define the following two variables for each vertex of a route r: $\Upsilon^{\rightarrow}_{v_i}$ is the battery charge that is needed to travel either from the previous visit to a recharging station or from the depot (in case no recharging visit is part of the route) to vertex v_i. $\Upsilon^{\leftarrow}_{v_i}$ is the battery charge that is needed to travel from v_i to either the next recharging station or the depot. The variables are calculated as follows:

$$\Upsilon^{\rightarrow}_{v_i} = \begin{cases} \max(b_{v_{i-1}v_i}(u_{v_i}), 0) & \text{if } v_{i-1} \in F'_0 \\ \max(\Upsilon^{\rightarrow}_{v_{i-1}} + b_{v_{i-1}v_i}(u_{v_i}), 0) & \text{if } v_{i-1} \notin F'_0 \end{cases} \qquad i = 1, \ldots, n+1,$$

$$\Upsilon^{\leftarrow}_{v_i} = \begin{cases} \max(b_{v_i v_{i+1}}(u_{v_{i+1}}), 0) & \text{if } v_{i+1} \in F'_{n+1} \\ \max(\Upsilon^{\leftarrow}_{v_{i+1}} + b_{v_i v_{i+1}}(u_{v_{i+1}}), 0) & \text{if } v_{i+1} \notin F'_{n+1} \end{cases} \qquad i = 0, \ldots, n.$$

The battery capacity violation of a solution S can be calculated by summing up the individual violations at every visit to a recharging station and on return to the depot:

$$L_{batt}(S) = \sum_{r_k \in S_E} \Big(\sum_{v_i \in Vert(r_k) \cap F'_{N+1}} \max\left(\Upsilon^{\rightarrow}_{v_i} - B, 0\right) \Big).$$

If energy consumption is independent of cargo load, changes in battery capacity violation can be calculated in $\mathcal{O}(1)$ for conventional neighborhood operators (see Schneider, Stenger, and Goeke, 2014). If load is considered, concatenation of partial routes or inserting a customer vertex between partial routes alter the energy consumption of all arcs preceding the point of route concatenation or vertex insertion. Thus, individual battery capacity violations at each recharging visit before and the next visit directly after the point of change are modified and therefore necessitate a recalculation. To lower the computational burden, we substitute a surrogate violation $\tilde{L}_{batt}(S)$ for the real battery capacity violation $L_{batt}(S)$. The surrogate assumes that violations at the preceding recharging visits remain unmodified. In this way, we are able to compute surrogates of the battery capacity violations in $\mathcal{O}(1)$. Note that the insertion of a recharging station can be handled in constant time without using surrogates.

Time window violations are calculated based on the principle of time travel described in (Nagata, Bräysy, and Dullaert, 2010; Schneider, Sand, and Stenger, 2013), which allows to compute the change in time window violation for conventional inter-route moves in $\mathcal{O}(1)$. In short, the violation of a time window is

counted once at the vertex v where the violation occurs and for the calculation of consecutive violations the vehicle is assumed to start service at the latest feasible moment l_v. Schneider, Stenger, and Goeke (2014) adapt this principle to the E-VRPTW. Here, if the partial route $\langle w, \ldots, v_{n+1} \rangle$ contains a recharging station z, i.e., $\langle w, \ldots, z, z+1, \ldots, v_{n+1} \rangle$, variables have to be recalculated for $\langle v, \ldots, z+1 \rangle$ in case of vertex insertion and for $\langle w, \ldots, z+1 \rangle$ for route concatenation.

In case of the E-VRPTWMF, for both vertex insertion and route concatenation, a recalculation from the beginning of the route to the first recharging station contained in the second partial route, i.e., of the partial route $\langle v_0, \ldots, z+1 \rangle$, becomes necessary. This is because the amount of required energy (affected by the change in mass along the route as explained above) is needed to calculate recharging times. We again use a surrogate $\tilde{L}_{tw}(S)$ to reduce computational effort and assume time window violations of vertices preceding the point of change to remain unmodified.

The generalized cost function using surrogates for both battery capacity violation and time window violation is denoted as $\tilde{f}_{gen}$.

4.5.3. Generation of Initial Solution

We generate an initial solution with $m = m_E + m_{IC}$. First, we build m ECV routes, of which m_{IC} routes are later converted to ICCV routes. Each route is initialized with a seed customer. The seed customers are determined by first sorting the customers in ascending order of their latest start time of service l_v and then selecting the first m entries. The remaining customers in the list are successively inserted into the existing routes at the cost-minimal position according to the generalized cost function $f_{gen}(S)$. As soon as a route becomes battery capacity infeasible, recharging visits are added at cost-minimal positions (according to $f_{gen}(S)$) until the route again becomes battery feasible.

After all customers are inserted into routes, we convert long routes with strong constraint violations to ICCV routes by removing recharging visits. More precisely, we sort the routes in descending order of their contribution to the generalized cost function and select the first m_{IC} entries for conversion.

4.5.4. Solution Improvement

The steps of the solution improvement phase are shown in Figure 4.2. First, we select the number of customers to be removed in the ALNS step. This number is commonly defined as percentage of the overall number of customers N and is usually drawn from a given interval $[\Omega_{min}, \Omega_{max}]$. In preliminary studies, we found that the size of this interval influences the quality of the solution and that a good choice for the interval is instance-dependent. Therefore, we use an adaptive mechanism to select promising intervals (in analogous fashion to the selection of destroy and repair operators). To this end, we split the original interval in ω subintervals without overlap. ALNS then chooses a subinterval according to the probabilities $\pi_i^{||}$, $i = 1, ..., \omega$. From the subinterval, we draw a random number δ of customers and/or stations. Afterwards, we consecutively apply a destroy and repair operator (Section 4.5.4.1).

The resulting solution is improved by a LS step (Section 4.5.4.2), which determines a locally optimal solution S' using dynamic penalties as described in Section 4.5.2. Solution acceptance is based on a simulated annealing (SA) criterion (Section 4.5.4.3). Finally, selection probabilities of the destroy and repair

operators, and of the intervals for the numbers of customers to remove are adjusted (Section 4.5.4.4). To find a feasible solution, the search is run for at most β_{feas} iterations. After a feasible solution is found, the search continues for another β_{obj} to improve the solution.

4.5.4.1. Destroy and Repair Operators

Our ALNS uses the following destroy operators:

Random removal removes arbitrary vertices. We implement two versions, the first only removes customers, the second removes customers and recharging visits.

Worst removal was introduced by Ropke and Pisinger (2006b) and aims at removing vertices that are unfavorable at their current position in a route. We sort all vertices in descending order of their contribution to the cost of the current solution, which is determined as the change in $\tilde{f}_{gen}$ that is caused by removing the respective vertex. From this list the vertex at position $\lfloor D \cdot b^{\chi_{worst}} \rfloor$ is chosen, where D is the size of the list, b is a uniform random number $\in [0, 1]$ and χ_{worst} a parameter to weight the impact that the change in objective value has on the selection. Again a version considering customers and a version considering customers and recharging visits is implemented.

Shaw removal was introduced in (Shaw, 1997). The idea is to remove customers that are similar to each other with respect to several criteria and are thus likely to be interchangeable. We define the relatedness between two customers i and j by geographical distance d_{ij}, difference in demand $|q_i - q_j|$ and difference of the earliest start time of service $|e_i - e_j|$. Each criterion is weighted with a parameter χ and normalized. The relatedness can be calculated as:

$$R(i,j) = \chi_d \frac{d_{ij}}{\max\limits_{i,j \in V}(d_{ij})} + \chi_q \frac{|q_i - q_j|}{\max\limits_{i \in V}(q_i) - \min\limits_{i \in V}(q_i)} + \chi_e \frac{|e_i - e_j|}{\max\limits_{i \in V}(e_i) - \min\limits_{i \in V}(e_i)}.$$

The first customer to be removed is selected randomly. Then, in each iteration, one customer is drawn randomly from the already removed customers and all non-removed customers are sorted in ascending order of their relatedness value to the selected customer (a small value of $R(i,j)$ corresponds to a high relatedness). From this list, the customer at position $\lfloor D \cdot b^{\chi_{shaw}} \rfloor$ is chosen, where D is the size of the list, b is a uniform random number $\in [0, 1]$ and χ_{shaw} a parameter to weight the impact that relatedness has on the selection. Shaw removal is only defined for customer vertices.

Cluster removal identifies and removes clusters of geographically close vertices and was proposed in (Ropke and Pisinger, 2006a). First, a route is selected at random and vertices contained in the route are separated into two clusters. To this end, Kruskal's algorithm for the minimal spanning tree problem is stopped when two subtrees remain. We randomly select one of the trees as cluster for removal. For the next clustering step, we determine the route closest to a randomly selected customer of the removed cluster. This process is repeated until the required number of vertices is removed. If the overall number of customers in a cluster exceeds the number of vertices that are still to be removed, we randomly select customers from the cluster until the given number is reached. Only one version of cluster removal exists, which does not distinguish between customers and recharging visits.

Station vicinity removal is a special type of the cluster removal operator, which aims at reordering customer visits in the vicinity of recharging stations. We observed that routes in these regions tend to be complex and intertwined because recharging and customer visits have to be scheduled in a manner respecting time window requirements and battery capacity level. To define the vicinity of a station, we randomly choose a radius r in the interval $[\chi^{min}_{radius} \cdot \max_{i,j \in V}(d_{ij}), \chi^{max}_{radius} \cdot \max_{i,j \in V}(d_{ij})]$, i.e., the radius r is determined instance-dependent as percentage χ^{min}_{radius} (χ^{max}_{radius}) of the maximal distance between two customers. Next, we randomly select a recharging station R and remove all customers i with a distance $d_{iR} < r$. This step is repeated until q customers are removed. We implement a second version that also removes all visits to the considered recharging stations.

The following repair operators are used in our ALNS:

Greedy insertion iteratively performs the best possible insertion. For each still unassigned vertex the increase of the surrogate cost function $\tilde{f}_{gen}$ for the insertion at every route position is evaluated and the vertex and position with minimal cost increase are chosen.

Regret insertion was described by Ropke and Pisinger (2006b) in the context of ALNS. The idea is to anticipate the future effect of an insertion operation. The k-regret value for each vertex is calculated as the difference between the insertion cost in the best route and in the k-best route, i.e., k describes the extent to which the future is anticipated. The cost of insertion is evaluated with $\tilde{f}_{gen}$ under the assumption that the insertion is performed at the best possible position within the route. We implement the regret-2 and regret-3 heuristic.

Greedy and Regret insertion are both realized in four variants: the first version only inserts customer vertices, the second version inserts customer vertices and recharging visits, and versions three and four additionally use a diversification penalty to evaluate the costs of inserting a vertex. The diversification penalty is based on the continuous diversification principle introduced in (Cordeau, Laporte, and Mercier, 2001) and later adapted to the E-VRPTW by Schneider, Stenger, and Goeke (2014).

GRASP insertion is based on the greedy randomized adaptive search (GRASP) metaheuristic introduced in (Feo and Resende, 1989). Analogous to greedy insertion, the cost increase for the insertion of all remaining vertices is calculated and the vertices are stored in a list of size D in ascending order of cost increase. Instead of selecting the best vertex for insertion, a random vertex of the first $\lfloor \chi_{grasp} \cdot D \rfloor$ best vertices is selected, where χ_{grasp} denotes the degree of randomness. GRASP only inserts customer vertices.

4.5.4.2. Local Search

The LS follows a best-improvement strategy using a composite neighborhood of Relocate (Savelsbergh, 1992), Exchange (Savelsbergh, 1992), 2-opt* (Potvin and Rousseau, 1995) and an operator for inserting and removing visits to recharging stations, called stationInRe (Schneider, Stenger, and Goeke, 2014). Relocate and Exchange are applied for intra- and inter-route moves, 2-opt* is only defined for inter-route moves. Moreover, Relocate is defined to handle recharging visits and 2-opt* allows to modify routes including recharging visits, while Exchange is explicitly not applied to recharging visits. 2-opt* is not

allowed between ECV and ICCV routes. Relocate moves inserting recharging visits into ICCV routes are forbidden and also stationInRe moves on ICCV routes.

Moves are realized by means of the generator arc principle introduced in (Toth and Vigo, 2003). Moves leading to the inclusion of an infeasible arc, i.e., moves that insert an arc from the set $A \setminus A^{-}$ into an ICCV route or moves inserting an arc from the set $A \setminus A_E^{-}$ into an ECV route, are not evaluated. As described in Section 4.5.2, we use a surrogate cost function whose evaluation is less demanding (see, e.g., Crainic et al., 1993; Gendreau and Potvin, 2010).

In each iteration of the LS, we determine a list $\mathcal{M}$ of the best ϵ solutions that can be generated in the composite neighborhood. The costs used to determine whether a solution is included in the list are either surrogate or exact cost values that were determined in previous iterations (Due to caching, exact values are available for moves that have been evaluated in previous iterations and that only affect routes which have not been changed since the exact evaluation). Then, for each solution with surrogate cost included in the list, we evaluate the exact cost and carry out the best of the ϵ moves based on exact costs.

The LS stops as soon as a local optimum (based on the current penalty factors) is reached.

4.5.4.3. Acceptance Criterion

In order to overcome local optima, ALNS uses an SA-based acceptance criterion (Kirkpatrick, Gelatt, and Vecchi, 1983). Besides always accepting an improving solution, SA accepts a new deteriorating solution with a certain probability that depends (i) on the difference between cost values of the new and the current solution and (ii) a temperature which follows a predefined cooling schedule.

As described above, we allow infeasible solutions and handle them with a dynamic penalty mechanism. Consequently, the cost value of a solution is strongly dependent on the current values of the penalty factors. The dynamic nature of the penalty factors raises the question which penalty factors to use in order to compare the cost value of the current and the new solution. Evaluating both solutions with different penalty factors, e.g., the ones that were valid at the time the solutions were created, can be misleading because differences in cost values may be solely caused by the difference in penalty factors.

Let $(\gamma_{cap}^{S}, \gamma_{tw}^{S}, \gamma_{batt}^{S})$ be the vector of penalty factors used for evaluating $f_{gen}(S)$ and $(\gamma_{cap}^{S'}, \gamma_{tw}^{S'}, \gamma_{batt}^{S'})$ the penalty factors used for evaluating $f_{gen}(S')$ at the time of solution creation. Then, in order to make solutions comparable, the cost values $f_{gen}^{eval}(S)$ and $f_{gen}^{eval}(S')$ to compute the acceptance probability are determined with penalty factors:

$$(\gamma_{cap}^{eval}, \gamma_{tw}^{eval}, \gamma_{batt}^{eval}) = (\frac{1}{2} \cdot (\gamma_{cap}^{S'} + \gamma_{cap}^{S}), \frac{1}{2} \cdot (\gamma_{tw}^{S'} + \gamma_{tw}^{S}), \frac{1}{2} \cdot (\gamma_{batt}^{S'} + \gamma_{batt}^{S})).$$

If solutions stay infeasible for longer periods, penalty factors and thus the absolute difference between cost values of solutions increase. To avoid the undesired effect that these (potentially better) solutions have a low chance to be accepted because of high penalty factors, we use the relative difference between cost values to calculate the acceptance probabilities:

$$\Delta f_{rel}(S', S) = \frac{f_{gen}^{eval}(S') - f_{gen}^{eval}(S)}{f_{gen}^{eval}(S)}.$$

Thus, we always accept improving solutions, while, given the current temperature T, deteriorating solutions are accepted with probability:

$$p(S, S', T) = e^{\frac{-\Delta f_{rel}(S', S)}{T}}.$$

The starting temperature is chosen in a fashion that a solution that deteriorates the current solution by $\iota\%$ is accepted with a probability of 50%. The temperature is decreased by a constant factor such that the temperature is below $T_{threshold} = 0.0001$ in the last 20% of iterations.

4.5.4.4. Adaptive Mechanism

Adaptive weight adjustment assesses the importance of each destroy and repair operator by modifying the probability π with which the respective operator is chosen based on the previous performance of the operator. In more detail, we periodically adjust the probabilities of the destroy operators (π_i^-), of the repair operators (π_i^+), and of the intervals from which the number of customers to remove is drawn ($\pi_i^{||}$).

Performance is measured by assigning scores σ_i to the destroy and repair operators and the intervals. If an operator or interval i is used in the current iteration of ALNS, we increase the respective score by σ_A if the following LS results in a new global best solution, by σ_B if a new improving solution is found, and by σ_C if a new deteriorating solution is found and accepted by the SA-based criterion. We consider a solution to be new if we did not obtain the cost value $f_{gen}(S')$ before (evaluated with a standard set of penalty factors $\gamma_{cap}, \gamma_{tw}, \gamma_{batt} = 1$).

After β_π iterations, we update the weights $\lambda_i^{||}$, λ_i^- and λ_i^+ to:

$$\lambda_i := w\frac{\sigma_i}{n_i} + (1 - w)\lambda_i,$$

where w denotes the weight of the current phase of evaluation, $(1 - w)$ the weight put on the previous success of the operator or interval, and n_i the number of times the operator or interval was selected during the last β_π iterations. Then, the scores of each operator or interval are reset 0. Subsequently, we determine the corresponding probabilities $\pi_i^{||}$, π_i^- and π_i^+ to:

$$\pi_i = \frac{\lambda_i}{\sum_i \lambda_i}.$$

4.6. Numerical Studies

We conduct numerical experiments on two different types of instances: First, we use newly generated E-VRPTWMF instances to investigate (i) the influence of using the surrogate objective function on the solution process, (ii) the effect of considering the actual distribution of load in comparison to assuming constant load over all traveled arcs, and (iii) the influence of different objective functions on the structure and costs of generated solutions and the share that ECVs contribute to this cost. Second, to assess the performance of ALNS in terms of solution quality and run-time, we conduct tests on available benchmark instances of the related problems VRPTW and E-VRPTW.

The section is structured as follows. The parameter tuning is presented in Section 4.6.1, the generation of the E-VRPTWMF benchmark instance in Section 4.6.2, the tests performed on these instances in Section 4.6.3, and the tests on the benchmark instances of related problems in Section 4.6.4.

4.6.1. Experimental Environment and Parameter Setting

All experiments are conducted on a desktop computer with an Intel Core i7 processor at 2.8 GHz, 8 GB of RAM, running Windows 7 Enterprise. Our ALNS is implemented as single-thread code in Java. The parameter setting of the ALNS is determined using a subset of 20 randomly selected E-VRPTWMF instances with 50 and 100 customers (details on the generation of E-VRPTWMF instances are given in Section 4.6.2). Parameter tests are carried out using the objective of minimizing traveled distance.

During the development of our algorithm, we identified (i) a set of parameters that have a stronger influence on the solution quality of the algorithm, and (ii) for each of these parameters a well-performing base setting. In the parameter tuning, we investigate the influence of modifying this base setting in two directions, i.e., three settings are tested per parameter. More precisely, we consecutively assess each of the parameters and always keep the best setting found for this parameter for the tuning of the remaining parameters. The order in which the parameters are tuned is determined randomly. Table 4.2 shows, for each parameter, the three tested settings and the deviation Δ_{f_d} of the best solution found in 10 runs with the respective setting to the result obtained with the best setting for this parameter. The best setting for each parameter is given in bold.

Table 4.2.: Results of different parameter settings on a subset of the E-VRPTWMF instances. The best setting for each of the parameters is marked in bold and used as final setting. Results for the tested settings for each parameter are given as deviation of the best solution achieved in 10 runs with the tested setting to the result obtained with the best setting for the respective parameter (Δ_{f_d}).

ALNS			
β_π	25	50	**75**
$\Delta_{f_d}(\%)$	0.03	0.02	0.0
$(\sigma_A, \sigma_B, \sigma_C)$	(20, 10, 10)	**(10, 20, 10)**	(5, 5, 10)
$\Delta_{f_d}(\%)$	0.0	0.0	0.10
w	0.2	0.4	**0.6**
$\Delta_{f_d}(\%)$	0.28	0.23	0.0
Ω_{max}	30%	40%	**50%**
$\Delta_{f_d}(\%)$	0.04	0.11	0.0
$(\chi^{min}_{radius}, \chi^{max}_{radius})$	(1%, 15%)	(5%, 15%)	**(5%, 20%)**
$\Delta_{f_d}(\%)$	0.13	0.01	0.00
ι	10%	**30%**	50%
$\Delta_{f_d}(\%)$	0.02	0.0	0.08
LS			
ϱ	1.2	1.5	**2.0**
$\Delta_{f_d}(\%)$	0.10	0.14	0.0
ϵ	**25**	50	75
$\Delta_{f_d}(\%)$	0.0	0.10	0.09

In the described fashion, we test the following parameters: the number of iterations after which the weights of ALNS are adjusted (β_π), the scores used in the adaptive weight adjustment ($\sigma_A, \sigma_B, \sigma_C$), the

reaction factor (w) of the weight adjustment, the maximal percentage of removed customers (Ω_{max}), the minimal and maximal radius for the station vicinity removal operator ($\chi^{min}_{radius}, \chi^{max}_{radius}$), the percentage of deterioration that is accepted at the beginning of SA (ι), the penalty update factor (ϱ), and the number of solutions in each iteration that are evaluated with the original objective function instead of the surrogate (ϵ). We observe that none of the tested settings leads to a significant deterioration of solution quality (the deviation is always below 0.28%). Consequently, our ALNS seems relatively insensitive to parameter variations.

In order to achieve a good tradeoff between run-time and solution quality, we set the maximal number of ALNS iterations to find a feasible solution to $\beta_{feas} = 2000$ and the number of iterations for further improvement to $\beta_{obj} = 750$ for all instances with less than 100 customers. For larger instances, we use $\beta_{obj} = 750/(0.01 \cdot |N|)^{1.5}$ in order to keep run-times acceptable. Table 4.3 summarizes the final parameter setting, which is used for all of the following experiments.

Table 4.3.: Final parameter setting of ALNS used in the numerical studies.

ALNS		**ALNS**		**LS**	
β_{feas}	2000	$(\Omega_{min}, \Omega_{max})$	(10%, 50%)	ϵ	25
β_{obj} ($\|N\| \leq 100$)	750	ω	5	$\gamma^0_{cap}, \gamma^0_{tw}, \gamma^0_{batt}$	100
β_{obj} ($\|N\| > 100$)	$750/(0.01 \cdot \|N\|)^{1.5}$	$(\chi^{min}_{radius}, \chi^{max}_{radius})$	(5%, 15%)	$\gamma^{min}_{cap}, \gamma^{min}_{tw}, \gamma^{min}_{batt}$	0.5
β_{π}	75	χ_{grasp}	0.2	$\gamma^{max}_{cap}, \gamma^{max}_{tw}, \gamma^{max}_{batt}$	$f_{gen}(S_{inital})$
$(\sigma_A, \sigma_B, \sigma_C)$	(10, 20, 10)	χ_{worst}	6	ϱ	2.0
w	0.6	χ_{shaw}	6	β_{pen}	2
ι	30%	(χ_d, χ_q, χ_e)	(6, 4, 5)		

4.6.2. Generation of E-VRPTWMF Instances

This section describes the generation of E-VRPTWMF test instances. The instances are based on the PRP instances proposed in (Demir, Bektaş, and Laporte, 2012). The customer locations in these instances represent real cities in the UK, customer demands and time windows are generated in random fashion. The benchmark consists of 9 instance sets grouped according to the size of the instances contained in each set (10–200 customers). Each set contains 20 instances with the respective size. Contrary to the PRP, speed is not a decision variable in E-VRPTWMF and we set a fixed speed that corresponds to the upper speed limit given in the PRP instances, i.e, we set $v_{ij} = 90\ km/h,\ \forall i, j \in V'_{0,N+1}$. In this way, we also ensure that all instances can be feasibly solved (with respect to time windows) when using a fleet of ICCVs.

The number of recharging stations is adapted to the number of customers N of the instances. More precisely, we locate $\lfloor 0.1 \cdot |N| \rfloor$ recharging stations at randomly drawn customer locations and one additional station at the depot. To determine the number of ECVs and ICCVs of each instance, we proceed as follows: We start with a total vehicle number m that corresponds to the number of ICCVs given in the PRP solutions of Demir, Bektaş, and Laporte (2012), and gradually substitute ICCVs with ECVs until either $m_E = \lfloor m/2 \rfloor$, or our ALNS is no longer able to find a feasible solution.

We use the physical constants, vehicle properties and coefficients provided in (Demir, Bektaş, and Laporte, 2012). The battery capacity is set to 80 kWh, which is a realistic for ECVs according to Davis and Figliozzi (2013). Efficiency parameters for engine and battery during discharge and recuperation are

determined as described in Section 4.3.1. The recharging power at stations is assumed equivalent to that of the Tesla supercharger network for passenger cars (Tesla Motors, Inc., 2014).

When using the objectives f_c or f_{cb}, we assume the following cost structure: Driver costs are equal for ICCVs and ECVs and correspond to the time spend en route. Hourly wages are given in (Demir, Bektaş, and Laporte, 2012). The diesel price, electricity price and costs for battery replacement used for calculating driving costs are taken from (Davis and Figliozzi, 2013). To determine the battery replacement cost on a per kilometer basis, we assume that a battery has to be replaced after 241350 kilometers (150000 miles) at a cost of 600 $\$/kWh$. Table 4.4 summarizes the data.

Table 4.4.: Data of E-VRPTWMF instances.

Notation	Value	Description	Source
g	$9.81\ m/s^2$	gravitational constant	Demir, Bektaş, and Laporte (2012)
ρ	$1.2041\ kg/m^3$	air density	Demir, Bektaş, and Laporte (2012)
A	$3.912\ m^2$	frontal surface	Demir, Bektaş, and Laporte (2012)
m_c	$6350\ kg$	curb mass	Demir, Bektaş, and Laporte (2012)
m_u	$1\ kg$	mass per unit	
Q	3650	capacity	Demir, Bektaş, and Laporte (2012)
c_r	0.01	coefficient rolling resistance	Demir, Bektaş, and Laporte (2012)
c_d	0.7	coefficient aerodynamic drag	Demir, Bektaş, and Laporte (2012)
k	$0.2\ kJ/(rev \cdot l)$	engine friction factor	Demir, Bektaş, and Laporte (2012)
ξ	1	fuel-to-air mass ratio	Demir, Bektaş, and Laporte (2012)
η	0.9	efficiency parameter diesel engines	Demir, Bektaş, and Laporte (2012)
η_{tf}	0.4	efficiency parameter drive train	Demir, Bektaş, and Laporte (2012)
κ	$44\ kJ/g$	heating value diesel fuel	Demir, Bektaş, and Laporte (2012)
ψ	$737\ l/g$	conversion factor	Demir, Bektaş, and Laporte (2012)
N	$33\ rev/s$	engine speed	Demir, Bektaş, and Laporte (2012)
D	$5\ l$	engine displacement	Demir, Bektaş, and Laporte (2012)
B	$80\ kWh$	battery capacity	Davis and Figliozzi (2013)
r	$120^{-1}\ h/kW$	recharging rate	Tesla Motors, Inc. (2014)
ϕ^d	1.184692	efficiency parameter (motor mode)	see Section 4.3.1
ϕ^r	0.846055	efficiency parameter (generator mode)	see Section 4.3.1
φ^d	1.112434	efficiency parameter discharging	see Section 4.3.1
φ^r	0.928465	efficiency parameter recuperation	see Section 4.3.1
c^D	$13.1368\ \$/h$	driver wages, conversion ratio $1.6421\ \$/£$	Demir, Bektaş, and Laporte (2012)
c^{IC}	$1.07\ \$/l$	diesel fuel cost	Davis and Figliozzi (2013)
c^E	$0.1106\ \$/kWh$	electricity cost	Davis and Figliozzi (2013)
c^B	$0.1989\ \$/km$	battery replacement cost	see above

4.6.3. Experiments on E-VRPTWMF Instances

We conducted tests with our ALNS for the complete set of instances described above. The detailed results are given in Appendix C as comparison values for researchers addressing the same instances in the future. In this section, we describe the numerical tests that investigate the research questions introduced above:

1. impact of using the surrogate objective function (Section 4.6.3.1)
2. effect of considering load in the calculation of energy consumption (Section 4.6.3.2)
3. influence of different objective functions and the resulting level of usage of ECVs (Section 4.6.3.3).

4.6.3.1. Influence of Surrogate Cost Function

As described in Section 4.5.4.2, the LS determines the best move in each iteration based on exactly evaluating a list $\mathcal{M}$ of promising candidate moves that were found using a surrogate cost function $\tilde{f}_{gen}$. We assess the impact of using the surrogate cost function (i) on the course of the search, and (ii) on the solution quality of our ALNS.

First, we conduct 50 iterations of ALNS on the 100-customer instance E-UK_100_03 with the objective of minimizing traveled distance. The size of the list $\mathcal{M}$ is set to $\epsilon = 50$. In each iteration, we calculate the best move based on the surrogate cost function, i.e., we fill $\mathcal{M}$ using the surrogate cost, then determine the exact costs of the entries in $\mathcal{M}$ and select the best entry. The quality of this move is assessed by determining its rank among the possible moves in this iteration if all moves were evaluated exactly (not only those in $\mathcal{M}$).

In Figure 4.3, we plot these ranks. Applications of destroy and repair operators are marked with an "x". Thus, we identify how often and to what degree we miss the best possible move. In 45% of iterations we choose the rank 1 move and in 59% of iterations we choose one of the top 3 moves. Only in 3.5% of iterations, the selected move is not among the top 10.

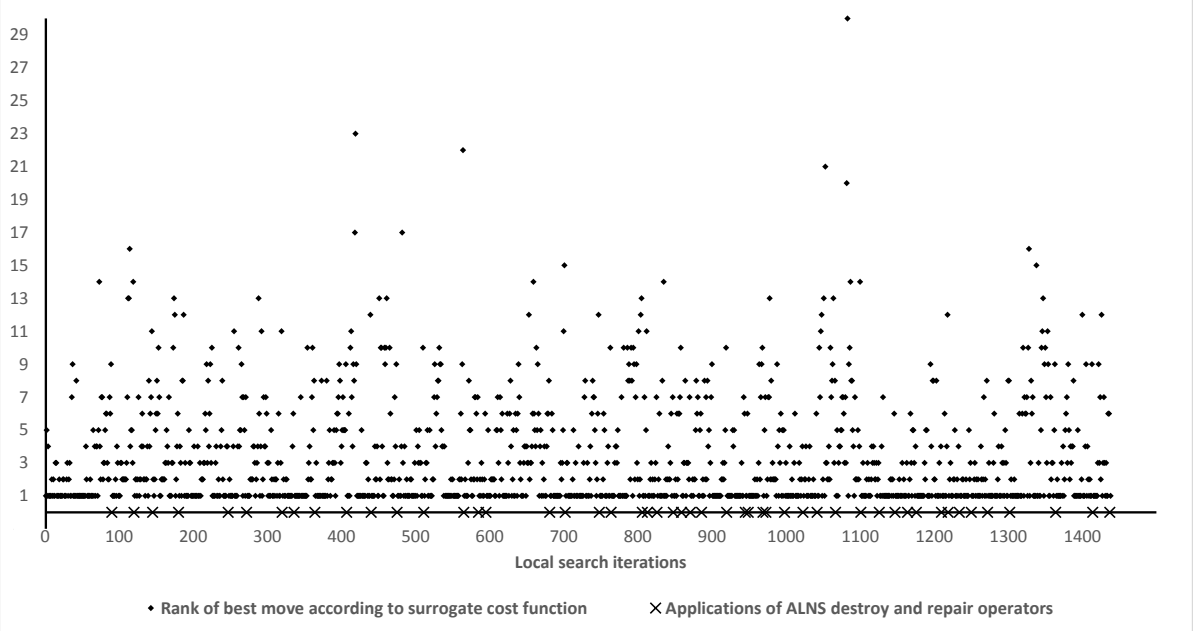

Figure 4.3.: Effect of using the surrogate cost function $\tilde{f}_{gen}$ to determine the best move in 50 ALNS iterations on the 100-customer instance E-UK_100_03. The figure depicts the rank of the selected best move among all possible moves evaluated with the exact cost function.

Second, we carry out additional experiments to investigate the effect of using the surrogate cost function on the solution quality of ALNS by comparing the solutions obtained with the surrogate and with the exact cost function. To keep computation times reasonable, the tests are run on a reduced subset of the E-VRPTWMF instances, which is obtained by randomly selecting 5 out of 20 instances for each problem size. Due to the very high run-times when using the exact cost function, we further have to restrict the size of the considered instances to maximally 75 customers for this study. Summarizing, we randomly select 5 of the 20 instances for the problem sizes $|N| = 5, 10, 15, 20, 25, 50, 75$ and conduct 10 runs for

each instance with the surrogate cost function and with the exact cost function, following the objective of minimizing traveled distance. The size of the list $\mathcal{M}$ is again set to $\epsilon = 50$.
For the exact cost function, Table 4.5 reports the best solution found in the 10 runs (f_d^{best}), the average solution quality (f_d^{avg}) and the average run-time in minutes (t_{avg}) as averages of groups of instances of the same size. For the surrogate cost function, the same measures are evaluated and the solution quality is reported as gaps Δ_d^{best} and Δ_d^{avg} to the solutions obtained with the exact cost function. The run-time using the surrogate cost function is presented using a run-time reduction factor $\mathcal{T}$, which is obtained by dividing the average run-time using the exact cost function by the average run-time using the surrogate cost function. At the bottom of the table, average results over the investigated instance groups are given. The results show that using the surrogate cost function has only minimal negative effect on the solution quality concerning the best as well as the average solution quality. However, the run-time savings using the surrogate are really impressive with a factor $\mathcal{T}$ of 194 for the 75-customer instances.

Table 4.5.: Comparison of the solution quality using the exact cost function and using the surrogate cost function. Gaps are reported in percent and run-times in minutes.

	exact cost function			surrogate cost function		
Inst. group	f_d^{best}	f_d^{avg}	t_{avg}	Δ_d^{best}	Δ_d^{avg}	$\mathcal{T}$
E-UK10	456.60	456.60	0.13	0.00	0.00	6.45
E-UK15	663.04	663.06	0.73	0.00	0.00	13.94
E-UK20	837.71	837.76	1.97	0.00	0.00	21.41
E-UK25	940.80	941.40	6.07	0.00	-0.03	27.81
E-UK50	1408.05	1415.55	100.08	0.08	0.38	107.94
E-UK75	1932.65	1939.69	419.49	0.15	0.50	194.04

4.6.3.2. Effect of Considering the Distribution of Load on Solution Quality and Cost Estimates

This study investigates the importance of considering the distribution of load when routing a mixed fleet of ICCVs and ECVs. To this end, we compare the quality of the solutions obtained based on the actual load distribution to the solutions obtained with the following two load estimates. First, the cautious load estimate is to assume that all vehicles are fully loaded during their whole trip ($\bar{u} = Q$). This is a worst-case estimate in some sense because it implies maximal energy consumption. This estimate guarantees that all obtained solutions are still feasible with respect to battery capacity and time windows if the actual load distribution is considered. The second load estimate assumes the average case that the vehicles travel half-loaded ($\bar{u} = Q/2$). This may result in solutions that are infeasible when taking into account the actual load due to either violations of battery capacity, i.e., ECVs that get stranded, or due to violations of time windows, which are caused by a wrong estimation of the recharging time of ECVs.

We are interested in:

- the quality of solutions S obtained with the estimates $S(\bar{u} = Q)$ and $S(\bar{u} = Q/2)$ when afterwards evaluating the costs considering the actual load (denoted as $f_c^{u_i}(S)$), and
- the error of determined costs when the solutions are obtained and evaluated based on the load estimates (denoted as $f_c^{\bar{u}=Q}(S)$ and $f_c^{\bar{u}=Q/2}(S)$). These cases are interesting as these are the cost estimates that will often be used in practical planning situations.

We conduct experiments addressing the above points on a reduced subset of the E-VRPTWMF instances

(5 randomly chosen instances out of 20 for each group), again, in order to keep computation times reasonable. For each instance, 10 ALNS runs are performed following the objective of minimizing costs without battery costs f_c and the reported results are based on the best of the 10 runs. Table 4.6 provides for each of the instances the number of ICCVs (m_{IC}) and ECVs (m_E), the cost of the solution obtained based on the actual load distribution and evaluated using the actual load ($f_c^{u_i}(S(u_i))$), and the gap (to the costs $f_c^{u_i}(S(u_i))$) of the solutions obtained considering load estimates and evaluated using (i) the actual load (columns $\Delta f_c^{u_i}(S(\bar{u}=Q))$ and $\Delta f_c^{u_i}(S(\bar{u}=Q/2))$), and (ii) the load estimate (columns $\Delta f_c^{\bar{u}=Q}(S(\bar{u}=Q))$ and $\Delta f_c^{\bar{u}=Q/2}(S(\bar{u}=Q/2))$). Instances for which the solution found with the estimate $\bar{u}=Q/2$ is not feasible if the actual load is considered are marked with a dash. At the bottom of the table, average results over the investigated instances are given.

As mentioned above, using the estimate $\bar{u}=Q$ guarantees feasibility, however, the obtained solutions show an average gap of 1.89% if the actual load distribution is used for evaluating costs. Further, cost evaluation based on this estimate leads to an overestimation of costs by more than 5% on average. On the other hand, using the estimate $\bar{u}=Q/2$ leads to infeasible solutions for 17 out of 45 instances and an average gap of approximately 1% over the instances for which a feasible solution is found. Costs based on this estimate underestimate real costs by -0.94%.

The results show that the quality of solutions obtained using estimates is clearly inferior to the quality of the solutions obtained considering the actual load. Moreover, costs evaluated with estimates strongly differ from real costs. Both deviations are stronger in the case of $\bar{u}=Q$. However, in the context of ECVs, using a load estimate other than this "worst-case estimate" may result in infeasible solutions due to stranded ECVs or time window violations caused by wrong assessment of the required recharging times of ECVs. This is different from ICCV fleet, where load estimates can only lead to wrong cost estimations and wrong routing decisions but not to infeasible solutions. Also note that it may happen that using a certain load estimate, it is not possible to find a feasible solution although a feasible solution exists for the actual load distribution. For example, this may be the case if time windows are very restrictive and ECVs have to travel with very little load in order to avoid recharging operations.

Overall, the study clearly demonstrates the importance of considering load information in the route planning of ECVs from a feasibility as well as a cost perspective. While the problem of stranded ECVs may be counteracted by allotting high safety margins, an adequate consideration of energy consumption allows to better utilize expensive battery capacity. Finally, we note that our ALNS is capable to solve all instances with the same number of vehicles as determined originally for the PPR and we can substitute approximately half of the ICCVs in the fleet ($m_E/(m_E+m_{IC})=47.2\%$) by ECVs.

4.6.3.3. Assessment of Different Objective Functions and Cost Contribution of ECVs

In this study, we compare the solutions obtained with the different objective functions of minimizing traveled distance (f_d), costs without battery depreciation cost (f_c) and costs including battery costs (f_{cb}) in terms of their performance regarding the different objectives. In the second part of the study, we investigate to what extent ECVs are employed when different objective functions are used.

We conduct 10 ALNS runs with each of the objective functions f_d, f_c and f_{cb} on the reduced instance set described above, and we assess the results of the best of the 10 runs for each instance.

Table 4.6.: Effect of considering the distribution of load on the solution quality and cost estimates. For each instance, the number of ICCVs (m_{IC}) and ECVs (m_E) and the results of the best of 10 ALNS runs are reported. $f_c^{u_i}(S(u_i))$ denotes the cost (in dollar) of the solution obtained based on the actual distribution of load and evaluated using the actual load, $\Delta f_c^{u_i}(S(\bar{u}=Q))$ and $\Delta f_c^{u_i}(S(\bar{u}=Q/2))$ the gaps (to the costs $f_c^{u_i}(S(u_i))$) obtained based on load estimates and evaluated using the actual load, and $\Delta f_c^{\bar{u}=Q}(S(\bar{u}=Q))$ and $\Delta f_c^{\bar{u}=Q/2}(S(\bar{u}=Q/2))$ the gaps obtained based on load estimates and evaluated using load estimates. Gaps are reported in percent. Instances with an infeasible solution are marked with a dash.

			$S(u_i)$	$S(\bar{u}=Q)$		$S(\bar{u}=Q/2)$	
Inst.	m_{IC}	m_E	$f_c^{u_i}$	$\Delta f_c^{u_i}$	$\Delta f_c^{\bar{u}=Q}$	$\Delta f_c^{u_i}$	$\Delta f_c^{\bar{u}=Q/2}$
E-UK10_04	1	1	207.14	0.25	4.20	0.25	0.46
E-UK10_09	1	1	188.32	0.06	4.23	0.06	0.39
E-UK10_13	1	1	207.96	0.32	4.52	0.32	0.31
E-UK10_18	1	1	159.86	5.61	9.47	5.61	0.10
E-UK10_20	1	1	165.06	7.60	13.15	7.98	1.35
E-UK15_01	2	0	313.46	0.00	3.38	0.00	-0.33
E-UK15_07	2	1	267.11	0.00	4.36	0.00	0.92
E-UK15_09	2	1	265.72	6.23	9.17	0.09	-4.88
E-UK15_12	2	1	319.79	0.96	5.33	0.96	0.63
E-UK15_15	1	1	231.21	12.64	18.01	0.00	-10.66
E-UK20_05	2	1	316.31	3.00	6.29	2.38	-0.79
E-UK20_06	2	1	366.85	0.36	3.64	0.04	-0.69
E-UK20_09	2	1	337.95	0.00	3.95	0.00	0.36
E-UK20_14	2	2	410.33	6.79	10.92	3.59	-2.74
E-UK20_15	2	1	338.95	0.09	4.07	0.00	0.19
E-UK25_10	2	2	367.30	2.74	6.50	0.26	-2.15
E-UK25_11	2	2	376.18	1.98	5.89	–	-1.54
E-UK25_12	2	2	426.02	2.15	6.15	–	-3.93
E-UK25_16	2	2	348.59	2.61	6.40	0.94	-1.41
E-UK25_19	2	2	441.59	4.07	8.28	–	-2.06
E-UK50_11	4	3	646.02	1.32	5.11	1.03	-0.04
E-UK50_12	4	3	586.88	0.74	4.17	0.24	-0.45
E-UK50_13	4	3	587.82	0.26	3.33	–	-1.21
E-UK50_16	4	3	582.00	0.96	3.95	1.12	-0.07
E-UK50_18	4	4	695.65	1.00	3.89	0.71	-0.63
E-UK75_02	6	5	854.35	2.37	5.20	0.54	-2.00
E-UK75_04	6	5	833.66	0.92	3.79	0.06	-0.76
E-UK75_09	5	5	933.15	1.32	4.69	–	-1.20
E-UK75_12	5	5	859.67	0.57	3.74	0.11	-0.62
E-UK75_17	6	5	914.59	1.09	4.18	–	-0.11
E-UK100_01	7	7	1247.03	0.07	2.91	–	-0.96
E-UK100_03	7	6	1120.41	0.80	3.90	–	-0.52
E-UK100_06	7	7	1237.11	1.59	4.74	0.47	-1.45
E-UK100_07	6	6	1079.63	0.86	3.92	–	-1.40
E-UK100_17	8	7	1308.21	1.43	4.48	–	-0.90
E-UK150_01	10	10	1527.75	1.77	4.35	–	-1.00
E-UK150_02	10	10	1758.03	2.16	5.14	–	-2.23
E-UK150_03	10	9	1535.77	0.88	3.40	0.87	-0.20
E-UK150_13	10	9	1692.97	1.42	4.32	–	-0.92
E-UK150_17	10	10	1728.11	3.09	6.24	–	-2.59
E-UK200_03	14	13	2212.82	0.82	3.68	-0.09	-1.09
E-UK200_05	14	13	2316.20	-0.06	2.69	–	-0.94
E-UK200_08	14	13	2217.84	2.10	4.91	–	0.19
E-UK200_13	13	12	2284.02	0.02	2.98	0.15	-0.10
E-UK200_16	14	13	2183.56	0.10	2.84	–	0.08
Avg.				1.89	5.34	0.99	-0.94

Table 4.7 reports the results as averages over the instance groups with different customer numbers $10, 15, 20, 25, 50, 75, 100, 150$ and 200. Let $S_x, x \in \{d, c, cb\}$ denote the solution obtained when minimizing with objective f_x and let $f_y(S), y \in \{d, c, cb\}$ denote the objective value of evaluating solution S with objective function f_y. The table reports the average values over the instance group for the solutions S_d obtained with the objective of minimizing traveled distance (columns $f_d(S_d)$, $f_c(S_d)$, and $f_{cb}(S_d)$), and the gaps (to these values) of the objective values of the solutions obtained when minimizing costs (columns $\Delta f_d(S_c)$, $\Delta f_c(S_c)$, and $\Delta f_{cb}(S_c)$) and when minimizing costs including battery costs (columns $\Delta f_d(S_{cb})$, $\Delta f_c(S_{cb})$, and $\Delta f_{cb}(S_{cb})$).

The results show that the ALNS is effective with regard to the different optimization objectives: The best results found with respect to a certain objective function are always found when optimizing according to this objective. The gaps of the solutions found with other objectives are partly significant with nearly 5%. In particular, the results show that the traditional objective of minimizing traveled distance does not provide satisfying solutions when costs for fuel (diesel and electricity), labor, and battery are considered. This is in line with the results of Bektaş and Laporte (2011) for pure ICCV fleets, who note similar effects considering the cost components fuel, labor and emission costs. If battery costs are included, the quality of distance-minimizing solutions rises as traveled distance becomes more expensive (there is only a gap of -3.84% for $\Delta f_{cb}(S_{cb})$ compared to a gap of -4.92% for $\Delta f_c(S_c)$).

In Table 4.8, we report the share that ECVs contribute to the traveled distance of the solutions obtained with the different objective functions, denoted as $d_E(f(S))$. If ICCVs and ECVs were used to the same extent, ECVs would contribute with $m_E/(m_E + m_{IC}) = 47.2\%$ to the traveled distance. We find that when minimizing traveled distance, ECVs travel a noticeably smaller part of the total distance because long routes will mostly be assigned to ICCVs in order to save detours. The share of traveled distance rises significantly when costs without battery costs are considered because ECV travel is cheaper than ICCV travel if only fuel and electricity consumption is considered. Finally, if costs including battery costs are considered, the contribution of ECVs decreases perceptibly because ECVs become relatively expensive if battery deprecation costs are added to the travel costs.

Table 4.7.: Comparison of the different objective functions f_d, f_c and f_{cb}. Results of the best of 10 ALNS runs, averaged over instance groups are reported. $S_x, x \in \{d, c, cb\}$ denotes the solution obtained when minimizing with objective f_x and $f_y(S), y \in \{d, c, cb\}$ denotes the objective value (in dollar, respectively km) of evaluating solution S with objective function f_y. The results for using the cost-minimizing objectives f_c and f_{cb} are given as gaps to the values obtained using the distance-minimizing objective f_d in percent.

	S_d			S_c			S_{cb}		
Inst. group	$f_d(S_d)$	$f_c(S_d)$	$f_{cb}(S_d)$	$\Delta f_d(S_c)$	$\Delta f_c(S_c)$	$\Delta f_{cb}(S_c)$	$\Delta f_d(S_{cb})$	$\Delta f_c(S_{cb})$	$\Delta f_{cb}(S_{cb})$
E-UK10	456.60	198.61	219.65	2.54	-6.50	-3.39	2.14	-2.95	-5.71
E-UK15	663.04	295.67	321.21	1.60	-5.53	-2.37	2.24	-2.82	-3.73
E-UK20	837.71	373.99	418.87	2.08	-5.37	-3.13	1.01	-4.65	-4.76
E-UK25	940.80	404.09	477.41	1.64	-3.02	-1.01	0.77	-2.19	-3.16
E-UK50	1409.00	665.59	766.10	4.16	-6.92	1.22	2.45	-4.52	-4.22
E-UK75	1935.64	937.70	1088.39	2.24	-6.23	-1.78	2.00	-4.19	-5.51
E-UK100	2666.67	1250.15	1475.69	2.84	-4.14	-0.43	1.91	-3.01	-3.46
E-UK150	3487.07	1705.31	2007.53	3.15	-3.34	0.15	1.47	-2.90	-3.04
E-UK200	4671.72	2316.99	2697.74	3.91	-3.20	0.79	3.57	-1.75	-0.94
Avg.	1896.47	905.35	1052.51	2.68	-4.92	-1.11	1.95	-3.22	-3.84

Table 4.8.: Share in percent that ECVs contribute to the traveled distance of the solutions obtained with the different objective functions.

Inst. group	$d_E(f_d(S_d))$	$d_E(f_d(S_c))$	$d_E(f_d(S_{cb}))$
E-UK10	23.92	29.80	16.18
E-UK15	19.92	25.76	16.57
E-UK20	25.66	29.42	23.94
E-UK25	39.32	42.57	35.71
E-UK50	35.30	53.23	33.82
E-UK75	39.18	48.11	33.00
E-UK100	42.76	49.83	39.41
E-UK150	43.34	50.47	41.07
E-UK200	40.92	49.31	41.11
Avg.	34.48	42.06	31.20

4.6.4. Performance of our ALNS on Instances of Related Problems

To assess the performance of the proposed ALNS, we conduct experiments on available benchmark instances for the related problems VRPTW and E-VRPTW, which are both special cases of the E-VRPTWMF. In the VRPTW, only ICCVs are available and no energy consumption is considered. In the E-VRPTW, only ECVs are available and instead of the energy consumption function described in Section 4.3, consumed energy is a linear function of the distance traveled. The following two sections present the detailed results.

4.6.4.1. Performance of ALNS on VRPTW Instances

We use the well-known Solomon 100-customer instances as VRPTW benchmark set (Solomon, 1987). The set contains 56 instances, which are grouped according to the distribution of customer locations—random (R), clustered (C), random-clustered (RC)—and the considered scheduling horizon. Heuristic methods for VRPTW use a hierarchical objective (minimize number of vehicles first, minimize traveled distance second). Our ALNS is not designed to minimize vehicle routes but to minimize traveled distance with a given number of vehicles. Therefore, we set the initial number of routes for each instance to the best known number given in the literature. If ALNS is not able to determine a feasible solution with the given number of vehicles after β_{feas} iterations, the number of vehicles is increased by one.

As comparison method for our ALNS, we use the currently best-performing VRPTW method in the literature, the memetic algorithm proposed by Nagata, Bräysy, and Dullaert (2010) (NBD). Note that for all Solomon instances the solution found by NBD corresponds to the best-known solution (BKS) in the literature (Vidal, Crainic, et al., 2013). Table 4.9 reports the best solution with respect to the vehicle number m and the distance traveled f out of 10 runs for our ALNS (out of five runs for NBD). Results are given as averages over the instance groups R1, R2, C1, C2, RC1 and RC2. Finally, the cumulative number of vehicles (CNV) and the cumulative traveled distance (CTD) over all instances are reported. ALNS achieves a gap to NBD of 0% concerning the CNV and a gap of 0.37% with regard to the CTD. Although a direct comparison of run-times between different platforms (using different operating systems, programming languages, and compilers) is always difficult, we state that ALNS has an average computation time of 87 seconds per run compared to 300 seconds of NBD. These results

prove the capability of ALNS to find high-quality VRPTW solutions in fast computation times. This is a prerequisite to be able to address the computationally demanding E-VRPTWMF instances.

Table 4.9.: Results of ALNS on Solomon benchmark instances in comparison to the currently best VRPTW heuristic from the literature by Nagata, Bräysy, and Dullaert (2010) (NBD). We report the number of vehicles m and the traveled distance f of the best of 10 runs for the ALNS, and of five runs for NBD. Average results for each instance group, the CNV and the CTD are given.

Inst. group	NBD (m | f)	ALNS (m | f)
R1	11.92 | 1210.34	11.92 | 1215.60
R2	2.73 | 951.03	2.73 | 958.02
C1	10.0 | 828.38	10.0 | 828.38
C2	3.0 | 589.86	3.0 | 589.86
RC1	11.5 | 1384.16	11.5 | 1388.76
RC2	3.25 | 1119.24	3.25 | 1123.37
CNV	405	405
CTD	57187	57397

4.6.4.2. Performance of ALNS on E-VRPTW Instances

Experiments for the E-VRPTW are conducted on the instance set presented in (Schneider, Stenger, and Goeke, 2014), where also a detailed description of the instances can be found. The set contains 56 instances which are based on the Solomon VRPTW instances but additionally contain 21 recharging stations. The vehicle fleet consists solely of ECVs with a limited battery capacity, whose energy consumption depends linearly on the distance traveled. Recharging times at stations are assumed a linear function of the required charge. The objective of E-VRPTW is hierarchical like in the VRPTW and we use the numbers of vehicles reported in (Schneider, Stenger, and Goeke, 2014) as start values for our ALNS. As comparison method, we use the ALNS of Hiermann, Puchinger, and Hartl (2014) (HPH), and the VNS/TS hybrid of Schneider, Stenger, and Goeke (2014) (SSG).

In Table 4.10, for each instance, we provide the previous BKS as reported in (Hiermann, Puchinger, and Hartl, 2014; Schneider, Stenger, and Goeke, 2014) in terms of number of vehicles m and traveled distance f. For the algorithms HPH, SSG, and our ALNS, we report the best result out of 10 runs: the vehicle number m and the gap of the traveled distance to the BKS ($\Delta_{best}(\%)$). Since our ALNS uses the same vehicle numbers as reported in (Schneider, Stenger, and Goeke, 2014), it is only reported once for SSG. Moreover, for HPH and our ALNS, we additionally provide the average run-time for each instance, for SSG only the average run-time over all instances is given. A precise comparison of run-times is only possible for ALNS and SSG because both algorithms were run on the same platform. However, although HPH was run on a different platform, we think that a rough comparison of run-times may be valid since all algorithms were run on modern computers with comparable computing power.

Column $\overline{\text{ALNS}}$ lists the overall best results that we found during the testing of our ALNS. The best solution for each instance is marked in bold. Moreover, we mark in italics those results for which the number of vehicles used by HPH and SSG/ALNS differ. Finally, the CNV, the averages over the gaps and the average run-times in minutes are reported at the bottom of the table. The average gap is the average over all instances, the corrected average gap only includes the instances for which all algorithms

use the same number of vehicles. We deem the latter measure more important due to the influence of the number of vehicles on the traveled distance. It may be quite easy to find a better distance when using one more vehicle, but this distorts the comparison between the algorithms with regard to traveled distance.

The results show that our ALNS outperforms HPH and SSG. In comparison to SSG, ALNS finds the same CNV and improves the average gap of the traveled distance from 0.49% to 0.11%. In comparison to HPH, ALNS is able to reduce the CNV by one vehicle and to improve the distance gap by 0.62%. The average run-time decreases from about 15 minutes for SSG and HPH to less than 3 minutes. Considering the solutions found during the overall testing, our ALNS is able to improve the previous BKS for 34 out of the 56 instances, for 20 instances the previous BKS is matched. On average, the previous BKS is improved by 0.35%.

4.7. Summary and Conclusion

We introduce the E-VRPTWMF, a VRP to optimize the routing of a mixed vehicle fleet consisting of ICCVs and ECVs. Contrary to existing VRPs for ECVs which assume energy consumption to be a linear function of traveled distance, we utilize a realistic energy consumption model that incorporates speed, gradient and load distribution. This is highly relevant in the context of ECVs, as energy consumption determines the maximal driving range of ECVs and the recharging times at stations. To address the problem, we develop an ALNS algorithm that is enhanced by a LS for intensification.

In numerical studies on newly designed E-VRPTWMF test instances, we find that consideration of the actual load strongly improves the quality of the generated solutions in comparison to solutions that are generated based on load estimates. Moreover, we find that a large number of solutions that are generated with "optimistic" load estimates are actually infeasible due to battery capacity or time window violations. We further show that our ALNS works effectively with all of the investigated cost functions and that the traditional objective of minimizing traveled distance fails to produce high-quality solutions if routing costs including fuel, labor and battery depreciation are considered. The choice of objective function additionally has a strong influence on the level of usage of the ECVs in the fleet. Finally, the performance of the developed algorithm is proven on benchmark instances of related problems: The ALNS provides convincing results in moderate run-times on the well-known Solomon VRPTW benchmark. On E-VRPTW instances, the ALNS outperforms all comparison methods with respect to both solution quality and run-time.

There are several interesting avenues for future research. From a modeling perspective, the real-world processes related to ECVs should be modeled in a more realistic fashion, see our discussion at the end of Section 4.4 on how to integrate topology and acceleration processes. Another important point in this respect is a more realistic modeling of the recharging processes. From the computational viewpoint, the development of exact solution methods for solving basic electric VRPs, and the design of efficient metaheuristics for solving rich variants featuring realistic energy consumption and recharging models as well as relevant side constraints of practical routing problems are both interesting topics.

Table 4.10.: Results of our ALNS on E-VRPTW instances in comparison to those of the ALNS of Hiermann, Puchinger, and Hartl (2014) (HPH), and of the VNS/TS of Schneider, Stenger, and Goeke (2014) (SSG). BKS denotes the previously best known solution.

	BKS		**HPH**			**SSG**		**ALNS**		$\overline{\textbf{ALNS}}$	
Inst.	m	f	m	Δ_{best}	t	m	Δ_{best}	Δ_{best}	t	f	Δ_f
c101	**12**	**1053.83**	**12**	**0.00**	9.16	**12**	**0.00**	**0.00**	1.62	**1053.83**	**0.00**
c102	**11**	1056.47	**11**	0.07	9.76	**11**	0.07	0.07	2.61	**1051.38**	**-0.48**
c103	**10**	1041.55	**10**	0.25	9.15	**10**	0.00	-0.26	3.61	**1034.86**	**-0.64**
c104	**10**	979.51	**10**	0.52	11.09	**10**	0.13	-0.76	4.89	**961.88**	**-1.80**
c105	**11**	**1075.37**	**11**	**0.00**	9.31	**11**	**0.00**	**0.00**	1.82	**1075.37**	**0.00**
c106	**11**	**1057.65**	**11**	**0.00**	9.44	**11**	0.02	**0.00**	2.44	**1057.65**	**0.00**
c107	**11**	**1031.56**	**11**	**0.00**	9.66	**11**	**0.00**	**0.00**	2.72	**1031.56**	**0.00**
c108	**10**	1100.32	**10**	0.83	9.21	**10**	0.00	-0.13	3.20	**1095.66**	**-0.42**
c109	**10**	1036.64	**10**	1.43	10.17	**10**	1.47	**-0.29**	2.84	**1033.67**	**-0.29**
c201	**4**	**645.16**	**4**	**0.00**	18.11	**4**	**0.00**	**0.00**	1.45	**645.16**	**0.00**
c202	**4**	**645.16**	**4**	0.21	21.13	**4**	**0.00**	**0.00**	2.49	**645.16**	**0.00**
c203	**4**	**644.98**	**4**	**0.00**	22.91	**4**	**0.00**	**0.00**	3.48	**644.98**	**0.00**
c204	**4**	**636.43**	**4**	0.30	19.70	**4**	**0.00**	**0.00**	3.45	**636.43**	**0.00**
c205	**4**	**641.13**	**4**	**0.00**	20.96	**4**	**0.00**	**0.00**	2.03	**641.13**	**0.00**
c206	**4**	**638.17**	**4**	**0.00**	23.26	**4**	**0.00**	**0.00**	2.33	**638.17**	**0.00**
c207	**4**	**638.17**	**4**	**0.00**	22.68	**4**	**0.00**	**0.00**	2.71	**638.17**	**0.00**
c208	**4**	**638.17**	**4**	**0.00**	22.30	**4**	**0.00**	**0.00**	2.61	**638.17**	**0.00**
r101	**18**	1663.04	**18**	**0.00**	8.80	**18**	0.57	0.46	1.58	1665.62	0.16
r102	**16**	1488.97	**16**	0.00	9.85	**16**	3.15	0.26	2.26	**1487.41**	**-0.10**
r103	**13**	1285.96	**13**	0.00	9.91	**13**	1.06	-0.31	2.62	**1271.35**	**-1.14**
r104	**11**	**1088.43**	**11**	0.86	8.73	**11**	0.00	0.21	2.43	**1088.43**	**0.00**
r105	**14**	1461.25	*15*	*-1.87*	9.31	***14***	*0.84*	*-0.51*	3.38	***1442.35***	***-1.29***
r106	**13**	1344.66	**13**	1.38	8.91	**13**	0.00	-1.13	2.64	**1324.10**	**-1.53**
r107	**12**	1154.52	**12**	0.94	9.16	**12**	0.00	-0.26	2.56	**1150.95**	**-0.31**
r108	**11**	**1050.04**	**11**	1.66	8.78	**11**	1.51	1.22	2.63	**1050.04**	**0.00**
r109	**12**	1294.05	*13*	*-3.72*	9.48	***12***	*0.00*	***-2.53***	3.76	***1261.31***	***-2.53***
r110	**11**	1126.74	**11**	2.56	9.15	**11**	1.49	1.30	3.69	**1119.50**	**-0.64**
r111	**12**	**1106.19**	**12**	1.29	9.34	**12**	1.62	1.61	2.71	**1106.19**	**0.00**
r112	**11**	1026.52	**11**	1.68	8.70	**11**	0.00	0.02	2.73	**1016.63**	**-0.96**
r201	**3**	**1264.82**	**3**	0.37	27.05	**3**	**0.00**	0.21	2.05	**1264.82**	**0.00**
r202	**3**	**1052.32**	**3**	0.15	27.60	**3**	**0.00**	0.05	2.71	**1052.32**	**0.00**
r203	**3**	895.91	**3**	0.14	27.81	**3**	1.89	0.35	3.14	**895.54**	**-0.04**
r204	**2**	788.67	**2**	0.00	17.86	**2**	0.24	-0.73	4.55	**779.49**	**-1.16**
r205	**3**	988.67	**3**	1.35	24.85	**3**	0.00	0.14	2.08	**987.36**	**-0.13**
r206	**3**	922.70	**3**	0.00	24.98	**3**	0.27	0.23	2.60	**922.19**	**-0.06**
r207	**2**	848.53	**2**	1.33	20.83	**2**	0.49	0.05	3.46	**845.26**	**-0.39**
r208	**2**	736.60	**2**	0.49	18.25	**2**	0.00	0.44	3.52	**736.12**	**-0.07**
r209	**3**	872.36	**3**	2.10	25.56	**3**	0.00	0.16	2.66	**867.05**	**-0.61**
r210	**3**	847.06	**3**	1.94	25.56	**3**	0.00	0.34	3.11	**846.20**	**-0.10**
r211	**2**	847.45	**2**	3.09	20.62	**2**	2.21	-1.41	3.53	**827.89**	**-2.31**
rc101	**16**	**1726.91**	**16**	**0.00**	8.05	**16**	0.24	0.47	1.45	**1726.91**	**0.00**
rc102	**14**	1659.53	***14***	***0.00***	8.57	*15*	*-6.32*	*-6.15*	1.88	*1552.08*	*-6.47*
rc103	**13**	1351.15	**13**	1.35	8.89	**13**	0.18	0.02	2.46	**1350.09**	**-0.08**
rc104	**11**	1229.82	**11**	0.00	8.79	**11**	1.58	-0.05	2.86	**1227.25**	**-0.21**
rc105	**14**	**1475.31**	**14**	0.23	7.88	**14**	0.55	0.62	1.48	**1475.31**	**0.00**
rc106	**13**	1436.61	**13**	0.00	7.63	**13**	0.25	0.17	1.66	**1427.21**	**-0.65**
rc107	**12**	1275.89	**12**	0.60	8.02	**12**	0.00	0.04	2.06	**1274.89**	**-0.08**
rc108	**11**	1204.87	**11**	0.00	7.68	**11**	2.82	-0.35	2.61	**1197.83**	**-0.58**
rc201	**4**	**1444.94**	**4**	1.34	23.02	**4**	0.16	0.13	1.77	**1444.94**	**0.00**
rc202	**3**	1412.91	**3**	1.71	25.12	**3**	0.00	0.45	3.24	**1410.74**	**-0.15**
rc203	**3**	1073.98	**3**	1.00	26.12	**3**	0.40	-0.01	3.88	**1055.19**	**-1.75**
rc204	**3**	885.35	**3**	1.96	26.45	**3**	0.44	0.80	4.08	**884.80**	**-0.06**
rc205	**3**	1282.58	**3**	0.00	23.42	**3**	3.05	0.56	3.19	**1273.55**	**-0.70**
rc206	**3**	1190.75	**3**	2.35	24.36	**3**	0.03	0.06	2.69	**1188.63**	**-0.18**
rc207	**3**	995.52	**3**	2.07	23.75	**3**	0.00	1.01	3.66	**985.03**	**-1.05**
rc208	**3**	837.82	**3**	1.20	24.64	**3**	0.03	**-0.18**	3.98	**836.29**	**-0.18**
CNV	**440**		442			441					
Avg. gap				0.59			0.36	**-0.06**			**-0.52**
Avg. gap (corr.)				0.73			0.49	**0.11**			**-0.35**
Avg. time				15.92			15.34	**2.78**			

Bibliography

Artmeier, A., J. Haselmayr, M. Leucker, and M. Sachenbacher (2010). The shortest path problem revisited: Optimal routing for electric vehicles. In: *KI 2010: Advances in Artificial Intelligence*. Vol. 6359. Lecture Notes in Computer Science. Springer, Berlin, pp. 309–316.

Barco, J., A. Guerra, L. Muñoz, and N. Quijano (2013). Optimal routing and scheduling of charge for electric vehicles: Case study. In: *CoRR* abs/1310.0145. URL: http://arxiv.org/abs/1310.0145.

Bektaş, T. and G. Laporte (2011). The pollution-routing problem. In: *Transportation Research Part B: Methodological* 45 (8), pp. 1232–1250.

Bousonville, T., A. Hartmann, T. Melo, and H. Kopfer (2011). Vehicle routing and refueling: The impact of price variations on tour length. In: *Logistikmanagement - Herausforderungen, Chancen, und Lösungen (Proceedings of LM 11)*. Vol. 2, pp. 83–101.

Conrad R., G. and A. Figliozzi M. (2011). The recharging vehicle routing problem. In: *Proceedings of the 2011 Industrial Engineering Research Conference*. Ed. by T. Doolen and E. Van Aken. Reno, USA.

Cordeau, J.-F., G. Laporte, and A. Mercier (2001). A unified tabu search heuristic for vehicle routing problems with time windows. In: *Journal of the Operational Research Society* 52 (8), pp. 928–936.

Crainic, T. G., M. Gendreau, P. Soriano, and M. Toulouse (1993). A tabu search procedure for multicommodity location/allocation with balancing requirements. In: *Annals of Operations Research* 41 (4), pp. 359–383.

Crevier, B., J.-F. Cordeau, and G. Laporte (2007). The multi-depot vehicle routing problem with inter-depot routes. In: *European Journal of Operational Research* 176 (2), pp. 756–773.

Davis, B. A. and M. A. Figliozzi (2013). A methodology to evaluate the competitiveness of electric delivery trucks. In: *Transportation Research Part E: Logistics and Transportation Review* 49 (1), pp. 8–23.

Dayarian, I., G. Crainic T., M. Gendreau, and W. Rei (2013). *An Adaptive Large Neighborhood Search Heuristic for a Multi-Period Vehicle Routing Problem*. Tech. rep. 2013-67. CIRRELT, Canada.

Dekker, R., J. Bloemhof, and I. Mallidis (2012). Operations research for green logistics: An overview of aspects, issues, contributions and challenges. In: *European Journal of Operational Research* 219 (3), pp. 671–679.

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

Demir, E., T. Bektaş, and G. Laporte (2011). A comparative analysis of several vehicle emission models for road freight transportation. In: *Transportation Research Part D: Transport and Environment* 16 (5), pp. 347–357.

– (2014a). A review of recent research on green road freight transportation. In: *European Journal of Operational Research* 237 (3), pp. 775–793.

– (2014b). The bi-objective pollution-routing problem. In: *European Journal of Operational Research* 232 (3), pp. 464–478.

Desaulniers, G., F. Errico, S. Irnich, and M. Schneider (2016). Exact algorithms for electric vehicle-routing problems with time windows. In: *Operations Research* 64 (6), pp. 1388–1405.

Erdoğan, S. and E. Miller-Hooks (2012). A green vehicle routing problem. In: *Transportation Research Part E: Logistics and Transportation Review* 48 (1), pp. 100–114.

Felipe, Á., M. T. Ortuño, G. Righini, and G. Tirado (2014). A heuristic approach for the green vehicle routing problem with multiple technologies and partial recharges. In: *Transportation Research Part E: Logistics and Transportation Review* 71, pp. 111–128.

Feng, W. and M. A. Figliozzi (2013). An economic and technological analysis of the key factors affecting the competitiveness of electric commercial vehicles: A case study from the USA market. In: *Transportation Research Part C: Emerging Technologies* 26, pp. 135–145.

Feo, T. A. and M. G. C. Resende (1989). A probabilistic heuristic for a computationally difficult set covering problem. In: *Operations Research Letters* 8 (2), pp. 67–71.

Gendreau, M., A. Hertz, and G. Laporte (1994). A tabu search heuristic for the vehicle routing problem. In: *Management Science* 40 (10), pp. 1276–1290.

Gendreau, M. and J.-Y. Potvin (2010). Tabu search. In: *Handbook of Metaheuristics*. Ed. by M. Gendreau and J.-Y. Potvin. Vol. 146. International Series in Operations Research & Management Science. Springer, pp. 41–59.

Guzzella, L. and A. Amstutz (2005). *The QSS toolbox manual*. Accessed May 10, 2013. URL: http://www.idsc.ethz.ch/Downloads/DownloadFiles/qss.

He, F., D. Wu, Y. Yin, and Y. Guan (2013). Optimal deployment of public charging stations for plug-in hybrid electric vehicles. In: *Transportation Research Part B: Methodological* 47, pp. 87–101.

Heineken International (2014). *Case Studies: Europe's largest electric truck will drive down emissions*. Sustainability Report 2013. Accessed July 17, 2014. URL: http://sustainabilityreport.heineken.com/Reducing-CO2-emissions/Case-studies/Europes-largest-electric-truck-will-drive-down-emissions/index.htm.

Hemmelmayr V., C., F. Doerner K., F. Hartl R., and S. Rath (2013). A heuristic solution method for node routing based solid waste collection problems. In: *Journal of Heuristics* 19 (2), pp. 129–156.

Hemmelmayr, V. C., J.-F. Cordeau, and G. Crainic T. (2012). An adaptive large neighborhood search heuristic for two-echelon vehicle routing problems arising in city logistics. In: *Computers & Operations Research* 39 (12), pp. 3215–3228.

Hiermann, G., J. Puchinger, and R. F. Hartl (2014). *The electric fleet size and mix vehicle routing problem with time windows and recharging stations*. Tech. rep. Accessed July 17, 2014. URL: http://prolog.univie.ac.at/research/publications/downloads/Hie_2014_638.pdf.

Hoke, A., A. Brissette, D. Maksimovic, A. Pratt, and K. Smith (2011). Electric vehicle charge optimization including effects of lithium-ion battery degradation. In: *Vehicle Power and Propulsion Conference (VPPC)*. Chicago, USA: IEEE, pp. 1–8.

International Energy Agency (2012). *EV city casebook: A look at the global electric vehicle movement*. Accessed July 17, 2014. URL: http://www.iea.org/publications/freepublications/publication/EVCityCasebook.pdf.

– (2013). *EV outlook: Understanding the electric vehicle landscape to 2020*. Accessed July 17, 2014. URL: http://www.iea.org/publications/freepublications/publication/GlobalEVOutlook_2013.pdf.

Jabali, O., T. van Woensel, and A. de Kok (2012). Analysis of travel times and CO2 emissions in time-dependent vehicle routing. In: *Production and Operations Management* 21 (6), pp. 1060–1074.

Kindervater, G. and M. Savelsbergh (1997). Vehicle routing: Handling edge exchanges. In: *Local Search in Combinatorial Optimization*. Ed. by E. Aarts and J. Lenstra. John Wiley & Sons, pp. 337–360.

Kirkpatrick, S., C. D. Gelatt, and M. P. Vecchi (1983). Optimization by simulated annealing. In: *Science* 220 (4598), pp. 671–680.

Kleindorfer, P. R., A. Neboian, A. Roset, and S. Spinler (2012). Fleet renewal with electric vehicles at La Poste. In: *Interfaces* 42 (5), pp. 465–477.

Kopfer, H. W., J. Schönberger, and H. Kopfer (2014). Reducing greenhouse gas emissions of a heterogeneous vehicle fleet. In: *Flexible Services and Manufacturing Journal* 26 (1-2), pp. 221–248.

Laporte, G., Y. Nobert, and M. Desrochers (1985). Optimal routing under capacity and distance restrictions. In: *Operations Research* 33 (5), pp. 1050–1073.

Mak, H.-Y., Y. Rong, and Z.-J. M. Shen (2013). Infrastructure planning for electric vehicles with battery swapping. In: *Management Science* 59 (7), pp. 1557–1575.

Marra, F., G. Y. Yang, C. Traholt, E. Larsen, C. N. Rasmussen, and Y. Shi (2012). Demand profile study of battery electric vehicle under different charging options. In: *Power and Energy Society General Meeting, 2012*. San Diego, USA: IEEE, pp. 1–7.

Montoya, A., C. Guéret, J. E. Mendoza, and J. G. Villegas (2014). *A modified multi-space sampling heuristic for the green vehicle routing problem*. Tech. rep. LARIS-EA 7315. Laboratoire Angevin de Recherche en Ingénierie des Sytèmes, Université d'Angers, France.

Nagata, Y., O. Bräysy, and W. Dullaert (2010). A penalty-based edge assembly memetic algorithm for the vehicle routing problem with time windows. In: *Computers & Operations Research* 37 (4), pp. 724–737.

National Renewable Energy Laboratory (2014). *Project startup: evaluating the performance of Frito Lay's electric delivery trucks*. Accessed July 23, 2014. URL: http://www.nrel.gov/docs/fy14osti/61455.pdf.

Nie, Y. and M. Ghamami (2013). A corridor-centric approach to planning electric vehicle charging infrastructure. In: *Transportation Research Part B: Methodological* 57, pp. 172–190.

Nie, Y. and Q. Li (2013). An eco-routing model considering microscopic vehicle operating conditions. In: *Transportation Research Part B: Methodological* 55, pp. 154–170.

Potvin, J.-Y. and J.-M. Rousseau (1995). An exchange heuristic for routeing problems with time windows. In: *Journal of the Operational Research Society* 46 (12), pp. 1433–1446.

Preis, H., S. Frank, and K. Nachtigall (2014). Energy-optimized routing of electric vehicles in urban delivery systems. In: *Operations Research Proceedings 2012 – Selected Papers of the International Annual Conference of the German Operations Research Society (GOR)*. Ed. by S. Helber, M. Breitner, D. Rösch, C. Schön, J.-M. Graf von der Schulenburg, P. Sibbertsen, M. Steinbach, S. Weber, et al. Cham: Springer International Publishing, pp. 583–588.

Ropke, S. and D. Pisinger (2006a). A unified heuristic for a large class of vehicle routing problems with backhauls. In: *European Journal of Operational Research* 171 (3), pp. 750–775.

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

Savelsbergh, M. (1992). The vehicle routing problem with time windows: Minimizing route duration. In: *ORSA Journal on Computing* 4 (2), pp. 146–154.

Sbihi, A. and R. W. Eglese (2010). Combinatorial optimization and green logistics. In: *Annals of Operations Research* 175 (1), pp. 159–175.

Schneider, M., B. Sand, and A. Stenger (2013). A note on the time travel approach for handling time windows in vehicle routing problems. In: *Computers & Operations Research* 40 (10), pp. 2564–2568.

Schneider, M., A. Stenger, and D. Goeke (2014). The electric vehicle-routing problem with time windows and recharging stations. In: *Transportation Science* 48 (4), pp. 500–520.

Schneider, M., A. Stenger, and J. Hof (2015). An adaptive VNS algorithm for vehicle routing problems with intermediate stops. In: *OR Spectrum* 37 (2), pp. 353–387.

Shaw, P. (1997). *A new local search algorithm providing high quality solutions to vehicle routing problems*. Tech. rep. Glasgow, Scotland: Department of Computer Science, University of Strathclyde.

– (1998). Using constraint programming and local search methods to solve vehicle routing problems. In: *Principles and Practice of Constraint Programming – CP98*. Ed. by M. Maher and J.-F. Puget. Vol. 1520. Lecture Notes in Computer Science. London, United Kingdom: Springer, pp. 417–431.

Solomon, M. (1987). Algorithms for the vehicle routing and scheduling problems with time window constraints. In: *Operations Research* 35 (2), pp. 254–265.

Suzuki, Y. (2012). A decision support system of vehicle routing and refueling for motor carriers with time-sensitive demands. In: *Decision Support Systems* 54 (1), pp. 758–767.

Tesla Motors, Inc. (2014). *Supercharger*. Accessed July 7, 2014. URL: http://www.teslamotors.com/supercharger.

Toth, P. and D. Vigo (2003). The granular tabu search and its application to the vehicle-routing problem. In: *INFORMS Journal on Computing* 15 (4), pp. 333–346.

Toth, P. and D. Vigo, eds. (2014). Vehicle Routing: Problems, Methods, and Applications. 2nd ed. MOS-SIAM Series on Optimization. Philadelphia, USA: SIAM.

van Keulen, T., B. de Jager, A. Serrarens, and M. Steinbuch (2010). Optimal energy management in hybrid electric trucks using route information. In: *Oil and Gas Science and Technololgy* 65 (1), pp. 103–113.

Vidal, T., T. G. Crainic, M. Gendreau, and C. Prins (2013). A hybrid genetic algorithm with adaptive diversity management for a large class of vehicle routing problems with time-windows. In: *Computers & Operations Research* 40, pp. 475–489.

Wang, Y.-W. and C.-C. Lin (2013). Locating multiple types of recharging stations for battery-powered electric vehicle transport. In: *Transportation Research Part E: Logistics and Transportation Review* 58, pp. 76–87.

Xiao, Y., Q. Zhao, I. Kaku, and Y. Xu (2012). Development of a fuel consumption optimization model for the capacitated vehicle routing problem. In: *Computers & Operations Research* 39 (7), pp. 1419–1431.

Zhang, J., J. Tang, and R. Fung (2011). A scatter search for multi-depot vehicle routing problem with weight-related cost. In: *Asia-Pacific Journal of Operational Research* 28 (3), pp. 323–348.

Chapter 5

Electric Vehicles for Pickups and Deliveries in Urban Areas

5.1. Introduction

Electric vehicles (EVs) gained popularity in policy-making, from car manufactures, and in the public interest as a means to reduce local and greenhouse gas emissions. Motivated by this trend, the vehicle-routing community recently began to spend considerable research effort on the development of models and algorithms for the use of EVs in goods distribution, see, e.g., the reviews of Demir, Bektaş, and Laporte (2014a) and Pelletier, Jabali, and Laporte (2016). Much of this work is dedicated to investigating variants of the classical vehicle-routing problem (VRP) in which customers are served with goods stored in a central depot. A practically important representative of this class is the VRP with time windows (VRPTW) where each customer needs to be served within an individual time interval. This problem was then extended in Schneider, Stenger, and Goeke (2014) to the electric VRPTW (E-VRPTW) in which EVs are used with a limited battery capacity that can be recharged en route. In the following, variants, extensions and solution methods for the E-VRPTW were widely studied, (see, e.g., Felipe et al., 2014; Preis, Frank, and Nachtigall, 2014; Bruglieri et al., 2015; Goeke and Schneider, 2015; Hiermann, Puchinger, Ropke, et al., 2016; Desaulniers et al., 2016; Keskin and Çatay, 2016; Montoya et al., 2017).

The pickup and delivery problem with time windows (PDPTW) generalizes the VRPTW. The PDPTW is about planning routes in which commodities are transported from multiple pickup to delivery locations under time restrictions. Besides urban courier services and less-than-truckload transportation, this problem has also important applications in passenger transport, where it is known as dial-a-ride-problem (Ropke and Cordeau, 2009). Today, third-party logistics providers gain increasing importance, and we see the advent of logistics marketplaces like Freightos, where excess transport capacity can be sold online (Deutsche Post DHL Group, 2016). This facilitates logistics systems where multiple requests are consolidated on one vehicle in order to reduce cost and lessen the impact that transportation has on the environment; however it also requires complex planning tools that model many-to-many relationships. Investigating EVs for these tasks, that often arise in urban areas, might be a promising avenue as the advantages of EVs especially come into effect in the context of intra-city logistics (Dekker, Bloemhof, and Mallidis, 2012).

To the best of our knowledge, a variant of the PDPTW that uses EVs is only studied in Grandinetti et al. (2016). They propose a model based on a three-indexed formulation that allows vehicles to be recharged at dedicated recharging stations. However, the vehicles must be fully charged whenever they

visit a recharging station. Further, the authors do not develop a solution method and only introduce toy instances with at most three requests and four recharging stations. This work focuses on this important research gap. We study the electric pickup and delivery problem with time windows and electric vehicles (PDPTW-EV) that extends the PDPTW by using EVs with a limited battery capacity that might be arbitrarily recharged at dedicated recharging stations. We consider a partial recharging option as it can have substantial advantages over mandatory full recharging, (see, e.g., Bruglieri et al., 2015; Desaulniers et al., 2016; Keskin and Çatay, 2016).

The contributions of this work are:

- We introduce a compact two-indexed formulation for the PDPTW-EV with the option of recharging an arbitrary amount (full or partial recharge) when visiting recharging stations.
- We develop the first solution method for the PDPTW-EV, a granular tabu search (GTS) algorithm that is able to solve the PDPTW-EV on realistic-sized instances. We contribute to this approach by computing an efficient recharging plan for EVs that finds a feasible solution if one exists and otherwise minimizes time window violations.
- In order to assess the performance of our metaheuristic, we also solve benchmark instances for the closely related PDPTW. On these instances, we show that GTS has an excellent performance. We attribute this to the following reasons: Local search methods proposed earlier in the literature use complex moves that simultaneously change pairs of pickup and delivery. We show that it is sufficient to use simple moves that change one vertex at a time, and to penalize violations of the pairing constraints in the objective function instead. This offers the following advantages: (i) the computational complexity of many typical moves is reduced from quadratic to linear, (ii) we can use other powerful moves introduced for VRPs, e.g., the two-opt*, without being concerned about splitting pickups and deliveries, and (iii) we are more flexible in traversing the solution space. Moreover, we propose rules to generate moves based on the granular arcs and rules to decide whether a move is tabu that are tailored to the type of move. We employ a simple move caching technique that further restricts the moves that are updated in every iteration, and we aim to compute moves efficiently by storing precalculated information about routes.
- Finally, we introduce a small and a larger set of instances for the PDPTW-EV. On the small instances, we compare the solution quality of GTS to the results obtained for the compact formulation. On the large instances, we show that interrupting the recharging process can noticeably decrease the number of vehicles required and the distance traveled because it allows to reach certain critical customers just in time.

The remainder of this work is organized as follows: In Section 5.2, we formally introduce the PDPTW-EV and our mathematical problem formulation. Section 5.3 details the GTS for solving PDPTW-EV. Section 5.4 describes the generation of instances for the PDPTW-EV and presents the numerical studies that we performed on these new instances and on instances for the PDPTW available from the literature. Finally, some conclusions drawn in Section 5.5 close this chapter.

5.2. The Pickup and Delivery Problem with Time Windows and Electric Vehicles

The PDPTW-EV asks that n transportation requests are fulfilled. A transportation request specifies one location where a quantity q of some commodity is picked up and another location where the commodity is delivered; each operation takes s units of time. In addition, requests are associated with time windows for pickup and delivery, i.e., each operation must take place at a time between e and l.

Each request is served by one vehicle from a homogeneous fleet $\mathcal{K}$ of EVs. The EVs are based at a single depot and have a limited capacity C to store commodities. Further, the EVs are associated with a maximum battery capacity Q. The EVs depart from the depot with a fully charged battery, and consume energy for traveling between locations. It is required that there is always sufficient energy left to reach the next recharging station or the depot. To this end, an arbitrary amount of energy (so that we not exceed Q) can be replenished at a set of recharging stations $\mathcal{F}$, i.e., partial recharging is permitted. Under real-world conditions, the recharging rate is usually only constant until the state-of-charge reaches around 80% of the battery capacity and then the recharging speed decreases. Montoya et al. (2017) approximate this nonlinear behaviour with piecewise linear functions and already investigated how this affects solutions to the E-VRPTW in terms of feasibility and quality. Integrating this behavior could be also interesting for future work, but for the time being, we assume that the recharging time is proportional to the energy charged, expressed by a constant recharging rate g.

The objective of the PDPTW-EV is to determine routes that begin and end at the depot, serve all requests, and respect all constraints such that first the total number of routes and second the total distance traveled is minimized. In the following, we provide a compact formulation and define the problem on a directed graph $\mathcal{G} = (\mathcal{V}'_{0,2n+1}, \mathcal{A})$: Let $\mathcal{V} = \mathcal{P} \cup \mathcal{D}$ be the set of vertices containing pickup $\mathcal{P}$ and delivery $\mathcal{D}$ locations, and let $i \in \mathcal{P}$ be the pickup vertex that represents the origin of a request with delivery vertex $i + n \in \mathcal{D}$. Visits to the depot at the beginning and the end of a route are represented by vertices 0 and $2n + 1$, respectively. If a set of vertices X includes the depot vertices, we indicate this with a subscript, i.e., $X_0 = X \cup \{0\}$. Then, let $\mathcal{F}' = \{2n + 2, ...\}$ be a set of dummy vertices that indicate visits to recharging stations $\mathcal{F}$, and $\mathcal{V}' = \mathcal{V} \cup \mathcal{F}'$ denote the joint set. Further, we define the set of arcs as $\mathcal{A} = \{(i, j) : i \in \mathcal{V}'_0, j \in \mathcal{V}'_{2n+1}, i \neq j\}$. Each arc $(i, j) \in \mathcal{A}$ is associated with positive distance d_{ij}, energy consumption γ_{ij} and a modified travel time $\hat{t}_{ij}$. If $i \in \mathcal{V}$, then $\hat{t}_{ij}$ includes the service time s_i at the pickup or delivery location, i.e., $\hat{t}_{ij} = t_{ij} + s_i$ with t_{ij} the time required to travel from i to j, and otherwise $\hat{t}_{ij} = t_{ij}$. Every vertex $i \in \mathcal{V}'_{0,2n+1}$ is associated with a demand q_i that is positive for $i \in \mathcal{P}$, negative for $i \in \mathcal{D}$, and zero for $i \in \mathcal{F}'_{0,2n+1}$. In addition, each vertex i is associated with a time window $[e_i, l_i]$. Pickup and delivery operations must begin within this time window, and if the EV arrives before e_i, it must wait until e_i. The time window at the depot specifies the planning horizon, and vehicles might arrive at any time at recharging stations, i.e., the time windows are not restrictive.

We decided to use a two-index formulation in order to keep the number of variables manageable for larger instances as we use the linear relaxation of the model in our heuristic. We therefore use route tracking variables Θ_i that were proposed in Furtadoa, Munaria, and Morabitoa (2017) for a two-index formulation of the PDPTW. These variables ensure the pairing constraints, i.e., that pickup and delivery location of the same request are on the same route. The idea is to find an individual identifier for every

route and to assign this identifier to every vertex $i \in \mathcal{V}'$ that is visited by this route. Let Θ_i be a unique positive and integer identifier for every vertex $i \in \mathcal{V}'$, we use the index of the vertex $\Theta_i = i$, and let $\Theta_{max} = \max_{i \in \mathcal{V}'}(\Theta_i)$ be the largest identifier, i.e., $\Theta_{max} = |\mathcal{V}'|$. Then, Θ_a of the first vertex a that is visited after the depot is chosen as the route's identifier. Further, we define the arrival time τ_i, the cumulated demand u_i, and the remaining battery charge y_i when arriving at vertex $i \in \mathcal{V}'_{0,2n+1}$. For each visit to a recharging station $i \in \mathcal{F}'$, we also define the amount of energy $\bar{y}_i$ that is charged. For each arc $(i,j) \in \mathcal{A}$, we define a binary decision variable x_{ij} that is equal to 1 when the arc is traversed and 0 otherwise. The resulting mixed-integer linear programming (MILP) (note that, for better readability, we tacitly assume $i \neq j$) is:

$$\min \quad \sum_{j \in \mathcal{V}'} x_{0j} \text{ first,} \tag{5.1}$$

$$\sum_{i \in \mathcal{V}'_0, j \in \mathcal{V}'_{2n+1}} d_{ij} x_{ij} \text{ second} \tag{5.2}$$

$$\text{s.t.} \sum_{j \in \mathcal{V}'_{2n+1}} x_{ij} = 1 \qquad i \in \mathcal{V} \tag{5.3}$$

$$\sum_{j \in \mathcal{V}'_{2n+1}} x_{ij} \leq 1 \qquad i \in \mathcal{F}' \tag{5.4}$$

$$\sum_{i \in \mathcal{V}'_0} x_{ij} - \sum_{i \in \mathcal{V}_{2n+1}} x_{ji} = 0 \qquad j \in \mathcal{V}' \tag{5.5}$$

$$u_i + q_i - C(1 - x_{ij}) \leq u_j \qquad i \in \mathcal{V}'_0 \quad j \in \mathcal{V}'_{2n+1} \tag{5.6}$$

$$0 \leq u_j \leq C \qquad j \in \mathcal{V}'_{0,2n+1} \tag{5.7}$$

$$y_i - \gamma_{ij} x_{ij} + Q(1 - x_{ij}) \geq y_j \qquad i \in \mathcal{V}_0 \quad j \in \mathcal{V}'_{2n+1} \tag{5.8}$$

$$y_i + \bar{y}_i - \gamma_{ij} x_{ij} + Q(1 - x_{ij}) \geq y_j \qquad i \in \mathcal{F}' \quad j \in \mathcal{V}'_{2n+1} \tag{5.9}$$

$$0 \leq \bar{y}_i \leq Q - y_i \qquad i \in \mathcal{F}' \tag{5.10}$$

$$0 \leq y_j \leq Q \qquad j \in \mathcal{V}'_{0,2n+1} \tag{5.11}$$

$$\Theta_j x_{0j} \leq \vartheta_j \qquad j \in \mathcal{V}' \tag{5.12}$$

$$\Theta_j x_{0j} + \Theta_{max}(1 - x_{0j}) \geq \vartheta_j \qquad j \in \mathcal{V}' \tag{5.13}$$

$$\vartheta_i - \Theta_{max}(1 - x_{ij}) \leq \vartheta_j \qquad i,j \in \mathcal{V}' \tag{5.14}$$

$$\vartheta_i + \Theta_{max}(1 - x_{ij}) \geq \vartheta_j \qquad i,j \in \mathcal{V}' \tag{5.15}$$

$$\vartheta_i = \vartheta_{i+n} \qquad i \in \mathcal{P} \tag{5.16}$$

$$\tau_i + \hat{t}_{ij} x_{ij} - l_{2n+1}(1 - x_{ij}) \leq \tau_j \qquad i \in \mathcal{V}_0 \quad j \in \mathcal{V}'_{2n+1} \tag{5.17}$$

$$\tau_i + \hat{t}_{ij} x_{ij} + g\bar{y}_i - (l_{2n+1} + gQ)(1 - x_{ij}) \leq \tau_j \qquad i \in \mathcal{F}' \quad j \in \mathcal{V}'_{2n+1} \tag{5.18}$$

$$e_j \leq \tau_j \leq l_j \qquad j \in \mathcal{V}'_{0,2n+1} \tag{5.19}$$

$$\tau_i + \hat{t}_{i,i+n} \leq \tau_{i+n} \qquad i \in \mathcal{P} \tag{5.20}$$

$$x_{ij} \in \{0,1\} \qquad i \in \mathcal{V}'_0 \quad j \in \mathcal{V}'_{2n+1} \tag{5.21}$$

The hierarchical objective function is given in (5.1) and (5.2). Constraints (5.3) ensure that each customer is visited exactly once and constraints (5.4) that each visit to a recharging station is optional.

Constraints (5.5) handle flow conservation at vertices. Constraints (5.6) and (5.7) guarantee that the capacity of vehicles to transport commodities is not exceeded. Constraints (5.8) update the battery charge when arriving at a vertex after departing from a customer or the depot, and constraints (5.9) when leaving from a recharging station. Constraints (5.10) limit the energy that can be charged at a visit to a recharging station, and constraints (5.11) restrict the level of charge at every vertex. Constraints (5.12)–(5.16) guarantee the pairing constraints. In particular, constraints (5.12) and (5.13) set the identifier of each route to the first vertex visited after the depot, and constraints (5.14) and (5.15) propagate the identifier to all vertices in that route. At last, constraints (5.16) ensure that the same route identifier is assigned to pickup and delivery of a request. Constraints (5.17) set the arrival time at the successor of a customer or the depot, and constraints (5.18) set the arrival time if the predecessor was a recharging station depending on the amount of energy charged. Constraints (5.19) ensure that all vertices are visited within their time windows. Finally, constraints (5.20) ensure that the pickup of a request is performed before the delivery. Note that in order to change the partial recharging option to a full recharging condition, we can simply replace constraints (5.10) with constraints $\bar{y}_i = Q - y_i$ for every $i \in \mathcal{F}'$.

5.3. Granular Tabu Search to Solve the PDPTW-EV

GTS is a metaheuristic that was introduced in Toth and Vigo (2003) for the capacitated vehicle-routing problem (CVRP). The core of the approach is that the full neighborhood of a solution can be restricted to a granular neighborhood provided a suitable sparse graph is generated a priori. The sparse graph then filters moves so that only promising moves are evaluated. The time saved can then, e.g., be invested into performing an increased number of iterations. The authors use a simple sparsification method based on the length of arcs. A side benefit they observed (this observation is in line with what we found during our testing activities) is that a suitable sized sparse graph, typically between 10–25% of the original graph, can even provide higher-quality solutions within the same number of iterations. For problems that are heavier-constrained than the CVRP, mechanisms to reduce the size of a neighborhood become even more crucial because the evaluation of a single move might already be computationally expensive.

The successful application of GTS to the CVRP inspired Schneider, Schwahn, and Vigo (2017) to investigate the influence of more sophisticated sparsification methods for the VRPTW. They found that using the reduced cost of arcs as given by the network relaxation of the VRP, works especially well. Following this direction of research, we also use reduced cost to decide which arcs are included in the sparse graph. We obtain the reduced cost by relaxing the integrality constraints of formulation (5.1)–(5.21) and solving the resulting linear program. Further, we carefully design appropriate rules to (i) generate moves based on two sets of arcs, i.e., a reduced set from which we remove clearly infeasible arcs and a second set that contains the granular arcs, and (ii) then to decide whether a move is tabu or not. In addition, following the example of Schneider, Schwahn, and Vigo (2017), we rely on the time-travel approach introduced for the VRPTW in Nagata, Bräysy, and Dullaert (2010). Time-travel is a way to consider temporary violations of time windows during the search. This strategy of relaxing restrictive time windows can be considered state-of-the-art in local search methods.

Now, we describe the components of our GTS. In a preprocessing phase, we detect infeasible arcs and create the sparse graph based on reduced cost (Section 5.3.1). During the search, we allow infeasible

solutions that are rated with a generalized objective function (Section 5.3.2). Note that the pairing constraints are also handled by the penalty mechanism of the objective function. In this way, we are able to use the simple neighborhood structures known for the CVRP and VRPTW. We use a recharging policy to determine the energy recharged at every visit to a recharging station (Section 5.3.3). Figure 5.1 gives an overview of the following steps: First, we generate an initial solution $\mathcal{S}$ (Section 5.3.4). Then, we create a move cache that constitutes the neighborhood of $\mathcal{S}$ (Section 5.3.5). The algorithm begins with a feasibility phase that lasts until we find a feasible solution using at most $|\mathcal{K}|$ vehicles. In this phase, we add every $\eta_{feas} = 50000$ iterations another EV to $\mathcal{K}$. In a second phase, we aim to minimize the traveled distance. This phase stops after $\eta_{dist} = 7500$ iterations without finding a solution that improves the currently best solution. In each iteration, we change $\mathcal{S}$ using a non-tabu move , and then we perform an update of the move cache (Section 5.3.6). In addition, we store every feasible route found in a route cache $\mathcal{R}$. Every $\eta_{cover} = 1000$ iterations, we combine these routes in order to find a new solution using the set covering formulation proposed for the VRP in Balinski and Quandt (1964) (Section 5.3.7).

```
1: S ← generateInitialSolution(K)
2: M ← generateMoveCache(S)
3: i ← 0
4: feasibilityPhase ← true
5: while feasibilityPhase ∨ (¬feasibilityPhase ∧ i < η_dist) do
6:    S ← applyNonTabuMove(M)
7:    M ← updateMoves(S)
8:    R ← storeRoutes(S)
9:    if η_cover iterations have passed then
10:      S ← solveSetCovering(R)
11:      M ← generateMoveCache(S)
12:   end if
13:   if feasibilityPhase then
14:      if ¬ feasible(S) then
15:         if i = η_feas then
16:            K ← addVehicle(K)
17:            i ← −1
18:         end if
19:      else
20:         feasibilityPhase ← false
21:         i ← −1
22:      end if
23:   else
24:      if isImproving(S) then
25:         i ← −1
26:      end if
27:   end if
28:   i ← i + 1
29: end while
```

Figure 5.1.: Overview of the GTS algorithm.

5.3.1. Preprocessing and Arc Sparsification

In the preprocessing step, we first derive tighter time windows, and reduce the set of arcs by identifying infeasible arcs that cannot be part of a feasible solution to the PDPTW-EV. Then, we create an additional

sparse graph that consists of arcs that are promising candidates for a high-quality solution (Toth and Vigo, 2003; Schneider, Schwahn, and Vigo, 2017).

(i) Shrinking time windows: We perform a preliminary step in order to increase the number of arcs that we can identify as infeasible (Dumas, Desrosiers, and Soumis, 1991). Because the pickup of a request must be served before the delivery, the time window of the pickup and the travel time to the delivery define a time interval during which we can arrive at the delivery and vice versa. Therefore, we redefine the time windows of delivery vertices $i \in \mathcal{D}$ to $e_i := \max(e_i, e_{i-n} + \hat{t}_{i-n,i})$ and $l_i := \min(l_i, l_{i-n} + \hat{t}_{i-n,i})$, and of pickup vertices $i \in \mathcal{P}$ to $e_i := \max(e_i, e_{i+n} - \hat{t}_{i,i+n})$ and $l_i := \min(l_i, l_{i+n} - \hat{t}_{i-n,i})$.

(ii) Generating the reduced graph In the following, we present the rules that we use in order to generate the reduced arc set $\mathcal{A}'$ from $\mathcal{A}$. Every arc $(i, j) \in \mathcal{A}$ that matches at least one of the following conditions is infeasible and therefore removed from $\mathcal{A}$:

$$(i \in \mathcal{P}) \wedge (j = 2n + 1), \tag{5.22}$$

$$(j \in \mathcal{D}) \wedge (i = 0), \tag{5.23}$$

$$(i \in \mathcal{D}, j \in \mathcal{P}) \wedge (j = i - n), \tag{5.24}$$

$$i, j \in \mathcal{V} \wedge \forall w, h \in \mathcal{F}'_{0,2n+1} : \gamma_{wi} + \gamma_{ij} + \gamma_{jh} > Q, \tag{5.25}$$

$$i, j \in \mathcal{V} \wedge e_i + \hat{t}_{ij} > l_j, \tag{5.26}$$

$$i \in \mathcal{P}, j \in \mathcal{V}' \wedge \max(e_j, e_i + \hat{t}_{ij}) + \hat{t}_{j,i+n} > l_{i+n}, \tag{5.27}$$

$$i \in \mathcal{V}', j \in \mathcal{D} \wedge \max(e_i, e_{j-n} + \hat{t}_{j-n,i}) + \hat{t}_{i,j} > l_j. \tag{5.28}$$

Conditions (5.22)–(5.24) are presented in Dumas, Desrosiers, and Soumis (1991) and relate to the order of pickup and delivery in a route. Note that for GTS, we dynamically reinsert the arcs removed by conditions (5.22) and (5.23) into $\mathcal{A}'$, whenever a route contains no other vertices than the depot. The reason is that otherwise we cannot insert any pickup or delivery into this route. Schneider, Stenger, and Goeke (2014) introduce conditions (5.25) that state that an arc can be removed if the battery capacity is insufficient to support traveling the arc itself and to and from it via recharging stations or the depot. Conditions (5.26) are derived from the time windows of i and j as described in Dumas, Desrosiers, and Soumis (1991). Cordeau (2006) presents conditions (5.27) and (5.28), which extend the previous conditions to triples of vertices, i.e., if it is not possible to arrive in time at a delivery vertex when departing from the pickup and making a detour to another vertex in between, both arcs connected to the detour vertex are infeasible.

In addition, we experimented with more complex rules, which extend conditions (5.27)–(5.28). In the first experiment, we considered the minimum additional time required for recharging when making a detour via a recharging station. The second experiment concerns rules described in Dumas, Desrosiers, and Soumis (1991) for sequences of two pickup and delivery pairs, i.e., if a sequence $\langle i, j, i + n, j + n \rangle$ and $\langle i, j, j + n, i + n \rangle$ with $i, j \in \mathcal{P}$, or if a sequence $\langle j - n, i - n, i, j \rangle$ and $\langle i - n, j - n, i, j \rangle$ with $i, j \in \mathcal{D}$ are temporally infeasible, we could remove arc (i, j). However, even though many arcs could

potentially be removed with these rules, our experiments indicate that the solution quality of GTS suffers when we remove these arcs.

(iii) Generating the sparse graph $\mathcal{A}'^{-}$: Contrary to other granular tabu search algorithms that rely on just one set of arcs, we work with two sets when creating moves, i.e., we keep the reduced set $\mathcal{A}'$ as it is, and create another sparse set $\mathcal{A}'^{-} \subseteq \mathcal{A}'$. To this end, we first solve the linear relaxation of formulation (5.1)–(5.21) with $0 \leq x_{ij} \leq 1$ if $(i,j) \in \mathcal{A}'$ and $x_{ij} = 0$ if $(i,j) \notin \mathcal{A}'$. In the linear relaxation, each x_{ij} is associated with reduced cost. The reduced cost indicate how much the objective function value would increase if the arc was included in the solution. Initially, we have an empty set $\mathcal{A}'^{-}$. We first add all arcs incident to the depot because, in our experience, keeping the complete set of depot arcs is beneficial to the search. Then, we iteratively add the arcs from $\mathcal{A}'$ that have the lowest reduced cost until $|\mathcal{A}'^{-}| = \min(|\mathcal{A}'|, 0.25|\mathcal{A}|)$.

5.3.2. Generalized Objective Function

We use the following generalized objective function $f_{gen}(\mathcal{S})$ (see, e.g., Gendreau, Hertz, and Laporte, 1994) in order to evaluate solution $\mathcal{S}$:

$$f_{gen}(\mathcal{S}) = f(\mathcal{S}) + \sigma_{cap} \cdot z_{cap}(\mathcal{S}) + \sigma_{batt} \cdot z_{batt}(\mathcal{S}) + \sigma_{tw} \cdot z_{tw}(\mathcal{S}) + \sigma_{pair} \cdot z_{pair}(\mathcal{S}) + \sigma_{prec} \cdot z_{prec}(\mathcal{S}). \quad (5.29)$$

$f(\mathcal{S})$ denotes the total traveled distance as given in equation (5.2). Violations of capacity $z_{cap}(\mathcal{S})$, battery capacity $z_{batt}(\mathcal{S})$, time windows $z_{tw}(\mathcal{S})$, pickup and delivery pairing $z_{pair}(\mathcal{S})$, and pickup and delivery precedence $z_{prec}(\mathcal{S})$ are rated by penalty factors σ_{cap}, σ_{batt}, σ_{tw}, σ_{pair}, and σ_{prec}, respectively.

The penalty factors are determined in such a way that they eventually guide GTS towards feasible solutions, but in order to traverse the solution space, temporarily allow infeasible solutions. If a violation of type x has been present $(z_x(\mathcal{S}) > 0)$ for two successive iterations, the penalty factor is updated to $\sigma_x := 1.2\sigma_x$, and similarly, if it has been absent $(z_x(\mathcal{S}) = 0)$ for two iterations, updated to $\sigma_x := \sigma_x / 1.2$. Further, all factors are restricted to a minimum and maximum value, i.e., $\sigma_x \in [0.1, 5000]$. The factor for time window violations is further restricted because its value can be quite large in comparison with those of the other violations, i.e., $\sigma_{tw} \in [0.1, 500]$. Initially, we randomize the penalty factors in the interval $\sigma_x \in [5, 1000]$.

We represent a solution as a set of routes $\mathcal{S} = \{r_k \mid k \in \mathcal{K}\}$. A route $r = \langle v_0 = 0, v_1, \ldots, v_{n(r)-1}, v_{n(r)} = 2n+1 \rangle$ is given as a sequence of $n(r)+1$ vertices. The sequence starts and ends at the depot. We reuse the notation already introduced to indicate sets specifically describing routes, e.g., the set $\mathcal{V}(r)$ contains the pickup and delivery vertices visited by route r, which do not necessarily belong to the same request, and $\mathcal{F}'(r)$ denotes visits to recharging stations. We denote the vertex at position o in route r with v_o. Each violation $z_x(\mathcal{S})$ is computed by calculating the sum of the individual violations $z_x(r)$ for each route $r \in \mathcal{S}$.

In the following, we give a formal description of the violations $z_x(r)$ and explain how they can be computed efficiently. For this purpose, each move evaluated during the search is expressed as a series of concatenations of partial sequences $a \oplus b \oplus \ldots \oplus c$ (see Section 5.3.5) that are derived from the routes involved. In turn, each concatenation $a \oplus b$ is evaluated by using stored values for a and b.

- We determine the capacity violation of a route r as the maximum cumulated demand exceeding the vehicle capacity for any sequence of vertices in r starting from the depot. We first define the cumulated demand $\Pi(a) = \sum_{i \in \mathcal{V}(a)} q_i$ of the pickup and delivery vertices $\mathcal{V}(a)$ served in sequence a (recall that $q_i > 0$ if $i \in \mathcal{P}$ and $q_i < 0$ if $i \in \mathcal{D}$). Now, let $a_{v_0 v_p}$ indicate a sub-sequence of a that begins at the first vertex of a and ends at position p. We calculate the maximum cumulated demand $\Phi(a) = \max_{p=0}^{n(a)} \Pi(a_{v_0 v_p})$. Then, the capacity violation of route r is $z_{cap}(r) = \max(\Phi(r) - C, 0)$. In order to efficiently calculate the capacity violation associated with a move, we require the maximum cumulated demand $\Phi(a \oplus b) = \max(\Phi(a), \Pi(a) + \Phi(b))$ and the cumulated demand $\Pi(a \oplus b) = \Pi(a) + \Pi(b)$. For example, assume that a route r is to be composed of three sequences $r = a \oplus b \oplus c$, then we recursively calculate $z_{cap}(r) = \max(\Phi((a \oplus b) \oplus c) - C, 0)$.

- In order to calculate the time window violation, we utilize the well-known concept of time travel (also called time warp) (Nagata, Bräysy, and Dullaert, 2010; Schneider, Sand, and Stenger, 2013; Vidal, Crainic, et al., 2013). The idea is that a vehicle arriving late at a vertex i, i.e., $\tau_i > l_i$, is assumed to begin service not at time τ_i but already at time l_i; we denote this the adjusted begin of service $\tau'_i = \min(\tau_i, l_i)$. The time of arrival at the successor j is calculated as $\tau_j = \max(e_j, \tau'_i + \hat{t}_{ij})$ if $i \in \mathcal{V}_0$ and $\tau_j = \max(e_j, \tau'_i + \hat{t}_{ij} + g\bar{y}_i)$ if $i \in \mathcal{F}'$. The total time window violation can now be computed as $z_{tw}(r) = \sum_{i \in \mathcal{V}'_{0,2n+1}(r)} \max(\tau_i - l_i, 0)$. Note that the arrival time depends on the variable amount of energy $\bar{y}_i$ recharged if the predecessor is a recharging station. The policy that we use in order to calculate $\bar{y}_i$ is explained in Section 5.3.3.

  Nagata, Bräysy, and Dullaert (2010) propose to efficiently calculate time window violations for simple moves by saving forward and backward slack variables for every vertex. The calculation in $\mathcal{O}(1)$ is limited to neighborhoods that change the route at only one position, e.g., the inter-route variant of relocate, exchange or two-opt. But, it can neither be applied to moves that alter a route at multiple positions as used by our GTS nor is it able to account for changes of the time required for recharging. Vidal, Crainic, et al. (2013) show that time window violations can also be calculated efficiently for the concatenation of arbitrary sequences if aggregated values for duration, time window violation, and earliest and latest begin of service are available for each of the considered sequences.

  We determine the time window violation $\tilde{z}_{tw}(r)$ as if no recharging took place as described in Vidal, Crainic, et al. (2013) and later add the additional time window violation caused by recharging under our policy $z^+_{tw}(r)$, i.e., $z_{tw}(r) = \tilde{z}_{tw}(r) + z^+_{tw}(r)$. In order to calculate $z^+_{tw}(r)$ (see Section 5.3.3), we need to isolate visits to recharging stations when we evaluate moves, i.e., visits to recharging stations should (when we begin the concatenation) only be contained alone in a sequence. We achieve this by splitting sequences before and after every visit to a recharging station. Therefore, the complexity of the time window calculation for a given move depends linearly on the number of sequences that are concatenated. This number depends in turn on the type of the move and also on the number of visits to recharging stations.

- To calculate the battery capacity violation of route r, we first define a variable $\Gamma(a)$ that represents the energy required to travel to the last vertex v_p of a sequence $a = \langle v_o, ..., v_{p-1}, v_p \rangle$. If sequence a contains at least one visit to a recharging station, $\Gamma(a)$ corresponds to the energy required from the last recharging station preceding v_p. If there is no recharging station in a, then $\Gamma(a)$ represents

the energy from v_o to v_p. Further, we define a function $\phi^<(i)$ that returns the recharging station preceding vertex i or the start depot if no recharging station exists in the sequence. The battery capacity violation $z_{batt}(r)$ of route r is the sum of violations at all recharging stations and the depot at the end of the route, i.e., $z_{batt}(r) = \sum_{i \in \mathcal{F}'_{2n+1}(r)} \max(\Gamma(a_{\phi^<(i),i}) - Q, 0)$.

Now, we formally introduce how $\Gamma(a)$ is calculated by concatenation. If a sequence a has only one vertex, i.e., $|a| = 1$, then we set $\Gamma(a) = 0$. For the concatenation of two sequences b and c, we obtain $\Gamma(b \oplus c) = \gamma_{v_{n(b)},v_{0(c)}} + \Gamma(c)$ if sequence b ends with a visit to a recharging station, i.e., $v_{n(b)} \in \mathcal{F}'$, and $\Gamma(b \oplus c) = \Gamma(b) + \gamma_{v_{n(b)},v_{0(c)}} + \Gamma(c)$ if $v_{n(b)} \notin \mathcal{F}'$. Note that this is only valid because we construct the sequences in such a way that sequence c contains no visit to a recharging station if $|c| > 1$.

- The violation of the pairing constraints $z_{pair}(r)$ is defined as the number of requests where either the pickup or the delivery is contained in route r, but not both. Let $\mathcal{P}^-(a)$ denote the set of pickups in sequence a where the corresponding delivery is not contained in a, i.e., $\mathcal{P}^-(a) = \{i \in \mathcal{P}(a) \mid (i+n) \notin \mathcal{D}(a)\}$ and the set of unmatched deliveries as $\mathcal{D}^-(a) = \{i \in \mathcal{D}(a) \mid (i-n) \notin \mathcal{P}(a)\}$, then $z_{pair}(r) = |\mathcal{P}^-(r)| + |\mathcal{D}^-(r)|$. The corresponding set for unmatched pickups resulting from concatenation is then calculated as $\mathcal{P}^-(a \oplus b) = (\mathcal{P}^-(a) \cup \mathcal{P}^-(b))/(\{i \in \mathcal{P}^-(a) \mid (i+n) \in \mathcal{D}^-(b)\} \cup \{i \in \mathcal{P}^-(b) \mid (i+n) \in \mathcal{D}^-(a)\})$. We calculate $\mathcal{D}^-(a \oplus b)$ in analogous fashion. The time complexity of evaluating the concatenation with respect to the pairing violation thus depends on the size of the sets, but in our experience, the sets typically become not that large and each check can be performed with low computational effort using appropriate data structures.
- The violation of the precedence constraints corresponds to the number of requests where pickup and delivery are contained in route r but the delivery is served before the pickup. We calculate $z_{prec}(r) = |\mathcal{P}^>(r)|$ with the help of set $\mathcal{P}^>(a) = \{i \in \mathcal{P}(a) \mid (i+n) \in \mathcal{D}(a) \wedge \tau_i > \tau_{i+n}\}$. For concatenation, we thus obtain $\mathcal{P}^>(a \oplus b) = \mathcal{P}^>(a) \cup \mathcal{P}^>(b) \cup \{i \in \mathcal{P}^-(b) \mid (i+n) \in \mathcal{D}^-(a)\}$ resulting in a complexity that again depends linearly on the size of the sets.

5.3.3. Recharging Policy

In order to generate a plan that determines the amount of energy recharged at every station in a route r, we propose a recharging policy with the following properties: First, it is complete, i.e., if it is possible

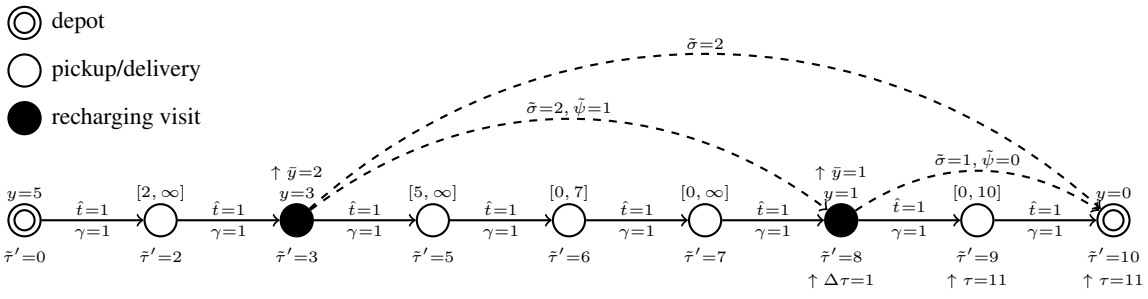

Figure 5.2.: Recharging policy applied to an example with recharging rate $g = 1$. Note that there exists no feasible solution. We determine the minimal time window violation to one unit. We indicate the changes caused by the recharging decisions with an arrow (↑).

to charge in such a way that the route is feasible with respect to energy consumption and time windows, our policy always provides a feasible charging plan. Second, it has a time complexity that is linear in the number of stations, i.e., $\mathcal{O}(|\mathcal{F}'(r)|)$ provided that aggregated information as described below is computed beforehand. Third, it makes sense from a practical point of view, i.e, if there exists no feasible charging plan, our policy minimizes violations of battery capacity first, and violations of time windows second.

Now, we briefly outline how our iterative policy works, and then we explain it in more detail: We charge, limited by the battery capacity, at every station first as much energy as required to reach the next station or the depot given the current state-of-charge. In addition, we charge as much as possible without adding any time window violation at any later vertex in the route. We begin at the first station in the route and then consider one station after another.

To compute the plan efficiently, we require slack and total waiting time between each pair of vertices in a route. Slack is the time that we can delay at the first vertex without causing an increase in time window violation at any subsequent vertex until we reach the second vertex. Total waiting time is the sum of all individual waiting times at vertices between the first (excluded) and the second (included) vertex. Let us now hypothetically assume that no recharging takes place. We need (i) the slack $\tilde{\sigma}_{w,h}$ between each station $w \in \mathcal{F}'(r)$ and the station $h \in \mathcal{F}'(r)$ that succeeds w, (ii) the total waiting time $\tilde{\psi}_{w,h}$, and (iii) the slack $\tilde{\sigma}_{w,v_{n(r)}}$ between each station $w \in \mathcal{F}'(r)$ and the depot $v_{n(r)}$. In addition, let $\tilde{\tau}'_i$ be the adjusted begin of service at vertex i before any charging takes place on the route, i.e., $\bar{y}_j = 0 \; \forall j \in \mathcal{F}'(r)$. Then, the total waiting time $\tilde{\psi}_{v_o v_p}$ between two vertices v_o and v_p at position o and p in a sequence $a = \langle v_o, ..., v_{p-1}, v_p \rangle$ is $\tilde{\psi}(a) = \tilde{\psi}_{v_o v_p} = \sum_{b=o+1}^{b \leq p} \max(e_{v_b} - \tilde{\tau}'_{v_{b-1}} - \hat{t}_{v_{b-1} v_b}, 0)$ for $p > o$ and 0 for $p = o$. We recursively define the slack $\tilde{\sigma}(a) = \tilde{\sigma}_{v_o v_p} = \min(\tilde{\sigma}_{v_o v_{p-1}}, \tilde{\psi}_{v_o v_p} + l_{v_p} - \tilde{\tau}'_{v_p})$ for $p > o$ and ∞ for $p = o$. The values resulting from the concatenation of two sequences a and b can be calculated as $\tilde{\psi}(a \oplus b) = \tilde{\psi}(a) + \tilde{\psi}(b) + \max(e_{v_{0(b)}} - \tilde{\tau}'_{v_{n(a)}} - \hat{t}_{v_{n(a)} v_{0(b)}}, 0)$ and $\tilde{\sigma}(a \oplus b) = \min(\tilde{\sigma}(a), \min(\tilde{\psi}(a \oplus v_{0(b)}) + \tilde{\sigma}(b), \tilde{\psi}(a \oplus v_{0(b)}) + l_{v_{0(b)}} - \tilde{\tau}'_{v_{0(b)}}))$.

In order to determine the amount of energy $\bar{y}$ to recharge and ultimately the increase in time window violation $z^+_{tw}(r)$ caused by recharging, we iteratively take the charging decisions into account. Let $w \in \mathcal{F}'(r)$ be the current station and $\phi^>(w)$ be the recharging station succeeding w or the end depot if no recharging station exists. Then, we charge, up to the available battery capacity, at least as much energy as required to reach the next station or the depot given the current state-of-charge. If the slack allows us to charge more than this minimum amount of energy, we charge the respective additional amount. In detail, we charge an amount of energy $\bar{y}_w = \min(\max(\Gamma(a_{w,\phi^>(w)}) - y_w, (\tilde{\sigma}_{w,v_{n(r)}} - \Delta\tau_w)/g), Q - y_w)$ in $\bar{t}_w = g\bar{y}_w$ units of time, with $\Gamma(a_{w,\phi^>(w)})$, the amount of energy required for the sequence until the next station or the depot, $\sigma_{w,v_{n(r)}}$ the slack between w and the depot, and $\Delta\tau_w$ the difference in arrival time at w that results from all previous recharging decisions for route r. The charge still left in the battery when arriving at w is calculated as $y_w = \max(y_{\phi^<(w)} + \bar{y}_{\phi^<(w)} - \Gamma(a_{\phi^<(w),w}), 0)$ if $\phi^<(w)$ is a station or $y_w = \max(Q - \Gamma(a_{\phi^<(w),w}), 0)$ if $\phi^<(w)$ is the start depot. Further, the delay at w is propagated from $\phi^<(w)$ as $\Delta\tau_w = \max(\min(\Delta\tau_{\phi^<(w)} + \bar{t}_{\phi^<(w)}, \tilde{\sigma}_{\phi^<(w),w}) - \tilde{\psi}_{\phi^<(w)w}, 0)$. The total increase of time window violation due to recharging is then calculated as $z^+_{tw}(r) = \sum_{j \in \mathcal{F}'(r)} \max(\bar{t}_j + \Delta\tau_j - \tilde{\sigma}_{j,\phi^>(j)}, 0)$.

In Figure 5.2, we give an example that illustrates the calculations. Starting with $y = 5$ units of charge at the depot, we have a charge of three units left when we arrive at the first recharging visit. In order to

reach the second recharging station, we need to charge at least one unit. Because two units of slack $\tilde{\sigma}$ (outer dashed arc) are available, we can safely recharge an amount $\bar{y}$ of two units in two units of time. Because one unit of waiting time $\tilde{\psi}$ is available between the two visits, recharging delays the arrival at the second recharging station $\Delta\tau$ by only one unit of time. At the second recharging station, we need to recharge one unit to reach the depot. This requires one additional unit of time. As only one unit of slack $\tilde{\sigma}$ is available, this results in a time window violation of one unit (we arrive late $\tau = 11$ at the last pickup/delivery location).

5.3.4. Generation of the Initial Solution

The following simple procedure is used to generate an initial solution. We create $|\mathcal{K}|$ routes that contain only the depot. Considering one vertex at a time, we greedily insert the vertices $i \in \mathcal{V}$ such that we minimize the generalized objective function $f_{gen}(\mathcal{S})$ as given by equation (5.29).

5.3.5. Reduced Neighborhood

In this section, we discuss our strategy to generate the neighborhood of a solution $\mathcal{S}$. Our aim is to keep the computational effort low by investigating only a part of the neighborhood that could be generated using the neighborhood structures implemented. At the same time, we want to ensure that we have a fair chance to access every part of the solution space. Therefore, we introduce a probabilistic element in the generation of the neighborhood, which can also help to diversify the search: We keep a move cache $\mathcal{M}$ that stores all moves that we have already evaluated with respect to $f_{gen}(\mathcal{S})$. When we generate the cache for the first time, we add all moves to $\mathcal{M}$ that can be generated from the neighborhood structures as detailed below. In every iteration, we select from $\mathcal{M}$ the move m that minimizes $f_{gen}(\mathcal{S})$ and is not tabu (see Section 5.3.6). Then, we perform move m and change the solution accordingly. Now, in order to update $\mathcal{M}$, we consider all moves from $\mathcal{M}$ that share at least one route associated with m. Some of these moves are now structurally invalid because the arcs involved do not longer correspond to the current solution, and for other moves the objective function value must be adjusted. We remove the former moves from $\mathcal{M}$. The latter moves, we sort according to their old objective function value. Then, we re-evaluate only the 100 best of these moves, and remove the others. Subsequently, we draw, with equal probability, one of our neighborhood structures, create all moves that are related to any of the routes modified by m and add them to $\mathcal{M}$ if they are not already present from the previous step.

In the following, we detail our neighborhood structures with the help of two exemplary routes r and s and two dedicated vertices i at position o and vertex j at position p, respectively:

$$r = \langle v_0, \ldots, i = v_o, \ldots, v_{n(r)} \rangle,$$
$$s = \langle v_0, \ldots, j = v_p, \ldots, v_{n(s)} \rangle.$$

- *Relocate* was introduced in Waters (1987). The idea is to move a single vertex $i \in \mathcal{V}'$ from position o in route r to another position $p > 0$ in route s. For the case of an intra-route move ($r = s$) and the case of an inter-route move ($r \neq s$), we obtain the following changed routes:

$$r = s \quad p < o \quad r' = \langle v_0, \ldots, v_{p-1} \rangle \oplus \langle i \rangle \oplus \langle v_p, \ldots, v_{o-1} \rangle \oplus \langle v_{o+1}, \ldots, v_{n(r)} \rangle,$$

$$
\begin{aligned}
p > o \quad & r' = \langle v_0, \ldots, v_{o-1} \rangle \oplus \langle v_{o+1}, \ldots, v_{p-1} \rangle \oplus \langle i \rangle \oplus \langle v_p, \ldots, v_{n(r)} \rangle, \\
r \neq s \quad & r' = \langle v_0, \ldots, v_{o-1} \rangle \oplus \langle v_{o+1}, \ldots, v_{n(r)} \rangle, \\
& s' = \langle v_0, \ldots, v_{p-1} \rangle \oplus \langle i \rangle \oplus \langle v_p, \ldots, v_{n(s)} \rangle.
\end{aligned}
$$

In this way, we create the following arcs $\mathcal{A}^{\oplus} = \mathcal{A}_1^{\oplus} \cup \mathcal{A}_2^{\oplus}$ with $\mathcal{A}_1^{\oplus} = \{(v_{p-1}, i), (i, v_p)\}$ at the new position of i and $\mathcal{A}_2^{\oplus} = \{(v_{o-1}, v_{o+1})\}$ at its old position. We aim to move a vertex only to feasible insertion positions, therefore we allow only moves where all arcs from $\mathcal{A}_1^{\oplus}$ are feasible, i.e., $\forall a \in \mathcal{A}_1^{\oplus} : a \in \mathcal{A}'$. The sparse graph should restrict the moves to a lesser extent than the reduced graph and gently guide GTS towards promising solutions. Therefore, we only generate moves where at least one arc from $\mathcal{A}_1^{\oplus}$ is contained in the sparse graph, i.e., $\exists a \in \mathcal{A}_1^{\oplus} : a \in \mathcal{A}'^{-}$. The removed arcs are $\mathcal{A}^{\ominus} = \{(v_{o-1}, i), (i, v_{o+1}), (v_{p-1}, v_p)\}$. We explain in the next section, how the removed arcs are relevant to the decision whether the move is tabu.

- *Exchange* was introduced in Savelsbergh (1992). Two vertices $i \in \mathcal{V}'$ and $j \in \mathcal{V}'$ are swapped. For intra-route moves, we only investigate the case that $o < p$. By this means, we avoid evaluating the same move twice.

$$
\begin{aligned}
r = s \quad & r' = \langle v_0, \ldots, v_{o-1} \rangle \oplus \langle j \rangle \oplus \langle v_{o+1}, \ldots, v_{p-1} \rangle \oplus \langle i \rangle \oplus \langle v_{p+1}, \ldots, v_{n(r)} \rangle, \\
r \neq s \quad & r' = \langle v_0, \ldots, v_{o-1} \rangle \oplus \langle j \rangle \oplus \langle v_{o+1}, \ldots, v_{n(r)} \rangle, \\
& s' = \langle v_0, \ldots, v_{p-1} \rangle \oplus \langle i \rangle \oplus \langle v_{p+1}, \ldots, v_{n(s)} \rangle.
\end{aligned}
$$

This creates arcs $\mathcal{A}^{\oplus} = \mathcal{A}_1^{\oplus} \cup \mathcal{A}_2^{\oplus}$ with $\mathcal{A}_1^{\oplus} = \{(v_{p-1}, i), (i, v_{p+1})\}$ and $\mathcal{A}_2^{\oplus} = \{(v_{o-1}, j), (j, v_{o+1})\}$. We restrict moves to cases where all arcs from $\mathcal{A}_1^{\oplus}$ and $\mathcal{A}_2^{\oplus}$ are feasible, i.e., $\forall a \in \mathcal{A}_1^{\oplus} \cup \mathcal{A}_2^{\oplus} : a \in \mathcal{A}'$, and at least one arc from each $\mathcal{A}_1^{\oplus}$ and $\mathcal{A}_2^{\oplus}$ is part of the sparse graph, i.e., $\exists a \in \mathcal{A}_1^{\oplus} : a \in \mathcal{A}'^{-} \wedge \exists a \in \mathcal{A}_2^{\oplus} : a \in \mathcal{A}'^{-}$. The removed arcs are $\mathcal{A}^{\ominus} = \{(v_{o-1}, i), (i, v_{o+1}), (v_{p-1}, j), (j, v_{p+1})\}$.

- *Two-opt** was proposed in Potvin and Rousseau (1995) as special case of the two-opt (Lin, 1965). It is suitable for problems with time windows because it preserves the orientation of routes. We split two routes ($r \neq s$) at $i \in \mathcal{V}_0'$ and $j \in \mathcal{V}_0'$ and then cross-connect them:

$$
\begin{aligned}
r' &= \langle v_0, \ldots, i \rangle \oplus \langle v_{p+1}, \ldots, v_{n(s)} \rangle, \\
s' &= \langle v_0, \ldots, j \rangle \oplus \langle v_{o+1}, \ldots, v_{n(r)} \rangle.
\end{aligned}
$$

This creates arcs $\mathcal{A}^{\oplus} = \{(i, v_{p+1}), (j, v_{o+1})\}$. We allow only moves where all arcs from $\mathcal{A}^{\oplus}$ are feasible, i.e., $\forall a \in \mathcal{A}^{\oplus} : a \in \mathcal{A}'$ and at least one arc from $\mathcal{A}^{\oplus}$ is part of the sparse arcs, i.e., $\exists a \in \mathcal{A}^{\oplus} : a \in \mathcal{A}'^{-}$. The removed arcs are $\mathcal{A}^{\ominus} = \{(i, v_{o+1}), (j, v_{p+1})\}$.

- *Swapping pairs* was introduced in Nanry and Barnes (2000) for the PDPTW. The idea is that pickup and delivery vertices of two requests are simultaneously exchanged. We only consider inter-route moves ($r \neq s$)) where pickup and delivery of the same request are on the same route. We exchange $i \in \mathcal{P}$ with $j \in \mathcal{P}$ and $i + n \in \mathcal{D}$ with $j + n \in \mathcal{D}$. Let $i + n$ be at position b and $j + n$ be at position u. We exemplarily show how the concatenation works for the case that both pickups

are served before their respective deliveries. The other variants can be derived analogously.

$$r' = \langle v_0, \ldots, v_{o-1} \rangle \oplus \langle j \rangle \oplus \langle v_{o+1}, \ldots, v_{b-1} \rangle \oplus \langle j+n \rangle \oplus \langle v_{b+1}, \ldots, v_{n(r)} \rangle,$$
$$s' = \langle v_0, \ldots, v_{p-1} \rangle \oplus \langle i \rangle \oplus \langle v_{p+1}, \ldots, v_{u-1} \rangle \oplus \langle i+n \rangle \oplus \langle v_{u+1}, \ldots, v_{n(s)} \rangle.$$

This creates arcs $\mathcal{A}_1^{\oplus} = \{(v_{p-1}, i), (i, v_{p+1})\}$, $\mathcal{A}_2^{\oplus} = \{(v_{o-1}, j), (j, v_{o+1})\}$, $\mathcal{A}_3^{\oplus} = \{(v_{u-1}, i+n), (i, v_{u+1})\}$, and $\mathcal{A}_4^{\oplus} = \{(v_{b-1}, j+n), (j, v_{b+1})\}$. We allow only moves where all arcs from $\mathcal{A}^{\oplus} = \bigcup_x \mathcal{A}_x^{\oplus}$ with $x \in X = \{1, 2, 3, 4\}$ are feasible, i.e., $\forall a \in \mathcal{A}^{\oplus} : a \in \mathcal{A}'$ and at least one arc from each $\mathcal{A}_x^{\oplus}$ is part of the sparse arcs, i.e., $\forall x \in X : (\exists a \in \mathcal{A}_x^{\oplus} : a \in \mathcal{A}'^{-})$. We remove arcs $\mathcal{A}^{\ominus} = \{(v_{o-1}, i), (i, v_{o+1}), (v_{p-1}, j), (j, v_{p+1}), (v_{b-1}, i+n), (i+n, v_{b+1}), (v_{u-1}, j+n), (j+n, v_{u+1})\}$.

- *stationInRe* was introduced in Schneider, Stenger, and Goeke (2014) and is used to insert or remove visits to recharging stations in intra-route fashion. We first describe the case that we insert a recharging station w before position o in route r:

  $$r' = \langle v_0, \ldots, v_{o-1} \rangle \oplus \langle w \rangle \oplus \langle v_o, \ldots, v_{n(r)} \rangle.$$

  In this way, we create two arcs $\mathcal{A}^{\oplus} = \{(v_{o-1}, w), (w, v_o)\}$ and remove one arc $\mathcal{A}^{\ominus} = \{(v_{o-1}, v_o)\}$. We opted for the following compromise to allow that visits to different stations are explored by GTS and at the same time the computational effort is kept low: We only consider the three closest feasible stations for insertion, i.e., w is from the set $\Lambda_{v_{o-1}, v_o} \subseteq \mathcal{F}'$ (this set can be precomputed for every combination of v_{o-1} and v_o). We generate Λ_{v_{o-1}, v_o} by considering all stations $h \in \mathcal{F}' : (v_{o-1}, h) \in \mathcal{A}' \wedge (h, v_o) \in \mathcal{A}'$, and then select the three stations h for which $d_{v_{o-1}h} + d_{hv_o} - d_{v_{o-1}v_o}$ is smallest. Note that only considering the station that causes the smallest total detour would be too limiting, as the remaining energy at v_{o-1} might be too low to reach this station. Finally, we consider the case that we remove a recharging station $j \in \mathcal{F}'$ from route s:

  $$s' = \langle v_0, \ldots, v_{p-1} \rangle \oplus \langle v_{p+1}, \ldots, v_{n(s)} \rangle.$$

  This creates a single arc $\mathcal{A}^{\oplus} = \{(v_{p-1}, v_{p+1})\}$ and removes two arcs $\mathcal{A}^{\ominus} = \{(v_{p-1}, j), (j, v_{p+1})\}$.

Additionally, we use a revoking criterion to quickly overcome solutions that contain infeasible arcs. If any removed arc is infeasible, i.e., $\exists a : a \in \mathcal{A}^{\ominus} \wedge a \notin \mathcal{A}'$, we create that move irrespective of whether the newly created arcs $a \in \mathcal{A}^{\oplus}$ are in the sparse graph or feasible (we prefer to explore solutions with different infeasible arcs to having a few infeasible arcs that persist for a long duration). Further, we do not select intra-route moves m that do not improve the current solution, i.e., $\Delta f_{gen}(m) \geq 0$ as we found that, being in a local optimum, there are typically more intra-route than inter-route moves that deteriorate the solution only slightly and are thus more likely to be selected. But, these do not contribute much in terms of exploring alternate search directions.

5.3.6. Tabu Criteria

During preliminary testing, we observed that a single tabu criterion is not equally well-suited for all neighborhood structures. For some, it was better to create a relationship between vertices and routes,

for others, it was more appropriate to take the arcs into account. Therefore, our tabu criterion depends on the type of move, and we use two lists of tabu items, $\Xi_1 = \{(i,r) \,|\, i \in \mathcal{V} \cup \mathcal{F}, r \in \mathcal{S}\}$ and $\Xi_2 = \{(i,j) \,|\, (i,j) \in \mathcal{A}\}$. In addition, it can be useful to distinguish whether a move improves or deteriorates the current solution in order to avoid cycles during the search and to guide GTS between feasible and infeasible solutions. Now, we first explain on which criteria we base the decision whether a move is tabu, and then which tabu items we generate when we perform a move:

First, we discuss how we treat the inter-route moves. Because the two-opt* reassigns multiple vertices to another route, it is more suitable and has a computational advantage to base the tabu definition on arcs and not on vertex-route pairs. Therefore, we set a two-opt* move tabu when at least one arc is tabu, i.e., $\exists a \in \mathcal{A}^{\oplus} : a \in \Xi_2$. For the other inter-route moves, using the first criterion helps to move both vertices, pickup and delivery, associated with a request to another route in order to respect the pairing constraints. Therefore, we say a relocate is tabu if $(i,s) \in \Xi_1$, an exchange is tabu if $(i,s) \vee (j,r) \in \Xi_1$, and a paired-exchange is tabu if $(i,s) \vee (i+n,s) \vee (j,r) \vee (j+n,r) \in \Xi_1$. In order to avoid that an intra-route move is too soon repeated, we consider it tabu if all created arcs $\mathcal{A}^{\oplus}$ are on the tabu list, i.e., $\forall a \in \mathcal{A}^{\oplus} : a \in \Xi_2$.

When we add tabu items to Ξ_1 and Ξ_2, we distinguish between moves m that improve $\mathcal{S}$ and moves that do not. The idea is that we do not want to repeat the former in the near future and do not want to immediately revert the latter, i.e., the rules for both cases complement each other. Now, we describe the first case $\Delta f_{gen}(m) < 0$. For intra-route moves and two-opt*, we add all created arcs $a \in \mathcal{A}^{\oplus}$ to Ξ_2. In addition, for two-opt* and stationInRe we also add the removed arcs $a \in \mathcal{A}^{\ominus}$ to Ξ_2. For relocate moves, we add (i,s), for exchange $\{(i,s),(j,r)\}$, and for paired-exchange $\{(i,s),(i+n,s),(j,r),(j+n,r)\}$ to Ξ_1. For the second case $\Delta f_{gen}(m) \geq 0$, we add for two-opt* the removed arcs $a \in \mathcal{A}^{\ominus}$ and also the added arcs $a \in \mathcal{A}^{\oplus}$ to Ξ_2. For relocate moves, we add (i,r), for exchange $\{(i,r),(j,s)\}$, and for paired-exchange $\{(i,r),(i+n,r),(j,s),(j+n,s)\}$ to Ξ_1. For the reasons explained in the previous section, we do not perform intra-route moves that deteriorate the current solution. When we add a tabu item to Ξ_1 or Ξ_2, we associate it with a tenure that we draw randomly from the interval $[4, 12]$. After that number of search iterations, we remove it from the list.

5.3.7. Set Covering

Periodically, we solve a set-covering problem involving all feasible routes that we found so far (see, e.g., Rochat and Taillard, 1995; Groër, Golden, and Wasil, 2011; Subramanian, Uchoa, and Ochi, 2013). In this way, we aim to find a new best solution and to continue the search from a better solution. For that reason, we store all feasible routes encountered in a set $\mathcal{R}$. Then, we associate each route $r \in \mathcal{R}$ with (i) a binary decision variable x_r that indicates whether the route is part of the new solution, (ii) coefficients b_{ri} that denote whether pickup $i \in \mathcal{P}$ is contained in r (because the routes are feasible, the delivery will also be included), and (iii) its contribution $f(r)$ to the objective function value, as defined in equation (5.29). In some instances, it can be difficult to obtain a feasible solution that serves all customers. Therefore, we allow solutions where not all requests are served, but we penalize them. To this end, let the binary decision variables $y_i = 1$ indicate that $i \in \mathcal{P}$ is not part of any selected route, and let ζ_i be a dynamically updated penalty factor. We set ζ_i to $10000(1.0 + 0.1\lambda_i)$ where λ_i is a variable that counts how many times the request was not served in a solution to the set-covering problem. In this way, we try to explore

different starting points to continue the search. We solve the following integer program with help of a commercial solver using a run-time limit of $(|\mathcal{R}|/100)$ seconds:

$$\min \quad \sum_{r\in\mathcal{R}} f(r)x_r + \sum_{i\in\mathcal{P}} \zeta_i y_i \tag{5.30}$$

$$\text{s.t.} \quad \sum_{r\in\mathcal{R}} b_{ri}x_r + y_i \geq 1 \qquad i \in \mathcal{P} \tag{5.31}$$

$$\sum_{r\in\mathcal{R}} x_r \leq |\mathcal{K}| \tag{5.32}$$

$$x_r \in \{0,1\} \qquad r \in \mathcal{R} \tag{5.33}$$

$$y_i \in \{0,1\} \qquad i \in \mathcal{P} \tag{5.34}$$

The objective is to minimize the sum of cost for routes and penalty for unserved requests as defined in (5.30). Constraints (5.31) ensure that each request is covered at least once by a route or that otherwise y is set. Constraints (5.32) limit the number of routes to the number of available vehicles, and constraints (5.33) and (5.34) define the range of variables.

When we restore $\mathcal{S}$ from the selected routes, we ensure that every request is included at most once in $\mathcal{S}$, and we add, one at a time, the requests associated with $y_i = 1$ in a greedy fashion such that $f_{gen}(\mathcal{S})$ is minimized. We continue the search from the new solution if the new solution is better than the previous best solution or if we have not obtained any feasible solution yet.

5.4. Numerical Studies

In this section, we present our numerical studies. First, we perform two series of experiments on the PDPTW-EV. To this end, we generated two sets of instances (Section 5.4.1.1). In the first series of experiments (Section 5.4.1.2), we solve our compact formulation on small-sized instances using Gurobi, and compare it to GTS in order to validate our solution method. In the second series (Section 5.4.1.3), we investigate on larger-sized instances how solution quality is affected when only full recharging is permitted instead of the option of partial recharging. Finally, we use GTS to solve the PDPTW and compare our method to other state-of-the-art heuristics on standard benchmark instances from the literature (Section 5.4.2).

We performed all tests on a desktop computer equipped with an AMD FX-6300 processor at 3.5 GHz with 8 GB of RAM and running Windows 10 Pro. We used Gurobi 6.5 as MILP solver to solve (i) the compact formulation (5.1)–(5.21), (ii) the set-covering problem (5.30)–(5.33), and (iii) the corresponding linear programming (LP) relaxation of (5.1)–(5.21) used to generate the granular arc set. GTS is implemented in Java. All calculations of GTS and Gurobi were performed using a single core, and the run-times are reported in seconds.

5.4.1. Experiments on PDPTW-EV

5.4.1.1. Generation of Test Instances

We generate two sets of instances: a set of 36 small instances with 3–9 requests and 2–8 recharging stations, and another set of 56 larger instances with 50 requests and 21 stations each.

The small instances are based on instances for the E-VRPTW introduced in Schneider, Stenger, and Goeke (2014). We determine pickup and delivery pairs in the following way that resembles the method proposed in Nanry and Barnes (2000): We first obtain an optimal solution to the corresponding VRPTW instance by solving each of the original instances with Gurobi, assuming that an unlimited battery capacity is available. Then, we randomly pair the customer vertices within each route of the solution in order to obtain pickup-delivery pairs. If their number is odd, we duplicate the last selected vertex. Finally, for each instance we take the characteristics of the vehicles and recharging stations from Schneider, Stenger, and Goeke (2014).

We generate the larger instances by combining the standard benchmark instances for the PDPTW introduced in Li and Lim (2001) and the instances for the E-VRPTW introduced in Schneider, Stenger, and Goeke (2014). As both sets are based on the same VRPTW instances, we use the pickup-delivery locations of the PDPTW instance and the vehicle characteristics and recharging stations of the E-VRPTW instance in order to generate a new instance. This is useful for benchmarking new solution methods because several of these instances are even today challenging for the majority of methods known from the literature. This difficulty is often governed by the properties of the instance, i.e., how the locations are distributed within the instances (randomly, clustered or randomly-clustered), the scheduling horizon (long or short), and the time windows (width and distribution).

During preliminary testing, we detected that we could not solve some of the new large instances when only full recharging is allowed. As we would like to compare that scenario to a partial recharging option, we repair these instances by increasing the battery capacity in steps of 10% until GTS finds a feasible solution.

5.4.1.2. Comparison between GTS and Compact Formulation

In this section, we investigate the performance of GTS in comparison to an exact solution approach. To this end, we perform 10 runs of GTS on the small-sized instances. Then, we restrict the run-time to one hour and solve them using Gurobi with the compact formulation introduced in Section 5.2. For this, we generate three visits for each recharging station. We think is a reasonable value for instances that contain at most 9 requests. Prior to this, in order to solve more instances optimally, we quickly generate a lower bound on the number of vehicles $|\mathcal{K}|$ by solving the PDPTW-relaxation of the PDPTW-EV, i.e., we set the battery capacity Q to a large value and remove the recharging stations from the problem. Now, we add the constraint $\sum_{j \in \mathcal{V}'} x_{0j} \geq |\mathcal{K}|$ to our formulation for the PDPTW-EV. In addition, we add the following simple symmetry breaking constraints that ensure that for any two visits associated with the same recharging station, we use the one with a lower identifier first: $\sum_{j \in \mathcal{V}'_{2n+1}} x_{ij} \geq \sum_{j \in \mathcal{V}'_{2n+1}} x_{wj} \quad \forall i, w \in \mathcal{F}' : \mathcal{F}(i) = \mathcal{F}(w) \wedge \Theta_i < \Theta_w$ with Θ_i the route identifier and $\mathcal{F}(i)$ a function

that returns the recharging station associated with visit i. Note that we also use the number of vehicles $|\mathcal{K}|$ that we get from solving the PDPTW instances as initial number for the feasibility phase of GTS.

Table 5.1 gives an overview of the results obtained for each of the three experiments, i.e., using Gurobi to solve the PDPTW-relaxation, using Gurobi to solve PDPTW-EV, and the best solution obtained for GTS in 10 runs. We report the instance name (Inst.), and for every experiment the number of vehicles used ($|\mathcal{K}|$) and the total distance traveled (f). We indicate optimal solutions to the PDPTW-EV in bold (assuming that no recharging station needs to be visited more than three times), and solutions in italics if GTS provided a better one than Gurobi. In addition, we provide the number of visits that took place to recharging stations ($|\tilde{\mathcal{F}}'|$). Further, we provide the run-time (t) for GTS as average over 10 runs, and for Gurobi as the time until an optimal solution was found, or we indicate–with TL–if the time limit of one hour was reached. In these cases, we also report the final lower bound obtained by Gurobi in terms of the number of vehicles ($|\mathcal{K}|_{\mathrm{L}}$) and the distance ($f_{\mathrm{L}}$). Note that the lower bound on the distance is only valid if upper and lower bounds on the number of vehicles are equal. For instances where this is guaranteed, we compare the lower bound for the PDPTW-EV to the lower bound that we can derive from the solution to the PDPTW. Then, we indicate the better of the two by underlining the corresponding value.

From the table, we see that Gurobi guarantees an optimal solution for 16 instances, and for another six instances at least the optimal number of vehicles. We further observe that GTS always matches the results obtained with the compact formulation. On four instances, the distance provided by GTS is even better (-0.29%, -2.32%, -5.82%, and -9.83%) than the upper bound reported by Gurobi. These four instances are also the only ones where the number of visits to recharging stations deviates between the two experiments. These very convincing results already clearly speak in favor of the ability of our recharging policy to handle the partial recharging aspect of the problem. In terms of run-time, GTS is significantly faster as it only requires on average about 17 compared to 2058 seconds for Gurobi, and when only considering the instances solved to optimality, 6 compared to 130 seconds. When we just compare the results between the two formulations, we conclude that PDPTW-EV is much more difficult to solve than PDPTW. All instances for the latter can easily be solved and need on average only nine seconds and at most three minutes. Often, also the final lower bound for PDPTW-EV after one hour of computation is still worse than for PDPTW.

5.4.1.3. Comparison between Full and Partial Recharging

This section intends to demonstrate that GTS is capable of solving instances with a practically important size in a reasonable run-time and that it makes adequate decisions on the amount recharged. Further, it provides some insight into the trade-off between the two extreme positions: vehicles are fully recharged at every visit to a recharging station or the amount recharged can be chosen freely. We perform for both settings 10 runs of GTS on the set of instances with 100 locations, and report the results in Table 5.2. Note that full recharging is implemented with a minor change to the recharging policy described in Section 5.3.3, i.e., we set $\bar{y}_w = Q - y_w$. We set the initial number of vehicles for the feasibility phase to the best value known for the corresponding instance of the PDPTW instance set, which we think is a reasonable lower bound.

For both settings, we report the best number of vehicles ($|\mathcal{K}|$), the corresponding minimum distance (f), the average run-time (t) and the number of visits to recharging stations $|\tilde{\mathcal{F}}'|$. Furthermore, we

Table 5.1.: Results on the small-sized instances obtained by Gurobi for the PDPTW-relaxation and for our compact formulation of PDPTW-EV, in comparison to the results of GTS.

	Gurobi (PDPTW)			Gurobi (PDPTW-EV)						GTS			
Inst.	$\|\mathcal{K}\|$	f	t	$\|\mathcal{K}\|$	f	$\|\tilde{\mathcal{F}}'\|$	$\|\mathcal{K}\|_{\mathrm{L}}$	f_{L}	t	$\|\mathcal{K}\|$	f	$\|\tilde{\mathcal{F}}'\|$	t
c101-6	2	240.00	0.1	**2**	**264.44**	3			1.3	**2**	**264.44**	3	5.7
c103-6	1	164.82	0.2	**1**	**175.37**	3			2.6	**1**	**175.37**	3	4.6
c206-6	1	236.51	0.1	**1**	**242.56**	3			4.5	**1**	**242.56**	3	5.5
c208-6	1	157.72	0.2	**1**	**158.48**	2			1.8	**1**	**158.48**	2	4.9
r104-6	1	132.81	0.1	2	191.62	3	1	203.70	TL	2	191.62	3	13.9
r105-6	2	151.15	0.1	**2**	**166.31**	3			1.0	**2**	**166.31**	3	4.9
r202-6	1	126.52	0.1	**1**	**128.78**	2			1.0	**1**	**128.78**	2	4.4
r203-6	1	178.05	0.2	**1**	**179.06**	3			1.2	**1**	**179.06**	3	5.0
rc105-6	2	227.18	0.1	**2**	**233.77**	3			1.6	**2**	**233.77**	3	5.2
rc108-6	2	245.87	0.1	**2**	**253.93**	3			1.1	**2**	**253.93**	3	5.6
rc204-6	1	172.03	0.1	**1**	**176.39**	3			2.7	**1**	**176.39**	3	4.8
rc208-6	1	162.67	0.2	**1**	**167.98**	2			1.5	**1**	**167.98**	2	4.7
c104-10	1	234.22	2.8	2	326.75	4	1	274.05	TL	2	326.75	4	22.0
c205-10	1	271.30	0.1	2	271.75	2	1	270.20	TL	2	271.75	2	22.9
r201-10	1	232.64	0.1	**1**	**241.51**	5			285.9	**1**	**241.51**	5	6.7
r203-10	1	213.65	14.3	**1**	**218.21**	4			7.5	**1**	**218.21**	4	5.5
rc108-10	2	329.79	0.5	4	468.09	4	2	352.01	TL	4	468.09	4	32.7
rc201-10	1	399.65	0.1	**1**	412.86	6	1	334.79	TL	**1**	412.86	6	7.3
rc205-10	1	446.06	0.2	2	424.32	5	1	336.93	TL	2	424.32	5	28.1
c101-12	3	366.33	0.1	**3**	**388.25**	5			839.4	**3**	**388.25**	5	7.7
c202-12	1	282.44	0.1	**1**	**304.06**	5			902.7	**1**	**304.06**	5	6.3
r102-12	3	241.14	0.1	**3**	**249.19**	3			16.3	**3**	**249.19**	3	6.6
r103-12	2	188.67	54.0	**2**	219.35	4	2	201.03	TL	**2**	219.35	4	8.1
rc102-12	3	384.75	0.1	4	429.50	2	3	415.91	TL	4	429.50	2	22.8
c103-16	2	344.06	0.4	3	380.67	5	2	311.61	TL	3	380.67	5	35.5
c106-16	2	274.35	0.2	3	365.88	4	2	305.93	TL	3	365.88	4	27.2
c202-16	1	534.34	22.4	2	532.96	7	1	350.45	TL	2	532.96	7	56.6
c208-16	1	315.90	0.3	2	328.30	4	1	284.90	TL	2	328.30	4	35.9
r105-16	3	312.41	0.1	4	396.64	5	3	318.93	TL	4	396.64	5	27.6
r202-16	1	405.63	11.3	2	467.79	9	1	305.57	TL	2	*440.56*	8	62.1
r209-16	1	299.69	14.2	**1**	341.04	7	1	288.08	TL	**1**	341.04	7	10.8
rc103-16	3	406.87	4.0	4	471.56	3	3	339.81	TL	4	471.56	3	27.5
rc108-16	3	358.78	149.9	**3**	398.51	4	3	380.86	TL	**3**	398.51	4	8.0
rc202-16	1	599.87	0.7	2	536.07	6	1	368.67	TL	2	*523.65*	8	53.1
rc204-16	1	314.29	56.0	**1**	427.52	6	1	244.09	TL	**1**	*385.48*	7	8.4
r102-18	5	377.76	0.2	**5**	421.14	7	5	372.48	TL	**5**	*419.90*	6	9.3
Avg.			9.3	2.1	315.57	4.1			2057.6	2.1	313.27	4.2	16.9

calculate gaps between the two settings in order to facilitate easier comparison, i.e., we report the absolute difference of the number of vehicles ($\Delta|\mathcal{K}|$) and visits ($\Delta|\tilde{\mathcal{F}}'|$), and the relative difference in distance (Δf) in percent.

Comparing the results, we find that there is no difference between the two settings in only four instances. In the other cases, full recharging is clearly inferior. On average, it requires another 0.5 vehicles and the distance is increased by 3.76%. This effect can be mainly attributed to instances from groups with a short scheduling horizon (lc1x, lr1x, and lrc1x) whereas for instances with a long scheduling horizon (lc2x, lr2x, and lrc2x) no additional vehicle is required for any instance and we observe only moderate detours. Further, we observe that the number of recharging visits decreases by 0.93 on average. From the driver's perspective, fewer stops might be desirable and increase the acceptance of the solution. From a cost perspective, recharging reduces the battery life, but it is difficult to tell how a reduced number of visits should be judged when considered on its own. Han, Han, and Aki (2014) propose a model to calculate the cost of battery wear and show that the cost is a function of the state-of-charge. They demonstrate that shallow as well as deep discharging should be avoided. If batteries are always fully recharged, it is cost efficient to actively use the middle range of the available capacity, i.e., the battery life is maximized if the battery is always discharged to about 30% and then recharged back to 100%. If the charging process can be interrupted, it is advantageous to always operate the battery at a low to medium state-of-charge. This, on the other hand, conflicts with the goal of providing a sufficient range and our prerequisite that vehicles are fully recharged at the depot during off-hours which makes certainly sense from a practical point of view.

Concerning run-time, GTS requires on average 162 and 222 seconds, for the partial and full recharging setting, respectively. The difference in run-time can be explained by the fact that it is more difficult to find a feasible solution with the second setting. Overall, we think the run-times are still moderate.

5.4.2. Experiments on PDPTW

For the PDPTW, instance sets that are currently studied in the literature were introduced in Li and Lim (2001) and are publicly available from `https://www.sintef.no/projectweb/top/pdptw/li-lim-benchmark`. We assess the performance of GTS on the benchmark set with 100 locations and compare our results to heuristics from the literature that report solutions for the same set and use a hierarchical objective function that minimizes the number of vehicles first and the distance second. The focus of more recent research on the PDPTW shifted to solving large instances, but we think this is beyond the scope of this work. Instead, we aim to demonstrate that our GTS is capable of effectively searching the solution space for instances that have a size that is realistic when EVs are first introduced for transportation services.

We compare our GTS to (i) a tabu-embedded simulated annealing approach introduced in Li and Lim (2001) that operates on feasible solutions only (TS+SA), (ii) a two-stage hybrid algorithm introduced in Bent and Hentenryck (2006) that first uses SA to minimize the number of vehicles and then LNS to minimize distance while preserving feasibility (SA+LNS), and (iii) an adaptive LNS that was proposed in Ropke and Pisinger (2006b) and which features an SA component for accepting solutions and a request bank to store requests that cannot be feasibly inserted (ALNS). We do not report results for (i) the genetic algorithm of Pankratz (2005) because it only minimizes distance and (ii) the generic VRP

Table 5.2.: Comparison between the partial and the full recharging option on newly-generated instances.

	Partial recharging allowed				Full recharging only				Gaps		
Inst.	$\|\mathcal{K}\|$	f	t	$\|\tilde{\mathcal{F}}'\|$	$\|\mathcal{K}\|$	f	t	$\|\tilde{\mathcal{F}}'\|$	$\Delta\|\mathcal{K}\|$	Δf (%)	$\Delta\|\tilde{\mathcal{F}}'\|$
lc101	11	954.57	50.72	9	11	989.20	74.02	7	0	3.63	-2
lc102	11	950.40	88.52	10	11	985.55	83.51	7	0	3.70	-3
lc103	10	998.21	308.08	10	11	963.51	305.95	7	1	-3.48	-3
lc104	10	902.35	210.84	8	10	960.19	270.44	8	0	6.41	0
lc105	10	843.32	52.60	8	11	974.22	177.48	7	1	15.52	-1
lc106	10	853.25	75.16	7	11	964.46	197.29	7	1	13.03	0
lc107	10	843.32	80.71	8	11	938.52	226.48	9	1	11.29	1
lc108	10	842.13	84.58	7	11	952.94	245.27	9	1	13.16	2
lc109	10	842.09	99.23	7	11	944.37	345.76	8	1	12.15	1
lc201	4	757.59	91.81	6	4	779.86	76.43	5	0	2.94	-1
lc202	4	706.53	86.09	6	4	744.21	113.12	6	0	5.33	0
lc203	4	689.61	103.38	5	4	740.88	137.86	4	0	7.44	-1
lc204	4	688.98	113.41	5	4	692.36	107.50	3	0	0.49	-2
lc205	4	685.47	69.99	5	4	690.06	82.62	4	0	0.67	-1
lc206	4	693.72	74.81	4	4	699.36	78.07	3	0	0.81	-1
lc207	4	660.97	70.86	4	4	660.97	82.61	4	0	0.00	0
lc208	4	649.47	73.27	3	4	649.47	60.87	3	0	0.00	0
lr101	20	1737.24	39.17	14	22	1855.17	169.80	11	2	6.79	-3
lr102	17	1517.09	106.36	13	19	1664.17	396.62	15	2	9.70	2
lr103	14	1424.11	140.58	15	15	1493.97	286.86	14	1	4.91	-1
lr104	12	1227.93	186.88	19	12	1273.53	226.47	16	0	3.71	-3
lr105	16	1715.32	220.16	31	19	1738.76	599.57	26	3	1.37	-5
lr106	14	1549.11	205.80	27	16	1548.48	441.57	19	2	-0.04	-8
lr107	13	1293.31	165.11	18	13	1359.75	178.90	17	0	5.14	-1
lr108	12	1222.85	178.85	17	13	1263.68	379.30	18	1	3.34	1
lr109	14	1458.36	172.08	24	15	1531.40	290.07	22	1	5.01	-2
lr110	13	1274.28	176.14	21	13	1366.76	341.82	20	0	7.26	-1
lr111	12	1319.23	288.83	22	13	1357.71	360.23	19	1	2.92	-3
lr112	12	1194.25	222.46	17	12	1293.14	373.50	17	0	8.28	0
lr201	4	1254.83	79.68	6	4	1256.35	94.22	6	0	0.12	0
lr202	3	1204.62	162.04	8	3	1210.06	156.93	7	0	0.45	-1
lr203	3	951.55	197.86	6	3	955.26	185.00	4	0	0.39	-2
lr204	2	866.87	551.78	6	2	895.36	478.15	4	0	3.29	-2
lr205	3	1056.94	144.55	8	3	1060.65	148.17	7	0	0.35	-1
lr206	3	933.03	154.49	4	3	933.03	204.85	4	0	0.00	0
lr207	2	915.02	345.54	4	2	915.03	242.44	5	0	0.00	1
lr208	2	738.31	182.80	4	2	739.41	243.97	4	0	0.15	0
lr209	3	937.82	165.06	6	3	945.49	177.51	5	0	0.82	-1
lr210	3	970.30	189.11	6	3	970.30	169.68	6	0	0.00	0
lr211	3	885.48	240.08	2	3	888.54	264.05	2	0	0.35	0
lrc101	18	1962.31	128.05	23	19	2063.67	168.54	21	1	5.17	-2
lrc102	17	1916.66	142.49	25	18	2050.31	212.58	24	1	6.97	-1
lrc103	13	1627.13	247.11	20	14	1704.20	267.95	19	1	4.74	-1
lrc104	12	1333.14	185.48	14	13	1470.23	326.23	15	1	10.28	1
lrc105	16	2048.31	210.14	21	17	2104.76	318.86	19	1	2.76	-2
lrc106	14	1740.33	159.69	23	16	1822.24	293.82	19	2	4.71	-4
lrc107	13	1452.14	146.36	16	14	1579.01	244.20	17	1	8.74	1
lrc108	12	1423.99	256.63	19	13	1470.01	308.02	19	1	3.23	0
lrc201	4	1414.51	93.06	7	4	1417.20	84.95	6	0	0.19	-1
lrc202	3	1383.22	177.22	6	3	1390.98	126.38	6	0	0.56	0
lrc203	3	1092.05	171.34	4	3	1101.83	191.54	5	0	0.90	1
lrc204	3	825.97	164.41	6	3	835.54	150.65	6	0	1.16	0
lrc205	4	1313.43	129.52	6	4	1314.39	122.99	5	0	0.07	-1
lrc206	3	1198.97	162.37	6	3	1214.19	162.55	5	0	1.27	-1
lrc207	3	1066.65	235.36	7	3	1080.79	191.55	6	0	1.33	-1
lrc208	3	865.65	196.88	6	3	877.10	170.43	6	0	1.32	0
Avg.			161.71				221.72		0.50	3.76	-0.93

solver described in Hasle and Kloster (2007) as, due to confidentiality reasons, no details on the solution method are given and we were not able to interpret the reported run-times; however, what we can say is that both methods miss best-known solution (BKS) in a few instances. We think that the mentioned algorithms sufficiently mark out the field of heuristics commonly applied to VRPs to offer a good basis for comparison.

Note that we initialize the feasibility phase with the best known value for the number of vehicles. As GTS starts with a low estimate of the number of vehicles and then increases this number if it fails to find a feasible solution, the solution quality would not be influenced if we started with a lower number. Clearly, the run-time would increase but we think that this elevated number would offer little insight and only complicate the comparison. Further, in many practical applications a reasonable range for the available vehicles is known (Ibaraki et al., 2008).

In Table 5.3, we report the best-known value (BKV) in the first row and then in the order of publication the following measures for the solution methods listed above: The number of runs performed (runs), the cumulative number of vehicles obtained in the best run (CNV_b), the corresponding cumulative traveled distance (CTD_b), as well as average values calculated over all runs (CNV_a and CTD_a). Further, we provide the average run-time (t) in seconds as given in the original paper. In order to enable a fairer comparison of run-times that takes the performance of the hardware into account, we assign the single-thread score (SCO) attributed by PassMark (`https://www.cpubenchmark.net`) to the respective CPU. We then use this score to convert the original time to a run-time (t') as if using our hardware. Note that we set the result reported for TS+SA in italics, because for one instance the distance reported is slightly lower than the optimal one.

Focusing on solution quality, we observe that the three last methods obtain results very close to each other, i.e., they all find the best known number of vehicles and only differ slightly in terms of distance. However, GTS is the only method that matches the BKS on every instance. Concerning average solution quality, GTS demonstrates a very convincing performance. It constantly provides the best number of vehicles whereas SA+LNS and ALNS use on average one additional vehicle. In terms of cumulative distance, GTS has on average a gap of only 0.11% compared to slightly increased values of 0.33% and 0.42% for ALNS and SA+LNS, respectively. In terms of run-time, ALNS clearly stands out with a converted time of only 16 seconds on average. On the other end of the range, SA+LNS is with 1031 seconds by far the slowest of the four methods, but it is set to always spend five minutes on each instance, therefore this is to be interpreted carefully. GTS uses on average 69 seconds. We think that this is a very reasonable value for a method based on local search, that is sufficient to serve most practical purposes.

Finally, we perform another experiment in order to investigate how changing the arc sparsification

Table 5.3.: Comparison of GTS to other heuristics from the literature.

	runs	CNV_b	CTD_b	CNV_a	CTD_a	t	SCO	t'
BKV		402	58059.50					
TS+SA	-	405	*58184.91*			769	200	109
SA+LNS	5	402	58061.82	403.0	58305.86	3600	403	1031
ALNS	10	402	58060.02	403.0	58249.42	66	344	16
GTS	10	402	58059.50	402.0	58126.15	69	1407	69

method influences solution quality (keeping the size of the sparse arc set constant). To this end, we generate the sparse arcs $\mathcal{A}'^{-}$ (i) with our base setting as described in Section 5.3.1 (denoted method **a**), (ii) with a simple procedure that sorts all arcs $(i, j) \in \mathcal{A}'$ by their length d_{ij} and then selects the shortest arcs (denoted **b**), and (iii) based on the arcs that are present in solutions to elementary shortest-path problems with resource constraints (ESPPRCs) (denoted **c**). The idea is that if in an solution only few vertices are visited between each pair of pickup and delivery, then promising arcs might be found in a restricted number of shortest paths from pickup to delivery. More precisely, for every pair of pickup and delivery, we solve 40 distinct ESPPRCs considering only the feasible arc set $\mathcal{A}'$. Each path must respect capacity and time windows but we allow that pairing and precedence constraints of the other vertices visited may be violated. We store all arcs present on the ESPPRCs, and then select the shortest arcs until $\mathcal{A}'^{-}$ has the desired size. If there are too few arcs available from the solutions to the ESPPRCs, we fill $\mathcal{A}'^{-}$ by selecting the shortest remaining edges from $\mathcal{A}'$.

In Figure 5.3, we illustrate how many arcs from the BKS to each instance are missing in $\mathcal{A}'^{-}$ when it is generated with each of the three methods (**a**, **b**, and **c**) described above. We find that for the majority of the 56 instances (33, 36, and 37 for **a**, **b**, and **c**, respectively) all arcs of the BKS are represented in $\mathcal{A}'^{-}$. For the other instances, the number of missing arcs is quite low and and does not exceed five. This seems to indicate that 25% of the size of $\mathcal{A}'$ is a reasonable value for the size of $\mathcal{A}'^{-}$. In total, 51, 40, and 36 arcs are missing for **a**, **b**, and **c**, respectively. This suggest that **c** should have been able to provide solutions closest to the BKS if the coverage of BKS arcs had been the decisive criterion.

However, as shown in Table 5.4, we find that **a** has the best performance of the three methods, i.e., in 10 runs, it provides at least once the BKS to every instance. In contrast, **b** and **c** miss the BKS for one instance. Further, **a** has also the best average solution quality in terms of vehicles (CNV_a) and distance (CTD_a); between **b** and **c**, **c** has the lower CNV_a and **b** the lower CTD_a. This clearly speaks for using the reduced cost as criterion to select the sparse arcs. But, on the other hand, **a** has also the highest run-time, i.e., 69 seconds compared to 40 seconds for **b** and **c** because the LP relaxation of formulation (5.1)–(5.21) needs to be solved. For our other studies, we chose **a** as base setting because we decided to favor solution quality over run-time in order to be able to provide high-quality solutions for the PDPTW-EV instances. Note that we annotate the run-time of **c** with † because the reported run-time does not include the time required to solve the ESPPRCs. For our experiment, we used Gurobi to solve the ESPPRCs a priori and then loaded the stored solutions. We think that with an efficient labeling algorithm the (currently prohibitively high) run-time for solving the ESPPRCs could be significantly decreased. But, because we opted for **a**, we refrained from implementing the labeling algorithm.

Table 5.4.: Comparison of three different methods for arc sparsification.

	runs	CNV_b	CTD_b	CNV_a	CTD_a	t
a (reduced cost)	10	402	58059.50	402.0	58126.15	69
b (shortest arcs)	10	402	58059.56	402.2	58377.25	40
c (ESPPRCs)	10	402	58059.56	402.0	58686.19	†40

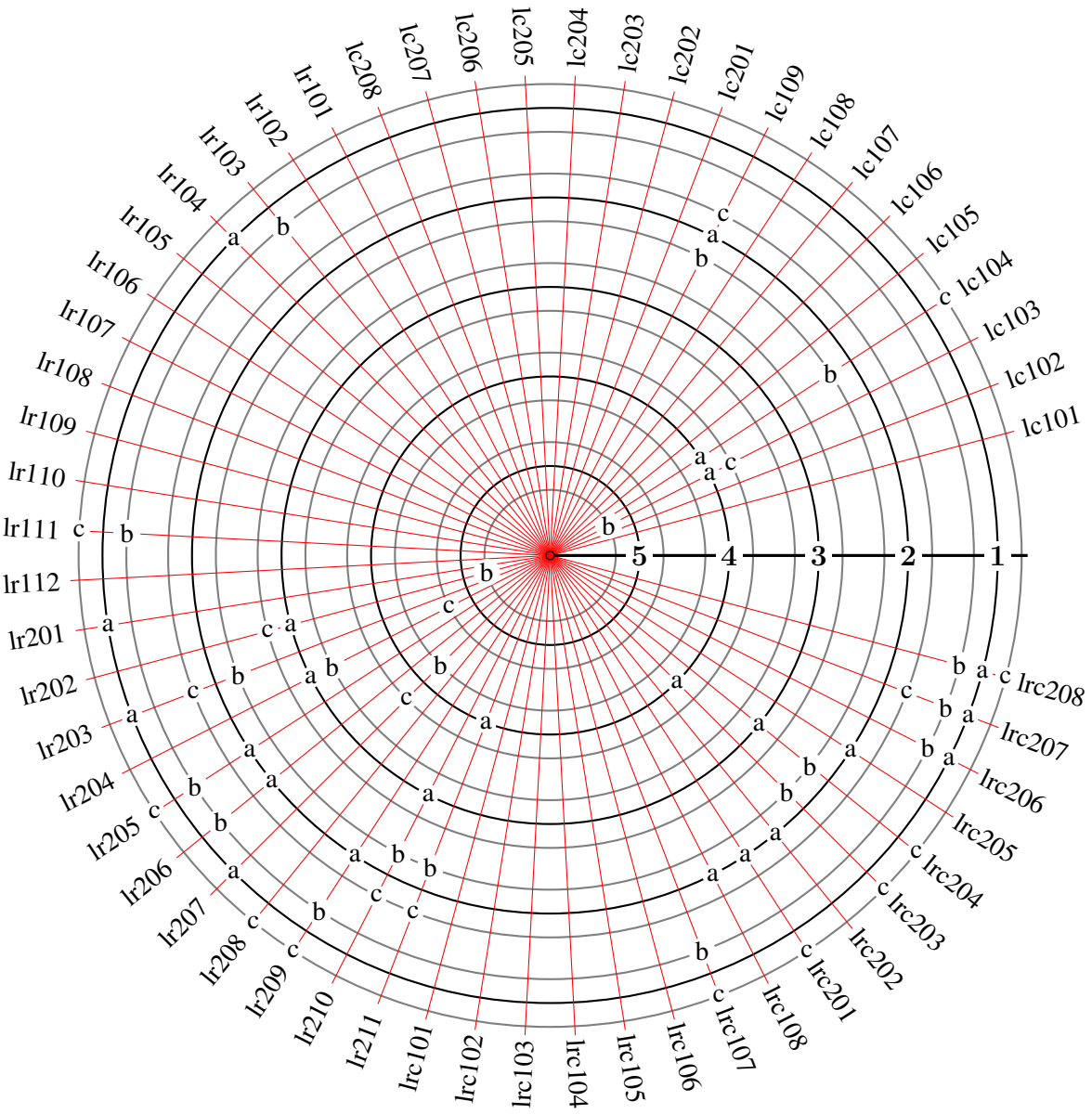

Figure 5.3.: Comparison of three different methods to generate the set of granular arcs. Arcs are sparsified based on reduced cost (a), shortest edges (b), and solutions to ESPPRCs (c). Then, for each instance the number of arcs that is present in the BKS but not present in the sparse arc set $\mathcal{A}'^{-}$ (generated once by each of thee three methods) is counted. These values are always between zero and five. We represent each value between one and five (omitting the zeros for clarity) by three circles, one for each method. Each instance is associated with a ray. In order to illustrate the distribution across the instances, we plot a, b, and c where the respective circle and ray intersect.

5.5. Summary and Conclusion

In this work, we introduce a variant of the PDPTW in which EVs are used for pickup and delivery operations. The vehicles have a limited battery capacity and their batteries can be recharged with an arbitrary amount of energy at dedicated recharging stations. Because of its relevance for real-world applications, we assume that this takes a time proportional to the amount of energy recharged. For testing purposes, we generate a set of small and another set of larger PDPTW-EV instances.

Further, we develop a novel GTS algorithm that features a policy for partial recharging. We validate our approach on small instances by comparing it to results obtained with Gurobi for our compact formulation. We find that the new formulation is considerably more difficult to solve than its PDPTW-relaxation. In contrast, the performance of GTS is excellent and matches or improves the solutions obtained with Gurobi within the time limit on every instance. On larger instances, we show that a partial recharging setting leads to a considerably reduced number of vehicles and total distance compared to a full recharging setting if the planning horizon is short. Finally, we demonstrate that GTS has a very convincing performance on benchmark instances for the PDPTW from the literature.

In future research, it might be worthwhile to examine the PDPTW-EV with an additional recharging infrastructure established at customer locations (Conrad and Figliozzi, 2011) in order to better utilize service and waiting times and reduce the additional time spent for recharging. In particular, it might be interesting to study this in combination with an accurate model, see, e.g., Barco et al. (2013) and Han, Han, and Aki (2014), to quantify the wear of the battery as a function of the state-of-charge and the amount of energy recharged.

Bibliography

Balinski, M. L. and R. E. Quandt (1964). On an integer program for a delivery problem. In: *Operations Research* 12 (2), pp. 300–304.

Barco, J., A. Guerra, L. Muñoz, and N. Quijano (2013). Optimal routing and scheduling of charge for electric vehicles: Case study. In: *CoRR* abs/1310.0145. URL: http://arxiv.org/abs/1310.0145.

Bent, R. and P. V. Hentenryck (2006). A two-stage hybrid algorithm for pickup and delivery vehicle routing problems with time windows. In: *Computers & Operations Research* 33 (4), pp. 875–893.

Bruglieri, M., F. Pezzella, O. Pisacane, and S. Suraci (2015). A variable neighborhood search branching for the electric vehicle routing problem with time windows. In: *Electronic Notes in Discrete Mathematics* 47, pp. 221–228.

Conrad R., G. and A. Figliozzi M. (2011). The recharging vehicle routing problem. In: *Proceedings of the 2011 Industrial Engineering Research Conference*. Ed. by T. Doolen and E. Van Aken. Reno, USA.

Cordeau, J.-F. (2006). A branch-and-cut algorithm for the dial-a-ride problem. In: *Operations Research* 54 (3), pp. 573–586.

Dekker, R., J. Bloemhof, and I. Mallidis (2012). Operations research for green logistics: An overview of aspects, issues, contributions and challenges. In: *European Journal of Operational Research* 219 (3), pp. 671–679.

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

Desaulniers, G., F. Errico, S. Irnich, and M. Schneider (2016). Exact algorithms for electric vehicle-routing problems with time windows. In: *Operations Research* 64 (6), pp. 1388–1405.

Deutsche Post DHL Group (2016). *Logistics trend radar*. DHL Customer Solutions & Innovation. Accessed December 16, 2017. Troisdorf, Germany. URL: http://www.dhl.com/en/about_us/logistics_insights/dhl_trend_research/trendradar.html#.Wf9MZLpFxZQ.

Dumas, Y., J. Desrosiers, and F. Soumis (1991). The pickup and delivery problem with time windows. In: *European Journal of Operational Research* 54 (1), pp. 7–22.

Felipe, Á., M. T. Ortuño, G. Righini, and G. Tirado (2014). A heuristic approach for the green vehicle routing problem with multiple technologies and partial recharges. In: *Transportation Research Part E: Logistics and Transportation Review* 71, pp. 111–128.

Furtadoa, M. G., P. Munaria, and R. Morabitoa (2017). Pickup and delivery problem with time windows: a new compact two-index formulation. In: *Operations Research Letters* 45 (4), pp. 334–341.

Gendreau, M., A. Hertz, and G. Laporte (1994). A tabu search heuristic for the vehicle routing problem. In: *Management Science* 40 (10), pp. 1276–1290.

Goeke, D. and M. Schneider (2015). Routing a mixed fleet of electric and conventional vehicles. In: *European Journal of Operational Research* 245 (1), pp. 81–99.

Grandinetti, L., F. Guerriero, F. Pezzella, and O. Pisacane (2016). A pick-up and delivery problem with time windows by electric vehicles. In: *International Journal of Productivity and Quality Management* 18 (2-3), pp. 403–423.

Groër, C., B. Golden, and E. Wasil (2011). A parallel algorithm for the vehicle routing problem. In: *INFORMS Journal on Computing* 23 (2), pp. 315–330.

Han, S., S. Han, and H. Aki (2014). A practical battery wear model for electric vehicle charging applications. In: *Applied Energy* 113, pp. 1100–1108.

Hasle, G. and O. Kloster (2007). Industrial vehicle routing. In: *Geometric Modelling, Numerical Simulation, and Optimization: Applied Mathematics at SINTEF*. Ed. by G. Hasle, K.-A. Lie, and E. Quak. Berlin, Heidelberg: Springer, pp. 397–435.

Hiermann, G., J. Puchinger, S. Ropke, and R. F. Hartl (2016). The electric fleet size and mix vehicle routing problem with time windows and recharging stations. In: *European Journal of Operational Research* 252 (3), pp. 995–1018.

Ibaraki, T., S. Imahori, K. Nonobe, K. Sobue, T. Uno, and M. Yagiura (2008). An iterated local search algorithm for the vehicle routing problem with convex time penalty functions. In: *Discrete Applied Mathematics* 156 (11), pp. 2050–2069.

Keskin, M. and B. Çatay (2016). Partial recharge strategies for the electric vehicle routing problem with time windows. In: *Transportation Research Part C: Emerging Technologies* 65, pp. 111–127.

Li, H. and A. Lim (2001). A metaheuristic for the pickup and delivery problem with time windows. In: *Proceedings 13th IEEE International Conference on Tools with Artificial Intelligence (ICTAI'01)*. Ed. by D. Moldovan. IEEE Press, pp. 160–167.

Lin, S. (1965). Computer solutions of the traveling salesman problem. In: *Bell System Technical Journal* 44 (10), pp. 2245–2269.

Montoya, A., C. Guéret, J. E. Mendoza, and J. G. Villegas (2017). The electric vehicle routing problem with nonlinear charging function. In: *Transportation Research Part B: Methodological* 103, pp. 87–110.

Nagata, Y., O. Bräysy, and W. Dullaert (2010). A penalty-based edge assembly memetic algorithm for the vehicle routing problem with time windows. In: *Computers & Operations Research* 37 (4), pp. 724–737.

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

Pankratz, G. (2005). A grouping genetic algorithm for the pickup and delivery problem with time windows. In: *OR Spectrum* 27 (1), pp. 21–41.

Pelletier, S., O. Jabali, and G. Laporte (2016). 50th anniversary invited article–Goods distribution with electric vehicles: Review and research perspectives. In: *Transportation Science* 50 (1), pp. 3–22.

Potvin, J.-Y. and J.-M. Rousseau (1995). An exchange heuristic for routeing problems with time windows. In: *Journal of the Operational Research Society* 46 (12), pp. 1433–1446.

Preis, H., S. Frank, and K. Nachtigall (2014). Energy-optimized routing of electric vehicles in urban delivery systems. In: *Operations Research Proceedings 2012 – Selected Papers of the International Annual Conference of the German Operations Research Society (GOR)*. Ed. by S. Helber, M. Breitner, D. Rösch, C. Schön, J.-M. Graf von der Schulenburg, P. Sibbertsen, M. Steinbach, S. Weber, et al. Cham: Springer International Publishing, pp. 583–588.

Rochat, Y. and É. D. Taillard (1995). Probabilistic diversification and intensification in local search for vehicle routing. In: *Journal of Heuristics* 1 (1), pp. 147–167.

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

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

Savelsbergh, M. (1992). The vehicle routing problem with time windows: Minimizing route duration. In: *ORSA Journal on Computing* 4 (2), pp. 146–154.

Schneider, M., B. Sand, and A. Stenger (2013). A note on the time travel approach for handling time windows in vehicle routing problems. In: *Computers & Operations Research* 40 (10), pp. 2564–2568.

Schneider, M., F. Schwahn, and D. Vigo (2017). Designing granular solution methods for routing problems with time windows. In: *European Journal of Operational Research* 263 (2), pp. 493–509.

Schneider, M., A. Stenger, and D. Goeke (2014). The electric vehicle-routing problem with time windows and recharging stations. In: *Transportation Science* 48 (4), pp. 500–520.

Subramanian, A., E. Uchoa, and L. S. Ochi (2013). A hybrid algorithm for a class of vehicle routing problems. In: *Computers & Operations Research* 40 (10), pp. 2519–2531.

Toth, P. and D. Vigo (2003). The granular tabu search and its application to the vehicle-routing problem. In: *INFORMS Journal on Computing* 15 (4), pp. 333–346.

Vidal, T., T. G. Crainic, M. Gendreau, and C. Prins (2013). A hybrid genetic algorithm with adaptive diversity management for a large class of vehicle routing problems with time-windows. In: *Computers & Operations Research* 40, pp. 475–489.

Waters, C. D. J. (1987). A solution procedure for the vehicle-scheduling problem based on iterative route improvement. In: *Journal of the Operational Research Society* 38 (9), pp. 833–839.

Chapter 6

Conclusion

6.1. Summary and Conclusion

This thesis investigates four variants of the vehicle-routing problem (VRP). First, it focuses on two well-established and fundamental problems that are highly relevant in many logistics applications, contributing to them from a methodological point of view. Then, this work turns towards two new problems that are aligned with the introduction of electric commercial vehicles (ECVs) and are practically relevant in the context of small-package shipping and urban logistics. Several methodological ideas and insights can be carried over to other VRPs as they are generic to some extent.

Chapter 2 studies the VRP with private fleet and common carrier (VRPPC). For the VRPPC, the first exact method is proposed, i.e., a branch-price-and-cut method (BPC). Routes are generated by solving shortest-path problems with resource constraints and restricted non-elementary paths, and heuristic pricers are used as acceleration technique. A hierarchical branching scheme ensures integrality. In addition, a large neighborhood search (LNS) is developed to solve the VRPPC heuristically. The LNS features a dynamic problem decomposition. In every iteration, subproblems are generated based on the current solution, and removal and insertion of customers are only performed on these subproblems. The solutions to modified subproblems are reintegrated into the solution to the original problem according to a criterion based on simulated annealing. Subsequently, a variable neighborhood descent improves the solutions of the original problem. The BPC provides tight lower bounds on large instances and can solve medium-sized instances with up to 75 customers to optimality. The LNS provides many new upper bounds, especially for instances with a heterogeneous vehicle fleet and is among the best heuristic methods for the VRPPC.

The high quality of the lower bounds for the VRPPC indicates that it should be within reach to solve several of the larger-sized instances to optimality if the pricing problems can be solved more efficiently. To this end, in the course of this thesis, two heuristics were tested (i) the LNS algorithm to heuristically solve the pricing subproblems in addition to using it for the complete problem and (ii) an iterated nearest-neighbor heuristic featuring a tabu criterion in order to explore different paths. Unfortunately, even though both heuristics worked in principle, they proved to be too slow in practice. The decomposition procedure used in the LNS, which generates subproblems based on routes, proved to be successful to solve the VRPPC. It would be interesting to learn whether LNS with decomposition is also suitable for other VRPs as it is quite easy to implement, possibly also using other decomposition strategies.

Chapter 3 investigates the consistent VRP (ConVRP), a multi-day problem in which driver consistency and arrival time consistency (ATC) must be respected when serving the customers. The first exact method, called cluster column generation (CCG), and an improved mixed-integer linear programming (MILP) formulation are proposed for the ConVRP. For CCG, a column does not represent a single route, which is usually a typical choice for VRPs, but a cluster of customers served by the same driver on all days of the planning horizon. Lower bounds are obtained by solving a traveling-salesman problem (TSP) for every customer cluster on each day. At a later stage, a MILP solver generates routes respecting ATC. As upper bounding procedure, an LNS is developed. An important element of the LNS is a re-optimization procedure in which routes are first inverted and then customers are relocated such that ATC is improved. New medium-sized instances are generated to test the performance of the exact method in a structured fashion, and most instances with up to 30 customers and five days can be solved to optimality. The LNS provides excellent upper bounds, and tested as stand-alone algorithm on large instances is clearly superior to all solution methods for the ConVRP suggested in the literature.

For medium-sized instances of ConVRP, CCG is very fast, and probably no substantial run-time improvements can be achieved or would be helpful in practice. However, the shortcoming of CCG is its limitation to medium-sized instances. CCG is mainly restricted by its memory requirements that increase exponentially with the instance size. The reason is that CCG depends on the complete enumeration of all customer clusters. If the clusters cannot be enumerated a priori, no meaningful lower bound can be provided. It would be desirable to develop an exact method that could theoretically be used to approach large instances. However, other column-generation formulations that keep the consistency requirements in the master problem seem to provide weaker lower bounds and are thus less suitable for a branch-and-cut approach. On the other hand, solving pricing problems over multiple days representing them as elementary shortest-path problems with resource constraints (ESPPRCs) might be not applicable. This makes further steps a challenging endeavor.

Chapter 4 introduces the electric VRP with mixed fleet (E-VRPTWMF). E-VRPTWMF uses realistic energy consumption functions for ECVs and internal combustion commercial vehicles and models the relationship between energy consumption and cargo load of a vehicle. The idea is that it might be desirable, in terms of saving energy, to serve customers with high demand early in a route even if a a detour needs to be accepted in exchange. An adaptive LNS is developed to solve E-VRPTWMF. Because the calculation of violations is computationally expensive when cargo load is taken into account, the ALNS uses a surrogate cost function to select promising moves. Then, the correct cost function only evaluates the promising moves. Numerical experiments performed on newly generated instances show that it is beneficial to consider the actual load and that the choice of objective function strongly influences the share of ECVs used. In addition, ALNS has better solution quality and run-time than all previous methods for the electric vehicle-routing problem with time windows and is competitive on instances for the vehicle-routing problem with time windows.

A lesson learned from the E-VRPTWMF is that a simplified objective function can be adequate if it is computationally too expensive to evaluate the neighborhood of a solution in detail. A promising subset of the neighborhood can then be evaluated with the real objective function to select the next solution from which the search is continued.

Chapter 5 proposes the electric pickup and delivery problem with time windows and electric vehicles (PDPTW-EV) with partial recharging option. As first solution method for the PDPTW-EV, a granular

tabu search (GTS) is designed. The GTS uses conventional VRP moves and acknowledges the coupling constraints of pickups and deliveries with a penalty term in the objective function. On new instances, it is demonstrated that partial recharging is advantageous compared to mandatory full recharging. Further, the GTS provides state-of-the-art solutions with moderate run-times for instances of the pickup and delivery problem with time windows.

Recent advances in the exact domain were made concerning VRPs with electric vehicles (EVs) and VRPs with a pickup and delivery structure, see Desaulniers et al. (2016) and Gschwind et al. (2018), respectively. Combining these with the high-quality upper bounds provided from the GTS, should allow a straight-forward implementation of an exact solution method for PDPTW-EV. Lower bounds could also indicate whether further heuristics need to be developed. When developing the GTS, the tabu rules were critical for the success of GTS, i.e., they need to be carefully adapted to the operators used. However, an yet open research question is how to approach larger-sized instances with GTS. GTS is very effective for medium-sized instances, however an inherent conflict of tabu search is that short tabu durations are typically too weak to move the search to different regions of the solution space if instances are large and long tabu durations make it difficult to comprehensively search the close region of a particular solution.

Finally, it can be remarked that relaxing constraint violations and accounting for them with penalties was always superior to restricting the search to feasible solutions. This was the case for all tests performed in the course of this thesis, even when feasibility was easy to restore. But, this should be interpreted with caution as there are might be limits to the extent to which this observation can be generalized.

6.2. Outlook

Several of the trends described in the introduction have not yet been sufficiently studied. One promising direction of research is to develop models for the simultaneous routing of drone-carrying platforms and drones. Until now, there is little literature regarding the optimization of such problems and, to the best of our knowledge, so far only the routing of a single platform serving as mobile basis for the drones has been considered, see, e.g., Murray and Chu (2015), Savuran and Karakaya (2016), and Ferrandez et al. (2016).

Two approaches to model the movement of drone-carrying platforms come to mind: (i) platforms are restricted to stops at dedicated locations, e.g., points on a grid or suitable waiting positions in a city, or (ii) they can move and stop freely. In the first scenario, the problem is closely related to the two-echelon capacitated VRP (2E-CVRP), see, e.g., Baldacci, Mingozzi, Roberti, and Calvo (2013). However, the 2E-CVRP typically only consider a small number of satellites whereas one can expect that there are many potential stops for platforms—this might drastically change how the problem needs to be approached from an algorithmic point of view. In addition, one typically abstracts from timing aspects in the 2E-CVRP but might need to take them into account when routing drones. This is especially the case if drones do not need to return to the platform from which they departed but can be exchanged between platforms. In the second scenario, the problem can be seen as related to the TSP with neighborhoods in which the tour must not exactly visit a customer but only traverse its neighborhood. For halting position available on the plane or in a street network, see, e.g., Arkin and Hassin (1994) and Shuttleworth et al. (2008), respectively. Note that so far the TSP with neighborhoods has not been studied for multiple tours.

The problem of placing and re-positioning micro-depots shares some similarities with the VRP with trailers and transshipments (Drexl, 2013). The micro-depots can be considered non-autonomous trailers and the lorries are tasked with relocating them. However, a major difference when operating mobile depots in urban areas is that the trucks are not allowed to visit customers; for that purpose there is another category of task vehicles—the load tricycles—that in turn can not pull the lorries.

Both problems described above could also be conceived as bilevel optimization problems. In bilevel optimization, a low-level problem is nested in a high-level problem, the decisions of the higher-level problem set the frame for the lower-level problem, that is optimized autonomously within these boundaries. Thus, the optimal decisions for the lower-level problem must be anticipated in the higher-level problem. For an overview, see, Colson, Marcotte, and Savard (2007). Recently, a solver for bilevel optimization problems, which is able to cope with integer and continuous variables on both levels, was proposed in Fischetti et al. (2017). As it was made publicly available, it could be interesting to use it as a starting point in order to investigate both problems more closely. In such a bilevel setting, the drones/tricycles would act as autonomous agents that can interact with the stops of the platforms/micro-depots, however the reaction of the agents need to be taken into account when planning the first-level routes.

With regard to the BPC for the VRPPC, one could contemplate to use a parallel architecture, e.g., a graphics processing unit, to obtain heuristics solutions for the pricing problem in shorter run-times, i.e., start several runs in parallel. Further, it is notable that large-sized instances for the heterogeneous fleet VRP could recently be solved in Sadykov, Uchoa, and Pessoa (2017). They use a bucket graph to organize the labels of the ESPPRCs that are solved in the pricing and thus are able to perform dominance check on labels more efficiently. As the problem studied is closely related to the VRPPC, this might be a promising direction to investigate. From a problem-oriented point of view, it could be promising to extend the VRPPC to a multi-period problem. In last-mile delivery, customers can often specify the preferred date of delivery, however it might also be acceptable that deliveries arrive one or several days late. This presents an opportunity to schedule the visits to customers over multiple days and to subcontract only those that cannot be profitably served on any day. This optimization problem can then, e.g., be embedded in a rolling-horizon decision making process and optimized in ongoing fashion.

Another opportunity presents itself in the context of the routing of EVs. The models discussed in this thesis assume that recharging stations are privately owned and their availability is not limited. Recently, Froger et al. (2017) are the first to introduce the notion of a limited number of chargers at recharging stations; but, they consider them to be fully at the disposal of the decision-maker. It could be interesting to study a scenario where electric vehicles share a public infrastructure of recharging stations complementing (or substituting) the privately operated infrastructure. In this setting, chargers might be occupied when EVs arrive, and one would either need to wait for the next free slot or visit another recharging station if enough energy is left. To this end, a model based on queueing theory could be developed in order to determine a decision policy. In addition, to improve the robustness of solutions, stochastic influences on the energy consumption, e.g., traffic congestion, temperature, and driving behavior, need to be incorporated into models for VRPs with EVs. So far, the effect of stochastic variables on the energy consumption of electric vehicles has only been considered in Jafari and Boyles (2017) in the context of finding the shortest path for an EV.

Finally, some general methodological ideas are presented that are related to the combination of artificial learning methods and optimization. Comparatively scarce work has been done related to the learning of

decision making in exact methods (Lodi and Zarpellon, 2017). The branching decisions influence the performance of branch-and-bound methods; however they are inherently heuristic. Learning approaches provide a data-driven foundation. For example, one could try to exploit information about the shape of the decision tree, e.g., in terms of depth, breadth and number of branches. Other sources of information are the already expanded nodes and the structure of the problem instances, i.e., the constraints and objective function coefficients. Two well-known techniques to improve branch-and-bound search are strong branching and pseudo-cost branching. Strong branching aims to look ahead by tentatively fixing fractional variables and observing the impact on the lower bound. However, as this is computationally expensive, artificial learning could present a chance to identify a promising subset of variables for strong branching. On the other hand, pseudo-cost branching looks back at the past success of a variable by assigning a score to each variable that captures information about the gain achieved by branching on this variable in terms of feasibility and improvement of the lower bound. As the pseudo-costs are a very simple measure, more complex methods, e.g., the training of neural nets, might offer a chance to improve the pseudo-cost strategy. Often, branch-and-bound methods will fail if the initial optimality gap is too large. Valid inequalities (cuts) can potentially strengthen the linear relaxation; however violated inequalities need to be identified by a separation procedure. For example, for the CVRP there exists an exponential number of capacity constraints, i.e., the minimum number of vehicles required to serve the demand of every subset of customers. In order to identify violated capacity constraints, one needs to check subsets of customers, and as there are many, one often resorts to heuristics to identify promising ones (Augerat et al., 1998). In addition, as often the inequalities have little impact on the lower bound but only increase the size of the linear program, one needs to decide how many and which cuts are added to the linear relaxation. Both tasks could be supported by artificial intelligence techniques.

Concerning the domain of metaheuristics, it is striking that LNS often provides high-quality solutions if the insertion and removals operators are adapted to the problem characteristics and include information related to the history of the search. One often finds a set of functions that rate moves based on simple expressions and work well; however determining them manually is often time consuming and it is likely that good ones are missed. It could be worthwhile to explore a direction of research where one automates this approach by specifying a function comprised of several weighted terms derived from the problem properties, e.g., for the case of the VRPPC, demand, cost of subcontracting, detour, and capacity violation. Then, one could try to find appropriate functions and randomization strategies. The idea is closely related to the idea of ALNS introduced in Ropke and Pisinger (2006b); however in the original setting one aims to determine a subset from a defined set of operators that work well on a specific instance and select these with a higher probability. The idea here is to first derive a set of operators from a generic function that work well over a broad range of instances for a given problem.

Bibliography

Arkin, E. M. and R. Hassin (1994). Approximation algorithms for the geometric covering salesman problem. In: *Discrete Applied Mathematics* 55 (3), pp. 197–218.

Augerat, P., J. Belenguer, E. Benavent, A. Corberán, and D. Naddef (1998). Separating capacity constraints in the CVRP using tabu search. In: *European Journal of Operational Research* 106 (2), pp. 546–557.

Baldacci, R., A. Mingozzi, R. Roberti, and R. W. Calvo (2013). An exact algorithm for the two-echelon capacitated vehicle routing problem. In: *Operations Research* 61 (2), pp. 298–314.

Colson, B., P. Marcotte, and G. Savard (2007). An overview of bilevel optimization. In: *Annals of Operations Research* 153 (1), pp. 235–256.

Desaulniers, G., F. Errico, S. Irnich, and M. Schneider (2016). Exact algorithms for electric vehicle-routing problems with time windows. In: *Operations Research* 64 (6), pp. 1388–1405.

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

Ferrandez, S. M., T. Harbison, T. Weber, R. Sturges, and R. Rich (2016). Optimization of a truck-drone in tandem delivery network using k-means and genetic algorithm. In: *Journal of Industrial Engineering and Management* 9 (2).

Fischetti, M., I. Ljubić, M. Monaci, and M. Sinnl (2017). A new general-purpose algorithm for mixed-integer bilevel linear programs. In: *Operations Research* 65 (6), pp. 1615–1637.

Froger, A., J. E. Mendoza, O. Jabali, and G. Laporte (2017). *A Matheuristic for the Electric Vehicle Routing Problem with Capacitated Charging Stations*. Tech. rep. 2017-31. CIRRELT, Canada.

Gschwind, T., S. Stefan Irnich, A.-K. Rothenbächer, and C. Tilk (2018). Bidirectional labeling in column-generation algorithms for pickup-and-delivery problems. In: *European Journal of Operational Research* 266 (2), pp. 521–530.

Jafari, E. and S. D. Boyles (2017). Multicriteria stochastic shortest path problem for electric vehicles. In: *Networks and Spatial Economics* 17 (3), pp. 1043–1070.

Lodi, A. and G. Zarpellon (2017). On learning and branching: A survey. In: *TOP* 25 (2), pp. 207–236.

Murray, C. C. and A. G. Chu (2015). The flying sidekick traveling salesman problem: Optimization of drone-assisted parcel delivery. In: *Transportation Research Part C: Emerging Technologies* 54, pp. 86–109.

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

Sadykov, R., E. Uchoa, and A. Pessoa (2017). *Enhanced Branch-Cut-and-Price Algorithm for Heterogeneous Fleet Vehicle Routing Problems*. Tech. rep. L-2017-7. Niterói, Brazil: Cadernos do LOGIS-UFF.

Savuran, H. and M. Karakaya (2016). Efficient route planning for an unmanned air vehicle deployed on a moving carrier. In: *Soft Computing* 20 (7), pp. 2905–2920.

Shuttleworth, R., B. L. Golden, S. Smith, and E. Wasil (2008). Advances in meter reading: Heuristic solution of the close enough traveling salesman problem over a street network. In: *The Vehicle Routing Problem: Latest Advances and New Challenges*. Ed. by B. Golden, S. Raghavan, and E. Wasil. Boston, MA, USA: Springer, pp. 487–501.

Appendices

Appendix A

Updated Results for the TS Algorithm of Côté and Potvin (2009)

Table A.1 provides updated results for the tabu search (TS) introduced in Côté and Potvin (2009). The coordinates of customers in instances of sets G and G-H are specified with a precision of four decimal places, but the results reported in Côté and Potvin (2009) are obtained treating them as integers. The authors repeated their tests with the higher precision and provided us with the results. Columns denoted f_{best} report the best objective value of TS obtained in tens runs and columns t(sec) the total run-time in seconds to perform tens runs using an I7 processor with 3.3 GHz. Note that the experiments in Potvin and Naud (2011) were also performed with integer coordinates. However, we cannot provide updated results for TS+ because we could not reach the author with the latest code version.

Table A.1.: Updated results for the TS of Côté and Potvin (2009).

	TS			TS	
Inst.	f_{best}	t(sec)	Inst.	f_{best}	t(sec)
CE			**CE-H**		
CE-01	1119.47	75	CE-H-01	1191.70	80
CE-02	1816.28	96	CE-H-02	1792.36	97
CE-03	1922.19	207	CE-H-03	1918.47	208
CE-04	2529.40	350	CE-H-04	2481.46	356
CE-05	3113.33	487	CE-H-05	3150.06	464
CE-06	1207.47	77	CE-H-06	1208.08	77
CE-07	2006.52	95	CE-H-07	2029.40	93
CE-08	2065.95	208	CE-H-08	1989.61	219
CE-09	2437.76	331	CE-H-09	2451.16	332
CE-10	3406.67	491	CE-H-10	3270.25	496
CE-11	2332.03	307	CE-H-11	2335.25	302
CE-12	1952.86	154	CE-H-12	1912.47	154
CE-13	2862.16	307	CE-H-13	2871.79	304
CE-14	2219.31	167	CE-H-14	1925.46	167
G			**G-H**		
G-01	14284.07	1306	G-H-01	14233.25	1345
G-02	19675.29	2484	G-H-02	18727.91	2618
G-03	25543.93	5918	G-H-03	26042.63	4895
G-04	36221.59	8481	G-H-04	36518.80	7677
G-05	14866.58	1917	G-H-05	15897.83	1530
G-06	22455.49	3384	G-H-06	20582.61	2585
G-07	24203.51	4794	G-H-07	24228.65	4436
G-08	30822.56	6154	G-H-08	28505.81	6413
G-09	1326.26	1085	G-H-09	1328.61	1079
G-10	1593.79	1733	G-H-10	1557.13	2036
G-11	2173.82	3073	G-H-11	2191.75	3018
G-12	2494.56	4901	G-H-12	2501.02	4836
G-13	2274.56	596	G-H-13	2237.86	657
G-14	2702.49	1066	G-H-14	2683.88	1124
G-15	3162.90	1921	G-H-15	3122.68	1989
G-16	3643.39	3002	G-H-16	3620.27	3123
G-17	1674.64	563	G-H-17	1702.66	607
G-18	2752.80	919	G-H-18	2754.05	940
G-19	3523.57	1654	G-H-19	3493.83	1657
G-20	4355.19	2378	G-H-20	4360.03	2383

Appendix B

Detailed results on Datasets A and D for the ConVRP

In Tables B.1–B.5, we report detailed results for large neighborhood search with set partitioning (LNS-SP)–25k and CCG on Datasets A and D. For LNS-SP–25k, we perform 10 runs on the instances of Dataset D, but only one run on those of Dataset A because the latter are easy to solve. We report the best objective function value UB_{init} obtained in the specified number of runs and the total time t_{heu} to perform all runs. Additionally, we report the time t_{clu} that we spend to generate the routes Φ and clusters $\hat{\Omega}$ in Step 2 of CCG. We calculate percentage gaps Δ_x of lower bound x to the best upper bound obtained, i.e., the minimum of UB_{final} and UB_{init}. Optimal solutions are marked in bold, and timeouts in italics. Finally, we indicate all cases of insufficient memory with OOM.

Table B.1.: Detailed results on small instances of Dataset A.

	LNS-SP		CCG													
Inst.	UB_{init}	t_{heu}	$\lvert\Phi\rvert$	ω'	t_{clu}	$z(LP_1)$	$\Delta_{z(LP_1)}$	$z(LP_2)$	$\Delta_{z(LP_2)}$	ω''	$z(\overline{SP})$	$\Delta_{z(\overline{SP})}$	$\lvert\overline{\Omega}\rvert$	UB_{final}	t_{noUB}	t_{tot}
1_10	**142.03**	5.1	305	744	0.1	142.03	100.0	142.03	100.0	5	142.03	100.0	2	**142.03**	0.1	5.2
2_10	**121.07**	2.5	760	951	0.1	118.15	97.6	121.07	100.0	9	121.07	100.0	2	**121.07**	0.1	2.6
3_10	**149.41**	2.6	1221	774	0.1	149.41	100.0	149.41	100.0	10	149.41	100.0	2	**149.41**	0.1	2.7
4_10	**150.89**	2.4	753	801	0.1	150.89	100.0	150.89	100.0	9	150.89	100.0	2	**150.89**	0.1	2.5
5_10	**132.31**	3.4	718	810	0.1	126.90	95.9	130.77	98.8	14	132.31	100.0	4	**132.31**	0.2	3.6
1_12	**171.02**	3.2	852	2150	0.1	169.14	98.9	170.93	99.9	19	171.02	100.0	2	**171.02**	0.1	3.3
2_12	**111.54**	3.0	657	3743	0.1	110.96	99.5	110.96	99.5	11	111.54	100.0	2	**111.54**	0.1	3.2
3_12	**145.69**	2.8	1573	3666	0.1	145.32	99.8	145.32	99.8	10	145.69	100.0	4	**145.69**	0.2	3.0
4_12	**166.37**	3.4	928	2317	0.1	161.49	97.1	164.01	98.6	47	166.37	100.0	13	**166.37**	0.5	3.9
5_12	**140.42**	3.1	1188	3412	0.1	138.36	98.5	139.75	99.5	11	140.42	100.0	5	**140.42**	0.2	3.3
Avg		3.2	896	1937	0.1		98.7		99.6	15		100.0	4		0.2	3.3

Table B.2.: Detailed results on new instances of Dataset D with 20 customers and route duration.

	LNS-SP		CCG													
Inst.	UB_{init}	t_{heu}	$\lvert\Phi\rvert$	ω'	t_{clu}	$z(LP_1)$	$\Delta_{z(LP_1)}$	$z(LP_2)$	$\Delta_{z(LP_2)}$	ω''	$z(\overline{SP})$	$\Delta_{z(\overline{SP})}$	$\lvert\overline{\Omega}\rvert$	UB_{final}	t_{noUB}	t_{tot}
6_19_0.5_0.4	**1247.04**	92.0	3948	357 551	0.2	1224.64	98.2	1228.09	98.5	89	1247.04	100.0	14	**1247.04**	0.8	92.8
6_19_0.5_0.6	**1228.12**	81.5	3948	357 551	0.2	1224.64	99.7	1228.09	100.0	13	1228.12	100.0	2	**1228.12**	0.4	81.9
6_19_0.5_0.8	**1228.12**	68.4	3948	357 551	0.2	1224.64	99.7	1228.09	100.0	13	1228.12	100.0	2	**1228.12**	0.4	68.8
6_19_0.5_1.0	**1228.12**	49.7	3948	357 551	0.2	1224.64	99.7	1228.09	100.0	13	1228.12	100.0	2	**1228.12**	0.4	50.0
6_19_0.7_0.4	**1992.41**	158.7	33 136	79 766	0.2	1950.73	97.9	1952.14	98.0	529	1992.41	100.0	24	**1992.41**	4.5	163.2
6_19_0.7_0.6	**1982.33**	93.5	33 136	79 766	0.2	1950.73	98.4	1952.14	98.5	267	1982.33	100.0	7	**1982.33**	1.2	94.7
6_19_0.7_0.8	**1974.50**	80.3	33 136	79 766	0.2	1950.73	98.8	1952.14	98.9	167	1974.5	100.0	4	**1974.5**	0.5	80.8
6_19_0.7_1.0	**1971.66**	63.6	33 136	79 766	0.2	1950.73	98.9	1952.14	99.0	118	1971.66	100.0	3	**1971.66**	0.4	64.0
6_19_0.9_0.4	**2159.29**	186.0	113 561	51 606	0.3	2149.89	99.6	2149.89	99.6	27	2159.29	100.0	9	**2159.29**	1.7	187.7
6_19_0.9_0.6	**2156.01**	170.4	113 561	51 606	0.2	2146.46	99.6	2149.03	99.7	23	2156.01	100.0	6	**2156.01**	0.9	171.2
6_19_0.9_0.8	**2154.30**	112.6	113 561	51 606	0.3	2132.27	99.0	2149.89	99.8	23	2154.3	100.0	5	**2154.3**	0.8	113.3
6_19_0.9_1.0	**2149.91**	70.5	113 561	51 606	0.3	2149.04	100.0	2149.04	100.0	11	2149.91	100.0	3	**2149.91**	0.3	70.8
7_20_0.5_0.4	**1395.44**	81.8	1456	88 035	0.1	1375.47	98.6	1375.47	98.6	74	1395.44	100.0	28	**1395.44**	0.9	82.8
7_20_0.5_0.6	**1378.31**	69.0	1456	88 035	0.2	1375.47	99.8	1375.47	99.8	29	1378.31	100.0	6	**1378.31**	0.3	69.3
7_20_0.5_0.8	**1377.14**	47.5	1456	88 035	0.2	1375.47	99.9	1375.47	99.9	37	1377.14	100.0	6	**1377.14**	0.3	47.8
7_20_0.5_1.0	**1375.50**	34.4	1456	88 035	0.2	1375.47	100.0	1375.47	100.0	18	1375.5	100.0	3	**1375.5**	0.2	34.6
7_20_0.7_0.4	**2117.73**	114.3	13 883	13 247	0.1	2015.68	95.2	2030.08	95.9	1124	2117.73	100.0	41	**2117.73**	11.7	126.0
7_20_0.7_0.6	**2030.11**	87.3	13 883	13 247	0.1	2030.08	100.0	2030.08	100.0	9	2030.11	100.0	3	**2030.11**	0.2	87.4
7_20_0.7_0.8	**2030.11**	69.2	13 883	13 247	0.1	2030.08	100.0	2030.08	100.0	9	2030.11	100.0	3	**2030.11**	0.2	69.3
7_20_0.7_1.0	**2030.11**	56.9	13 883	13 247	0.1	2030.08	100.0	2030.08	100.0	9	2030.11	100.0	3	**2030.11**	0.1	57.1
7_20_0.9_0.4	**2339.56**	122.3	24 364	10 175	0.1	2329.8	99.6	2329.8	99.6	10	2339.56	100.0	5	**2339.56**	0.4	122.7
7_20_0.9_0.6	**2334.67**	91.8	24 364	10 175	0.1	2329.8	99.8	2329.8	99.8	11	2334.67	100.0	4	**2334.67**	0.3	92.0
7_20_0.9_0.8	**2334.44**	80.9	24 364	10 175	0.1	2314.73	99.2	2327.25	99.7	24	2334.44	100.0	4	**2334.44**	0.3	81.2
7_20_0.9_1.0	**2329.84**	63.5	24 364	10 175	0.1	2329.8	100.0	2329.8	100.0	7	2329.84	100.0	4	**2329.84**	0.2	63.7
8_20_0.5_0.4	**1391.32**	101.4	6089	672 447	0.4	1349.13	97.0	1356.94	97.5	231	1391.32	100.0	57	**1391.32**	18.1	119.5
8_20_0.5_0.6	**1370.20**	117.7	6089	672 447	0.3	1349.32	98.5	1356.94	99.0	32	1370.2	100.0	17	**1370.2**	1.4	119.1
8_20_0.5_0.8	**1356.96**	100.3	6089	672 447	0.4	1356.94	100.0	1356.94	100.0	6	1356.96	100.0	2	**1356.96**	0.6	100.8
8_20_0.5_1.0	**1356.96**	47.4	6089	672 447	0.4	1356.94	100.0	1356.94	100.0	6	1356.96	100.0	2	**1356.96**	0.5	47.9
8_20_0.7_0.4	**2001.81**	137.2	44 334	245 517	0.3	1911.73	95.5	1911.73	95.5	236	2001.81	100.0	57	**2001.81**	123.1	260.3
8_20_0.7_0.6	**1929.03**	122.1	44 334	245 517	0.3	1911.73	99.1	1911.73	99.1	27	1929.03	100.0	3	**1929.03**	1.3	123.4
8_20_0.7_0.8	**1917.23**	105.1	44 334	245 517	0.3	1911.73	99.7	1911.73	99.7	21	1917.23	100.0	3	**1917.23**	0.6	105.7
8_20_0.7_1.0	**1911.73**	64.8	44 334	245 517	0.3	1911.73	100.0	1911.73	100.0	6	1911.73	100.0	2	**1911.73**	0.5	65.3
8_20_0.9_0.4	**2338.99**	257.6	219 512	132 131	0.5	2256.59	96.5	2273.16	97.2	1088	2338.83	100.0	36	**2338.83**	63.8	321.4
8_20_0.9_0.6	**2274.02**	112.7	219 512	132 131	0.4	2274.02	100.0	2274.02	100.0	9	2274.02	100.0	3	**2274.02**	0.5	113.2
8_20_0.9_0.8	**2274.02**	105.4	219 512	132 131	0.4	2274.02	100.0	2274.02	100.0	9	2274.02	100.0	3	**2274.02**	0.6	105.9
8_20_0.9_1.0	**2274.02**	80.7	219 512	132 131	0.5	2274.02	100.0	2274.02	100.0	9	2274.02	100.0	3	**2274.02**	0.6	81.3
Avg.		97.2	51 143	183 386	0.2		99.1		99.3	120		100.0	11		6.6	103.8

Table B.3.: Detailed results on new instances of Dataset D with 20 customers and without route duration.

	LNS-SP		CCG													
Inst.	UB_{init}	t_{heu}	$\lvert\Phi\rvert$	ω'	t_{clu}	$z(LP_1)$	$\Delta_{z(LP_1)}$	$z(LP_2)$	$\Delta_{z(LP_2)}$	ω''	$z(\overline{SP})$	$\Delta_{z(\overline{SP})}$	$\lvert\overline{\Omega}\rvert$	UB_{final}	t_{noUB}	t_{tot}
6_19_0.5_0.4	**1247.04**	92.9	4310	453 087	0.2	1214.53	97.4	1214.53	97.4	535	1247.04	100.0	25	**1247.04**	14.8	107.6
6_19_0.5_0.6	**1221.59**	88.9	4310	453 087	0.2	1214.53	99.4	1214.53	99.4	11	1221.59	100.0	2	**1221.59**	0.6	89.5
6_19_0.5_0.8	**1216.89**	85.1	4310	453 087	0.2	1214.53	99.8	1214.53	99.8	14	1216.89	100.0	2	**1216.89**	0.5	85.6
6_19_0.5_1.0	**1214.56**	53.6	4310	453 087	0.2	1214.53	100.0	1214.53	100.0	14	1214.56	100.0	2	**1214.56**	0.4	54.0
6_19_0.7_0.4	**1914.25**	194.8	78 895	332 602	0.3	1861.8	97.3	1866.31	97.5	601	1914.25	100.0	37	**1914.25**	307.6	502.4
6_19_0.7_0.6	**1874.14**	175.1	78 895	332 602	0.3	1866.31	99.6	1866.31	99.6	18	1874.14	100.0	4	**1874.14**	1.1	176.2
6_19_0.7_0.8	**1874.14**	151.4	78 895	332 602	0.3	1866.31	99.6	1866.31	99.6	18	1874.14	100.0	4	**1874.14**	1.3	152.7
6_19_0.7_1.0	**1866.33**	81.5	78 895	332 602	0.3	1866.31	100.0	1866.31	100.0	8	1866.33	100.0	2	**1866.33**	0.5	82.0
6_19_0.9_0.4	2151.30	200.2	372 185	262 770	0.7	2058.06	95.7	2066.55	96.1	3832	2087.16	97.0	4	2151.40	*7201.0*	7401.1
6_19_0.9_0.6	**2116.19**	175.0	372 185	262 770	0.7	2058.06	97.3	2066.55	97.7	530	2116.19	100.0	10	**2116.19**	1328.4	1503.5
6_19_0.9_0.8	2096.99	149.4	372 185	262 770	0.8	2058.06	98.2	2066.55	98.6	200	2096.62	100.0	4	**2096.62**	48.4	197.8
6_19_0.9_1.0	**2066.58**	73.7	372 185	262 770	0.8	2066.55	100.0	2066.55	100.0	11	2066.58	100.0	2	**2066.58**	0.9	74.6
7_20_0.5_0.4	**1383.02**	89.1	5453	596 939	0.3	1322.03	95.6	1322.03	95.6	2521	1383.02	100.0	244	**1383.02**	139.4	228.5
7_20_0.5_0.6	**1355.65**	81.4	5453	596 939	0.3	1322.03	97.5	1322.03	97.5	228	1355.65	100.0	42	**1355.65**	3.5	84.9
7_20_0.5_0.8	**1332.60**	88.3	5453	596 939	0.3	1322.03	99.2	1322.03	99.2	29	1332.6	100.0	8	**1332.60**	0.7	89.0
7_20_0.5_1.0	**1322.04**	45.4	5453	596 939	0.3	1322.03	100.0	1322.03	100.0	14	1322.04	100.0	2	**1322.04**	0.5	45.8
7_20_0.7_0.4	**2023.34**	119.6	77 850	159 129	0.2	1969.14	97.3	1969.14	97.3	448	2023.34	100.0	52	**2023.34**	14.2	133.8
7_20_0.7_0.6	**1980.89**	89.8	77 850	159 129	0.2	1969.14	99.4	1969.14	99.4	24	1980.89	100.0	6	**1980.89**	0.7	90.5
7_20_0.7_0.8	**1969.17**	86.4	77 850	159 129	0.2	1969.14	100.0	1969.14	100.0	11	1969.17	100.0	3	**1969.17**	0.3	86.7
7_20_0.7_1.0	**1969.17**	56.9	77 850	159 129	0.2	1969.14	100.0	1969.14	100.0	11	1969.17	100.0	3	**1969.17**	0.3	57.2
7_20_0.9_0.4	**2230.16**	145.6	182 168	116 396	0.4	2223.26	99.7	2223.26	99.7	13	2230.16	100.0	3	**2230.16**	0.6	146.2
7_20_0.9_0.6	**2228.29**	111.9	182 168	116 396	0.3	2223.26	99.8	2223.26	99.8	9	2228.29	100.0	3	**2228.29**	0.5	112.4
7_20_0.9_0.8	**2228.06**	99.7	182 168	116 396	0.3	2223.26	99.8	2223.26	99.8	11	2228.06	100.0	3	**2228.06**	0.5	100.1
7_20_0.9_1.0	**2223.26**	68.8	182 168	116 396	0.3	2223.26	100.0	2223.26	100.0	6	2223.26	100.0	3	**2223.26**	0.4	69.2
8_20_0.5_0.4	1378.12	127.7	8955	1 048 575	0.4	1258.1	91.3	1258.1	91.3	331 833	1285.84	93.3	3	1378.22	*7200.4*	7328.1
8_20_0.5_0.6	**1308.13**	124.1	8955	1 048 575	0.4	1258.1	96.2	1258.1	96.2	43	1308.13	100.0	15	**1308.13**	2817.0	2941.2
8_20_0.5_0.8	**1292.97**	109.0	8955	1 048 575	0.4	1258.1	97.3	1258.1	97.3	11	1292.97	100.0	3	**1292.97**	410.9	519.9
8_20_0.5_1.0	**1258.10**	70.2	8955	1 048 575	0.4	1258.1	100.0	1258.1	100.0	4	1258.1	100.0	1	**1258.10**	1.4	71.5
8_20_0.7_0.4	1946.20	168.6	100 329	1 048 017	0.7	1818.77	93.5	1825.08	93.8	10 649	1825.08	93.8	2	1946.30	*7200.2*	7368.8
8_20_0.7_0.6	**1846.00**	234.8	100 329	1 048 017	0.7	1825.08	98.9	1825.08	98.9	32	1846	100.0	4	**1846.00**	206.6	441.4
8_20_0.7_0.8	**1835.84**	208.6	100 329	1 048 017	0.7	1825.08	99.4	1825.08	99.4	32	1835.84	100.0	2	**1835.84**	63.5	272.1
8_20_0.7_1.0	**1825.09**	102.7	100 329	1 048 017	0.6	1825.08	100.0	1825.08	100.0	15	1825.09	100.0	2	**1825.09**	54.4	157.0
8_20_0.9_0.4	2267.22	281.3	1 076 210	1 035 699	2.4	2144.42	94.6	2172.15	95.8	7591	2172.15	95.8	2	2267.32	*7199.6*	7481.0
8_20_0.9_0.6	**2181.90**	255.4	1 076 210	1 035 699	2.4	2172.15	99.6	2172.15	99.6	7	2181.9	100.0	4	**2181.90**	6.3	261.7
8_20_0.9_0.8	**2181.90**	242.6	1 076 210	1 035 699	2.4	2172.15	99.6	2172.15	99.6	7	2181.9	100.0	4	**2181.90**	6.6	249.2
8_20_0.9_1.0	**2172.15**	122.4	1 076 210	1 035 699	2.4	2172.15	100.0	2172.15	100.0	9	2172.15	100.0	2	**2172.15**	3.3	125.7
Avg.		129.2	211 817	561 468	0.6		98.4		98.5	9982		99.4	14		951.0	1080.2

Table B.4.: Detailed results on new instances of Dataset D with 30 customers and with route duration.

	LNS-SP		CCG													
Inst.	UB_{init}	t_{heu}	$\lvert\Phi\rvert$	ω'	t_{clu}	$z(LP_1)$	$\Delta_{z(LP_1)}$	$z(LP_2)$	$\Delta_{z(LP_2)}$	ω''	$z(\overline{SP})$	$\Delta_{z(\overline{SP})}$	$\lvert\overline{\Omega}\rvert$	UB_{final}	t_{noUB}	t_{tot}
6_29_0.5_0.4	**1835.77**	136.6	45 910	28 967 792	46.8	1830.85	99.7	1831.19	99.8	17	1835.77	100.0	8	**1835.77**	60.6	197.1
6_29_0.5_0.6	**1835.63**	155.2	45 910	28 967 792	46.9	1830.85	99.7	1831.19	99.8	15	1835.63	100.0	11	**1835.63**	61.6	216.8
6_29_0.5_0.8	**1835.02**	141.7	45 910	28 967 792	46.8	1830.77	99.8	1831.19	99.8	24	1835.02	100.0	8	**1835.02**	61.4	203.1
6_29_0.5_1.0	**1831.25**	95.6	45 910	28 967 792	47.0	1831.19	100.0	1831.19	100.0	16	1831.25	100.0	3	**1831.25**	60.3	155.9
6_29_0.7_0.4	**2575.00**	258.4	352 396	2 512 344	14.8	2516.63	97.7	2516.63	97.7	311	2575.00	100.0	29	**2575.00**	71.2	329.6
6_29_0.7_0.6	**2524.04**	199.8	352 396	2 512 344	14.7	2516.63	99.7	2516.63	99.7	10	2524.04	100.0	3	**2524.04**	15.7	215.5
6_29_0.7_0.8	**2521.56**	172.7	352 396	2 512 344	14.7	2516.63	99.8	2516.63	99.8	12	2521.56	100.0	3	**2521.56**	15.4	188.2
6_29_0.7_1.0	**2516.67**	124.7	352 396	2 512 344	14.7	2516.63	100.0	2516.63	100.0	7	2516.67	100.0	3	**2516.67**	15.3	140.0
6_29_0.9_0.4	**3008.48**	313.5	1 736 175	764 444	23.8	2893.42	96.2	2904.11	96.5	11 302	3008.48	100.0	37	**3008.48**	267.9	581.4
6_29_0.9_0.6	**2962.44**	273.5	1 736 175	764 444	23.7	2893.42	97.7	2904.11	98.0	1146	2962.44	100.0	6	**2962.44**	27.1	300.6
6_29_0.9_0.8	**2958.34**	204.7	1 736 175	764 444	23.7	2893.42	97.8	2904.11	98.2	790	2958.34	100.0	6	**2958.34**	26.4	231.1
6_29_0.9_1.0	**2954.36**	130.5	1 736 175	764 444	23.8	2893.42	97.9	2904.11	98.3	677	2954.36	100.0	4	**2954.36**	25.5	155.9
7_30_0.5_0.4	**2106.96**	148.7	8602	2 084 316	12.5	2055.42	97.6	2075.81	98.5	1162	2106.96	100.0	56	**2106.96**	28.4	177.2
7_30_0.5_0.6	**2084.88**	151.9	8602	2 084 316	12.5	2074.9	99.5	2075.81	99.6	43	2084.88	100.0	12	**2084.88**	13.4	165.3
7_30_0.5_0.8	**2077.73**	128.5	8602	2 084 316	12.5	2075.8	99.9	2075.8	99.9	24	2077.73	100.0	6	**2077.73**	13.0	141.4
7_30_0.5_1.0	**2076.22**	87.1	8602	2 084 316	12.5	2076.18	100.0	2076.18	100.0	6	2076.22	100.0	4	**2076.22**	12.9	100.1
7_30_0.7_0.4	**2986.74**	218.1	108 313	108 933	8.2	2871.66	96.1	2919.77	97.8	7433	2986.74	100.0	42	**2986.74**	128.6	346.7
7_30_0.7_0.6	**2945.62**	181.7	108 313	108 933	8.1	2874.05	97.6	2919.77	99.1	528	2945.62	100.0	8	**2945.62**	16.6	198.3
7_30_0.7_0.8	**2944.57**	152.8	108 313	108 933	8.2	2874.05	97.6	2919.77	99.2	446	2944.57	100.0	5	**2944.57**	13.3	166.1
7_30_0.7_1.0	**2944.57**	119.7	108 313	108 933	8.2	2874.05	97.6	2919.77	99.2	446	2944.57	100.0	5	**2944.57**	14.3	134.0
7_30_0.9_0.4	**3257.22**	207.0	164 734	77 960	20.0	3218.71	98.8	3235.59	99.3	34	3257.22	100.0	5	**3257.22**	20.9	227.9
7_30_0.9_0.6	**3255.35**	183.4	164 734	77 960	19.9	3235.59	99.4	3235.59	99.4	22	3255.35	100.0	5	**3255.35**	20.6	204.1
7_30_0.9_0.8	**3254.27**	166.3	164 734	77 960	20.2	3247.74	99.8	3247.74	99.8	12	3254.27	100.0	5	**3254.27**	20.9	187.1
7_30_0.9_1.0	**3247.81**	131.8	164 734	77 960	19.3	3247.74	100.0	3247.74	100.0	12	3247.81	100.0	5	**3247.81**	19.6	151.4
8_30_0.5_0.4	**1994.22**	219.5	75 635	111 325 308	171.3	1953.18	97.9	1953.18	97.9	348	1994.22	100.0	61	**1994.22**	299.6	519.0
8_30_0.5_0.6	**1953.60**	221.0	75 635	111 325 308	172.3	1953.18	100.0	1953.18	100.0	13	1953.60	100.0	3	**1953.60**	222.1	443.0
8_30_0.5_0.8	**1953.18**	234.1	75 635	111 325 308	171.7	1953.18	100.0	1953.18	100.0	9	1953.18	100.0	3	**1953.18**	216.0	450.0
8_30_0.5_1.0	**1953.18**	135.0	75 635	111 325 308	171.7	1953.18	100.0	1953.18	100.0	9	1953.18	100.0	3	**1953.18**	216.0	350.9
8_30_0.7_0.4	**2704.92**	402.0	1 097 114	7 694 121	59.5	2673.06	98.8	2673.06	98.8	172	2704.92	100.0	23	**2704.92**	69.7	471.6
8_30_0.7_0.6	**2673.06**	367.8	1 097 114	7 694 121	59.2	2673.06	100.0	2673.06	100.0	8	2673.06	100.0	3	**2673.06**	62.0	429.8
8_30_0.7_0.8	**2673.06**	334.9	1 097 114	7694 121	58.9	2673.06	100.0	2673.06	100.0	8	2673.06	100.0	3	**2673.06**	61.8	396.7
8_30_0.7_1.0	**2673.06**	191.3	1 097 114	7 694 121	59.1	2673.06	100.0	2673.06	100.0	8	2673.06	100.0	3	**2673.06**	61.9	253.2
8_30_0.9_0.4	3183.11	381.1	4 243 573	3 022 173	56.4	3031.64	95.2	3031.64	95.2	29 396	3067.43	96.4	12	3183.21	*7199.0*	7580.1
8_30_0.9_0.6	**3040.10**	385.1	4 243 573	3 022 173	56.2	3031.64	99.7	3031.64	99.7	14	3040.10	100.0	3	**3040.10**	57.9	443.0
8_30_0.9_0.8	**3031.64**	373.6	4 243 573	3 022 173	55.6	3031.64	100.0	3031.64	100.0	10	3031.64	100.0	3	**3031.64**	56.9	430.5
8_30_0.9_1.0	**3031.64**	189.1	4 243 573	3 022 173	55.6	3031.64	100.0	3031.64	100.0	10	3031.64	100.0	3	**3031.64**	56.8	245.9
Avg.		208.8	870 272	17 395 265	45.9		98.9		99.2	1514		99.9	11		267.0	475.8

Table B.5.: Detailed results on new instances of Dataset D with 30 customers and without route duration.

	LNS-SP		CCG													
Inst.	UB_{init}	t_{heu}	$\lvert\Phi\rvert$	ω'	t_{clu}	$z(LP_1)$	$\Delta_{z(LP_1)}$	$z(LP_2)$	$\Delta_{z(LP_2)}$	ω''	$z(\overline{SP})$	$\Delta_{z(\overline{SP})}$	$\lvert\overline{\Omega}\rvert$	UB_{final}	t_{noUB}	t_{tot}
6_29_0.5_0.4	**1808.47**	136.6	107 422	266 052 583	126.3	1777.59	98.3	1777.59	98.3	891	1808.47	100.0	47	**1808.47**	282.1	418.7
6_29_0.5_0.6	**1797.64**	155.2	107 422	266 052 583	125.5	1777.59	98.9	1777.59	98.9	51	1797.64	100.0	24	**1797.64**	277.5	432.7
6_29_0.5_0.8	**1786.60**	141.7	107 422	266 052 583	125.2	1777.59	99.5	1777.59	99.5	34	1786.6	100.0	7	**1786.6**	253.8	395.5
6_29_0.5_1.0	**1777.63**	95.6	107 422	266 052 583	131.0	1777.59	100.0	1777.59	100.0	12	1777.63	100.0	2	**1777.63**	263.2	358.9
6_29_0.7_0.4	2521.96	258.4	4 372 042	120 585 951	90.2	2485.79	98.7	2494.71	99.1	615	2518.52	100.0	11	**2518.52**	108.1	366.5
6_29_0.7_0.6	**2504.13**	199.8	4 372 042	120 585 951	90.4	2494.71	99.6	2494.71	99.6	22	2504.13	100.0	3	**2504.13**	99.5	299.4
6_29_0.7_0.8	**2498.45**	172.7	4 372 042	120 585 951	90.3	2494.71	99.9	2494.71	99.9	9	2498.45	100.0	3	**2498.45**	98.9	271.7
6_29_0.7_1.0	**2494.75**	124.7	4 372 042	120 585 951	90.3	2494.71	100.0	2494.71	100.0	12	2494.75	100.0	3	**2494.75**	99.6	224.3
6_29_0.9_0.4	2917.67	313.5	43 645 944	50 553 707	209.8	2824.76	96.8	2833.25	97.1	8799	2848.88	97.6	3	2917.77	*7199.4*	7512.9
6_29_0.9_0.6	2862.65	273.5	43 645 944	50 553 707	291.5	2824.76	98.8	2833.25	99.1	193	2858.7	100.0	14	**2858.7**	341.1	614.7
6_29_0.9_0.8	**2835.91**	204.7	43 645 944	50 553 707	295.6	2833.25	99.9	2833.25	99.9	13	2835.91	100.0	4	**2835.91**	299.7	504.4
6_29_0.9_1.0	**2833.28**	130.5	43 645 944	50 553 707	295.1	2833.25	100.0	2833.25	100.0	9	2833.28	100.0	3	**2833.28**	299.8	430.2
7_30_0.5_0.4	1998.04	148.7	79 883	145 846 224	73.8	1922.96	96.2	1922.96	96.2	15 212	1989.62	99.6	265	1998.04	*7253.7*	7402.4
7_30_0.5_0.6	**1958.23**	151.9	79 883	145 846 224	71.1	1922.96	98.2	1922.96	98.2	253	1958.23	100.0	44	**1958.23**	128.1	280.0
7_30_0.5_0.8	**1923.01**	128.5	79 883	145 846 224	71.1	1922.96	100.0	1922.96	100.0	9	1923.01	100.0	3	**1923.01**	80.6	209.0
7_30_0.5_1.0	**1923.01**	87.1	79 883	145 846 224	71.3	1922.96	100.0	1922.96	100.0	9	1923.01	100.0	3	**1923.01**	80.7	167.8
7_30_0.7_0.4	**2887.70**	218.1	2 240 806	7 831 629	22.6	2807.24	97.2	2824.32	97.8	9432	2887.7	100.0	137	**2887.7**	596.1	814.2
7_30_0.7_0.6	**2824.32**	181.7	2 240 806	7 831 629	23.0	2824.32	100.0	2824.32	100.0	15	2824.32	100.0	4	**2824.32**	23.8	205.5
7_30_0.7_0.8	**2824.32**	152.8	2 240 806	7 831 629	22.4	2824.32	100.0	2824.32	100.0	15	2824.32	100.0	4	**2824.32**	23.2	176.0
7_30_0.7_1.0	**2824.32**	119.7	2 240 806	7 831 629	22.7	2824.32	100.0	2824.32	100.0	15	2824.32	100.0	4	**2824.32**	23.4	143.1
7_30_0.9_0.4	3147.24	207.0	5 226 739	3 925 527	48.1	3118.09	99.1	3126.92	99.4	102	3145.37	100.0	6	**3145.37**	50.4	257.4
7_30_0.9_0.6	**3145.29**	183.4	5 226 739	3 925 527	48.5	3119.53	99.2	3126.92	99.4	68	3145.29	100.0	6	**3145.29**	50.1	233.5
7_30_0.9_0.8	**3131.55**	166.3	5 226 739	3 925 527	48.3	3126.92	99.9	3126.92	99.9	17	3131.55	100.0	5	**3131.55**	48.9	215.1
7_30_0.9_1.0	**3126.95**	131.8	5 226 739	3 925 527	48.2	3126.92	100.0	3126.92	100.0	13	3126.95	100.0	5	**3126.95**	48.6	180.4
8_30_0.5_0.4	1956.30	219.5						OOM								219.5
8_30_0.5_0.6	1905.55	221.0						OOM								221.0
8_30_0.5_0.8	1905.55	234.1						OOM								234.1
8_30_0.5_1.0	1874.87	135.0						OOM								135.0
8_30_0.7_0.4	2616.57	402.0	20 799 154	821 136 477	756.6	2540.33	97.1	2549.22	97.4	9200	2549.22	97.4	2	2616.67	*7200.6*	7602.6
8_30_0.7_0.6	**2574.86**	367.8	20 799 154	821 136 477	741.5	2549.22	99.0	2549.22	99.0	129	2574.86	100.0	12	**2574.86**	1353.4	1721.2
8_30_0.7_0.8	**2552.02**	334.9	20 799 154	821 136 477	729.1	2549.22	99.9	2549.22	99.9	16	2552.02	100.0	2	**2552.02**	1155.2	1490.1
8_30_0.7_1.0	**2549.24**	191.3	20 799 154	821 136 477	732.5	2549.22	100.0	2549.22	100.0	11	2549.24	100.0	2	**2549.24**	1146.3	1337.6
8_30_0.9_0.4	3087.07	381.1						OOM								381.1
8_30_0.9_0.6	2967.49	385.1						OOM								385.1
8_30_0.9_0.8	2961.19	373.6						OOM								373.6
8_30_0.9_1.0	2947.71	189.1						OOM								189.1
Avg.		208.8	10 924 570	202 276 014	196.1		99.1		99.2	1613		99.8	22		1031.6	1011.2

Appendix C

Detailed results on E-VRPTWMF

To enable researchers that address the E-VRPTWMF in the future to compare the performance of their algorithms with ours, we provide results on the complete set of E-VRPTWMF instances introduced in Section 4.6.2. The reported results are based on the best of 10 ALNS runs. As objective function, we used the minimization of traveled distance f_d (in km).

Table C.1.: Overview of results for the complete E-VRPTWMF instance set. Results for the best of 10 ALNS runs under the objective of distance minimization are reported. We provide the number of ICCVs (m_{IC}), the number of ECVs (m_E), the distance (f_d) and the average run-time (t) in minutes.

Inst.	m_{IC}	m_E	f_d	t	Inst.	m_{IC}	m_E	f_d	t	Inst.	m_{IC}	m_E	f_d	t
E-UK10_01	2	1	408.13	0.03	E-UK25_01	3	1	672.89	0.59	E-UK100_01	14	7	2820.29	7.02
E-UK10_02	2	1	527.13	0.02	E-UK25_02	4	2	812.97	0.22	E-UK100_02	13	6	2668.59	6.74
E-UK10_03	2	1	493.05	0.02	E-UK25_03	3	1	467.84	0.48	E-UK100_03	13	6	2476.95	7.55
E-UK10_04	2	1	495.66	0.02	E-UK25_04	3	1	611.53	0.34	E-UK100_04	14	7	2351.57	8.72
E-UK10_05	2	1	454.86	0.03	E-UK25_05	4	2	790.33	0.22	E-UK100_05	14	7	2281.46	6.62
E-UK10_06	2	1	595.80	0.02	E-UK25_06	4	2	715.26	0.25	E-UK100_06	14	7	2713.13	6.02
E-UK10_07	2	1	510.30	0.02	E-UK25_07	3	1	795.60	0.24	E-UK100_07	12	6	2412.81	6.80
E-UK10_08	2	1	581.21	0.02	E-UK25_08	3	1	874.11	0.18	E-UK100_08	13	6	2546.89	6.44
E-UK10_09	2	1	451.99	0.02	E-UK25_09	4	2	679.11	0.22	E-UK100_09	13	6	2191.43	6.71
E-UK10_10	2	1	569.74	0.02	E-UK25_10	4	2	895.39	0.30	E-UK100_10	12	6	2473.94	6.75
E-UK10_11	2	1	761.50	0.02	E-UK25_11	4	2	887.59	0.53	E-UK100_11	15	7	2736.90	6.11
E-UK10_12	2	1	472.83	0.07	E-UK25_12	4	2	1034.18	0.19	E-UK100_12	12	6	2353.43	6.84
E-UK10_13	2	1	524.57	0.02	E-UK25_13	4	2	511.47	0.48	E-UK100_13	13	6	2614.28	6.48
E-UK10_14	2	1	397.75	0.01	E-UK25_14	4	2	927.99	0.21	E-UK100_14	14	7	2902.97	6.16
E-UK10_15	2	1	291.37	0.12	E-UK25_15	3	1	879.95	0.17	E-UK100_15	15	7	2989.12	6.00
E-UK10_16	2	1	443.61	0.03	E-UK25_16	4	2	818.78	0.24	E-UK100_16	12	6	2200.01	6.43
E-UK10_17	2	1	387.52	0.06	E-UK25_17	4	2	1277.86	0.17	E-UK100_17	15	7	2907.88	6.28
E-UK10_18	2	1	394.53	0.03	E-UK25_18	3	1	942.12	0.18	E-UK100_18	13	6	2487.61	7.69
E-UK10_19	2	1	414.46	0.06	E-UK25_19	4	2	1068.07	0.19	E-UK100_19	13	6	2262.53	6.67
E-UK10_20	2	1	416.23	0.03	E-UK25_20	4	2	825.93	0.16	E-UK100_20	14	7	2899.56	6.06
E-UK15_01	2	0	709.01	0.03	E-UK50_01	7	3	1357.86	1.05	E-UK150_01	20	10	3046.28	17.67
E-UK15_02	2	1	517.16	0.06	E-UK50_02	7	3	1406.41	1.18	E-UK150_02	20	10	3831.61	12.18
E-UK15_03	3	1	751.86	0.08	E-UK50_03	7	3	1448.90	2.20	E-UK150_03	19	9	3132.22	15.38
E-UK15_04	3	1	757.34	0.08	E-UK50_04	8	4	1764.76	1.25	E-UK150_04	21	10	3703.23	10.88
E-UK15_05	2	1	891.94	0.06	E-UK50_05	6	3	1568.67	1.16	E-UK150_05	20	10	3276.74	12.06
E-UK15_06	3	1	551.15	0.13	E-UK50_06	8	4	1283.86	2.05	E-UK150_06	21	10	3195.89	14.89
E-UK15_07	3	1	604.83	0.10	E-UK50_07	7	3	1193.06	1.74	E-UK150_07	21	10	3856.46	10.89
E-UK15_08	2	1	431.17	0.07	E-UK50_08	7	3	1271.94	1.10	E-UK150_08	20	10	3447.95	11.63
E-UK15_09	3	1	637.57	0.07	E-UK50_09	7	3	1611.98	0.98	E-UK150_09	20	10	3789.48	10.51
E-UK15_10	2	1	568.48	0.08	E-UK50_10	7	3	1564.27	1.95	E-UK150_10	20	10	3616.08	10.24
E-UK15_11	2	0	644.95	0.08	E-UK50_11	7	3	1473.79	1.15	E-UK150_11	20	10	3818.09	11.33
E-UK15_12	3	1	779.42	0.05	E-UK50_12	7	3	1318.96	1.45	E-UK150_12	21	10	4018.53	10.47
E-UK15_13	3	1	593.14	0.08	E-UK50_13	7	3	1324.20	0.91	E-UK150_13	19	9	3686.79	10.73
E-UK15_14	3	1	846.94	0.04	E-UK50_14	7	3	1542.86	1.73	E-UK150_14	20	10	3729.52	10.42
E-UK15_15	2	1	584.35	0.06	E-UK50_15	6	3	1373.09	1.26	E-UK150_15	19	9	3107.91	11.53
E-UK15_16	2	1	556.12	0.06	E-UK50_16	7	3	1322.54	1.63	E-UK150_16	20	10	3772.39	12.16
E-UK15_17	3	1	693.26	0.04	E-UK50_17	7	3	938.50	1.01	E-UK150_17	20	10	3805.71	11.24
E-UK15_18	3	1	825.43	0.05	E-UK50_18	8	4	1601.10	1.05	E-UK150_18	20	10	3739.99	11.78
E-UK15_19	2	1	376.11	0.18	E-UK50_19	7	3	1347.43	1.26	E-UK150_19	20	10	4161.73	10.78
E-UK15_20	3	1	469.97	0.16	E-UK50_20	7	3	1573.49	1.24	E-UK150_20	20	10	4051.19	11.47
E-UK20_01	3	1	785.47	0.13	E-UK75_01	11	5	2227.65	2.73	E-UK200_01	28	14	4616.77	17.11
E-UK20_02	3	1	881.53	0.13	E-UK75_02	11	5	1803.68	3.00	E-UK200_02	24	12	4478.89	17.44
E-UK20_03	3	1	456.73	0.25	E-UK75_03	10	5	1965.21	2.91	E-UK200_03	27	13	4420.04	16.07
E-UK20_04	3	1	800.39	0.18	E-UK75_04	11	5	1689.82	3.72	E-UK200_04	26	13	4175.85	16.45
E-UK20_05	3	1	718.34	0.11	E-UK75_05	10	5	2009.95	2.98	E-UK200_05	27	13	4868.83	15.37
E-UK20_06	3	1	839.43	0.15	E-UK75_06	11	5	2085.90	2.83	E-UK200_06	27	13	4044.02	16.26
E-UK20_07	3	1	522.23	0.46	E-UK75_07	11	5	2201.08	2.49	E-UK200_07	27	13	4334.31	16.40
E-UK20_08	3	1	700.59	0.14	E-UK75_08	10	5	2334.70	3.52	E-UK200_08	27	13	4679.70	24.72
E-UK20_09	3	1	796.50	0.09	E-UK75_09	10	5	2155.98	2.84	E-UK200_09	25	12	4026.65	16.91
E-UK20_10	3	1	696.79	0.35	E-UK75_10	11	5	2232.32	2.97	E-UK200_10	28	14	4903.01	15.23
E-UK20_11	3	1	932.86	0.12	E-UK75_11	10	5	1427.25	2.79	E-UK200_11	27	13	4130.06	16.12
E-UK20_12	3	1	794.58	0.12	E-UK75_12	10	5	1971.57	2.97	E-UK200_12	25	12	4845.70	15.86
E-UK20_13	3	1	764.83	0.11	E-UK75_13	10	5	2263.37	3.08	E-UK200_13	25	12	4847.07	16.45
E-UK20_14	4	2	1045.96	0.11	E-UK75_14	10	5	2146.74	3.18	E-UK200_14	27	13	4406.88	16.12
E-UK20_15	3	1	788.34	0.11	E-UK75_15	10	5	2346.76	2.65	E-UK200_15	25	12	4766.98	16.04
E-UK20_16	3	1	829.14	0.09	E-UK75_16	10	5	2132.85	2.73	E-UK200_16	27	13	4520.69	15.98
E-UK20_17	3	1	873.35	0.14	E-UK75_17	11	5	2057.84	2.60	E-UK200_17	26	13	5006.00	14.74
E-UK20_18	3	1	838.96	0.10	E-UK75_18	10	5	1882.89	3.09	E-UK200_18	27	13	4453.07	20.28
E-UK20_19	3	1	812.21	0.11	E-UK75_19	10	5	1865.06	2.22	E-UK200_19	25	12	3999.15	15.30
E-UK20_20	3	1	873.35	0.11	E-UK75_20	11	5	2028.89	2.21	E-UK200_20	27	13	4825.55	15.44

Formalities

Complete Bibliography

Achterberg, T., T. Koch, and A. Martin (2005). Branching rules revisited. In: *Operations Research Letters* 33 (1), pp. 42–54.

Adulyasak, Y., J.-F. Cordeau, and R. Jans (2014). Optimization-based adaptive large neighborhood search for the production routing problem. In: *Transportation Science* 48 (1), pp. 20–45.

Ahammed, F. and P. Moscato (2011). Evolving L-systems as an intelligent design approach to find classes of difficult-to-solve traveling salesman problem instances. In: *Applications of Evolutionary Computation. EvoApplications 2011*. Ed. by C. Di Chio, S. Cagnoni, C. Cotta, M. Ebner, A. Ekárt, A. I. Esparcia-Alcázar, J. J. Merelo, F. Neri, et al. Vol. 6624. Lecture Notes in Computer Science. Berlin, Heidelberg: Springer, pp. 1–11.

Ajanovic, A. and R. Haas (2016). Dissemination of electric vehicles in urban areas: Major factors for success. In: *Energy* 115 (Part 2), pp. 1451–1458.

Applegate, D. L., R. E. Bixby, V. Chvatal, and W. J. Cook (2007). The Traveling Salesman Problem: A Computational Study (Princeton Series in Applied Mathematics). Princeton, NJ, USA: Princeton University Press.

Arkin, E. M. and R. Hassin (1994). Approximation algorithms for the geometric covering salesman problem. In: *Discrete Applied Mathematics* 55 (3), pp. 197–218.

Artmeier, A., J. Haselmayr, M. Leucker, and M. Sachenbacher (2010). The shortest path problem revisited: Optimal routing for electric vehicles. In: *KI 2010: Advances in Artificial Intelligence*. Vol. 6359. Lecture Notes in Computer Science. Springer, Berlin, pp. 309–316.

Augerat, P., J. Belenguer, E. Benavent, A. Corberán, and D. Naddef (1998). Separating capacity constraints in the CVRP using tabu search. In: *European Journal of Operational Research* 106 (2), pp. 546–557.

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

Baldacci, R., A. Mingozzi, R. Roberti, and R. W. Calvo (2013). An exact algorithm for the two-echelon capacitated vehicle routing problem. In: *Operations Research* 61 (2), pp. 298–314.

Balinski, M. L. and R. E. Quandt (1964). On an integer program for a delivery problem. In: *Operations Research* 12 (2), pp. 300–304.

Barco, J., A. Guerra, L. Muñoz, and N. Quijano (2013). Optimal routing and scheduling of charge for electric vehicles: Case study. In: *CoRR* abs/1310.0145. URL: http://arxiv.org/abs/1310.0145.

Barks, C. (1953). Walt Disney's Comics and Stories. Vol. 149. New York, NY, USA: Dell Comics.

Bektaş, T. and G. Laporte (2011). The pollution-routing problem. In: *Transportation Research Part B: Methodological* 45 (8), pp. 1232–1250.

Bent, R. and P. V. Hentenryck (2006). A two-stage hybrid algorithm for pickup and delivery vehicle routing problems with time windows. In: *Computers & Operations Research* 33 (4), pp. 875–893.

Berg, P., S. Isaacs, and K. Blodgett (2016). *Airborne fulfillment center utilizing unmanned aerial vehicles for item delivery*. US Patent 9,305,280. April 5, 2016.

Bernon, M., J. Cullen, and J. Gorst (2016). Online retail returns management: Integration within an omni-channel distribution context. In: *International Journal of Physical Distribution & Logistics Management* 46 (6/7), pp. 584–605.

Bogdanski, R. (2017). *Innovationen auf der letzten Meile - Bewertung der Chancen für die nachhaltige Stadtlogistik von morgen - Nachhaltigkeitsstudie 2017*. Ed. by Bundesverband Paket & Express Logistik. Accessed 7/24/2017. URL: www.biek.de/index.php/studien.html?file=tl_files/biek/Nachhaltigkeitsstudie%202017/BIEK_Nachhaltigkeitsstudie_2017.pdf.

Bolduc, M.-C., J. Renaud, and F. Boctor (2007). A heuristic for the routing and carrier selection problem. In: *European Journal of Operational Research* 183 (2), pp. 926–932.

Bolduc, M.-C., J. Renaud, F. Boctor, and G. Laporte (2008). A perturbation metaheuristic for the vehicle routing problem with private fleet and common carriers. In: *Journal of the Operational Research Society* 59 (6), pp. 776–787.

Bousonville, T., A. Hartmann, T. Melo, and H. Kopfer (2011). Vehicle routing and refueling: The impact of price variations on tour length. In: *Logistikmanagement - Herausforderungen, Chancen, und Lösungen (Proceedings of LM 11)*. Vol. 2, pp. 83–101.

Bruglieri, M., F. Pezzella, O. Pisacane, and S. Suraci (2015). A variable neighborhood search branching for the electric vehicle routing problem with time windows. In: *Electronic Notes in Discrete Mathematics* 47, pp. 221–228.

Christofides, N., A. Mingozzi, and P. Toth (1979). The vehicle routing problem. In: *Combinatorial Optimization*. Ed. by N. Christofides, A. Mingozzi, P. Toth, and C. Sandi. Chichester, UK: Wiley, pp. 315–338.

Chu, C.-W. (2005). A heuristic algorithm for the truckload and less-than-truckload problem. In: *European Journal of Operational Research* 165 (3), pp. 657–667.

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

Colson, B., P. Marcotte, and G. Savard (2007). An overview of bilevel optimization. In: *Annals of Operations Research* 153 (1), pp. 235–256.

Conrad R., G. and A. Figliozzi M. (2011). The recharging vehicle routing problem. In: *Proceedings of the 2011 Industrial Engineering Research Conference*. Ed. by T. Doolen and E. Van Aken. Reno, USA.

Contardo, C. and R. Martinelli (2014). A new exact algorithm for the multi-depot vehicle routing problem under capacity and route length constraints. In: *Discrete Optimization* 12, pp. 129–146.

Cordeau, J.-F., G. Laporte, and A. Mercier (2001). A unified tabu search heuristic for vehicle routing problems with time windows. In: *Journal of the Operational Research Society* 52 (8), pp. 928–936.

Cordeau, J.-F. (2006). A branch-and-cut algorithm for the dial-a-ride problem. In: *Operations Research* 54 (3), pp. 573–586.

Côté, J.-F. and J.-Y. Potvin (2009). A tabu search heuristic for the vehicle routing problem with private fleet and common carrier. In: *European Journal of Operational Research* 198 (2), pp. 464–469.

Council of Supply Chain Management Professionals (2013). *Supply chain management terms and glossary*. Accessed July 20, 2017. URL: http://cscmp.org/CSCMP/Educate/SCM_Definitions_and_Glossary_of_Terms/CSCMP/Educate/SCM_Definitions_and_Glossary_of_Terms.aspx.

Crainic, T. G., M. Gendreau, P. Soriano, and M. Toulouse (1993). A tabu search procedure for multicommodity location/allocation with balancing requirements. In: *Annals of Operations Research* 41 (4), pp. 359–383.

Crevier, B., J.-F. Cordeau, and G. Laporte (2007). The multi-depot vehicle routing problem with inter-depot routes. In: *European Journal of Operational Research* 176 (2), pp. 756–773.

Dabia, S., S. Ropke, T. van Woensel, and T. de Kok (2013). Branch and price for the time-dependent vehicle routing problem with time windows. In: *Transportation Science* 47 (3), pp. 380–396.

Davis, B. A. and M. A. Figliozzi (2013). A methodology to evaluate the competitiveness of electric delivery trucks. In: *Transportation Research Part E: Logistics and Transportation Review* 49 (1), pp. 8–23.

Dayarian, I., G. Crainic T., M. Gendreau, and W. Rei (2013). *An Adaptive Large Neighborhood Search Heuristic for a Multi-Period Vehicle Routing Problem*. Tech. rep. 2013-67. CIRRELT, Canada.

Dekker, R., J. Bloemhof, and I. Mallidis (2012). Operations research for green logistics: An overview of aspects, issues, contributions and challenges. In: *European Journal of Operational Research* 219 (3), pp. 671–679.

Dembski, W. A. and R. J. Marks II (2009). Conservation of information in search: Measuring the cost of success. In: *IEEE Transactions on Systems, Man, and Cybernetics - Part A: Systems and Humans* 39 (5), pp. 1051–1061.

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

Demir, E., T. Bektaş, and G. Laporte (2011). A comparative analysis of several vehicle emission models for road freight transportation. In: *Transportation Research Part D: Transport and Environment* 16 (5), pp. 347–357.

– (2014a). A review of recent research on green road freight transportation. In: *European Journal of Operational Research* 237 (3), pp. 775–793.

– (2014b). The bi-objective pollution-routing problem. In: *European Journal of Operational Research* 232 (3), pp. 464–478.

Desaulniers, G., F. Errico, S. Irnich, and M. Schneider (2016). Exact algorithms for electric vehicle-routing problems with time windows. In: *Operations Research* 64 (6), pp. 1388–1405.

Desrosiers, J. and M. E. Lübbecke (2005). A primer in column generation. In: *Column Generation*. Ed. by G. Desaulniers, J. Desrosiers, and M. M. Solomon. Boston, MA, USA: Springer, pp. 1–32.

Deutsche Post DHL Group (2016). *Logistics trend radar*. DHL Customer Solutions & Innovation. Accessed December 16, 2017. Troisdorf, Germany. URL: http://www.dhl.com/en/about_us/logistics_insights/dhl_trend_research/trendradar.html#.Wf9MZLpFxZQ.

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

Dumas, Y., J. Desrosiers, and F. Soumis (1991). The pickup and delivery problem with time windows. In: *European Journal of Operational Research* 54 (1), pp. 7–22.

Erdoğan, S. and E. Miller-Hooks (2012). A green vehicle routing problem. In: *Transportation Research Part E: Logistics and Transportation Review* 48 (1), pp. 100–114.

European Commission (2014). *Communication from the Commission to the European Parliament, the Council, the European Economic and Social Committee and the Committee of the regions. A policy framework for climate and energy in the period from 2020 to 2030, COM(2014)15, SWD(2014) 16 final*. Accessed July 25, 2017. URL: http://eur-lex.europa.eu/legal-content/EN/ALL/?uri=CELEX:52014DC0015.

– (2016). EU Transport in figures: Statistical pocketbook. Luxembourg: Publications Office of the European Union.

European Environment Agency (2017a). *Data viewer on greenhouse gas emissions and removals, sent by countries to UNFCCC and the EU Greenhouse Gas Monitoring Mechanism (EU Member States)*. Accessed July 25, 2017. URL: http://www.eea.europa.eu/data-and-maps/data/data-viewers/greenhouse-gases-viewer.

– (2017b). *Greenhouse gas emissions from transport*. Accessed July 25, 2017. URL: https://www.eea.europa.eu/data-and-maps/indicators/transport-emissions-of-greenhouse-gases/transport-emissions-of-greenhouse-gases-10.

Feillet, D., M. Gendreau, and L.-M. Rousseau (2007). New refinements for the solution of vehicle routing problems with branch and price. In: *Information Systems and Operational Research* 45 (4), pp. 239–256.

Feillet, D. (2010). A tutorial on column generation and branch-and-price for vehicle routing problems. In: *4OR - A Quarterly Journal of Operations Research* 8 (4), pp. 407–424.

Feillet, D., T. Garaix, F. Lehuédé, O. Péton, and D. Quadri (2014). A new consistent vehicle routing problem for the transportation of people with disabilities. In: *Networks* 63 (3), pp. 211–224.

Felipe, Á., M. T. Ortuño, G. Righini, and G. Tirado (2014). A heuristic approach for the green vehicle routing problem with multiple technologies and partial recharges. In: *Transportation Research Part E: Logistics and Transportation Review* 71, pp. 111–128.

Feng, W. and M. A. Figliozzi (2013). An economic and technological analysis of the key factors affecting the competitiveness of electric commercial vehicles: A case study from the USA market. In: *Transportation Research Part C: Emerging Technologies* 26, pp. 135–145.

Feo, T. A. and M. G. C. Resende (1989). A probabilistic heuristic for a computationally difficult set covering problem. In: *Operations Research Letters* 8 (2), pp. 67–71.

Ferrandez, S. M., T. Harbison, T. Weber, R. Sturges, and R. Rich (2016). Optimization of a truck-drone in tandem delivery network using k-means and genetic algorithm. In: *Journal of Industrial Engineering and Management* 9 (2).

Fischetti, M., I. Ljubić, M. Monaci, and M. Sinnl (2017). A new general-purpose algorithm for mixed-integer bilevel linear programs. In: *Operations Research* 65 (6), pp. 1615–1637.

Froger, A., J. E. Mendoza, O. Jabali, and G. Laporte (2017). *A Matheuristic for the Electric Vehicle Routing Problem with Capacitated Charging Stations*. Tech. rep. 2017-31. CIRRELT, Canada.

Furtadoa, M. G., P. Munaria, and R. Morabitoa (2017). Pickup and delivery problem with time windows: a new compact two-index formulation. In: *Operations Research Letters* 45 (4), pp. 334–341.

Gaskell, T. J. (1967). Bases for vehicle fleet scheduling. In: *Journal of the Operational Research Society* 18 (3), pp. 281–295.

Gendreau, M., A. Hertz, and G. Laporte (1994). A tabu search heuristic for the vehicle routing problem. In: *Management Science* 40 (10), pp. 1276–1290.

Gendreau, M. and J.-Y. Potvin (2010). Tabu search. In: *Handbook of Metaheuristics*. Ed. by M. Gendreau and J.-Y. Potvin. Vol. 146. International Series in Operations Research & Management Science. Springer, pp. 41–59.

Glover, F. (1986). Future paths for integer programming and links to artificial intelligence. In: *Computers & Operations Research* 13 (5), pp. 533–549.

Goeke, D. and M. Schneider (2015). Routing a mixed fleet of electric and conventional vehicles. In: *European Journal of Operational Research* 245 (1), pp. 81–99.

Golden, B., E. Wasil, J. Kelly, and I.-M. Chao (1998). Fleet management and logisitics. In: *The impact of metaheuristics on solving the vehicle routing problem: Algorithms, problem sets, and computational results*. Ed. by T. G. Grainic and G. Laporte. Springer, pp. 33–56.

Grandinetti, L., F. Guerriero, F. Pezzella, and O. Pisacane (2016). A pick-up and delivery problem with time windows by electric vehicles. In: *International Journal of Productivity and Quality Management* 18 (2-3), pp. 403–423.

Groër, C., B. Golden, and E. Wasil (2009). The consistent vehicle routing problem. In: *Manufacturing & Service Operations Management* 11 (4), pp. 630–643.

– (2011). A parallel algorithm for the vehicle routing problem. In: *INFORMS Journal on Computing* 23 (2), pp. 315–330.

Gschwind, T., S. Stefan Irnich, A.-K. Rothenbächer, and C. Tilk (2018). Bidirectional labeling in column-generation algorithms for pickup-and-delivery problems. In: *European Journal of Operational Research* 266 (2), pp. 521–530.

Guzzella, L. and A. Amstutz (2005). *The QSS toolbox manual*. Accessed May 10, 2013. URL: http://www.idsc.ethz.ch/Downloads/DownloadFiles/qss.

Han, S., S. Han, and H. Aki (2014). A practical battery wear model for electric vehicle charging applications. In: *Applied Energy* 113, pp. 1100–1108.

Hasle, G. and O. Kloster (2007). Industrial vehicle routing. In: *Geometric Modelling, Numerical Simulation, and Optimization: Applied Mathematics at SINTEF*. Ed. by G. Hasle, K.-A. Lie, and E. Quak. Berlin, Heidelberg: Springer, pp. 397–435.

Hawkins, T. R., O. M. Gausen, and A. H. Strømman (2012). Environmental impacts of hybrid and electric vehicles: A review. In: *The International Journal of Life Cycle Assessment* 17 (8), pp. 997–1014.

He, F., D. Wu, Y. Yin, and Y. Guan (2013). Optimal deployment of public charging stations for plug-in hybrid electric vehicles. In: *Transportation Research Part B: Methodological* 47, pp. 87–101.

Heineken International (2014). *Case Studies: Europe's largest electric truck will drive down emissions*. Sustainability Report 2013. Accessed July 17, 2014. URL: http://sustainabilityreport.heineken.com/Reducing-CO2-emissions/Case-studies/Europes-largest-electric-truck-will-drive-down-emissions/index.htm.

Hemmelmayr V., C., F. Doerner K., F. Hartl R., and S. Rath (2013). A heuristic solution method for node routing based solid waste collection problems. In: *Journal of Heuristics* 19 (2), pp. 129–156.

Hemmelmayr, V. C., J.-F. Cordeau, and G. Crainic T. (2012). An adaptive large neighborhood search heuristic for two-echelon vehicle routing problems arising in city logistics. In: *Computers & Operations Research* 39 (12), pp. 3215–3228.

Hiermann, G., J. Puchinger, and R. F. Hartl (2014). *The electric fleet size and mix vehicle routing problem with time windows and recharging stations*. Tech. rep. Accessed July 17, 2014. URL: http://prolog.univie.ac.at/research/publications/downloads/Hie_2014_638.pdf.

Hiermann, G., J. Puchinger, S. Ropke, and R. F. Hartl (2016). The electric fleet size and mix vehicle routing problem with time windows and recharging stations. In: *European Journal of Operational Research* 252 (3), pp. 995–1018.

Ho, Y. and D. Pepyne (2002). Simple explanation of the no-free-lunch theorem and its implications. In: *Journal of Optimization Theory and Applications* 115 (3), pp. 549–570.

Hoke, A., A. Brissette, D. Maksimovic, A. Pratt, and K. Smith (2011). Electric vehicle charge optimization including effects of lithium-ion battery degradation. In: *Vehicle Power and Propulsion Conference (VPPC)*. Chicago, USA: IEEE, pp. 1–8.

Ibaraki, T., S. Imahori, K. Nonobe, K. Sobue, T. Uno, and M. Yagiura (2008). An iterated local search algorithm for the vehicle routing problem with convex time penalty functions. In: *Discrete Applied Mathematics* 156 (11), pp. 2050–2069.

International Energy Agency (2012). *EV city casebook: A look at the global electric vehicle movement*. Accessed July 17, 2014. URL: http://www.iea.org/publications/freepublications/publication/EVCityCasebook.pdf.

– (2013). *EV outlook: Understanding the electric vehicle landscape to 2020*. Accessed July 17, 2014. URL: http://www.iea.org/publications/freepublications/publication/GlobalEVOutlook_2013.pdf.

Irnich, S. and G. Desaulniers (2005). Shortest path problems with resource constraints. In: *Column Generation*. Ed. by G. Desaulniers, J. Desrosiers, and M. Solomon. New York, NY: Springer. Chap. 2, pp. 33–65.

Irnich, S., G. Desaulniers, J. Desrosiers, and A. Hadjar (2010). Path-reduced costs for eliminating arcs in routing and scheduling. In: *INFORMS Journal on Computing* 22 (2), pp. 297–313.

Jabali, O., T. van Woensel, and A. de Kok (2012). Analysis of travel times and CO2 emissions in time-dependent vehicle routing. In: *Production and Operations Management* 21 (6), pp. 1060–1074.

Jafari, E. and S. D. Boyles (2017). Multicriteria stochastic shortest path problem for electric vehicles. In: *Networks and Spatial Economics* 17 (3), pp. 1043–1070.

Jepsen, M., B. Petersen, S. Spoorendonk, and D. Pisinger (2008). Subset-row inequalities applied to the vehicle-routing problem with time windows. In: *Operations Research* 56 (2), pp. 497–511.

Jochem, P., C. Doll, and W. Fichtner (2016). External costs of electric vehicles. In: *Transportation Research Part D: Transport and Environment* 42, pp. 60–76.

Juan, A., C. Méndez, J. Faulin, J. de Armas, and S. Grasman (2016). Electric vehicles in logistics and transportation: A survey on emerging environmental, strategic, and operational challenges. In: *Energies* 9 (2), pp. 86–107.

Keskin, M. and B. Çatay (2016). Partial recharge strategies for the electric vehicle routing problem with time windows. In: *Transportation Research Part C: Emerging Technologies* 65, pp. 111–127.

Kindervater, G. and M. Savelsbergh (1997). Vehicle routing: Handling edge exchanges. In: *Local Search in Combinatorial Optimization*. Ed. by E. Aarts and J. Lenstra. John Wiley & Sons, pp. 337–360.

Kirkpatrick, S., C. D. Gelatt, and M. P. Vecchi (1983). Optimization by simulated annealing. In: *Science* 220 (4598), pp. 671–680.

Kleindorfer, P. R., A. Neboian, A. Roset, and S. Spinler (2012). Fleet renewal with electric vehicles at La Poste. In: *Interfaces* 42 (5), pp. 465–477.

Kopfer, H. W., J. Schönberger, and H. Kopfer (2014). Reducing greenhouse gas emissions of a heterogeneous vehicle fleet. In: *Flexible Services and Manufacturing Journal* 26 (1-2), pp. 221–248.

Kovacs, A. A., B. L. Golden, R. F. Hartl, and S. N. Parragh (2015). The generalized consistent vehicle routing problem. In: *Transportation Science* 49 (4), pp. 796–816.

Kovacs, A. A., R. F. Hartl, S. N. Parragh, and B. L. Golden (2014). Vehicle routing problems in which consistency considerations are important: A survey. In: *Networks* 64 (3), pp. 192–213.

Kovacs, A. A., S. N. Parragh, and R. F. Hartl (2014). A template-based adaptive large neighborhood search for the consistent vehicle routing problem. In: *Networks* 63 (1), pp. 60–81.

– (2015). The multi-objective generalized consistent vehicle routing problem. In: *European Journal of Operational Research* 247 (2), pp. 441–458.

Krajewska, M. A. and H. Kopfer (2009). Transportation planning in freight forwarding companies: Tabu search algorithm for the integrated operational transportation planning problem. In: *European Journal of Operational Research* 197 (2), pp. 741–751.

Lanz, M. and R. Tuokko (2017). Concepts, methods and tools for individualized production. In: *Production Engineering* 11 (2), pp. 205–212.

Laporte, G., Y. Nobert, and M. Desrochers (1985). Optimal routing under capacity and distance restrictions. In: *Operations Research* 33 (5), pp. 1050–1073.

Lebeau, P., C. Macharis, and J. van Mierlo (2016). Exploring the choice of battery electric vehicles in city logistics: A conjoint-based choice analysis. In: *Transportation Research Part E: Logistics and Transportation Review* 91, pp. 245–258.

Lebeau, P., C. Macharis, J. van Mierlo, and K. Lebeau (2015). Electrifying light commercial vehicles for city logistics? A total cost of ownership analysis. In: *European Journal of Transport and Infrastructure Research* 15 (4), pp. 551–569.

Li, F., B. Golden, and E. Wasil (2005). Very large-scale vehicle routing: New test problems, algorithms, and results. In: *European Journal of Operational Research* 32 (5), pp. 1165–1179.

Li, H. and A. Lim (2001). A metaheuristic for the pickup and delivery problem with time windows. In: *Proceedings 13th IEEE International Conference on Tools with Artificial Intelligence (ICTAI'01)*. Ed. by D. Moldovan. IEEE Press, pp. 160–167.

Lian, K., A. B. Milburn, and R. L. Rardin (2016). An improved multi-directional local search algorithm for the multi-objective consistent vehicle routing problem. In: *IIE Transactions* 48 (10), pp. 975–992.

Lin, S. (1965). Computer solutions of the traveling salesman problem. In: *Bell System Technical Journal* 44 (10), pp. 2245–2269.

Liu, R., Z. Jiang, X. Liu, and F. Chen (2010). Task selection and routing problems in collaborative truckload transportation. In: *Transportation Research Part E: Logistics and Transportation Review* 46 (6), pp. 1071–1085.

Lodi, A. and G. Zarpellon (2017). On learning and branching: A survey. In: *TOP* 25 (2), pp. 207–236.

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

Lysgaard, J. (2003). *CVRPSEP: A package of separation routines for the Capacitated Vehicle Routing Problem*. Working Paper 03-04. Aarhus, Denmark: Department of Management Science and Logistics, Aarhus School of Business.

Mak, H.-Y., Y. Rong, and Z.-J. M. Shen (2013). Infrastructure planning for electric vehicles with battery swapping. In: *Management Science* 59 (7), pp. 1557–1575.

Marra, F., G. Y. Yang, C. Traholt, E. Larsen, C. N. Rasmussen, and Y. Shi (2012). Demand profile study of battery electric vehicle under different charging options. In: *Power and Energy Society General Meeting, 2012*. San Diego, USA: IEEE, pp. 1–7.

Masson, R., F. Lehuédé, and O. Péton (2013). An adaptive large neighborhood search for the pickup and delivery problem with transfers. In: *Transportation Science* 47 (3), pp. 344–355.

Montoya, A., C. Guéret, J. E. Mendoza, and J. G. Villegas (2014). *A modified multi-space sampling heuristic for the green vehicle routing problem*. Tech. rep. LARIS-EA 7315. Laboratoire Angevin de Recherche en Ingénierie des Sytèmes, Université d'Angers, France.

– (2017). The electric vehicle routing problem with nonlinear charging function. In: *Transportation Research Part B: Methodological* 103, pp. 87–110.

Murray, C. C. and A. G. Chu (2015). The flying sidekick traveling salesman problem: Optimization of drone-assisted parcel delivery. In: *Transportation Research Part C: Emerging Technologies* 54, pp. 86–109.

Nagata, Y., O. Bräysy, and W. Dullaert (2010). A penalty-based edge assembly memetic algorithm for the vehicle routing problem with time windows. In: *Computers & Operations Research* 37 (4), pp. 724–737.

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

National Renewable Energy Laboratory (2014). *Project startup: evaluating the performance of Frito Lay's electric delivery trucks*. Accessed July 23, 2014. URL: http://www.nrel.gov/docs/fy14osti/61455.pdf.

Nie, Y. and M. Ghamami (2013). A corridor-centric approach to planning electric vehicle charging infrastructure. In: *Transportation Research Part B: Methodological* 57, pp. 172–190.

Nie, Y. and Q. Li (2013). An eco-routing model considering microscopic vehicle operating conditions. In: *Transportation Research Part B: Methodological* 55, pp. 154–170.

Nitta, N., F. Wu, J. T. Lee, and G. Yushin (2015). Li-ion battery materials: Present and future. In: *Materials Today* 18 (5), pp. 252–264.

Nordelöf, A., M. Messagie, A.-M. Tillman, M. Ljunggren Söderman, and J. van Mierlo (2014). Environmental impacts of hybrid, plug-in hybrid, and battery electric vehicles—What can we learn from life cycle assessment? In: *The International Journal of Life Cycle Assessment* 19 (11), pp. 1866–1890.

Pankratz, G. (2005). A grouping genetic algorithm for the pickup and delivery problem with time windows. In: *OR Spectrum* 27 (1), pp. 21–41.

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

Pecin, D., C. Contardo, G. Desaulniers, and E. Uchoa (2017). New enhancements for the exact solution of the vehicle routing problem with time windows. In: *INFORMS Journal on Computing* 29 (3), pp. 489–502.

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

Pelletier, S., O. Jabali, and G. Laporte (2016). 50th anniversary invited article–Goods distribution with electric vehicles: Review and research perspectives. In: *Transportation Science* 50 (1), pp. 3–22.

Potvin, J.-Y. and M.-A. Naud (2011). Tabu search with ejection chains for the vehicle routing problem with private fleet and common carrier. In: *Journal of the Operational Research Society* 62 (2), pp. 326–336.

Potvin, J.-Y. and J.-M. Rousseau (1995). An exchange heuristic for routeing problems with time windows. In: *Journal of the Operational Research Society* 46 (12), pp. 1433–1446.

Preis, H., S. Frank, and K. Nachtigall (2014). Energy-optimized routing of electric vehicles in urban delivery systems. In: *Operations Research Proceedings 2012 – Selected Papers of the International Annual Conference of the German Operations Research Society (GOR)*. Ed. by S. Helber, M. Breitner, D. Rösch, C. Schön, J.-M. Graf von der Schulenburg, P. Sibbertsen, M. Steinbach, S. Weber, et al. Cham: Springer International Publishing, pp. 583–588.

Quak, H., N. Nesterova, and T. van Rooijen (2016). Possibilities and barriers for using electric-powered vehicles in city logistics practice. In: *Tenth International Conference on City Logistics 17-19 June 2015, Tenerife, Spain*. Ed. by E. Taniguchi and G. Russell. Vol. 12. Transportation Research Procedia. Elsevier, pp. 157–169.

Reichel, J. (2016). Lastenrad-Projekt: UPS gewinnt Nachhaltigkeitspreis. In: *Logistra - News*. Accessed 7/24/2017. URL: http://www.logistra.de/news-nachrichten/nfz-fuhrpark-lagerlogistik-intralogistik/7687/maerkte-amp-trends/lastenrad-projekt-ups-gewinnt-nachhaltigkeitsp.

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

Roberti, R. and A. Mingozzi (2014). Dynamic ng-path relaxation for the delivery man problem. In: *Transportation Science* 48 (3), pp. 413–424.

Rochat, Y. and É. D. Taillard (1995). Probabilistic diversification and intensification in local search for vehicle routing. In: *Journal of Heuristics* 1 (1), pp. 147–167.

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

Ropke, S. and D. Pisinger (2006a). A unified heuristic for a large class of vehicle routing problems with backhauls. In: *European Journal of Operational Research* 171 (3), pp. 750–775.

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

Sadykov, R., E. Uchoa, and A. Pessoa (2017). *Enhanced Branch-Cut-and-Price Algorithm for Heterogeneous Fleet Vehicle Routing Problems*. Tech. rep. L-2017-7. Niterói, Brazil: Cadernos do LOGIS-UFF.

Sarstedt, M. and E. Mooi (2014). A concise guide to market research: The process, data, and methods using IBM SPSS statistics. In: Berlin Heidelberg, Germany: Springer. Chap. Cluster Analysis, pp. 273–324.

Savelsbergh, M. (1992). The vehicle routing problem with time windows: Minimizing route duration. In: *ORSA Journal on Computing* 4 (2), pp. 146–154.

Savuran, H. and M. Karakaya (2016). Efficient route planning for an unmanned air vehicle deployed on a moving carrier. In: *Soft Computing* 20 (7), pp. 2905–2920.

Sbihi, A. and R. W. Eglese (2010). Combinatorial optimization and green logistics. In: *Annals of Operations Research* 175 (1), pp. 159–175.

Schneider, M., B. Sand, and A. Stenger (2013). A note on the time travel approach for handling time windows in vehicle routing problems. In: *Computers & Operations Research* 40 (10), pp. 2564–2568.

Schneider, M., F. Schwahn, and D. Vigo (2017). Designing granular solution methods for routing problems with time windows. In: *European Journal of Operational Research* 263 (2), pp. 493–509.

Schneider, M., A. Stenger, and D. Goeke (2014). The electric vehicle-routing problem with time windows and recharging stations. In: *Transportation Science* 48 (4), pp. 500–520.

Schneider, M., A. Stenger, and J. Hof (2015). An adaptive VNS algorithm for vehicle routing problems with intermediate stops. In: *OR Spectrum* 37 (2), pp. 353–387.

Schrimpf, G., J. Schneider, H. Stamm-Wilbrandt, and G. Dueck (2000). Record breaking optimization results using the ruin and recreate principle. In: *Journal of Computational Physics* 159 (2), pp. 139–171.

Shaw, P. (1997). *A new local search algorithm providing high quality solutions to vehicle routing problems*. Tech. rep. Glasgow, Scotland: Department of Computer Science, University of Strathclyde.

– (1998). Using constraint programming and local search methods to solve vehicle routing problems. In: *Principles and Practice of Constraint Programming – CP98*. Ed. by M. Maher and J.-F. Puget. Vol. 1520. Lecture Notes in Computer Science. London, United Kingdom: Springer, pp. 417–431.

Shuttleworth, R., B. L. Golden, S. Smith, and E. Wasil (2008). Advances in meter reading: Heuristic solution of the close enough traveling salesman problem over a street network. In: *The Vehicle Routing Problem: Latest Advances and New Challenges*. Ed. by B. Golden, S. Raghavan, and E. Wasil. Boston, MA, USA: Springer, pp. 487–501.

Solomon, M. (1987). Algorithms for the vehicle routing and scheduling problems with time window constraints. In: *Operations Research* 35 (2), pp. 254–265.

Spiegel, J., M. McKenna, G. Lakshman, and P. Nordstrom (2013). *Method and system for anticipatory package shipping*. US Patent 8,615,473. December 24, 2013.

Stenger, A., M. Schneider, and D. Goeke (2013). The prize-collecting vehicle routing problem with single and multiple depots and non-linear cost. In: *EURO Journal on Transportation and Logistics* 2 (1-2), pp. 57–87.

Stenger, A., D. Vigo, S. Enz, and M. Schwind (2013). An adaptive variable neighborhood search algorithm for a vehicle routing problem arising in small package shipping. In: *Transportation Science* 47 (1), pp. 64–80.

Stevens, G. C. and M. Johnson (2016). Integrating the supply chain ... 25 years on. In: *International Journal of Physical Distribution & Logistics Management* 46 (1), pp. 19–42.

Subramanian, A., E. Uchoa, and L. S. Ochi (2013). A hybrid algorithm for a class of vehicle routing problems. In: *Computers & Operations Research* 40 (10), pp. 2519–2531.

Subramanyam, A. and C. E. Gounaris (2016). A branch-and-cut framework for the consistent traveling salesman problem. In: *European Journal of Operational Research* 248 (2), pp. 384–395.

– (2017). A decomposition algorithm for the consistent traveling salesman problem with vehicle idling. In: *Transportation Science*. DOI: 10.1287/trsc.2017.0741.

Sungur, I., Y. Ren, F. Ordóñez, M. Dessouky, and H. Zhong (2010). A model and algorithm for the courier delivery problem with uncertainty. In: *Transportation Science* 44 (2), pp. 193–205.

Suzuki, Y. (2012). A decision support system of vehicle routing and refueling for motor carriers with time-sensitive demands. In: *Decision Support Systems* 54 (1), pp. 758–767.

Tang, L. and X. Wang (2006). Iterated local search algorithm based on very large-scale neighborhood for prize-collecting vehicle routing problem. In: *The International Journal of Advanced Manufacturing Technology* 29 (11), pp. 1246–1258.

Tarantilis, C., F. Stavropoulou, and P. Repoussis (2012). A template-based tabu search algorithm for the consistent vehicle routing problem. In: *Expert Systems with Applications* 39 (4), pp. 4233–4239.

Tesla Motors, Inc. (2014). *Supercharger*. Accessed July 7, 2014. URL: http://www.teslamotors.com/supercharger.

Tilk, C., A.-K. Rothenbächer, T. Gschwind, and S. Irnich (2017). Asymmetry matters: Dynamic half-way points in bidirectional labeling for solving shortest path problems with resource constraints faster. In: *European Journal of Operational Research* 261 (2), pp. 530–539.

Toth, P. and D. Vigo (2003). The granular tabu search and its application to the vehicle-routing problem. In: *INFORMS Journal on Computing* 15 (4), pp. 333–346.

Toth, P. and D. Vigo, eds. (2014). Vehicle Routing: Problems, Methods, and Applications. 2nd ed. MOS-SIAM Series on Optimization. Philadelphia, USA: SIAM.

Tsurukawa, N., S. Prakash, and A. Manhart (2011). *Social impact of artisanal cobalt mining in Katanga, Democratic Republic of Congo*. Tech. rep. Accessed December 13, 2017. Freiburg, Germany: ÖkoInstitut e.V. URL: https://www.oeko.de/oekodoc/1294/2011-419-en.pdf.

van Keulen, T., B. de Jager, A. Serrarens, and M. Steinbuch (2010). Optimal energy management in hybrid electric trucks using route information. In: *Oil and Gas Science and Technololgy* 65 (1), pp. 103–113.

van Mierlo, J., M. Messagie, and S. Rangaraju (2017). Comparative environmental assessment of alternative fueled vehicles using a life cycle assessment. In: *World Conference on Transport Research - WCTR 2016 Shanghai. 10-15 July 2016*. Vol. 25. Transportation Research Procedia. Elsevier, pp. 3435–3445.

Vidal, T., T. G. Crainic, M. Gendreau, and C. Prins (2013). A hybrid genetic algorithm with adaptive diversity management for a large class of vehicle routing problems with time-windows. In: *Computers & Operations Research* 40, pp. 475–489.

Vidal, T., N. Maculan, L. S. Ochi, and P. H. V. Penna (2016). Large neighborhoods with implicit customer selection for vehicle routing problems with profits. In: *Transportation Science* 50 (2), pp. 720–734.

Wainwright, I. (2015). *TfL Freight and fleet presentation to UKNMB. Presentation on sustainable delivery and servicing - Lessons from London*. Accessed July 25, 2017. URL: http://www.ukroadsliaisongroup.org/download.cfm/docid/31DA53DC-664B-4164-87603AFECDFAE6C7.

Wang, Y.-W. and C.-C. Lin (2013). Locating multiple types of recharging stations for battery-powered electric vehicle transport. In: *Transportation Research Part E: Logistics and Transportation Review* 58, pp. 76–87.

Waters, C. D. J. (1987). A solution procedure for the vehicle-scheduling problem based on iterative route improvement. In: *Journal of the Operational Research Society* 38 (9), pp. 833–839.

Weinberg, B. and E. G. Talbi (2004). NFL theorem is unusable on structured classes of problems. In: *Proceedings of the 2004 Congress on Evolutionary Computation*. Vol. 1. IEEE, pp. 220–226.

Wolpert, D. H. and W. G. Macready (1997). No free lunch theorems for optimization. In: *IEEE Transactions on Evolutionary Computation* 1 (1), pp. 67–82.

Wulfsberg, J. P., T. Redlich, and F.-L. Bruhns (2011). Open production: Scientific foundation for co-creative product realization. In: *Production Engineering* 5 (2), pp. 127–139.

Xiao, Y., Q. Zhao, I. Kaku, and Y. Xu (2012). Development of a fuel consumption optimization model for the capacitated vehicle routing problem. In: *Computers & Operations Research* 39 (7), pp. 1419–1431.

Yellow, P. (1970). A computational modification to the savings method of vehicle scheduling. In: *Operational Research Quarterly* 21, pp. 281–283.

Zhang, J., J. Tang, and R. Fung (2011). A scatter search for multi-depot vehicle routing problem with weight-related cost. In: *Asia-Pacific Journal of Operational Research* 28 (3), pp. 323–348.

Office: Kackertstraße 7 – 52072 Aachen
☎ +49 241 80 96188 • ✉ goeke@dpo.rwth-aachen.de

Dominik Goeke

Education

06/2012–05/2018 **PhD student**, *TU Kaiserslautern*.

08/2010–01/2011 **Semester abroad**, *Linköping University (Sweden)*.

10/2006–03/2012 **Diploma**, *TU Kaiserslautern*, Industrial engineering.

Work Experience

since 07/2017 **Research Associate**, *Deutsche Post Chair - Optimization of Distribution Networks, RWTH Aachen*.

10/2016 **Research Visit**, *Vrije Universiteit Amsterdam*.

06/2012–06/2017 **Research Associate**, *Chair of Business Information Systems and Operations Research, TU Kaiserslautern*.

Office: Kackertstraße 7 – 52072 Aachen
☎ +49 241 80 96188 • ✉ goeke@dpo.rwth-aachen.de

Publications

- D. Goeke, R. Roberti, and M. Schneider (forthcoming). "Exact and Heuristic Solution of the Consistent Vehicle-Routing Problem". In: *Transportation Science*.
- D. Goeke, T. Gschwind, and M. Schneider (2017). *Upper and lower bounds for the vehicle-routing problem with private fleet and common carrier*. Tech. rep. DPO-2017-08. Manuscript submitted for publication. Aachen, Germany: Deutsche Post Lehrstuhl - Optimization of Distribution Networks. URL: http://www.dpo.rwth-aachen.de/global/show_document.asp?id=aaaaaaaaaydqff.
- D. Goeke (2017). *The Pickup and Delivery Problem with Time Windows and Electric Vehicles*. Tech. rep. DPO-2017-07. Manuscript submitted for publication. Aachen, Germany: Deutsche Post Lehrstuhl - Optimization of Distribution Networks. URL: http://www.dpo.rwth-aachen.de/global/show_document.asp?id=aaaaaaaaaaxubza.
- D. Goeke, M. Moeini, and D. Poganiuch (2017). "A Variable Neighborhood Search heuristic for the maximum ratio clique problem". In: *Computers & Operations Research* 87, pp. 283–291.
- J. Hof, M. Schneider, and D. Goeke (2017). "Solving the battery swap station location-routing problem with capacitated electric vehicles using an AVNS algorithm for vehicle-routing problems with intermediate stops". In: *Transportation Research Part B: Methodological* 97, pp. 102–112.
- D. Goeke and M. Schneider (2015). "Routing a mixed fleet of electric and conventional vehicles". In: *European Journal of Operational Research* 245.1, pp. 81–99.
- O. Wendt and D. Goeke (2014). "Revenue Management für Tourenplanungsprobleme". In: *Management integrativer Leistungserstellung: Festschrift für Hans Corsten, Betriebswirtschaftliche Schriften*. Ed. by R. Gössinger and G. Zäpfel. Vol. 168. Berlin: Duncker & Humblot, pp. 455–485.
- M. Schneider, A. Stenger, and D. Goeke (2014). "The Electric Vehicle-Routing Problem with Time Windows and Recharging Stations". In: *Transportation Science* 48.4, pp. 500–520.
- A. Stenger, M. Schneider, and D. Goeke (2013). "The prize-collecting vehicle routing problem with single and multiple depots and non-linear cost". In: *EURO Journal on Transportation and Logistics* 2.1, pp. 57–87.