Metaheuristics for Vehicle-Routing Problems Arising in Sustainable Logistics

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

D i s s e r t a t i o n

vorgelegt von

Dipl.-Wirtsch.-Ing. Julian Hof

Tag der mündlichen Prüfung:	17. Dezember 2018
Dekan:	Prof. Dr. Jan Wenzelburger
Vorsitzender:	Prof. Dr. Matthias Baum
Berichterstatter:	1. Prof. Dr. Oliver Wendt
	2. Prof. Dr. Daniele Vigo

D 386
(2018)

Contents

List of Figures

List of Tables

List of Abbreviations

2E-MTVRP-SS two-echelon multiple-trip VRP with satellite synchronization

AGVNS adaptive general VNS

ALNS adaptive LNS

AVNS adaptive VNS

BEV battery electric vehicle

BKS best-known solution

BSS battery swap station

BSS-EV-LRP BSS LRP with capacitated electric vehicles

CVRP capacitated VRP

DBCA density-based clustering algorithm

DV delivery vehicle

E-VRPTW electric VRP with time windows

EVRPRF electric VRP with recharging facilities

G-VRP green VRP

GRASP greedy randomized adaptive search procedure

IPCC Intergovernmental Panel on Climate Change

LNS large neighborhood search

LRP location-routing problem

MCWS modified Clarke and Wright savings algorithm

MDVRPI multi-depot VRP with inter-depot routes

OR operations research

PR path relinking

PVRP-IF periodic VRP with intermediate facilities

SA simulated annealing

SDG sustainable development goal

SV support vehicle

TS tabu search

UB upper bound

UN United Nations

VND variable neighborhood descent

VNS variable neighborhood search

VRP vehicle-routing problem

VRPDDP VRP with divisible deliveries and pickups

VRPDDPTW VRPDDP with time windows

VRPIRF VRP with intermediate replenishment facilities

VRPIS VRP with intermediate stops

VRPMS VRP with multiple synchronization constraints

VRPPC VRP with private fleet and common carriers

VRPRMDDP VRP with restricted mixing of divisible deliveries and pickups

VRPRMDP VRP with restricted mixing of deliveries and pickups

VRPSPD VRP with simultaneous pickup and delivery

VRPSPDTL VRPSPD with time limit

VRPSPDTW VRPSPD with time windows

VRPTW VRP with time windows

VRPTWMD VRP with time windows and mobile depots

Chapter 1

Introduction

Section 1.1 provides the background and the motivation for the problems addressed in this thesis that arise in the context of sustainable logistics operations. Section 1.2 presents the structure of this work highlighting the contribution of each chapter.

1.1. Background

The worldwide economic growth of the last century has given rise to a vast consumption of goods while globalization has led to large streams of goods all over the world. The production, transportation, storage, and consumption of all these goods, however, have created grave environmental problems. Global warming, caused by large-scale emissions of greenhouse gases, is today one of the major environmental concerns (Dekker et al., 2013). According to a recently published report by the Intergovernmental Panel on Climate Change (IPCC), the global greenhouse gas emissions have to be lowered by at least 45% by 2030 in comparison with the levels observed in 2010, reaching "net zero" around 2050, to have a likely chance of limiting the increase in global mean temperature to 1.5°C, which is vital to mitigate environmental changes like extreme weather, rising sea levels, and the decline of Arctic sea ice. The IPCC calls for "rapid, far-reaching and unprecedented changes in all aspects of society" (IPCC, 2018).

Moreover, according to the United Nations (UN), more people globally are expected to join the middle class over the next two decades. While beneficial for individual prosperity, this will further increase demand for already constrained natural resources. The "material footprint" of an economy refers to the total amount of raw materials extracted globally—across the entire supply chain—to meet that economy's final consumption demand. People rely on such materials to meet basic needs—for food, clothing, water, shelter, infrastructure, and numerous other aspects of life. Across much of the developing world, an increase in the material footprint is inevitable to enhance the living standards of growing populations. At the same time, it is crucial to reduce reliance on raw materials and increase their recycling to decrease pressure and impact on the environment. Should the global population reach 9.6 billion by 2050, the equivalent of almost three planets could be required to provide the natural resources needed to sustain current lifestyles (UN, 2018).

The 2030 Agenda for Sustainable Development proclaimed by the UN articulates these issues of mitigating climate change and of ensuring sustainable consumption and production patterns in sustainable development goals (SDGs)[1] No 13 and No 12, respectively.

[1]https://www.un.org/sustainabledevelopment/

The logistics sector represents an important lever for achieving these goals. On the one hand, it belongs to the major contributors to the world's greenhouse gas emissions and plays a key role in reducing the dependency of our economy on non-renewable and especially emission-intensive energy sources (Smokers et al., 2014). In 2016, transportation in particular accounted for 21% of the greenhouse gas emissions worldwide (International Energy Agency, 2018). In the light of the world population prospect mentioned above, a dramatically rising demand for products and thus a strong growth in goods transportation can be expected. Without measures to improve the efficiency and to reduce the carbon intensity of the transport sector, this will strongly increase the use of fossil fuels and the associated emissions of greenhouse gases (Smokers et al., 2014).

Logistics service providers have recognized the use of battery electric vehicles (BEVs) to be an effective means to make their operations more climate-efficient. Among them the Deutsche Post DHL Group which could achieve a 30% improvement in carbon efficiency in 2016 compared to the 2007 baseline thanks to an environmental protection program called GoGreen, which especially relies on the deployment of the StreetScooter, the group's own BEV for delivering mail and parcels (Appel, 2017). By the end of 2017, already 5000 StreetScooters were on the road in Germany and over the long term, DHL plans the transition of their entire fleet to BEVs (Bönnighausen, 2017; Appel, 2017).

However, the large-scale adoption of BEVs still faces a number of hurdles to overcome. While poor driving range and limited attractiveness have long been the main bottlenecks for BEV uptake, with prices declining and ranges expanding, the management consulting firm McKinsey predicts that if consumers purchase BEVs at the expected rates in the next five to ten years, a lack of recharging infrastructure might become the remaining obstacle to a widespread adoption (Knupfer et al., 2017).

The base-case scenario of a consumer survey of buyers considering BEVs conducted by McKinsey in 2016, suggests that approximately 140 million BEVs could be on the road by 2030 in China, the European Union, and the United States. The more aggressive scenario even sees this number doubled. Besides varying BEV adoption levels across regions, structural aspects lead to a highly localized demand for recharging stations. For example, a city like Los Angeles with many single-family low-rise homes which have parking garages will have extremely different recharging-infrastructure needs in comparison with New Delhi, where unorganized street parking prevails. The structural limitations of highly dense urban cities, which are characterized by larger proportions of on-street and large-commercial-garage parking, are the catalysts for increased public-charging demand (Knupfer et al., 2017).

In environments where the recharging infrastructure is not (yet) existent, mobile recharging possibilities seem to be an interesting alternative. Recently, the Italian company E-GAP[2] began to offer mobile recharging for BEVs. Upon request per mobile application, a recharging van equipped with fast recharging technology meets with a BEV in need for a recharge at its current or a predefined future parking location.

On the other hand, transport-related efforts contributing to achieving a responsible production and consumption as pursued by SDG No 12 may be found in the context of reverse logistics. Reverse logistics can be defined as the process of planning, implementing, and controlling backward flows of raw materials, in-process inventory, packaging and finished goods, from a manufacturing, distribution, or consumption point, to a point of recovery or point of proper disposal (Dekker et al., 2013). Especially

[2]http://www.e-gap.com

the recovery aspect is in line with SDG No 12. By optimizing the re-use of products, the utilization of raw materials is reduced and waste processing becomes more efficient. The extreme case of a fully circular production (the cradle-to-cradle ideal as envisioned by McDonough and Braungart, 2010) would produce zero waste (Smokers et al., 2014).

To effectively plan these logistics operations and to exploit their full potential with respect to achieving the environmental goals, techniques from the domain of operations research (OR) are reasonably applied to design corresponding decision support systems. In the scientific literature, transportation tasks are frequently represented as vehicle-routing problems (VRPs, Vigo and Toth, 2014). The basic VRP aims at determining a set of delivery routes each starting and ending at a single depot to visit a set of customers such that the total travel cost are minimized. Numerous variants of the VRP incorporating real-world constraints and conditions have been proposed, among them the capacitated VRP (CVRP), where vehicles are characterized by a limited freight capacity (see, e.g., Irnich, Toth, and Vigo, 2014) and the VRP with time windows (VRPTW), where customers are associated with specific time intervals within which their service must start (see, e.g., Bräysy and Gendreau, 2005a; Bräysy and Gendreau, 2005b). The CVRP and its extensions are $\mathcal{NP}$-hard and thus only relatively small-sized instances can be solved by means of exact algorithms. Therefore, numerous metaheuristic solution methods have been proposed to provide solutions for instances of sizes faced in practical applications (Laporte, Ropke, and Vidal, 2014).

To consider the utilization of BEVs in the route planning, corresponding routing models need to adequately address the characteristics of BEVs. For example, despite improvements in battery technology in recent years, the maximum driving range of BEVs might not always be sufficient to perform typical delivery tours in one run or to reach customers located far away from the depot. Therefore, visits to recharging or battery swap stations (BSSs) need to be properly integrated into the route planning process, especially if the respective infrastructure is sparse.

The task of planning the BEV infrastructure, i.e., determining the appropriate locations for recharging stations or BSSs may be reasonably modeled as location-routing problem (LRP) in which decisions on the location of facilities are jointly taken with decisions on the routing of vehicles. LRPs arise from the observation that taking these types of decisions independently of one another may lead to highly suboptimal planning results, even if the location decisions are aimed for the long term (Drexl and Schneider, 2014).

The deployment of mobile means for recharging or swapping the vehicle battery leads to the interdependence between the vehicle routes. This results from the necessity for the vehicles to meet at a location in order to perform the transfer operation. Thus, in addition to the usual task covering constraints, further synchronization requirements between the vehicles arise. Consequently, corresponding problems are referred to as VRPs with multiple synchronization constraints (VRPMSs) in the scientific literature (Drexl, 2012). While in many VRP variants, the vehicle routes are independent of each other, modifications to a route in a VRPMS might render the entire solution infeasible in the worst case.

Finally, as described above, reverse logistics means that in addition to the distribution of goods to the customers, e.g., re-usable packaging and goods to be recycled or remanufactured have to be transported in the reverse direction. If both tasks have to be performed simultaneously at the customer locations, a routing problem in the literature commonly referred to as VRP with simultaneous pickup and delivery (VRPSPD) arises (see, e.g, Dethloff, 2001). In the classical CVRP, the vehicle load monotonically

decreases along a route depending on the delivered customer demands. This is not true for VRPSPDs where the additional existence of pickup demands results in a rather fluctuating load profile, which calls for the implementation of sophisticated capacity evaluation procedures in respective solution methods.

1.2. Organization and Contribution

This thesis addresses the above described challenges for sustainable logistics operations and investigates (1) the integration of intermediate stops in the route planning of transportation vehicles, which especially becomes relevant when BEVs with limited driving range are considered, (2) the combined planning of the battery replacement infrastructure and of the routing for BEVs, (3) the use of mobile load replenishment or refueling possibilities in environments where the respective infrastructure is not available, and (4) the additional consideration of the flow of goods from the end user backwards to the point of origin for the purpose of, e.g., recapturing value or proper disposal. We[3] utilize models and solution methods from the domain of OR to gain insights into the investigated problems and thus to support managerial decisions with respect to these issues. Note that, due to the diversity of the problems addressed in this thesis, we refrain from providing an entirely unified notation. Instead, we introduce the notation used in each chapter anew thus allowing each chapter to be read on its own.

Chapter 2 introduces some basic concepts that are relevant for the problems and approaches addressed in this thesis. As the CVRP lies at the core of all problems investigated in this work, we provide a thorough description of the problem. In addition, we present the basic principles of the metaheuristic approaches that we adapt in this work.

In Chapter 3, we introduce a route planning model that considers intermediate stops for the transportation vehicles. The *vehicle-routing problem with intermediate stops* (VRPIS) features a set of intermediate facilities at which vehicles may stop en-route to replenish the goods to be delivered or to refuel. With respect to the first application case, intermediate replenishment stops are used in distribution systems where the products to be delivered are stored at several facilities with the goal of reloading the delivery vehicles without having to return to a central depot. The second aspect is especially relevant when considering the utilization of BEVs. Then, the intermediate facilities may represent recharging stations that allow to recharge the vehicle battery or battery swap stations at which vehicles can swap their depleted battery for a fully charged one. In contrast to the obligatory customer visits, these stops are optional and aim at keeping the vehicles operational.

As VRPIS extends the $\mathcal{NP}$-hard CVRP by several combinatorial aspects, we develop an adaptive variable neighborhood search (AVNS) heuristic to provide solutions for the VRPIS. We assess the competitiveness of the proposed approach on benchmark instances from the literature for two special cases of the VRPIS, namely for the green VRP and the VRP with intermediate replenishment facilities. Moreover, we consider an additional special case of the VRPIS, the electric VRP with recharging facilities (EVRPRF). We design two sets of small and large EVRPRF instances based on well-known CVRP benchmarks. On the small instances, we assess the performance of our AVNS in comparison to the commercial solver IBM ILOG CPLEX[4]. Finally, we analyze the effect of the problem-specific

[3]To improve readability, the first person plural is used to indicate the author of this thesis throughout the work as commonly done in the scientific literature.

[4]https://www.ibm.com/analytics/cplex-optimizer

components that explicitly address intermediate facilities on solution quality and computation time. Chapter 3 has been published in a similar form in Schneider, Stenger, and Hof (2015).

In Chapter 4, we study the *battery swap station location-routing problem with capacitated electric vehicles* (BSS-EV-LRP) that has been introduced by Yang and Sun (2015). In the BSS-EV-LRP, a homogeneous fleet of BEVs with limited driving range and capacity is stationed at a single depot and has to serve a set of customers with given demands. To stay operational, the BEVs can stop at BSSs to perform a replacement of the depleted vehicle battery. From an algorithmic point of view, BSSs can thus be modeled as intermediate facilities. However, in the BSS-EV-LRP, an additional strategic component has to be considered: While in the VRPIS, the number and locations of intermediate facilities are given ex ante, the BSS-EV-LRP calls for the determination of the BSS locations to open from a set of candidate locations with given construction cost.

Using the AVNS algorithm introduced in Chapter 3, we show how algorithms designed for VRPIS can generally be extended to address the BSS-EV-LRP. In numerical studies, we investigate the performance of our extended AVNS and its components on the BSS-EV-LRP benchmark from the literature and on a newly introduced instance set for the BSS-EV-LRP, which is more meaningful with regard to the necessity of utilizing BSSs. We use this new set to additionally investigate the effect of varying construction cost on the location of BSSs. The contents of Chapter 4 have been published in a similar form in Hof, Schneider, and Goeke (2017).

Chapter 5 introduces the *vehicle-routing problem with time windows and mobile depots* (VRPTWMD). The VRPTWMD is characterized by a fleet of delivery vehicles (DVs) and a fleet of support vehicles (SVs). Depending on the application context, an SV may either serve as a mobile depot to restore the load capacity of the DVs for the customer demand to be delivered or the DVs' fuel capacity thus acting as a mobile refueling station. To serve in one of the described ways, an SV must meet with a DV at a location, and both vehicles must stay at that location until the capacity transfer terminates. The possible meeting points may encompass specific customer locations and dedicated transfer locations. In the former case, a transfer may only take place when the DV serves the respective customer. The order in which a DV performs the service at a customer and the transfer with the SV takes place is a decision variable.

We develop an adaptive large neighborhood search (ALNS) heuristic combined with a path relinking (PR) approach, called ALNS-PR, to address the VRPTWMD. Our ALNS-PR makes use of problem-specific components that explicitly consider the existence of resource transfers and is able to evaluate fuel, load, and time window violations in constant time. We generate two sets of VRPTWMD instances: A set of small-sized instances that we can solve exactly with the commercial solver CPLEX to assess the performance of ALNS-PR on the VRPTWMD and a set of more realistically sized instances that we use to analyze the usefulness of certain components of our algorithm and the effect of different problem characteristics on the structure of the identified solutions. In addition, we evaluate the competitiveness of our algorithm on benchmark instances for the related two-echelon multiple-trip VRP with satellite synchronization.

In Chapter 6, we investigate a class of *vehicle-routing problems with simultaneous pickup and delivery* (VRPSPD). In VRPSPDs, customers may require (i) delivery service of goods originating at the depot, and (ii) pickup service for goods which need to be returned to the depot. Consequently, route planning tasks arising in the context of reverse logistics may be adequately represented as VRPSPDs. In addition to the standard VRPSPD, we investigate (i) the VRPSPD with time limit (VRPSPDTL),

which imposes a maximum duration on the vehicle routes, (ii) the VRPSPD with time windows (VRPSPDTW), where customers are associated with time intervals in which their service must start, (iii) the VRP with divisible deliveries and pickups (VRPDDP), which allows to satisfy a customer's pickup and delivery requests in two separate visits, (iv) the previously unstudied VRP with restricted mixing of divisible deliveries and pickups (VRPRMDDP), which additionally requires that a certain percentage of the vehicle capacity must remain unoccupied when both types of demand are simultaneously loaded, and (v) the previously unstudied VRPDDP with time windows (VRPDDPTW). In order to study the suitability of our solution method for an even larger variety of problems, we perform additional experiments on instances for the VRP with mixed deliveries and pickups (VRPMDP) and the VRP with restricted mixing of deliveries and pickups (VRPRMDP), which represent special cases of the VRPSPD(TL) and the VRPRMDDP, respectively, in which each customer is associated with only one type of demand.

We again develop an ALNS-PR hybrid to provide solutions for the investigated variants and special cases of the VRPSPD. Besides approaches with general validity for VRPs, we implement algorithmic components that explicitly take the load characteristics of VRPSPDs into account. In extensive numerical studies, we analyze the benefits of the algorithmic innovations and the competitiveness of our algorithm on benchmark instances from the literature. The contents of this chapter will appear as Hof and Schneider (forthcoming).

Finally, we summarize the findings of this thesis and provide an outlook on future research opportunities in Chapter 7.

Chapter 2

Fundamental Concepts

In this section, we provide some fundamental concepts that are relevant for the problems investigated in this thesis. In Section 2.1, we introduce the capacitated vehicle-routing problem (CVRP) which builds the foundation of all problems addressed in this work. We give a thorough description and provide an exact mathematical formulation of the problem.

Section 2.2 lays the foundations for the solution methods developed in this thesis. To this end, we present the basic principles of the metaheuristic paradigms that have been adapted in this work.

2.1. The Capacitated Vehicle-Routing Problem

The CVRP is the most studied version of the VRP. Although the CVRP itself is primarily relevant in the academic context, it lies at the core of most VRP variants studied in the literature. The CVRP seeks to determine a cost-minimal set of routes performed by a fleet of homogeneous vehicles located at a single depot. Each route starts and ends at the depot and the cumulated demand of all customers visited on a route must not exceed the capacity of the associated vehicle. Each customer is served by exactly one vehicle (Irnich, Toth, and Vigo, 2014).

In the following, we provide a formal definition of the CVRP as mixed-integer program. Let $\mathcal{C} = \{1, ..., n\}$ denote the set of n customers and let 0 and $n + 1$ denote instances of the same depot corresponding to the start and end of each vehicle route, respectively. Moreover, let $\mathcal{V}$ denote the set containing the depot and the customers. We use indices 0 and $n+1$ to indicate which depot instances are contained in the set, i.e., $\mathcal{V}_0 = \{0\} \cup \mathcal{C}$, $\mathcal{V}_{n+1} = \{n + 1\} \cup \mathcal{C}$, and $\mathcal{V}_{0,n+1} = \{0\} \cup \{n + 1\} \cup \mathcal{C}$.

The CVRP can then be defined on a complete directed graph $\mathcal{G} = (\mathcal{V}_{0,n+1}, \mathcal{A})$ with the set of arcs $\mathcal{A} = \{(i, j) : i, j \in \mathcal{V}_{0,n+1}, i \neq j\}$. Each arc $(i, j) \in \mathcal{A}$ is associated with a travel distance c_{ij} and a travel time t_{ij}. Each customer $i \in \mathcal{C}$ has a nonnegative demand u_i. A homogeneous fleet of m vehicles with capacity q is based at the depot.

Moreover, the following decision variables are used in the model: a_i specifies the time and l_i the load level on arrival at vertex i. Finally, binary decision variables x_{ij} take value 1 if vertex j is visited after vertex i and 0 otherwise. Based on this notation, we formulate the mixed-integer program of the CVRP as follows:

$$min \sum_{i \in \mathcal{V}_0} \sum_{j \in \mathcal{V}_{n+1}} c_{ij} x_{ij} \tag{2.1}$$

$$\sum_{i \in \mathcal{V}_0 \setminus \{j\}} x_{ij} = 1 \qquad \forall j \in \mathcal{C} \tag{2.2}$$

$$\sum_{i \in \mathcal{V}_0 \setminus \{j\}} x_{ij} - \sum_{i \in \mathcal{V}_{n+1} \setminus \{j\}} x_{ji} = 0 \qquad \forall j \in \mathcal{C} \tag{2.3}$$

$$\sum_{j \in \mathcal{V}_{n+1}} x_{0j} \leqslant m \tag{2.4}$$

$$a_i + t_{ij} x_{ij} - M(1 - x_{ij}) \leqslant a_j \qquad \forall i \in \mathcal{V}_0, j \in \mathcal{V}_{n+1}, i \neq j \tag{2.5}$$

$$0 \leqslant a_i \leqslant \mathbb{R} \qquad \forall i \in \mathcal{V}_{0,n+1} \tag{2.6}$$

$$l_j \leqslant l_i - u_i + q(1 - x_{ij}) \qquad \forall i \in \mathcal{V}_0, j \in \mathcal{V}_{n+1}, i \neq j \tag{2.7}$$

$$0 \leqslant l_i \leqslant q \qquad \forall i \in \mathcal{V}_{0,n+1} \tag{2.8}$$

$$x_{ij} \in \{0, 1\} \qquad \forall i \in \mathcal{V}_0, j \in \mathcal{V}_{n+1}, i \neq j \tag{2.9}$$

The goal of the CVRP is to minimize the total distance traveled expressed by objective function (2.1). Constraints (2.2) guarantee that each customer is visited exactly once. Flow conservation is provided via Constraints (2.3) by requiring that at each customer, the number of incoming arcs is equal to the number of outgoing arcs. The number of arcs leaving the depot is limited to the number of available vehicles in Constraints (2.4). Constraints (2.5) prevent the formation of subtours by establishing time feasibility for arcs leaving customers and the depot. The range of the arrival time is defined in Constraints (2.6). Feasibility with respect to the vehicle load is ensured by Constraints (2.7) and (2.8). Finally, the binary variables are defined in (2.9).

2.2. Metaheuristic Solution Methods for VRPs

Even the basic CVRP proves to be very challenging as it degenerates to the $\mathcal{NP}$-hard one-dimensional bin-packing problem if all arc weights are zero (Coffman, Garey, and Johnson, 1984). Despite considerable effort put in the development of sophisticated mathematical programming decomposition algorithms since the VRP was first introduced in Dantzig and Ramser (1959), only relatively small instances can be solved optimally and run-times highly vary. However, instances faced in practical applications are often large and characterized by numerous real-world constraints which calls for the development of efficient metaheuristic solution methods that are able to obtain adequate solutions in reasonable and predictable computation times (Laporte, Ropke, and Vidal, 2014).

Metaheuristics are typically high-level strategies which guide underlying problem-specific heuristics to enhance their performance. The main goal is to avoid the disadvantages of purely greedy local search procedures by intelligently combining different concepts for exploring and exploiting the search space and thus allowing the search to escape from local optima. This may encompass the acceptance of worsening moves during the search or the generation of new starting solutions for the local search not entirely at random but in an intelligent, biased way (Osman and Laporte, 1996; Gendreau and Potvin, 2005). In contrast to exact methods, metaheuristics are approximation algorithms as they do not guarantee to obtain the global optimum of an optimization problem. Instead, they aim at achieving a reasonable trade-off between solution quality and computational effort.

Metaheuristics may be classified with respect to the number of solutions considered at each iteration. We distinguish between (i) single-solution and (ii) population-based metaheuristics. While in single-

solution methods, only a single solution is modified at any time in the search, population-based methods aim at concurrently evolving a multiplicity of solutions (Gendreau and Potvin, 2005).

The solution methods presented in this thesis are composed of components from both categories of metaheuristics. In the following, we first provide the foundations for the single-solution approaches adapted in this work, namely simulated annealing (SA), variable neighborhood search (VNS), and large neighborhood search (LNS) in Sections 2.2.1, 2.2.2, and 2.2.3, respectively. Finally, we describe the basic concept of path relinking (PR), the population-based approach utilized in this work, in Section 2.2.4.

2.2.1. Simulated Annealing

SA, first proposed in Kirkpatrick, Gelatt, and Vecchi (1983) and Černý (1985), is a randomized local search procedure that implements a strategy to escape from local optima by accepting worse solutions with a certain probability which is decreased in the course of the search.

The algorithm is inspired by the physical annealing process used to obtain low-energy states of solids. In condensed matter physics, annealing denotes a process in which a solid is first melted by increasing its temperature and then allowed to cool very slowly until it achieves its most regular crystal lattice configuration possible (i.e., its minimum lattice energy state), and thus is free of crystal defects (Nikolaev and Jacobson, 2010). In a combinatorial optimization context, a solution corresponds to a state of the physical system and the objective function value to the associated energy (Gendreau and Potvin, 2005).

A pseudocode representation of the SA algorithm is shown in Figure 2.1 (Blum and Roli, 2003). At each iteration, the current solution $\mathcal{S}$ is modified by randomly selecting a solution $\mathcal{S}'$ from the neighborhood $\mathcal{N}(\mathcal{S})$ of $\mathcal{S}$. A neighborhood of a solution is typically implicitly defined in form of a class of transformations, i.e., neighborhood operators applicable to a solution. If the new solution $\mathcal{S}'$ improves on $\mathcal{S}$, it is always accepted and replaces $\mathcal{S}$ as the starting point for the subsequent iteration. Otherwise, the new solution is accepted according to the Metropolis criterion (Metropolis et al., 1953), where the probability of acceptance is related to the magnitude of the cost increase and a temperature parameter ϑ:

$$e^{\frac{-(f(\mathcal{S}')-f(\mathcal{S}))}{\vartheta}}.$$

Consequently, a solution is more likely to be accepted if the temperature is high and the magnitude of the cost increase is low. SA starts with an initial temperature ϑ_0 which is incrementally decreased according to a predefined cooling schedule where a certain number of iterations are performed at each temperature level. Assuming an infinite number of iterations, it can be proven that SA–contrary to most heuristics–asymptotically converges to a global optimum. Finite-time implementations, however, do not provide such a guarantee which compromises the practical benefit of this observation (Gendreau and Potvin, 2005).

2.2.2. Variable Neighborhood Search

VNS has been introduced by Mladenović and Hansen (1997) and follows the idea of exploring increasingly distant neighborhoods of a solution to escape from local optima. More precisely, the procedure

$\mathcal{S} \leftarrow$ generateInitialSolution()
$\vartheta \leftarrow \vartheta_0$
while termination conditions not satisfied **do**
  $\mathcal{S}' \leftarrow$ generateRandomSolution($\mathcal{N}(\mathcal{S})$)
  if $f(\mathcal{S}') < f(\mathcal{S})$ **then**
    $\mathcal{S} \leftarrow \mathcal{S}'$
  else
    Accept $\mathcal{S}'$ as new solution with probability $e^{\frac{-(f(\mathcal{S}')-f(\mathcal{S}))}{\vartheta}}$
  end if
  update(ϑ)
end while

Figure 2.1.: Simulated annealing in pseudocode.

moves to the next neighborhood as soon as the local optimum of the current neighborhood is obtained. Figure 2.2 provides an overview of the VNS paradigm in pseudocode (Hansen et al., 2010).

Define the set of neighborhood structures $\mathcal{N}_\kappa$ with $\kappa = 1, ..., \kappa^{max}$
Generate initial solution $\mathcal{S}$
$\kappa \leftarrow 1$
while termination conditions not satisfied **do**
  {*Shaking*}
  $\mathcal{S}' \leftarrow$ generateRandomSolution($\mathcal{N}_\kappa(\mathcal{S})$)
  {*Local Search*}
  $\mathcal{S}'' \leftarrow$ performLocalDescent($\mathcal{S}'$)
  if $f(\mathcal{S}'') < f(\mathcal{S})$ **then**
    $\mathcal{S} \leftarrow \mathcal{S}''$
    $\kappa \leftarrow 1$
  else
    $\kappa \leftarrow \kappa \mod \kappa^{max} + 1$
  end if
end while

Figure 2.2.: Variable neighborhood search in pseudocode.

Given a set of predefined neighborhood structures $\{N_\kappa \mid \kappa = 1, ..., \kappa^{max}\}$, a perturbation phase (also called shaking) generates a random solution $\mathcal{S}'$ in the first neighborhood of the current solution $\mathcal{S}$ from which a local descent is subsequently performed. If the obtained local optimum $\mathcal{S}''$ does not improve on the incumbent, the next neighborhood structure is selected and the procedure repeated. The search is restarted from the first neighborhood when either an improving solution has been found or every neighborhood structure has been explored. VNS terminates as soon as a stopping condition, e.g., a maximum number of iterations without improvement, is met (Gendreau and Potvin, 2005).

2.2.3. Large Neighborhood Search

LNS was originally proposed by Shaw (1998) and is based on a ruin-and-recreate principle. More precisely, at each iteration, a destroy method is applied to destruct a typically large portion of the current solution and a repair method subsequently rebuilds the destroyed solution. The destroy method generally contains an element of stochasticity to favor the recreation of different parts of a solution at every invocation of the method. The neighborhood $\mathcal{N}(\mathcal{S})$ of a solution $\mathcal{S}$ is then defined as the set of solutions that can be reached by successively applying the destroy and the repair method to $\mathcal{S}$. If the solution $\mathcal{S}'$ obtained in this way improves on the current one $\mathcal{S}$, $\mathcal{S}'$ replaces $\mathcal{S}$ as the starting point for the subsequent iteration. Otherwise, $\mathcal{S}$ is again destroyed and rebuilt. Figure 2.3 shows the LNS procedure in pseudocode (Pisinger and Ropke, 2010).

```
Generate initial solution S
while termination conditions not satisfied do
  S' ← repairSolution(destroySolution(S))
  if f(S') < f(S) then
    S ← S'
  end if
end while
```

Figure 2.3.: Large neighborhood search in pseudocode.

Contrary to many local-search-based methods, the basic LNS metaheuristic does not search the entire neighborhood of a solution but rather samples the neighborhood. However, the large neighborhood allows LNS to navigate the solution space and thus to overcome local optima easily, even if the problem instance is tightly constrained (Pisinger and Ropke, 2010).

2.2.4. Path Relinking

PR was originally proposed by Glover (1997) as a complementary intensification strategy that seeks to discover improving solutions on the trajectories between elite solutions obtained by another metaheuristic like, e.g., tabu search (Resende et al., 2010). To this end, starting from an initial solution, PR generates new solutions by successively incorporating characteristics of a guiding solution into the current solution. In other words, by selecting moves that introduce attributes contained in the guiding solution, a path in the neighborhood space leading toward the guiding solution is created. Instead of merely encouraging the selection of moves that decrease the diversity between both solutions, PR subordinates all other considerations to the goal of introducing the attributes of the guiding solution (Laguna and Marti, 1999).

For the purpose of illustration, Figure 2.4 shows two exemplary paths, i.e., sequences of moves that connect a high-quality solution A to a high-quality solution B found during the search. The solid line indicates the path obtained by applying a typical local-search-based heuristic that seeks to minimize the objective function value but implements some strategy to escape from local optima. The dashed line depicts the path followed by the PR procedure. The paths differ because PR prioritizes the incorporation of attributes of the guiding solution. In this way, improving solutions may be discovered that are not in the neighborhood of the solutions visited by the original path (solution C).

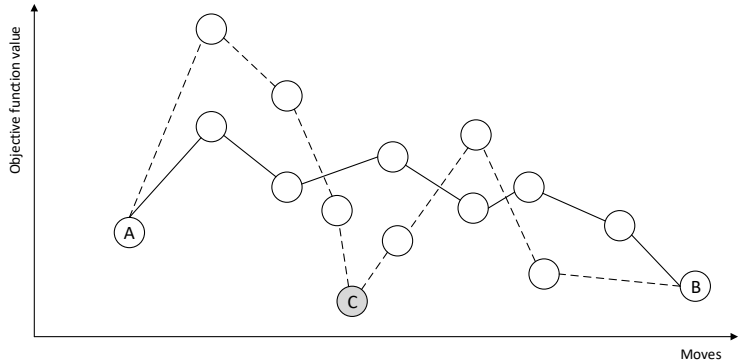


Figure 2.4.: Example of a path obtained via the path relinking procedure in comparison to the path followed by a local-search-based approach (Laguna and Marti, 1999).

For this reason, PR may reasonably complement metaheuristics that generate a sequence of locally optimal feasible solutions like SA, VNS, ALNS, or as initially mentioned, tabu search. To hybridize

PR with these approaches, a diverse pool of high-quality solutions found during the search, called elite set, is usually employed. Depending on the specific PR implementation, each locally optimal solution obtained in the course of the search is relinked with one or more solutions contained in the elite set. The elite set is initially empty and limited in size. Each locally optimal solution produced by the metaheuristic and each solution returned by PR is considered as a candidate for inclusion in the elite set. If the set is not yet filled, any solution that differs from the solutions already contained in the set may be added. Otherwise, a simple strategy may consist in accepting a solution to be included if it improves on the worst solution currently contained in the elite set. The new solution then replaces the previous worst solution (Resende et al., 2010).

Chapter 3

Considering Intermediate Stops in Route Planning

3.1. Introduction

Intermediate stops have to be considered in many practical vehicle routing applications, e.g., for (i) replenishment of the goods to be delivered, (ii) refueling (or recharging in case of battery electric vehicles), or (iii) unloading of collected goods or disposal of waste. These stops differ from regular customer stops in two aspects: First, they are optional, and second, they depend on the state of the vehicle with respect to load and fuel level (which decides the latest possible moment at which an intermediate stop has to occur). Contrary to optional customer stops, e.g., in vehicle-routing problems (VRPs) with profits (Archetti, Speranza, and Vigo, 2014), intermediate stops are not directly related to customer service or profit maximization but aim at keeping the vehicle operational.

As mentioned above, major applications of intermediate stops are good replenishment, refueling and waste disposal, which are detailed in the following. Intermediate replenishment stops are used in distribution systems with several facilities storing the products to be delivered (Angelelli and Mansini, 2002; Crevier, Cordeau, and Laporte, 2007; Tarantilis, Zachariadis, and Kiranoudis, 2008). The aim is to avoid returning to a central depot in order to reload the delivery vehicle. Concrete applications can be found in the distribution of heating oil (Prescott-Gagnon, Desaulniers, and Rousseau, 2014), road maintenance (Amaya, Langevin, and Trépanier, 2007), or in city logistics, where city freighters may visit satellite facilities in order to be replenished (Crainic, Ricciardi, and Storchi, 2009).

Intermediate stops for unloading operations are common in waste collection or snow ploughing. Here, intermediate disposal sites need to be visited, at the latest, when the maximal capacity of the vehicle is reached (see, e.g., Kim, Kim, and Sahoo, 2006; Benjamin and Beasley, 2010).

Intermediate refueling stops occur in several practical applications. For example, several companies keep contracts with gas station chains in order to get special rates at the respective stations, which makes it profitable to consider refueling stops in the route planning. Without such contracts, this is not the case because the network of gas stations is generally quite dense in developed countries. The refueling topic gains further relevance by the strong growth in alternative fuels, namely biodiesel, ethanol, hydrogen, methanol, natural gas, or propane, for which only a sparse infrastructure is existent. Finally, battery electric vehicles (BEVs) need to stop to recharge during longer routes due to their limited driving range (Conrad and Figliozzi, 2011; Schneider, Stenger, and Goeke, 2014). BEV technologies have recently gained importance due to city logistics concepts, which aim at reducing the negative external effects of urban freight transportation. In this context, BEVs seem to be a very good choice as they have no local emissions, operate very efficiently at the stop-and-go level and have low

noise levels (Wang and Lin, 2013). Moreover, BEVs are defined to be emission-free by EU regulation No 510/2011 and are therefore a major means to comply with laws and regulations on emissions.

We are the first to introduce the VRP with intermediate stops (VRPIS), a routing model that considers visits to intermediate facilities in order to keep vehicles operational. The necessity to visit an intermediate facility depends on the fuel and/or the load level of a delivery vehicle. The time spent at a facility is defined as a function of the fuel and load level on arrival. By abstracting from the actual purpose of the intermediate stop, be it for replenishment, refueling, or disposal, our problem definition comprises several problems with specific applications proposed in the literature. The objective of VRPIS is to minimize the total costs composed of travel costs and fixed costs for the deployment of vehicles.

VRPIS extends the $\mathcal{NP}$-hard capacitated VRP (CVRP) by several combinatorial aspects, which makes exact methods unsuitable for solving large problem instances in fast computation time. We develop a heuristic solution approach, namely an adaptive variable neighborhood search (AVNS), which combines ideas of VNS (Hansen and Mladenović, 2001) and adaptive large neighborhood search (ALNS, see Pisinger and Ropke, 2007). This method has been successfully applied to single and multi-depot routing problems (see Stenger et al., 2013; Stenger, Schneider, and Goeke, 2013). To assess the performance of our AVNS, we perform computational studies on benchmark instances of VRPIS variants previously studied in the literature.

Moreover, we investigate the electric VRP with recharging facilities (EVRPRF) as a special case of VRPIS. EVRPRF models the routing decision of logistics service providers employing BEVs. The driving range of a vehicle is restricted by the maximum battery capacity and a distance-related energy consumption along the route, which determines the battery charge. We simplify several real-world characteristics and do not consider the influence of vehicle load, vehicle speed, and grades on energy consumption. The battery can be recharged at any of the available recharging facilities. For this problem, we generate a set of small benchmark instances, which are used to assess the solution quality of our method in comparison to the commercial solver CPLEX. In addition, detailed results for a set of large instances are provided.

This chapter is organized as follows. In Section 3.2, we review the related literature. Section 3.3 presents the problem description and the mathematical models of VRPIS and of the special case EVRPRF. The AVNS solution method is detailed in Section 3.4. Computational tests to assess the performance of the proposed method are described in Section 3.5. The chapter is summarized and concluded in Section 3.6.

3.2. Literature Review

This section gives short reviews of the following strands of literature related to VRPIS and EVRPRF: (i) VRPs with intermediate replenishment or disposal stops, (ii) VRPs with refueling or recharging stops, and (iii) refueling problems occurring in other application areas.

Crevier, Cordeau, and Laporte (2007) introduce the multi-depot VRP with inter-depot routes (MD-VRPI), which considers intermediate depots at which vehicles can be replenished with goods during the course of a route. The authors develop a heuristic procedure that combines ideas from adaptive

memory programming (Rochat and Taillard, 1995), tabu search (TS) and integer programming. Although the multi-depot case is described, all proposed benchmark instances consider only one depot at which the vehicle fleet is stationed. Therefore, Tarantilis, Zachariadis, and Kiranoudis (2008) rename the problem to VRP with intermediate replenishment facilities (VRPIRF) and we adopt this acronym for the remainder of this thesis. They propose a hybrid guided local search heuristic that follows a three-step procedure. First, an initial solution is constructed by means of a cost-savings heuristics. Second, a VNS algorithm is applied using a TS in the local search phase. Third, the solution is further improved by means of a guided local search.

Prescott-Gagnon, Desaulniers, and Rousseau (2014) propose three metaheuristics to solve a VRP arising in heating oil distribution, considering intra-route replenishments, heterogeneous vehicles, optional customer visits and time windows. The authors design a TS, an LNS based on the TS and a column generation heuristic and report computational results obtained on test instances derived from a real-world dataset. Other problems similar to VRPIRF arise in the collection of waste (see, e.g., Angelelli and Mansini, 2002; Kim, Kim, and Sahoo, 2006; Coene, Arnout, and Spieksma, 2010; Benjamin and Beasley, 2010), in snow clearance (Perrier, Langevin, and Campbell, 2007), or in road maintenance and marking (Amaya, Langevin, and Trépanier, 2007; Salazar-Aguilar, Langevin, and Laporte, 2013). A recent review of the literature on waste collection can be found in Beliën, De Boeck, and Van Ackere (2014).

Hemmelmayr et al. (2013) study the periodic VRP with intermediate facilities (PVRP-IF) in the context of waste collection. The authors introduce a hybrid solution approach consisting of a VNS using dynamic programming to insert intermediate facilities. The solution procedure is also applied to the VRPIRF problem instances provided by Crevier, Cordeau, and Laporte (2007) and Tarantilis, Zachariadis, and Kiranoudis (2008) and is able to outperform both approaches.

The literature on routing problems with refueling stops is still relatively scarce. Conrad and Figliozzi (2011) present the recharging VRP, in which vehicles with limited range have the possibility of recharging en route at certain customer locations. The recharging time is assumed to be fixed. The impact of maximum driving range, recharging time, and time window existence is studied using a selection of the VRP with time windows (VRPTW) instances of Solomon (1987). Moreover, bounds are formulated to predict average tour lengths. Erdoğan and Miller-Hooks (2012) propose the green VRP (G-VRP), which considers a limited fuel capacity of the vehicles and the possibility of refueling at facilities along the route with a fixed refueling time. Neither capacity restrictions nor time window constraints are considered. The authors propose two heuristics to solve the G-VRP. The first is a modified Clarke and Wright savings algorithm (MCWS) which creates routes by establishing feasibility through the insertion of refueling facilities, merging feasible routes according to savings, and removing redundant facilities. The second heuristic is a density-based clustering algorithm (DBCA) designed as cluster-first and route-second approach.

Schneider, Stenger, and Goeke (2014) develop a hybrid heuristic approach that combines VNS with TS to address the electric VRP with time windows (E-VRPTW). Contrary to the EVRPRF studied in this chapter, their E-VRPTW includes time windows but features no maximal route duration constraints. Moreover, their objective is hierarchical and inspired by the objective function used in heuristic methods for the VRPTW: They first minimize the number of employed vehicles and only minimize traveled distance second, whereas we follow the objective of minimizing total costs composed of travel costs and fixed vehicle costs. Their VNS/TS is able to significantly improve on the results

of Erdoğan and Miller-Hooks (2012) on the G-VRP instances and achieves convincing results on the VRPIRF instances, although the method is not specifically designed for this type of problem.

Refueling problems are also investigated in other application areas, e.g., the refueling of aircraft or locomotives. In Barnes et al. (2004), tanker aircraft stationed at several bases have to be assigned to receiver aircraft in order to perform refueling in midair. Raviv and Kaspi (2012) deal with the optimal refueling schedule of locomotives pulling trains, i.e., the determination of the yards at which the locomotives have to be refueled.

3.3. Problem Definition

This section presents a mixed-integer program of the VRPIS (Section 3.3.1) and derives the formulation of the special case EVRPRF (Section 3.3.2).

3.3.1. The Vehicle-Routing Problem with Intermediate Stops

We start with the introduction of some necessary notation. Let $\mathcal{C} = \{1, ..., n\}$ denote the set of n customers and let $\mathcal{F}$ denote the set of facilities. The set $\mathcal{F}$ itself comprises the set of refueling facilities $\mathcal{F}_F$, the set of replenishment or unloading/disposal facilities (in the following denoted as loading facilities) $\mathcal{F}_L$, and the set $\mathcal{F}_{FL}$ of facilities where both refueling and loading is possible (from now on referred to as combined facilities), i.e., $\mathcal{F} = \mathcal{F}_F \cup \mathcal{F}_L \cup \mathcal{F}_{FL}$. We use a set of dummy vertices $\mathcal{F}'$ to allow several visits to the facilities in $\mathcal{F}$ (see, e.g., Bard et al., 1998; Schneider, Stenger, and Goeke, 2014). Further, let vertices 0 and $n + 1$ denote instances of the same depot representing the start and end of each vehicle route. To indicate which depot instances are included in a fictive set $\mathcal{X}$, the respective depot instances are used as indices, i.e., $\mathcal{X}_0 = \mathcal{X} \cup \{0\}$, $\mathcal{X}_{n+1} = \mathcal{X} \cup \{n + 1\}$ and $\mathcal{X}_{0,n+1} = \mathcal{X} \cup \{0\} \cup \{n + 1\}$. Finally, let $\mathcal{V}' = \mathcal{C} \cup \mathcal{F}'$ denote the set of all customers and visits to facilities.

Then, VRPIS can be defined on the complete directed graph $\mathcal{G} = (\mathcal{V}'_{0,n+1}, \mathcal{A})$ with the set of arcs $\mathcal{A} = \{(i, j) : i, j \in \mathcal{V}'_{0,n+1}, i \neq j\}$. Arcs $(i, j) \in \mathcal{A}$ are associated with a cost c_{ij}, a distance d_{ij}, and a travel time t_{ij}. A homogeneous fleet of m vehicles with load capacity q, fuel capacity p, and fixed costs per use c^{fix} is stationed at the depot. Fuel capacity is expressed in distance units and denotes the distance that can be traveled with maximum fuel level.

Each customer $i \in \mathcal{C}$ has a positive demand u_i and service time t^s_i. Each facility visit $j \in \mathcal{F}'$ is associated with a docking time t^d_j, which marks the time span between the arrival at the facility and the beginning of the actual refueling and/or loading process. The time span for the refueling and loading process is determined by functions $\Phi^f(f_j)$ and $\Phi^l(l_j)$ respectively. It may depend on the fuel level f_j (the cargo level l_j) on arrival at the facility. Visiting a facility $j \in \mathcal{F}_F$ completely refills the fuel tank of a vehicle and vehicles are fully replenished or unloaded at facilities in $\mathcal{F}_L$. For combined facilities, we assume that vehicles are fully loaded and are simultaneously refueled during the time span occupied by the loading process. The increase in fuel during that time span is given by the function $\Theta(l_j)$, which depends on the loading time and thus on the load level l_j on arrival at the facility. A refueling process at a combined facility that takes longer than the loading time is modeled by a visit to a refueling facility in $\mathcal{F}_F$ with the same location as the combined facility. Each facility is assumed to have an unlimited fuel and cargo capacity, respectively, and can be simultaneously used

by any number of vehicles. This assumption seems adequate for many real-world scenarios but clearly represents a simplification for scenarios in which capacity and loading possibilities are constrained.

To represent working hour restrictions of real-world applications, we assume that the arrival time of all vehicles at depot instance $n+1$ may not exceed the maximum route duration t^{max}. The following variables are used in the model: a_j specifies the time on arrival at vertex j, f_j the fuel level, and l_j the load level. Binary decision variables x_{ij} take value 1 if vertex j is visited after vertex i and 0 otherwise. Thus, the mixed-integer program of VRPIS is as follows (Table 3.1 summarizes the notation):

Sets	
$0, n+1$	instances of the depot
$\mathcal{C}$	set of customers $= \{1, ..., n\}$
$\mathcal{F}'_F$	set of visits to refueling facilities
$\mathcal{F}'_L$	set of visits to loading facilities
$\mathcal{F}'_{FL}$	set of visits to combined facilities
$\mathcal{F}'$	set of visits to all intermediate facilities, $\mathcal{F}' = \mathcal{F}'_F \cup \mathcal{F}'_L \cup \mathcal{F}'_{FL}$
$\mathcal{V}'$	set of all customers and visits to facilities $\mathcal{C} \cup \mathcal{F}'$
$\mathcal{V}'_{0,n+1}$	set of all vertices $\mathcal{V}'_{0,n+1} = \mathcal{V}' \cup \{0\} \cup \{n+1\}$
$\mathcal{V}'_0$	set of all vertices excluding depot instance $n+1$, $\mathcal{V}'_0 = \{0\} \cup \mathcal{V}'$
$\mathcal{V}'_{n+1}$	set of all vertices excluding depot instance 0, $\mathcal{V}'_{n+1} = \{n+1\} \cup \mathcal{V}'$
$\mathcal{A}$	set of arcs, $\mathcal{A} = \{(i,j) : i,j \in \mathcal{V}'_{0,n+1}, i \neq j\}$
Parameters	
c^{fix}	fixed cost per used vehicle
c_{ij}	travel costs between vertices i and j
d_{ij}	distance between vertices i and j
u_i	demand of customer i ($u_i = 0$ if $i \notin \mathcal{C}$)
m	number of available vehicles
p	maximal fuel capacity of a vehicle expressed as possible range without refueling
q	maximal loading capacity of a vehicle
t_{ij}	travel time between vertices i and j
t^{max}	maximum route duration
t^d_i	docking time at intermediate facility i
t^s_i	service time at customer i ($t^s_i = 0$ if $i \notin \mathcal{C}$)
Functions	
$\Theta(l_i)$	function returning the amount that is refueled at vertex i during the loading process, given the current load level l_i
$\Phi^f(f_i)$	function returning the refueling time at a refueling facility depending on fuel level f_i
$\Phi^l(l_i)$	function returning the loading time at a loading facility depending on the current load level l_i
Decision variables	
a_i	decision variable specifying the arrival time at vertex i
f_i	decision variable specifying the fuel level at vertex i expressed in distance units
l_i	decision variable specifying the load level at vertex i
x_{ij}	binary decision variable indicating if arc $(i,j) \in \mathcal{A}$ is traversed

Table 3.1.: Sets, parameters, functions, and decision variables of the VRPIS model.

$$\min \sum_{i \in \mathcal{V}'_0} \sum_{j \in \mathcal{V}'_{n+1}, i \neq j} c_{ij} x_{ij} + \sum_{j \in \mathcal{V}'} c^{fix} x_{0j} \tag{3.1}$$

$$\sum_{i \in \mathcal{V}'_0, i \neq j} x_{ij} = 1 \qquad \forall j \in \mathcal{C} \tag{3.2}$$

$$\sum_{i \in \mathcal{V}'_0, i \neq j} x_{ij} \leqslant 1 \qquad \forall j \in \mathcal{F}' \tag{3.3}$$

$$\sum_{j \in \mathcal{V}'} x_{0j} \leqslant m \tag{3.4}$$

$$\sum_{i \in \mathcal{V}'_0, i \neq j} x_{ij} - \sum_{i \in \mathcal{V}'_{n+1}, i \neq j} x_{ji} = 0 \qquad \forall j \in \mathcal{V}' \tag{3.5}$$

$$0 \leqslant a_i \leqslant t^{max} \qquad \forall i \in \mathcal{V}'_{0,n+1} \tag{3.6}$$

$$a_i + (t_{ij} + t_i^s)x_{ij} - t^{max}(1 - x_{ij}) \leqslant a_j \qquad \forall i \in \mathcal{C} \cup \{0\}, j \in \mathcal{V}'_{n+1}, i \neq j \tag{3.7}$$

$$a_i + (t_{ij} + t_i^d)x_{ij} + \Phi^f(f_i) - t^{max}(1 - x_{ij}) \leqslant a_j \qquad \forall i \in \mathcal{F}'_F, j \in \mathcal{V}'_{n+1}, i \neq j \tag{3.8}$$

$$a_i + (t_{ij} + t_i^d)x_{ij} + \Phi^l(l_i) - t^{max}(1 - x_{ij}) \leqslant a_j \qquad \forall i \in \mathcal{F}'_L \cup \mathcal{F}'_{FL}, j \in \mathcal{V}'_{n+1}, i \neq j \tag{3.9}$$

$$0 \leqslant f_j \leqslant f_i - d_{ij}x_{ij} + p(1 - x_{ij}) \qquad \forall i \in \mathcal{C} \cup \mathcal{F}'_L, j \in \mathcal{V}'_{n+1}, i \neq j \tag{3.10}$$

$$0 \leqslant f_j \leqslant p - d_{ij}x_{ij} \qquad \forall i \in \mathcal{F}'_F \cup \{0\}, j \in \mathcal{V}'_{n+1}, i \neq j \tag{3.11}$$

$$0 \leqslant f_j \leqslant f_i + \Theta(l_i) - d_{ij}x_{ij} + p(1 - x_{ij}) \qquad \forall i \in \mathcal{F}'_{FL}, j \in \mathcal{V}'_{n+1}, i \neq j \tag{3.12}$$

$$f_i + \Theta(l_i) \leqslant p \qquad \forall i \in \mathcal{F}'_{FL} \tag{3.13}$$

$$0 \leqslant l_j \leqslant l_i - u_i x_{ij} + q(1 - x_{ij}) \qquad \forall i \in \mathcal{C} \cup \mathcal{F}'_F, j \in \mathcal{V}'_{n+1}, i \neq j \tag{3.14}$$

$$0 \leqslant l_j \leqslant q - u_i x_{ij} \qquad \forall i \in \{0\} \cup \mathcal{F}'_L \cup \mathcal{F}'_{FL}, j \in \mathcal{V}'_{n+1}, i \neq j \tag{3.15}$$

$$x_{ij} \in \{0, 1\} \qquad \forall i \in \mathcal{V}'_0, j \in \mathcal{V}'_{n+1}, i \neq j \tag{3.16}$$

The goal of the VRPIS is to minimize the sum of the total travel cost and the fixed vehicle cost, expressed by the objective function (3.1). Constraints (3.2) ensure that every customer must be visited, while optional intermediate stops are ensured by Constraints (3.3). Constraints (3.4) guarantee that the number of routes does not exceed the number of available vehicles. Flow conservation is given by Constraints (3.5). Constraints (3.6) limit the arrival time at each vertex to the maximum route duration. Time feasibility for arcs leaving customers or the depot is defined by Constraints (3.7). The same is ensured for arcs leaving refueling facilities and loading facilities in Constraints (3.8) and (3.9). Constraints (3.10) control the fuel feasibility for arcs leaving customers or loading facilities and Constraints (3.11) guarantee that a refueling facility is left in a completely refueled state. The fuel increase during loading at combined facilities is defined in Constraints (3.12). Constraints (3.13) guarantee that no refueling beyond the maximal fuel capacity is possible at combined facilities. Constraints (3.14) control the load feasibility for arcs leaving customers or refueling facilities. Constraints (3.15) ensure that vehicles leave loading facilities and the depot in a fully loaded state. Binary decision variables are defined in Constraints (3.16).

3.3.2. New Special VRPIS Case: The Electric Vehicle-Routing Problem with Recharging Facilities

As described above, we investigate the EVRPRF as a special case of the VRPIS. The VRPIS model is transformed into a formulation of the EVRPRF as follows:

- No intermediate cargo loading takes place. Therefore, no loading facilities are present in the EVRPRF and sets $\mathcal{F}_L$ and $\mathcal{F}_{FL}$ are empty and Constraints (3.9), (3.12), and (3.13) can be

neglected. To allow for recharging at the depot, dummy instances of 0 are now contained in the set $\mathcal{F}'_F$.

- We assume a linear recharging process of the vehicle battery, depending on a given average recharging speed g. Since the fuel capacity is expressed as the maximum travel range without refueling, g describes the increase of range per time unit. Thus, the function for the refueling time is defined as: $\Phi^f(f_i) = \frac{p-f_i}{g}$.

3.3.3. Special Cases from the Literature: G-VRP and VRPIRF

To assess the performance of our AVNS, we conduct tests on instances of the special VRPIS cases G-VRP and VRPIRF and compare our results to those presented in the literature (see Section 3.5).

G-VRP can be addressed as special case of VRPIS as follows:

- As no loading at intermediate facilities is considered in G-VRP, all related constraints of VRPIS are omitted.
- c^{fix} is set to zero because no vehicle cost are considered in G-VRP.
- The refueling time $\Phi^f(f_i)$ is set to zero and docking time t_i^d is set to the fixed service time of the G-VRP instances.

VRPIRF can be addressed as special case of VRPIS as follows:

- Since the VRPIRF instances do not consider refueling possibilities, all refueling related constraints of VRPIS are omitted.
- Loading time $\Phi^l(l_i)$ is set to zero because only a fixed docking time t_i^d occurs when visiting a loading facility.
- The maximum route duration t^{max} is reduced by t_i^d in order to account for a docking operation at the depot at the end of a route, which is considered in the VRPIRF.
- No vehicle deployment costs are considered in the VRPIRF, so c^{fix} is set to zero.

Table 3.2 clarifies the relation between VRPIS and the special cases considered in this chapter by comparing the properties of each problem.

3.4. An Adaptive Variable Neighborhood Search Algorithm for the VRPIS

In this section, we describe in detail our AVNS algorithm for solving VRPIS. AVNS follows the VNS diversification paradigm of searching in increasingly large neighborhoods (for a detailed introduction

	EVRPRF	VRPIRF	G-VRP	**VRPIS**
fixed vehicle cost	✓			✓
refueling possible	✓		✓	✓
loading possible		✓		✓
fuel/load dependent service times	✓			✓
fixed docking time	✓	✓	✓	✓

Table 3.2.: Relation between VRPIS, the introduced special case EVRPRF as well as the special cases from the literature VRPIRF and G-VRP.

to standard VNS, see Hansen and Mladenović, 2001). However, routes and vertices involved in the shaking step of AVNS are not selected entirely at random but are determined by problem-specific rules and the past search performance of these rules. AVNS has previously provided promising results for MDVRP, VRP with private fleet and common carriers (VRPPC), multi-depot VRPPC (Stenger et al., 2013) and the prize collecting VRP with non-linear cost (Stenger, Schneider, and Goeke, 2013).

The choice of AVNS is motivated by two factors. First, the high complexity of VRPIS makes it necessary to use an algorithm with strong diversification possibilities. Pretests have shown that classical local-search-based algorithms, like TS, often get stuck in local traps from which they are not able to escape. By contrast, the shaking step of our AVNS modifies up to four routes and moves sequences of up to six vertices in one iteration, which proved to be of vital importance to find promising solutions. Moreover, VNS algorithms presented in the literature have previously shown convincing performance on VRPs with intermediate stops (see, e.g., Tarantilis, Zachariadis, and Kiranoudis, 2008). Second, to ensure acceptable runtimes on large problem instances, a high efficiency of the search is required. The adaptive mechanism of AVNS takes into account the problem-specific characteristics of VRPIS and adapts based on the recent search performance. Thus, it efficiently guides the search to improving solutions. To sum up, combining the strong diversification of VNS with an adaptive mechanism results in a highly efficient heuristic, characterized by short computing times and high-quality results.

A pseudocode overview of the AVNS is given in Figure 3.1. First, the set of neighborhood structures $\{\mathcal{N}_\kappa \mid \kappa = 1, ..., \kappa^{max}\}$ is defined. Next, an initial solution $\mathcal{S}$ is constructed by means of a modified version of the savings algorithm by Clarke and Wright (1964), which considers the insertion of intermediate stops (Section 3.4.1), and the solution is subsequently improved by a local search (see Section 3.4.3).

In the AVNS component, a guided shaking step is used to diversify the search, producing a random solution $\mathcal{S}'$ within the κ-th neighborhood of $\mathcal{S}$ (Section 3.4.2.1). The adaptive mechanism is characterized by problem-specific selection methods for the routes and vertices to be shaken instead of an entirely random selection. Besides methods which have proven their effectiveness in previous works, we design specific methods which take the characteristics of intermediate stops into account (Section 3.4.2.2). Each of the selection methods is chosen according to a probability, which is dynamically updated during the search depending on the performance of the method (Section 3.4.2.3).

Subsequently, a greedy local search procedure is applied to obtain the local optimum $\mathcal{S}''$ (Section 3.4.3). In this step, classical operators as well as operators which are able to rearrange intermediate stops are used. If $\mathcal{S}''$ is accepted, it replaces $\mathcal{S}$ and κ is reset to one. Otherwise, $\mathcal{S}''$ is discarded and κ is increased by one, i.e., the next neighborhood is selected. We reset to the overall best solution after a certain number of iterations without improvement. The search is stopped after a given number of iterations without improvement of the best solution.

As described above, we adapt the algorithmic framework of AVNS to the specifics of VRPIS by incorporating problem-specific knowledge into the selection methods of our adaptive component and into the local search component. Numerical tests have proven the positive effects of these novel methods on the solution quality and run-time of our algorithm (see Section 3.5.3.3 for details).

Define the neighborhood structures $\mathcal{N}_\kappa$ with $\kappa = 1, ..., \kappa^{max}$
Generate initial solution $\mathcal{S}$
Improve initial solution by local search
$\kappa \leftarrow 1$
repeat
  {*Adaptive Shaking*}
  Select route and vertex selection method and generate $\mathcal{S}' \in \mathcal{N}_\kappa(\mathcal{S})$
  {*Local Search*}
  $\mathcal{S}'' \leftarrow$ localSearch($\mathcal{S}'$)
  {*Acceptance Decision*}
  if accept($\mathcal{S}''$) **then**
    $\mathcal{S} \leftarrow \mathcal{S}''$
    $\kappa \leftarrow 1$
  else
    $\kappa \leftarrow \kappa \mod \kappa^{max} + 1$
  end if
  Update weights of route and vertex selection methods
until given number of iterations without improvement reached

Figure 3.1.: Pseudocode of the AVNS heuristic for solving VRPIS.

3.4.1. Initialization with Modified Savings Algorithm

A modification of the savings algorithm, introduced by Clarke and Wright (1964), is used to quickly generate initial vehicle routes that include intermediate stops. We allow the initial solution to be infeasible with respect to fuel, load, or duration constraints. The steps of our modified savings algorithm are the following:

1. Generate back-and-forth tours for all customers. If such a tour is already infeasible concerning fuel, perform the cost-optimal insertion of a refueling facility into the respective route (see Section 3.4.3 for details).
2. Evaluate potential cost savings for merging each pair of routes and sort the merge moves in decreasing order.
3. Out of the remaining merge moves, select the two routes with highest cost savings and merge them if the maximum route duration is not exceeded. If no merge with positive cost savings exists, stop.
4. Evaluate the resulting route:
   a) If fuel or load violations emerge in the resulting route, try to resolve them by adding intermediate facilities at the optimal position.
   b) If the facility insertion leads to a duration violation, cancel the previous merging and continue with Step 3.
   c) If the resulting route starts or ends with an intermediate facility, i.e., no merging according to customer-related cost savings can be performed at this position, try to connect the facility with one of the remaining routes such that the cost increase is minimized and all constraints are still met (see Figure 3.2).
5. Continue with Step 3

The resulting number of routes may exceed the number of available vehicles. In this case, the route with the smallest cumulated customer demand is dissolved and its customers are inserted into the remaining routes at the cost-optimal position. Load capacity, fuel capacity, and duration violations are handled by means of a penalizing cost function, see Section 3.4.4. The process of dissolving routes is repeated until the required number of vehicles is reached. Subsequently, the solution is improved by a local search step (see Section 3.4.3).

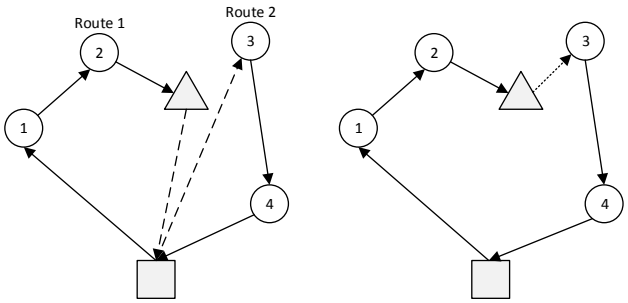


Figure 3.2.: Merging of routes that start or end with an intermediate facility. Removed arcs are shown with dashed lines, inserted arcs with dotted lines.

3.4.2. The Adaptive Shaking

In the shaking step of our AVNS, new solutions are generated according to predefined neighborhood structures (Section 3.4.2.1). Problem-specific methods are used for the selection of the routes and vertices to be involved in the shaking (Section 3.4.2.2). The algorithm guides the shaking step by adapting the selection probabilities of these methods according to their previous performance during the search (Section 3.4.2.3).

3.4.2.1. Shaking Neighborhoods

Similar to Stenger et al. (2013), two operators are employed in order to generate neighboring solutions: A sequence relocation and a cyclic exchange operator, originally introduced by Thompson and Orlin (1989). The cyclic exchange moves vertices between routes in a cyclic fashion. It is characterized by two parameters: the number of routes involved Ω and the maximum number of vertices to be exchanged Γ^{max}.

For each route k, the vertex sequence $\Psi^{k}_{j_k,\Gamma_k}$ with start vertex j_k and length Γ_k is transferred to route $k+1$ at the former position of sequence $\Psi^{k+1}_{j_{k+1},\Gamma_{k+1}}$. In Figure 3.3, the cyclic exchange operator is depicted with $\Omega = 3$ routes, exchanging $\Gamma_1 = 1$, $\Gamma_2 = 2$, and $\Gamma_3 = 2$ vertices. Note that if the total number of existing routes gets below the number of routes to cycle, Ω is reduced accordingly. Similarly, Γ_k has to be adjusted if it exceeds the number of vertices of route k, denoted with $|\mathcal{V}_k|$.

Sequence relocation represents a restriction of the cyclic exchange operator. A vertex sequence is relocated from one route to another, and the latter keeps all of its former vertices. Thus, $\Gamma^{max} = 0$ applies for the second route.

Table 3.3 shows the neighborhood structures employed within the search. After six sequence relocation neighborhoods, the search continues with 18 cyclic exchange neighborhoods, considering $\Omega = 2$ to $\Omega = 4$ routes between which up to $\Gamma^{max} = 6$ vertices can be transferred. Sequence lengths with up to $\Gamma^{max} = 4$ vertices are randomly chosen within the interval $[0, \min(\Gamma^{max}, |\mathcal{V}_k|)]$. Sequence lengths with more than four customers are defined to be fixed.

3.4.2.2. Selection Methods

Instead of determining the routes and vertices to be involved in the shaking entirely at random, the AVNS algorithm guides the shaking step to a certain extent. For this purpose, several methods are

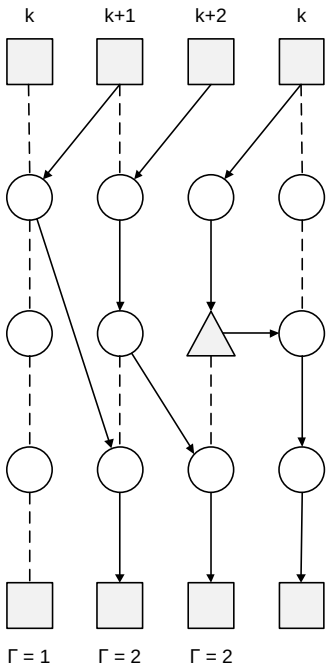


Figure 3.3.: Example of a cyclic exchange with three routes.

κ	Type	Ω	Γ^{max}	κ	Type	Ω	Γ^{max}
1	sequence relocation	2	1	13	cyclic exchange	3	1
2	sequence relocation	2	2	14	cyclic exchange	3	2
3	sequence relocation	2	3	15	cyclic exchange	3	3
4	sequence relocation	2	4	16	cyclic exchange	3	4
5	sequence relocation	2	5	17	cyclic exchange	3	5
6	sequence relocation	2	6	18	cyclic exchange	3	6
7	cyclic exchange	2	1	19	cyclic exchange	4	1
8	cyclic exchange	2	2	20	cyclic exchange	4	2
9	cyclic exchange	2	3	21	cyclic exchange	4	3
10	cyclic exchange	2	4	22	cyclic exchange	4	4
11	cyclic exchange	2	5	23	cyclic exchange	4	5
12	cyclic exchange	2	6	24	cyclic exchange	4	6

Table 3.3.: Neighborhood structures examined within the shaking step of the AVNS.

implemented which bias the route and vertex sequence selection. On the one hand, we use methods which have proven their effectiveness in previous works on routing problems. On the other hand, we design problem-specific methods which take into account the new characteristics of intermediate stops, e.g., the associated detours required. Each of the methods is chosen with a certain probability, which is dynamically updated during the search depending on the success of the method in former iterations. The selection methods and the adaptive mechanism are detailed in the following.

Route Selection The first of the Ω_κ routes of the current neighborhood $\mathcal{N}_\kappa$ is chosen according to one of the following five route selection methods:

1. *Random*: The probability of being selected is equal for every route.
2. *Route length*: The probability of a route for being selected is proportional to the associated travel distance. The intention is to remove vertices from long routes and reinsert them into shorter routes in order to reduce the total costs.
3. *Route length per demand unit*: The selection probability of a route is proportional to the relation of the total distance and the cumulated demand of a route. This criterion shall lead to an improvement of inefficient routes.
4. *Facility density*: The probability is proportional to the ratio of the number of intermediate stops to the number of customers within a route. The goal is to favor routes that possibly contain redundant facility visits.
5. *Facility detour*: The probability is proportional to the total detour resulting from intermediate stops. This is intended to reduce the associated detours and thus the overall costs.

After choosing the first route by means of one of the procedures above, the other routes to be involved in the shaking are iteratively determined as follows: The next route is randomly chosen from the set of all remaining routes that are spatially closer than a predefined threshold d^{max} to the previously selected route (cp. Stenger et al., 2013).

Vertex Sequence Selection Once the routes to be involved are determined, the vertex sequences to be removed from each route must be identified. The following three methods are used for this selection decision:

1. *Random*: Each vertex sequence is chosen with the same probability.
2. *Distance to next route*: The probability of selecting a vertex sequence is inversely proportional to the distance of the sequence to the route into which it will be inserted. This is measured by the sum of the vertex distances to the center of gravity of the target route.
3. *Distance to neighboring vertices*: The probability of selecting a sequence is proportional to the distance of the sequence to the surrounding vertices. It is given by the sum of the distance between the first vertex and its predecessor and the distance between the last vertex and its successor. Removing a sequence which is far apart from the other vertices of the route can reduce the total costs.

3.4.2.3. Adaptive Mechanism

At each shaking step, the choice of the route and vertex selection methods is based on probabilities. Each method is assigned the same probability at the beginning of the search. The probability of each

method is then dynamically updated in the course of the search depending on its success in improving the current solution. To select the methods, we use the roulette-wheel selection procedure as proposed by Pisinger and Ropke (2007) for ALNS. Given h selection methods, each method s is assigned a weight w_s. The probability of selecting method s is then defined by $w_s / \sum_{i=1}^{h} w_i$. After γ AVNS iterations, the weight of each method is updated based on its success during these iterations. The performance of a method is measured by a scoring system. A score of nine is added to the total score of a method whenever it achieved a new overall best solution, a score of three if the current solution was improved and a score of one if the solution is worse than the current one but accepted according to the acceptance criterion. If $\emptyset_s$ denotes the current score of method s and χ_s the number of applications of the method since the last weight update, then the new weight is calculated as $w_s = w_s(1-\rho) + \rho\frac{\emptyset_s}{\chi_s}$. The system parameter $\rho \in [0,1]$ allows to control to what extent the past value of the weight influences the new one. The values $\emptyset_s$ and χ_s are reset to zero after each update.

3.4.3. Local Search

The solution generated within the shaking step is subsequently improved by several greedy local search procedures. All operators are implemented such that the first improving move is accepted.

First, potential fuel or load violations within a route are handled by adding visits to intermediate facilities. If the distance between two consecutive refueling facility visits exceeds the fuel capacity of the vehicle, the fuel level drops below zero at a certain point. Hence, at least one refueling facility must be visited before this point. Let ϕ denote the position of the last visited refueling facility and σ that of the last vertex reachable from there. The best insertion position is therefore determined within the path $\{\phi+1, ..., \sigma+1\}$. For each possible position, the cost for inserting the closest refueling facility $i \in \mathcal{F}_F$ is calculated. The insertion with the lowest cost increase is performed, but in this step, insertions leading to feasible solutions are always preferred to infeasible solutions. The insertion of loading facilities is carried out in analogous fashion.

In a second step, we aim to improve the routing by means of the following operators, which are applied in random order. The 2-opt operator replaces two edges by two new ones (Lin, 1965). A restricted variant of the Or-Opt exchange (Or, 1976) replaces three existing edges by three new ones such that a sequence of three vertices is relocated (Stenger et al., 2013). The intra-route relocate operator moves a customer to a different position within a route (Savelsbergh, 1992). This operator is also defined for moving facilities. Finally, a facility replacement operator evaluates for each facility visit of each route whether replacing the facility visit with a visit to a different facility decreases the routing costs.

This block is followed by an application of a facility removal operator, which aims at removing redundant facility visits. In a final step, we apply two inter-route operators. The inter-route relocate operator moves a customer from its current route to another, and the exchange operator interchanges two customers between two routes (Savelsbergh, 1992).

3.4.4. Penalty Determination

Tightly constrained problems often let the local search get stuck in local optima quickly. It is therefore reasonable to temporarily allow constraint violations and impose penalty costs on infeasible solutions (see, e.g., Cordeau, Gendreau, and Laporte, 1997; Vidal et al., 2012). We define the total penalty

costs of a solution as $Cost^{penalty} = \delta^C \cdot \upsilon^C + \delta^D \cdot \upsilon^D + \delta^U \cdot \upsilon^U$, with δ^C denoting the penalty factor for capacity violations, υ^C the capacity violation of the solution, δ^D the duration penalty factor, υ^D the duration violation of the solution, δ^U the fuel penalty factor, and υ^U the fuel violation of the solution.

All penalty factors are initialized to δ^0 and dynamically varied within the interval $[\delta^{min}, \delta^{max}]$. After a given number of local search iterations η^+ with a violation of the respective constraint, the penalty factor is increased by factor δ^{update}. Analogously, after η^- feasible iterations, the penalty factor is reduced by factor δ^{update}. Preliminary tests showed that choosing different values for η^+ and η^- limits cycling of the local search, especially in small-sized problems.

3.4.5. Acceptance Decision

The solution $\mathcal{S}''$ obtained by the local search procedure is compared to the current solution $\mathcal{S}$. If $\mathcal{S}''$ is accepted, it replaces $\mathcal{S}$ as initial solution and κ is reset to one. Standard VNS implementations usually model the local search as a simple descent step, i.e., $\mathcal{S}''$ is only accepted if it is improving on $\mathcal{S}$. We use a criterion inspired by simulated annealing (SA) to control solution acceptance. This approach was originally proposed by Hemmelmayr, Doerner, and Hartl (2009) and also applied in Stenger et al. (2013).

Improving solutions are always accepted, while non-improving ones are accepted with probability

$$e^{\frac{-(f(\mathcal{S}'')-f(\mathcal{S}))}{\vartheta}}.$$

The temperature parameter ϑ is decreased from its initial value ϑ^0 by factor ϑ^- after every AVNS iteration. After ϵ non-improving main iterations, the current solution is reset to the best solution found so far. Solution diversification is increased by resetting ϑ to ϑ^0 after ϵ solution resets.

3.5. Computational Studies

This section presents the computational studies to examine the effectiveness of the AVNS. We perform tests on available instances developed for the routing problems G-VRP (Erdoğan and Miller-Hooks, 2012) and VRPIRF (Crevier, Cordeau, and Laporte, 2007; Tarantilis, Zachariadis, and Kiranoudis, 2008), which are both special cases of VRPIS. In addition, we design two sets of benchmark instances for the EVRPRF introduced in Section 3.3.2. A set of small instances is used to assess the quality of our solutions by comparing them to the solutions obtained with the commercial solver CPLEX. A set of large instances is used to prove the ability of our algorithm to deal with realistically-sized problems in terms of computational effort. Detailed results are provided to enable a comparison with future methods developed for the EVRPRF. To the best of our knowledge, our numerical studies cover all special cases of the VRPIS investigated in the literature.

Section 3.5.1 describes the test environment and the parameter setting. The computational results obtained on the special cases of the VRPIS from the literature are presented in Section 3.5.2. Section 3.5.3 details the generation of EVRPRF instances and the results obtained on this benchmark.

3.5.1. Computational Environment and Parameter Setting

The AVNS is implemented as single-thread code in Java. Tests are conducted on a desktop computer with an Intel Core i5 2.67 GHz processor with 4 GB RAM, running Windows 7 Professional. All numerical tests are carried out with the same parameter setting, which was determined during the development and testing of our algorithm.

To determine this parameter setting, we follow the approach described in Ropke and Pisinger (2006b). As test instances, we selected a reasonably large subset of the test instances of all VRPIS special cases. Then, we use the parameter setting that we have found during the development of our algorithm as basis for the tuning. Here, we stepwise refine the value of each parameter. In detail, we adjust the value of a single parameter while all remaining parameters are fixed. With every parameter setting, we perform 20 runs on the selected subset of test instances. The setting which produces the best average result is kept and the procedure is repeated with the next parameter. The resulting parameter setting is reported in Table 3.4.

In detail, the table shows the setting for the number of iterations after which the probabilities are updated (γ), the parameter ρ, which weighs the old weight and the new scores in the weight update of the selection methods within the adaptive mechanism, the initial (δ^0), minimal (δ^{min}), and maximal (δ^{max}) penalty factors, the penalty update factor (δ^{update}), the numbers of iterations after which the penalty costs are decreased (η^-) and increased (η^+), the initial temperature (ϑ^0), the temperature reduction factor (ϑ^-), and the number of resets of the current solution to the best solution found after which the temperature is reset to its initial value (ϵ).

AVNS		**Penalties**		**SA**	
γ	30	δ^0	1000	ϑ^0	50
ρ	0.3	δ^{min}	10	ϑ^-	0.9995
		δ^{max}	10000	ϵ	4
		δ^{update}	1.5		
		η^-	2		
		η^+	3		

Table 3.4.: Overview of the final parameter setting of AVNS chosen for the numerical studies.

In order to achieve reasonable runtimes on the investigated test instances, we set the maximum number of iterations without improvement (ω) and the number of non-improving iterations after which the current solution is reset to the best solution found (μ) as follows: For EVRPRF, we set ω equal to 2000 and μ to 50, for VRPIRF, we use $\omega = 500$ and $\mu = 25$. Additional tests on VRPIRF showed that a higher iteration number does not significantly improve the solution quality.

3.5.2. Experiments on Problems with Intermediate Stops from the Literature

To assess the performance of our AVNS, we conduct tests on instances of the special VRPIS cases G-VRP and VRPIRF and compare our results to those presented in the literature.

3.5.2.1. Green-VRP

The benchmark instances designed for the G-VRP (see Section 3.2) consist of five sets. Four sets contain ten instances each comprising 20 customers (which are either uniformly distributed or clustered) and between two and ten refueling facilities. The fifth set represents a case study conducted

by the authors and consists of twelve instances involving between 111 and 500 customers and 21 to 28 facilities. Note that customers that either cannot be served within the maximum route duration or whose service requires visiting more than one refueling stop must be identified and removed from the test instances. The geographical coordinates given in the instances have to be converted to distances between vertices by means of the Haversine formula using an average earth radius of 4182.45 miles.

Tables 3.5 and 3.6 show the results of AVNS on the small and the large G-VRP instances, respectively. We compare our results to those of the MCWS and DBCA heuristics of Erdoğan and Miller-Hooks (2012) and the VNS/TS of Schneider, Stenger, and Goeke (2014). For each problem instance, we report the problem name and the best-known solution (BKS) provided by either Erdoğan and Miller-Hooks (2012) or Schneider, Stenger, and Goeke (2014). For the MCWS and DBCA of Erdoğan and Miller-Hooks (2012), we give only the result of the better of the two algorithms for each instance. It was originally determined as the best of multiple runs (f^b in the table), but the exact number of runs is not given in the paper. For the VNS/TS of Schneider, Stenger, and Goeke (2014) and the AVNS, f^b corresponds to the best solution found in ten runs. For all algorithms, we further provide the percentage gap of f^b to the BKS (Δ^b) and the number of served customers (n). For VNS/TS and AVNS, we also display the average computing time of ten runs (t^a) in minutes, for the algorithms of Erdoğan and Miller-Hooks (2012), no runtimes were reported. The run-times of VNS/TS and AVNS are directly comparable as both algorithms are coded in Java and executed on the same computer. Moreover, we report the average solution quality of the ten runs for the AVNS (f^a). Finally, averages of the runtimes and relative gaps to the BKS over the complete set of instances are given at the end of the table.

For the small G-VRP instances (Table 3.5), AVNS finds the BKS for all instances. Thus, the AVNS is able to clearly outperform the methods of Erdoğan and Miller-Hooks (2012), even if for each instance only the best results provided by either MCWS or DBCA are considered. For the comparison with the VNS/TS of Schneider, Stenger, and Goeke (2014), note that the large gap of -19.05% to the BKS for instance S2_10i6s is not meaningful because Schneider, Stenger, and Goeke (2014) identify one more customer to be reachable for this instance. However, even disregarding this instance, AVNS yields better solution quality than VNS/TS for one instance and is able to match the quality for all other instances. Moreover, compared to the VNS/TS approach, it runs nearly four times as fast on average. The results further prove the robustness of the developed algorithm: For the large majority of instances, the average solution quality of the ten runs f^a is equal to the quality of the best run f^b; for the remaining instances, the gap is quite small.

On the large-sized G-VRP instances (Table 3.6), the AVNS algorithm finds new BKS for all instances and achieves an average gap to the previous BKS of more than 1%. Moreover, the speed of the AVNS is remarkable, using approximately 4% of the runtime of the VNS/TS of Schneider, Stenger, and Goeke (2014).

3.5.2.2. VRP with Intermediate Replenishment Facilities

The VRPIRF (see Section 3.2) considers intermediate replenishment facilities for the goods to be delivered. We run tests on two instance sets. The set of Crevier, Cordeau, and Laporte (2007) comprises 22 instances consisting of 48-216 customers, three to six intermediate facilities, and four to six available vehicles. The customers are clustered around the facilities. The set of Tarantilis, Zachariadis,

		MCWS/DBCA			VNS/TS				**AVNS**				
Inst.	BKS	n	f^b	Δ^b(%)	n	f^b	Δ^b(%)	t^a(min)	n	f^b	Δ^b(%)	f^a	t^a(min)
20c3sU1	**1797.49**	20	1797.51	0.00	20	**1797.49**	0.00	0.69	20	**1797.49**	0.00	**1797.49**	0.16
20c3sU2	**1574.77**	20	1613.53	2.46	20	**1574.77**	0.00	0.64	20	**1574.78**	0.00	**1574.78**	0.15
20c3sU3	**1704.48**	20	1964.57	15.26	20	**1704.48**	0.00	0.64	20	**1704.48**	0.00	**1704.48**	0.13
20c3sU4	**1482.00**	20	1487.15	0.35	20	**1482.00**	0.00	0.65	20	**1482.00**	0.00	**1482.00**	0.17
20c3sU5	**1689.37**	20	1752.73	3.75	20	**1689.37**	0.00	0.67	20	**1689.37**	0.00	**1689.37**	0.18
20c3sU6	**1618.65**	20	1668.16	3.06	20	**1618.65**	0.00	0.67	20	**1618.65**	0.00	**1618.65**	0.15
20c3sU7	**1713.66**	20	1730.45	0.98	20	**1713.66**	0.00	0.64	20	**1713.66**	0.00	**1713.66**	0.19
20c3sU8	**1706.50**	20	1718.67	0.71	20	**1706.50**	0.00	0.67	20	**1706.50**	0.00	**1706.50**	0.16
20c3sU9	**1708.81**	20	1714.43	0.33	20	**1708.81**	0.00	0.66	20	**1708.82**	0.00	**1708.82**	0.19
20c3sU10	**1181.31**	20	1309.52	10.85	20	**1181.31**	0.00	0.64	20	**1181.31**	0.00	**1181.31**	0.23
20c3sC1	**1173.57**	20	1300.62	10.83	20	**1173.57**	0.00	0.62	20	**1173.57**	0.00	**1173.57**	0.38
20c3sC2	**1539.97**	19	1553.53	0.88	19	**1539.97**	0.00	0.58	19	**1539.97**	0.00	**1539.97**	0.21
20c3sC3	**880.20**	12	1083.12	23.05	12	**880.20**	0.00	0.25	12	**880.20**	0.00	**880.20**	0.15
20c3sC4	**1059.35**	18	1091.78	3.06	18	**1059.35**	0.00	0.53	18	**1059.35**	0.00	1077.71	0.23
20c3sC5	**2156.01**	19	2190.68	1.61	19	**2156.01**	0.00	0.60	19	**2156.01**	0.00	**2156.01**	0.14
20c3sC6	**2758.17**	17	2883.71	4.55	17	**2758.17**	0.00	0.71	17	**2758.17**	0.00	**2758.17**	0.14
20c3sC7	**1393.99**	6	1701.40	22.05	6	**1393.99**	0.00	0.18	6	**1393.99**	0.00	**1393.99**	0.04
20c3sC8	**3139.72**	18	3319.74	5.73	18	**3139.72**	0.00	0.62	18	**3139.72**	0.00	**3139.72**	0.08
20c3sC9	**1799.94**	19	1811.05	0.62	19	**1799.94**	0.00	0.60	19	**1799.94**	0.00	**1799.94**	0.16
20c3sC10	**2583.42**	15	2644.11	2.35	15	**2583.42**	0.00	0.45	15	**2583.42**	0.00	2600.39	0.09
S1_2i6s	**1578.12**	20	1614.15	2.28	20	**1578.12**	0.00	0.71	20	**1578.12**	0.00	**1578.12**	0.16
S1_4i6s	**1397.27**	20	1541.46	10.32	20	**1397.27**	0.00	0.75	20	**1397.27**	0.00	**1397.27**	0.16
S1_6i6s	**1560.49**	20	1616.20	3.57	20	**1560.49**	0.00	0.73	20	**1560.49**	0.00	**1560.49**	0.20
S1_8i6s	**1692.32**	20	1882.54	11.24	20	**1692.32**	0.00	0.74	20	**1692.32**	0.00	**1692.32**	0.17
S1_10i6s	**1173.48**	20	1309.52	11.59	20	**1173.48**	0.00	0.71	20	**1173.48**	0.00	**1173.48**	0.24
S2_2i6s	**1633.10**	20	1645.80	0.78	20	**1633.10**	0.00	0.75	20	**1633.10**	0.00	**1633.10**	0.19
S2_4i6s	**1505.06**	19	1505.06	0.00	19	1532.96	1.85	0.88	19	**1505.07**	0.00	**1505.07**	0.14
S2_6i6s	**2431.33**	20	3115.10	28.12	20	**2431.33**	0.00	0.78	20	**2431.33**	0.00	**2431.33**	0.13
S2_8i6s	**2158.35**	16	2722.55	26.14	16	**2158.35**	0.00	0.57	16	**2158.35**	0.00	**2158.35**	0.09
S2_10i6s	1958.46	16	1995.62	1.90	17	1958.46	0.00	0.61	16	1585.46	-19.05	1585.46	0.15
S1_4i2s	**1582.20**	20	1582.20	0.00	20	**1582.21**	0.00	0.63	20	**1582.21**	0.00	**1582.21**	0.13
S1_4i4s	**1460.09**	20	1580.52	8.25	20	**1460.09**	0.00	0.68	20	**1460.09**	0.00	**1460.09**	0.16
S1_4i6s	**1397.27**	20	1541.46	10.32	20	**1397.27**	0.00	0.75	20	**1397.27**	0.00	**1397.27**	0.16
S1_4i8s	**1397.27**	20	1561.29	11.74	20	**1397.27**	0.00	0.82	20	**1397.27**	0.00	**1397.27**	0.17
S1_4i10s	**1396.02**	20	1529.73	9.58	20	**1396.02**	0.00	0.85	20	**1396.02**	0.00	**1396.02**	0.23
S2_4i2s	**1059.35**	18	1117.32	5.47	18	**1059.35**	0.00	0.51	18	**1059.35**	0.00	1069.42	0.23
S2_4i4s	**1446.08**	19	1522.72	5.30	19	**1446.08**	0.00	0.60	19	**1446.08**	0.00	1449.17	0.21
S2_4i6s	**1434.14**	20	1730.47	20.66	20	**1434.14**	0.00	0.69	20	**1434.14**	0.00	1445.35	0.20
S2_4i8s	**1434.14**	20	1786.21	24.55	20	**1434.14**	0.00	0.75	20	**1434.14**	0.00	**1434.14**	0.20
S2_4i10s	**1434.13**	20	1729.51	20.60	20	**1434.13**	0.00	0.78	20	**1434.13**	0.00	1455.31	0.24
Avg.				**8.12**			**0.05**	**0.65**			**-0.48**		**0.17**

Table 3.5.: Results of AVNS on the small-sized G-VRP instances by Erdoğan and Miller-Hooks (2012). The results are compared to those of the MCWS/DCBA of Erdoğan and Miller-Hooks (2012) and the VNS/TS of Schneider, Stenger, and Goeke (2014). BKS denotes the previously best known solution. f^b denotes the best solution found (for VNS/TS and AVNS in ten runs), Δ^b the gap to the BKS in percent and n the number of served customers. For VNS/TS and AVNS, the average computing time in minutes is given (t^a). For AVNS, we additionally report the average solution quality of ten runs (f^a). Numbers in bold indicate the best solution found.

		MCWS/DBCA			VNS/TS				**AVNS**				
Inst.	BKS	n	f^b	Δ^b(%)	n	f^b	Δ^b(%)	t^a(min)	n	f^b	Δ^b(%)	f^a	t^a(min)
111c_21s	4797.15	109	5626.64	17.29	109	4797.15	0.00	21.76	109	**4770.47**	-0.56	4791.53	1.78
111c_22s	4802.16	109	5610.57	16.83	109	4802.16	0.00	23.56	109	**4776.81**	-0.53	4797.31	1.94
111c_24s	4786.96	109	5412.48	13.07	109	4786.96	0.00	21.90	109	**4767.14**	-0.41	4790.84	2.16
111c_26s	4778.62	109	5408.38	13.18	109	4778.62	0.00	25.12	109	**4767.14**	-0.24	4782.60	2.04
111c_28s	4799.15	109	5331.93	11.10	109	4799.15	0.00	24.17	109	**4765.52**	-0.70	4781.26	1.73
200c	8963.46	190	10413.59	16.18	192	8963.46	0.00	76.65	192	**8886.00**	-0.86	8970.14	3.61
250c	10800.18	235	11886.61	10.06	237	10800.18	0.00	120.90	237	**10487.15**	-2.90	10531.20	3.67
300c	12594.77	281	14229.92	12.98	283	12594.77	0.00	182.23	283	**12374.49**	-1.75	12514.78	4.94
350c	14323.02	329	16460.30	14.92	329	14323.02	0.00	232.03	329	**14103.66**	-1.53	14271.56	7.11
400c	16850.21	378	19099.04	13.35	378	16850.21	0.00	305.12	378	**16697.21**	-0.91	16839.23	12.70
450c	18521.23	424	21854.17	18.00	424	18521.23	0.00	525.52	424	**18310.60**	-1.14	18512.47	13.19
500c	21170.90	471	24517.08	15.81	471	21170.90	0.00	356.01	471	**20609.67**	-2.65	20874.50	19.51
Avg.				**14.40**			**0.00**	**159.58**			**-1.18**		**6.20**

Table 3.6.: Results of AVNS on the large-sized G-VRP instances by Erdoğan and Miller-Hooks (2012). The results are compared to those of the MCWS/DCBA of Erdoğan and Miller-Hooks (2012) and the VNS/TS of Schneider, Stenger, and Goeke (2014). BKS denotes the previously best-known solution. f^b denotes the best solution found (for VNS/TS and AVNS in ten runs), Δ^b the gap to the BKS and n the number of served customers. For VNS/TS and AVNS, the average computing time in minutes is given (t^a). For AVNS, we additionally report the average solution quality of ten runs (f^a). Numbers in bold indicate the best solution found.

and Kiranoudis (2008) contains 54 instances with 50-175 customers, three to eight facilities, and two to eight vehicles.

Tables 3.7 and 3.8 show the results of our AVNS on the instances of Crevier, Cordeau, and Laporte (2007), compared to the results of these authors (CCL), those of Tarantilis, Zachariadis, and Kiranoudis (2008) (TZK), of the VNS/TS of Schneider, Stenger, and Goeke (2014), and the VNS of Hemmelmayr, Doerner, Hartl, and Rath (2013) (HDHR). For each instance, we report the instance name and the previously known BKS as determined by the four comparison methods. In Table 3.7, we report for all algorithms the average solution quality of ten runs (f^a), the gap of the average solution to the BKS (Δ^a) and the average computing time in minutes (t^a). Finally, averages of the runtimes and the gaps to the BKS over the complete set of instances are given at the end of the table. Note that a direct comparison of run-times is only valid for AVNS and VNS/TS. The other methods are partly coded in different programming languages and were run on different platforms to obtain the reported computation times. The best solution quality obtained by any of the methods on each instance is indicated in bold.

In Table 3.8, we report the best solution found (f^b) and the gap of the best solution to the BKS (Δ^b). The best solution reported by Crevier, Cordeau, and Laporte (2007), Schneider, Stenger, and Goeke (2014), Hemmelmayr et al. (2013), and our AVNS are based on ten runs, those of Tarantilis, Zachariadis, and Kiranoudis (2008) are the best solutions ever found with the final parameter setting, which we indicated with an asterisk. Finally, we report for our AVNS the best solutions found during the overall testing in column $\overline{\text{AVNS}}$.

Concerning the entire instance set of Crevier, Cordeau, and Laporte (2007), a comparison of AVNS is only possible with CCL, VNS/TS, and HDHR because TZK only provide solutions for the first subset of the instances. AVNS is able to improve on the solution quality of CCL and VNS/TS concerning the best as well as the average quality. Moreover, the runtimes of AVNS are clearly faster than those of VNS/TS. AVNS is not able to match the solution quality of HDHR, which is superior to all comparison methods in terms of solution quality.

In Table 3.9, the results of AVNS on the test instances of Tarantilis, Zachariadis, and Kiranoudis

		CCL			TZK			VNS/TS			HDHR			**AVNS**		
Inst.	BKS	f^a	Δ^a(%)	t^a(min)	f^a	Δ^a(%)	t^a(min)	f^a	Δ^a(%)	t^a(min)	f^a	Δ^a(%)	t^a(min)	f^a	Δ^a(%)	t^a(min)
a1	**1179.79**	1211.28	2.67	4.58	1189.70	0.84	3.40	1197.59	1.51	1.82	1180.57	0.07	1.42	1184.57	0.41	0.64
b1	**1217.07**	1232.67	1.28	9.17	1225.08	0.66	7.80	1226.80	0.80	7.14	**1217.07**	0.00	6.39	1218.21	0.09	4.19
c1	**1866.76**	1893.01	1.41	36.22	1898.92	1.72	34.20	1925.21	3.13	33.93	1867.96	0.06	20.40	1925.41	3.14	32.98
d1	**1059.43**	1076.31	1.59	8.55	1064.29	0.46	5.90	1063.08	0.35	1.82	**1059.43**	0.00	1.57	1061.50	0.20	0.55
e1	**1309.12**	1311.60	0.19	13.52	1309.12	0.00	8.70	1343.90	2.66	7.29	**1309.12**	0.00	6.22	1312.75	0.28	5.08
f1	**1570.41**	1601.54	1.98	41.41	1585.83	0.98	38.80	1619.80	3.14	34.61	1573.05	0.17	25.60	1601.40	1.97	34.99
g1	**1181.13**	1202.00	1.77	55.22	1190.21	0.77	5.80	1190.72	0.81	4.21	1183.32	0.19	3.38	1183.75	0.22	1.69
h1	**1545.50**	1598.51	3.43	32.07	1577.54	2.07	11.10	1582.33	2.38	18.03	1548.61	0.20	14.61	1567.22	1.41	14.08
i1	**1922.18**	1976.11	2.81	51.01	1956.17	1.77	42.50	2004.35	4.27	45.62	1923.52	0.07	33.58	1974.97	2.75	35.11
j1	**1115.78**	1161.77	4.12	58.90	1128.86	1.17	5.50	1120.65	0.44	4.24	**1115.78**	0.00	2.78	1116.82	0.09	2.02
k1	**1576.36**	1618.45	2.67	64.61	1591.74	0.98	12.10	1601.67	1.61	18.11	1577.96	0.10	14.56	1600.42	1.53	10.74
l1	**1863.28**	1917.08	2.89	104.27	1904.39	2.21	51.40	1932.82	3.73	46.14	1869.70	0.34	35.48	1916.07	2.83	40.59
Avg.			**2.23**	**39.96**		**1.14**	**18.93**		**2.07**	**18.58**		**0.10**	**13.83**		**1.24**	**15.22**
a2	**997.94**	1005.16	0.72	6.40				1002.58	0.47	1.80	**997.94**	0.00	1.23	**997.94**	0.00	0.72
b2	**1291.19**	1333.20	3.25	14.70				1324.84	2.61	7.35	**1291.19**	0.00	6.41	1300.42	0.72	4.83
c2	**1715.60**	1792.46	4.48	61.70				1760.81	2.63	18.05	1715.84	0.01	15.01	1741.55	1.51	18.32
d2	**1856.84**	1898.21	2.23	40.50				1908.34	2.77	35.10	1860.92	0.22	30.14	1903.15	2.49	30.64
e2	**1919.38**	1995.75	3.98	73.80				1993.47	3.86	59.12	1922.81	0.18	49.31	1957.80	2.00	41.60
f2	**2230.32**	2312.15	3.67	162.20				2325.34	4.26	89.86	2233.43	0.14	71.24	2313.08	3.71	42.80
g2	**1152.92**	1185.93	2.86	29.50				1161.50	0.74	4.14	1153.17	0.02	3.71	1158.21	0.46	2.20
h2	**1575.28**	1611.75	2.32	160.80				1610.48	2.23	18.35	**1575.28**	0.00	15.66	1586.24	0.70	21.20
i2	**1919.74**	1998.20	4.09	322.40				1969.98	2.62	47.58	1922.24	0.13	41.92	1971.27	2.68	41.10
j2	**2247.70**	2325.18	3.45	256.90				2330.34	3.68	91.30	2250.21	0.11	73.38	2303.67	2.49	41.93
Avg.			**3.10**	**112.89**					**2.59**	**37.27**		**0.08**	**30.80**		**1.68**	**24.53**
Tot. Avg.			**2.63**	**73.11**					**2.30**	**27.07**		**0.09**	**21.55**		**1.44**	**19.46**

Table 3.7.: Average solution quality of AVNS on the VRPIRF instances of Crevier, Cordeau, and Laporte (2007). Results are compared to those of Crevier, Cordeau, and Laporte (2007) (CCL), Tarantilis, Zachariadis, and Kiranoudis (2008) (TZK), of the VNS/TS of Schneider, Stenger, and Goeke (2014), and the VNS of Hemmelmayr, Doerner, Hartl, and Rath (2013) (HDHR). f^a denotes the average solution found in ten runs, Δ^a the gap to BKS and t^a the average computing time in minutes. Numbers in bold indicate the best solution found.

		CCL		TZK		VNS/TS		HDHR		**AVNS**		$\overline{\textbf{AVNS}}$	
Inst.	BKS	f^b	Δ^b(%)	f^{b*}	Δ^{b*}(%)	f^b	Δ^b(%)	f^b	Δ^b(%)	f^b	Δ^b(%)	f	Δ(%)
a1	**1179.79**	1203.39	2.00	**1179.79**	0.00	**1179.79**	0.00	**1179.79**	0.00	**1179.79**	0.00	**1179.79**	0.00
b1	**1217.07**	**1217.07**	0.00	**1217.07**	0.00	**1217.07**	0.00	**1217.07**	0.00	**1217.07**	0.00	**1217.07**	0.00
c1	**1866.76**	1888.22	1.15	1883.05	0.87	1897.30	1.64	**1866.76**	0.00	1893.53	1.43	1882.46	0.84
d1	**1059.43**	**1059.43**	0.00	**1059.43**	0.00	1060.10	0.06	**1059.43**	0.00	**1059.43**	0.00	**1059.43**	0.00
e1	**1309.12**	**1309.12**	0.00	**1309.12**	0.00	**1309.12**	0.00	**1309.12**	0.00	**1309.12**	0.00	**1309.12**	0.00
f1	**1570.41**	1592.25	1.39	1572.17	0.11	1584.06	0.87	**1570.41**	0.00	1579.89	0.60	1577.63	0.46
g1	**1181.13**	1190.93	0.83	**1181.13**	0.00	1181.99	0.07	**1181.13**	0.00	**1181.13**	0.00	**1181.13**	0.00
h1	**1545.50**	1566.75	1.37	1547.24	0.11	1566.19	1.34	**1545.50**	0.00	1555.52	0.65	1553.75	0.53
i1	**1922.18**	1945.73	1.23	1925.99	0.20	1953.39	1.62	**1922.18**	0.00	1956.70	1.80	1934.08	0.62
j1	**1115.78**	1144.41	2.57	1117.20	0.13	**1115.78**	0.00	**1115.78**	0.00	**1115.78**	0.00	**1115.78**	0.00
k1	**1576.36**	1586.92	0.67	1580.39	0.26	1586.64	0.65	**1576.36**	0.00	1591.81	0.98	1577.98	0.10
l1	**1863.28**	1897.74	1.85	1880.60	0.93	1902.72	2.12	**1863.28**	0.00	1907.15	2.35	1894.69	1.69
Avg.			**1.09**		**0.22**		**0.70**		**0.00**		**0.65**		**0.35**
a2	**997.94**	1000.24	0.23			**997.94**	0.00	**997.94**	0.00	**997.94**	0.00	**997.94**	0.00
b2	**1291.19**	1307.28	1.25			1301.21	0.78	**1291.19**	0.00	**1291.19**	0.00	**1291.19**	0.00
c2	**1715.60**	1751.45	2.09			1732.19	0.97	**1715.60**	0.00	1730.14	0.85	**1715.60**	0.00
d2	**1856.84**	1877.03	1.09			1892.62	1.93	**1856.84**	0.00	1878.89	1.19	1874.12	0.93
e2	**1919.38**	1974.13	2.85			1940.52	1.10	**1919.38**	0.00	1943.61	1.26	1937.84	0.96
f2	**2230.32**	2298.51	3.06			2292.40	2.78	**2230.32**	0.00	2292.84	2.80	2268.54	1.71
g2	**1152.92**	1162.58	0.84			1158.21	0.46	**1152.92**	0.00	1158.21	0.46	**1152.92**	0.00
h2	**1575.28**	1593.40	1.15			1597.41	1.41	**1575.28**	0.00	1576.86	0.10	1576.86	0.10
i2	**1919.74**	1978.70	3.07			1934.09	0.75	**1919.74**	0.00	1945.24	1.33	1944.74	1.30
j2	**2247.70**	2303.01	2.46			2293.40	2.03	**2247.70**	0.00	2281.86	1.52	2281.86	1.52
Avg.			**1.81**				**1.22**		**0.00**		**0.95**		**0.65**
Tot. Avg.			**1.42**				**0.94**		**0.00**		**0.79**		**0.49**

Table 3.8.: Comparison of the results on the VRPIRF instances of Crevier, Cordeau, and Laporte (2007) to those of TZK, CCL, VNS/TS, and HDHR based on the best run. Note that the value given in column f^{b*} for TZK corresponds to the best solution ever found with the final parameter setting. For all other methods, f^b refers to the best out of ten runs. Δ^b denotes the gap to BKS. We also provide the best solutions found during our overall testing in column $\overline{\text{AVNS}}$. Numbers in bold indicate the best solution found.

(2008) are compared to those of TZK, the VNS/TS of Schneider, Stenger, and Goeke (2014), and HDHR. The reported measures are the same as in Tables 3.7 and 3.8. Concerning the average gap to the BKS, the AVNS is able to improve on the results of TZK and VNS/TS based on average as well as best solution quality. AVNS is not able to match the solution quality of HDHR, which again outperforms all other methods concerning solution quality. AVNS is able to provide two new BKS during the ten test runs and four new BKS during the overall testing. Run-times are observably faster than those of VNS/TS.

3.5.3. Experiments on EVRPRF Instances

In this section, we conduct numerical studies on EVRPRF instances. Section 3.5.3.1 describes the generation of the EVRPRF instances in more detail. Section 3.5.3.2 presents the computational results of our AVNS on the new instances.

3.5.3.1. Generation of EVRPRF Instances

Our EVRPRF instances are based on the benchmark instances for the CVRP introduced by Christofides and Eilon (1969) and Golden et al. (1998). In order to generate valid EVRPRF instances, the following adjustments are made: The service time t_i^s of each customer $i \in \mathcal{C}$ is set to ten time units. The battery capacity p of each vehicle is equal to the amount of electricity required to travel 60% of the average route length of a high-quality solution of the respective CVRP instance. The CVRP solutions are taken from the website http://neumann.hec.ca/chairedistributique/data/vrp/old/ for the instances of

		TZK				VNS/TS				HDHR				**AVNS**				$\overline{\textbf{AVNS}}$	
Inst.	BKS	f^{b}*	Δ^{b}*(%)	Δ^{a}(%)	t^{a}(min)	f^{b}	Δ^{b}(%)	Δ^{a}(%)	t^{a}(min)	f^{b}	Δ^{b}(%)	Δ^{a}(%)	t^{a}(min)	f^{b}	Δ^{b}(%)	Δ^{a}(%)	t^{a}(min)	f	Δ(%)
50c3d2v	**2209.83**	**2209.83**	0.00	2.27	2.85	**2209.83**	0.00	0.10	1.82	**2209.83**	0.00	0.00	6.62	**2209.83**	0.00	0.00	0.80	**2209.83**	0.00
50c3d4v	**2368.33**	**2368.33**	0.00	2.18	2.23	**2368.33**	0.00	0.89	1.84	**2368.33**	0.00	0.00	1.76	**2368.33**	0.00	0.00	0.83	**2368.33**	0.00
50c3d6v	**2999.29**	3000.88	0.05	2.18	2.74	**2999.29**	0.00	0.81	1.88	**2999.29**	0.00	0.00	1.15	**2999.29**	0.00	0.01	0.52	**2999.29**	0.00
50c5d2v	**2608.25**	**2608.25**	0.00	2.87	1.54	**2608.25**	0.00	1.01	2.00	**2608.25**	0.00	0.00	7.64	**2608.25**	0.00	0.00	0.73	**2608.25**	0.00
50c5d4v	**3086.58**	**3086.58**	0.00	1.23	2.07	**3086.58**	0.00	0.00	1.98	**3086.58**	0.00	0.00	2.04	**3086.58**	0.00	0.11	0.88	**3086.58**	0.00
50c5d6v	**3548.88**	3552.00	0.09	0.99	3.04	3552.00	0.09	0.37	2.02	**3548.88**	0.00	0.23	1.28	**3548.88**	0.00	0.27	0.41	**3548.88**	0.00
50c7d2v	**3353.08**	**3353.08**	0.00	2.37	3.16	**3353.83**	0.02	3.10	2.38	**3353.08**	0.00	0.00	14.11	3370.85	0.53	1.29	0.86	3370.85	0.53
50c7d4v	**3380.27**	3381.57	0.04	2.67	3.36	**3380.27**	0.00	0.58	2.10	**3380.27**	0.00	0.00	4.76	3398.28	0.53	0.93	1.25	3398.28	0.53
50c7d6v	**4074.44**	4097.80	0.57	0.83	3.42	**4074.44**	0.00	0.95	2.07	**4074.44**	0.00	0.36	1.50	**4074.44**	0.00	0.24	0.41	**4074.44**	0.00
75c3d2v	**2678.80**	**2678.80**	0.00	0.57	4.50	2692.76	0.52	1.68	4.67	**2678.80**	0.00	0.00	23.79	**2678.80**	0.00	0.53	2.89	**2678.8**	0.00
75c3d4v	**2746.74**	**2746.74**	0.00	1.95	3.38	**2746.74**	0.00	0.17	4.33	**2746.74**	0.00	0.00	7.13	**2746.74**	0.00	0.00	3.86	**2746.74**	0.00
75c3d6v	**3393.89**	3454.71	1.79	3.11	4.89	3448.64	1.61	2.21	4.34	**3393.89**	0.00	0.00	3.31	3419.49	0.75	1.33	1.97	3419.49	0.75
75c5d2v	**3373.69**	**3373.69**	0.00	2.98	3.29	3386.64	0.38	2.34	5.09	**3373.69**	0.00	0.00	29.45	3374.76	0.03	1.18	3.27	**3373.69**	0.00
75c5d4v	**3553.46**	3568.35	0.42	2.86	3.54	3569.82	0.46	1.05	4.42	**3553.46**	0.00	0.21	4.09	**3553.46**	0.00	1.16	2.62	**3553.46**	0.00
75c5d6v	**4184.65**	4198.61	0.33	2.00	4.18	4215.30	0.73	2.38	4.55	**4184.65**	0.00	0.00	3.17	4193.86	0.22	0.25	1.56	4193.86	0.22
75c7d2v	**3569.02**	**3569.02**	0.00	2.41	5.38	3581.32	0.34	1.63	5.06	**3569.02**	0.00	0.00	25.84	**3569.02**	0.00	0.24	2.47	**3569.02**	0.00
75c7d4v	**3822.10**	3830.43	0.22	2.35	5.51	3830.43	0.22	1.92	4.61	**3822.10**	0.00	1.07	5.43	**3822.10**	0.00	0.13	2.60	**3822.1**	0.00
75c7d6v	**4239.76**	**4239.76**	0.00	2.02	4.29	4244.35	0.11	0.75	4.80	4263.24	0.55	0.55	2.55	4263.24	0.55	0.97	1.54	**4239.76**	0.00
100c3d3v	**3123.51**	**3123.51**	0.00	1.10	7.01	3127.65	0.13	2.33	7.94	**3123.51**	0.00	0.00	14.66	3180.66	1.83	2.51	9.64	3176.39	1.69
100c3d5v	**3548.44**	3552.50	0.11	2.48	7.31	3548.75	0.01	0.30	7.62	**3548.75**	0.01	0.10	4.58	**3548.44**	0.00	0.06	4.18	**3548.44**	0.00
100c3d7v	**4235.31**	**4239.83**	0.11	0.93	6.62	4268.34	0.78	2.45	7.92	**4235.31**	0.00	0.28	4.03	4274.72	0.93	1.51	3.49	4274.72	0.93
100c5d3v	**4053.95**	**4053.95**	0.00	1.06	7.88	**4053.95**	0.00	1.58	8.49	**4053.95**	0.00	0.00	24.23	**4053.95**	0.00	0.57	10.59	**4053.95**	0.00
100c5d5v	**4413.17**	**4413.17**	0.00	2.69	7.20	4424.81	0.26	5.52	7.70	**4413.17**	0.00	0.00	3.74	**4413.17**	0.00	0.00	2.62	**4413.17**	0.00
100c5d7v	**5142.52**	5148.98	0.13	0.69	7.72	**5142.52**	0.00	0.28	7.93	**5142.52**	0.00	0.00	2.67	**5142.52**	0.00	0.15	2.09	**5142.52**	0.00
100c7d3v	**4207.79**	4216.47	0.21	0.82	8.53	4242.38	0.82	1.83	8.87	**4207.79**	0.00	0.00	30.27	**4207.79**	0.00	0.40	14.83	**4207.79**	0.00
100c7d5v	**4414.69**	4462.51	1.08	2.45	8.79	4448.15	0.76	1.76	8.00	**4414.69**	0.00	0.59	4.60	4440.36	0.58	1.17	5.12	4439.55	0.56
100c7d7v	**4847.79**	4897.47	1.02	2.59	8.35	4916.62	1.42	4.89	8.10	**4847.79**	0.00	0.42	3.33	4850.21	0.05	1.20	1.90	4850.21	0.05
125c4d3v	**3916.02**	3920.05	0.10	1.28	8.73	3966.61	1.29	3.73	13.23	**3916.02**	0.00	0.00	31.29	**3916.02**	0.00	0.75	11.13	**3916.02**	0.00
125c4d5v	**4308.44**	4315.68	0.17	1.47	9.00	**4308.44**	0.00	0.45	12.33	**4308.44**	0.00	0.00	7.62	**4308.44**	0.00	0.00	5.89	**4308.44**	0.00
125c4d7v	4668.77	4763.49	2.03	3.54	8.40	4694.32	0.55	2.61	12.54	4769.75	2.16	2.19	4.77	**4664.38**	-0.09	2.67	3.52	**4664.38**	-0.09
125c6d3v	**4063.25**	**4064.20**	0.02	0.80	9.19	4117.41	1.33	3.42	13.56	**4063.25**	0.00	0.00	28.08	**4063.25**	0.00	1.37	12.51	**4063.25**	0.00
125c6d5v	**4760.47**	4826.71	1.39	4.13	8.33	4786.74	0.55	1.61	13.09	**4760.47**	0.00	0.09	8.79	4780.61	0.42	1.47	7.53	4770.52	0.21
125c6d7v	**5164.03**	5325.28	3.12	5.86	9.18	5221.52	1.11	2.55	12.89	**5164.03**	0.00	0.70	3.78	5358.54	3.77	4.44	3.57	5203.67	0.77
125c8d3v	**4534.14**	4553.28	0.42	3.16	10.23	4574.82	0.90	1.94	14.98	**4534.14**	0.00	0.16	39.65	4545.44	0.25	1.11	23.82	4541.82	0.17
125c8d5v	**4947.00**	5045.65	1.99	3.40	9.64	**4958.26**	0.23	3.88	13.38	**4947.00**	0.00	1.92	10.33	5071.39	2.51	3.17	10.79	4954.11	0.14
125c8d7v	**5335.79**	5416.96	1.52	2.16	9.34	5397.86	1.16	2.59	13.38	**5335.79**	0.00	0.07	5.74	5372.92	0.70	1.59	4.94	5359.52	0.44
150c4d3v	**4049.48**	**4049.48**	0.00	0.01	9.71	4072.95	0.58	3.05	21.84	**4049.48**	0.00	0.13	46.10	4100.31	1.26	1.79	37.67	4095.45	1.14
150c4d5v	4618.23	4638.72	0.44	1.89	8.19	4622.77	0.10	1.05	19.11	**4618.23**	0.00	0.16	14.52	**4612.05**	-0.13	0.49	15.92	**4612.05**	-0.13
150c4d7v	**5118.41**	5176.50	1.13	2.45	8.00	5163.02	0.87	1.70	19.06	**5118.41**	0.00	0.45	7.51	5147.84	0.57	1.54	6.83	5138.05	0.38
150c6d3v	**4057.09**	**4057.09**	0.00	0.15	9.96	4066.71	0.24	1.45	22.07	**4057.09**	0.00	0.02	48.35	4062.22	0.13	1.07	36.59	4062.22	0.13
150c6d5v	**4855.29**	**4872.08**	0.35	0.89	10.23	4931.13	1.56	2.77	21.16	**4855.29**	0.00	0.09	30.57	4943.20	1.81	3.94	26.30	4928.68	1.51
150c6d7v	**5695.26**	5768.29	1.28	3.89	10.73	5840.52	2.55	3.31	20.40	**5695.26**	0.00	0.01	10.19	5722.09	0.47	2.98	11.37	5722.09	0.47
150c8d3v	**4641.30**	4653.90	0.27	2.07	10.18	4689.13	1.03	3.94	22.67	**4641.30**	0.00	0.00	51.81	4708.21	1.44	3.30	34.50	4652.13	0.23
150c8d5v	**5065.11**	5113.77	0.96	2.07	11.62	5116.55	1.02	2.67	19.60	**5065.11**	0.00	0.00	11.48	**5065.11**	0.00	0.68	16.01	**5065.11**	0.00
150c8d7v	**5605.83**	5665.23	1.06	1.06	12.01	5648.32	0.76	1.56	19.67	**5605.83**	0.00	0.16	5.06	5620.73	0.27	1.33	8.44	5613.07	0.13
175c4d4v	**4703.25**	**4706.76**	0.07	1.68	21.74	4720.36	0.36	1.67	28.69	**4703.25**	0.00	0.13	42.08	4747.63	0.94	1.55	35.10	4730.6	0.58
175c4d6v	**4816.54**	**4835.64**	0.40	2.99	23.01	4863.88	0.98	2.91	26.71	**4816.54**	0.00	0.01	17.43	4888.01	1.48	4.81	20.26	4888.01	1.48
175c4d8v	**5830.63**	5943.28	1.93	3.49	18.40	5853.90	0.40	1.78	27.35	**5830.63**	0.00	0.36	13.99	5836.97	0.11	2.08	19.13	5836.97	0.11
175c6d4v	**4999.43**	5025.51	0.52	2.17	21.51	5011.01	0.23	2.43	29.28	**4999.43**	0.00	0.22	55.83	5000.48	0.02	1.25	35.82	5000.48	0.02
175c6d6v	**5292.88**	5431.34	2.62	2.73	22.54	5382.57	1.69	3.60	27.43	**5292.88**	0.00	0.10	14.38	5446.37	2.90	3.82	13.49	5323.95	0.59
175c6d8v	6037.20	6090.01	0.87	2.16	25.81	6066.10	0.48	1.97	27.97	6037.20	0.00	0.06	7.09	6045.58	0.14	1.54	10.19	**6025.08**	-0.20
175c8d4v	**5747.73**	5878.58	2.28	4.93	24.90	5840.25	1.61	3.61	29.83	**5747.73**	0.00	0.01	44.23	5866.54	2.07	2.92	30.52	5866.54	2.07
175c8d6v	**5925.40**	5989.63	1.08	3.91	25.21	5968.99	0.74	3.35	27.78	**5925.40**	0.00	0.10	15.21	5985.42	1.01	4.48	12.50	5940.74	0.26
175c8d8v	6773.77	6943.63	2.51	4.45	26.70	6840.04	0.98	4.15	27.98	6773.77	0.00	0.69	6.77	6792.15	0.27	3.75	8.58	**6766.54**	-0.11
Avg.			**0.64**	**2.25**	**9.54**		**0.59**	**2.09**	**12.79**		**0.05**	**0.22**	**15.19**		**0.52**	**1.41**	**10.13**		**0.29**

Table 3.9.: Comparison of the AVNS results on the VRPIRF instances by Tarantilis, Zachariadis, and Kiranoudis (2008) to those of TZK, VNS/TS and HDHR. BKS denotes the previously best-known solution. Note that the value given in column f^{b}* for TZK corresponds to the best solution ever found with the final parameter setting. For all other methods, f^{b} refers to the best solution out of ten runs and Δ^{b} to the gap to BKS. The gap of the average solution found to the BKS is given by Δ^{a} and t^{a} denotes the average computing time in minutes. Additionally, we provide the best solutions found during our overall testing in column $\overline{\text{AVNS}}$. Numbers in bold indicate the best solution found.

Christofides and Eilon (1969) and from the paper of Mester and Bräysy (2007) for the instances of Golden et al. (1998). The fixed costs per vehicle c^{fix} are calculated by dividing the objective function value of the respective high-quality CVRP solution by the number of vehicles employed in this solution (rounded up to the next multiple of 20). The recharging speed g is set such that recharging the amount p takes 30 time units. Due to the additional time-consumption of visiting recharging facilities, the maximum route durations given by Christofides and Eilon (1969) and Golden et al. (1998) are increased by t_i^s multiplied with the average number of customers per route in the corresponding high-quality CVRP solution.

Further, each problem instance is complemented with eleven recharging facilities, of which one is located at the depot. The docking time t_i^d of each recharging facility $i \in \mathcal{F}_F$ is set to five time units. The procedure for locating the recharging facilities is illustrated in Figure 3.4. First, we generate a circle around the depot with a radius that corresponds to the maximal distance d^{max} of any customer to the depot. From this circle, we create a circular ring using the radii $r_1 = 0.3\, d^{max}$ and $r_2 = 0.8\, d^{max}$ and divide this ring into ten sectors of identical size. Within each of these circle ring sectors, we iteratively draw possible locations for the recharging facility in a random fashion until the following two criteria are met: (i) the location does not coincide with a customer location, and (ii) the distance of the possible location to all previously placed recharging facilities exceeds a given threshold. This threshold is continuously decreased after a certain number of the generated random points have not met this criterion.

In this way, we generate a total of 34 large EVRPRF instances, 14 based on the instances of Christofides and Eilon (1969) and 20 based on those of Golden et al. (1998). In addition, we create a set of small problem instances as follows: For each of the large EVRPRF instances of Christofides and Eilon (1969), we generate four small instances by (i) drawing five, ten, 15, and 20 customers of the original instances and removing the remaining ones, and (ii) solving the thus generated instances with our AVNS and removing the recharging facilities that are not used in the produced solutions. In this way, 56 small instances are generated, which are denoted by the identifier of the underlying CVRP instance (CE plus instance number) followed by the number of customers (#C) and the number of facilities (#F) in the instance. For example, CE-01-05C2F denotes the instance obtained from the CVRP instance CE-01, containing five customers and two facilities.

3.5.3.2. Results on the EVRPRF Instances

We solve the small EVRPRF instances by means of our AVNS and compare our results to those of the commercial solver CPLEX. Ten AVNS runs are conducted for each problem instance. CPLEX is given a time limit of 7200 seconds for each instance and we generate three dummy vertices for each recharging facility to represent visits to the facility. The results are presented in Table 3.10. For CPLEX, we report the solution f and the runtime t in seconds. If CPLEX terminates before the end of the time limit, the given solution is optimal. Otherwise, the result corresponds to the best upper bound found within the time limit. For the AVNS, we give the best solution found in the ten runs (f^b), the relative gap of this solution to the CPLEX solution (Δ^b), the average solution (f^a), the gap of the average solution to CPLEX (Δ^a) and the average runtime in seconds (t^a).

While CPLEX is only able to solve twelve out of 56 instances to optimality, AVNS is able to provide high-quality solutions with an average runtime of just above one second. Concerning the best solution,

	CPLEX		AVNS				
Inst.	f	t(s)	f^b	Δ^b(%)	f^a	Δ^a(%)	t^a(s)
CE-01-05C2F	292.55	914	292.55	0.00	292.55	0.00	0.23
CE-02-05C1F	195.78	24	195.78	0.00	195.78	0.00	0.17
CE-03-05C1F	225.49	34	225.49	0.00	225.49	0.00	0.16
CE-04-05C2F	207.09	3988	207.09	0.00	207.09	0.00	0.23
CE-05-05C2F	261.72	68	261.72	0.00	261.72	0.00	0.39
CE-06-05C2F	382.72	51	382.72	0.00	382.72	0.00	0.36
CE-07-05C2F	508.96	91	508.96	0.00	508.96	0.00	0.26
CE-08-05C2F	227.70	75	227.70	0.00	227.70	0.00	0.19
CE-09-05C3F	389.60	217	389.60	0.00	389.60	0.00	0.33
CE-10-05C2F	322.42	17	322.42	0.00	322.42	0.00	0.40
CE-11-05C3F	435.71	72	435.71	0.00	435.71	0.00	0.33
CE-12-05C4F	286.50	7200	286.50	0.00	286.50	0.00	0.37
CE-13-05C4F	494.33	7200	494.33	0.00	494.33	0.00	0.47
CE-14-05C5F	322.63	7200	322.63	0.00	322.63	0.00	0.61
CE-01-10C2F	439.92	7200	439.92	0.00	439.92	0.00	0.99
CE-02-10C4F	433.39	7200	433.39	0.00	433.39	0.00	0.67
CE-03-10C4F	312.87	7200	312.87	0.00	312.87	0.00	0.76
CE-04-10C5F	338.90	7200	338.90	0.00	338.90	0.00	1.98
CE-05-10C6F	397.11	1343	397.11	0.00	397.11	0.00	2.21
CE-06-10C4F	544.12	7200	544.12	0.00	544.12	0.00	0.38
CE-07-10C3F	533.65	7200	533.65	0.00	533.65	0.00	0.35
CE-08-10C4F	429.21	7200	429.21	0.00	430.70	0.35	0.48
CE-09-10C5F	736.75	7200	736.75	0.00	736.75	0.00	0.47
CE-10-10C5F	645.73	7200	645.73	0.00	645.73	0.00	0.41
CE-11-10C3F	428.14	7200	428.14	0.00	428.14	0.00	1.03
CE-12-10C6F	499.53	7200	499.53	0.00	499.66	0.03	1.00
CE-13-10C4F	447.75	7200	447.75	0.00	447.75	0.00	1.38
CE-14-10C7F	417.64	7200	417.64	0.00	417.64	0.00	2.68
CE-01-15C4F	491.70	7200	491.70	0.00	491.70	0.00	1.00
CE-02-15C5F	586.13	7200	586.13	0.00	587.43	0.22	0.69
CE-03-15C3F	498.00	7200	498.00	0.00	498.00	0.00	1.22
CE-04-15C5F	488.38	7200	488.38	0.00	488.38	0.00	1.87
CE-05-15C6F	489.54	7200	489.54	0.00	489.54	0.00	1.40
CE-06-15C5F	611.58	7200	611.58	0.00	611.58	0.00	0.64
CE-07-15C5F	876.38	7200	876.38	0.00	876.38	0.00	0.46
CE-08-15C4F	605.26	7200	605.26	0.00	607.97	0.45	0.86
CE-09-15C5F	754.09	7200	754.09	0.00	754.09	0.00	0.54
CE-10-15C5F	515.30	7200	515.30	0.00	515.30	0.00	0.58
CE-11-15C3F	483.14	7200	483.14	0.00	483.14	0.00	2.10
CE-12-15C7F	595.82	7200	595.82	0.00	595.82	0.00	1.94
CE-13-15C4F	459.06	7200	459.06	0.00	459.06	0.00	3.19
CE-14-15C6F	428.83	7200	428.83	0.00	428.83	0.00	1.80
CE-01-20C6F	718.64	7200	718.64	0.00	718.64	0.00	1.05
CE-02-20C6F	637.97	7200	637.41	-0.09	637.41	-0.09	1.08
CE-03-20C4F	526.62	7200	526.62	0.00	526.62	0.00	2.26
CE-04-20C6F	509.02	7200	509.02	0.00	509.02	0.00	2.74
CE-05-20C5F	526.41	7200	526.41	0.00	526.41	0.00	2.19
CE-06-20C5F	602.24	7200	602.24	0.00	602.24	0.00	1.00
CE-07-20C6F	909.96	7200	895.52	-1.59	915.45	0.60	0.53
CE-08-20C4F	806.08	7200	802.46	-0.45	803.19	-0.36	0.71
CE-09-20C5F	919.69	7200	773.20	-15.93	773.20	-15.93	0.57
CE-10-20C7F	893.96	7200	889.74	-0.47	889.74	-0.47	0.96
CE-11-20C3F	662.05	7200	662.05	0.00	662.05	0.00	2.96
CE-12-20C7F	622.46	7200	622.46	0.00	622.46	0.00	2.18
CE-13-20C4F	697.26	7200	697.26	0.00	697.26	0.00	2.57
CE-14-20C8F	614.58	7200	600.98	-2.21	602.98	-1.89	2.53
Avg.		**5780.25**		**-0.37**		**-0.31**	**1.09**

Table 3.10.: Comparison of the AVNS results on the generated small-sized EVRPRF instances with CPLEX. For CPLEX, f denotes the objective function value and t the total runtime in seconds. The maximum duration for CPLEX was set to 7200 seconds. For AVNS, f^b denotes the best solution found in ten runs, f^a the average solution of ten runs, Δ^b and Δ^a the gaps to the CPLEX solution, and t^a the average computing time in seconds.

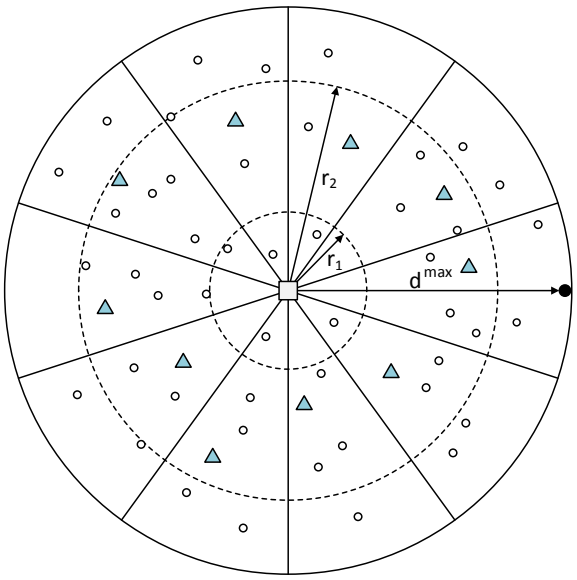


Figure 3.4.: Locating recharging facilities.

the quality of all optimal CPLEX solutions and all CPLEX upper bounds is matched or improved. The robustness of our AVNS is again proven by a negative average gap of the average AVNS solutions to the CPLEX solutions.

Finally, we provide the results of our AVNS on the large EVRPRF instances in Table 3.11. The instances are denoted by the identifier of the underlying CVRP instance (CE or G, respectively, plus instance number) followed by the number of customers (#C) in the instance. The same measures as for the AVNS results on the small instances are reported. Here, no results for comparison are available, however, we want to provide researchers tackling the EVRPRF in the future with the possibility to compare their results with those of our AVNS. Moreover, we can show that our AVNS is able to provide solutions to the large instances within reasonable runtimes.

	AVNS				AVNS		
Inst.	f^b	f^a	t^a(min)	Inst.	f^b	f^a	t^a(min)
CE-01-050C	1148.73	1148.73	3.31	G-01-240C	12363.17	12476.44	23.81
CE-02-075C	1895.52	1903.06	3.60	G-02-320C	18253.91	18390.20	40.34
CE-03-100C	1822.43	1825.80	11.81	G-03-400C	22504.96	24069.90	41.68
CE-04-150C	2344.28	2353.74	14.19	G-04-480C	29334.38	29787.80	41.67
CE-05-199C	3164.90	3180.26	19.01	G-05-200C	12995.39	13135.66	39.31
CE-06-050C	1053.82	1056.98	1.70	G-06-280C	17135.69	17206.53	41.67
CE-07-075C	2040.79	2046.01	1.47	G-07-360C	20650.82	21833.96	41.68
CE-08-100C	1703.03	1709.49	4.74	G-08-440C	25211.37	25861.50	41.69
CE-09-150C	2354.09	2375.32	5.67	G-09-255C	1564.96	1573.69	41.62
CE-10-199C	2856.10	2885.45	8.20	G-10-323C	1863.69	1880.54	39.63
CE-11-120C	2250.00	2273.21	13.86	G-11-399C	2232.24	2259.98	41.74
CE-12-100C	1960.85	1976.11	3.98	G-12-483C	2540.14	2583.65	41.80
CE-13-120C	2232.42	2318.04	9.53	G-13-252C	2000.50	2034.28	15.00
CE-14-100C	1808.34	1834.67	6.06	G-14-320C	2455.81	2477.32	19.95
				G-15-396C	3591.95	3633.86	28.64
				G-16-480C	4264.65	4294.46	35.73
				G-17-240C	1667.67	1682.53	14.77
				G-18-300C	2251.71	2273.29	18.27
				G-19-360C	3655.12	3690.33	34.29
				G-20-420C	4603.23	4633.80	34.82
Avg.			**7.65**				**33.90**

Table 3.11.: Results of AVNS on large EVRPRF instances. f^b corresponds to the best solution found in ten runs and f^a denotes the average solution of ten runs. The average computing time in minutes is given by t^a.

3.5.3.3. Influence of Problem-Specific Components

Finally, Table 3.12 shows a comparison of our AVNS to an AVNS without the problem-specific components addressing intermediate stops (denoted as AVNS-without) on the VRPIRF instances of Crevier, Cordeau, and Laporte (2007). For each instance, we report the instance name and the BKS. Moreover, we provide the average solution found in the ten runs (f^a), the gap of the average solution to the BKS (Δ^a) and the average run time in minutes (t^a) for both methods. Finally, averages of the runtimes and the gaps to the BKS over the complete set of instances are given at the end of the table. The results show that adding the problem-specific components clearly improves the solution quality while notably reducing run-times.

3.6. Conclusion

This chapter presents an adaptive variable neighborhood search (AVNS) to address the vehicle-routing problem with intermediate stops (VRPIS), in which vehicles are required to stop at certain facilities along their route in order to remain operational. The competitiveness of the proposed approach is demonstrated on benchmark instances from the literature designed for the green VRP and the VRP with intermediate replenishment facilities, which both represent special cases of VRPIS. Our AVNS algorithm shows a convincing performance compared to the methods from the literature and is able to obtain numerous new best solutions.

As a special case of the VRPIS, we additionally consider the electric VRP with recharging facilities (EVRPRF). We design two sets of small and large EVRPRF instances based on well-known CVRP benchmarks. On the small instances, our AVNS, using a runtime of approximately one second, is able to match or improve all results obtained by the commercial solver CPLEX within a time limit of two hours.

		AVNS			**AVNS** − without		
Inst.	BKS	f^a	Δ^a(%)	t^a(min)	f^a	Δ^a(%)	t^a(min)
a1	**1179.79**	1184.57	0.41	0.64	1186.85	0.60	0.78
b1	**1217.07**	1218.21	0.09	4.19	1219.47	0.20	8.99
c1	**1866.76**	1925.41	3.14	32.98	1923.19	3.02	37.85
d1	**1059.43**	1061.50	0.20	0.55	1062.24	0.27	1.08
e1	**1309.12**	1312.75	0.28	5.08	1312.46	0.25	5.95
f1	**1570.41**	1601.40	1.97	34.99	1611.08	2.59	39.62
g1	**1181.13**	1183.75	0.22	1.69	1187.66	0.55	2.67
h1	**1545.50**	1567.22	1.41	14.08	1571.66	1.69	25.43
i1	**1922.18**	1974.97	2.75	35.11	1977.47	2.88	42.05
j1	**1115.78**	1116.82	0.09	2.02	1119.00	0.29	2.73
k1	**1576.36**	1600.42	1.53	10.74	1598.56	1.41	27.30
l1	**1863.28**	1916.07	2.83	40.59	1921.14	3.11	41.95
Avg.			**1.24**	**15.22**		**1.40**	**19.70**
a2	**997.94**	**997.94**	0.00	0.72	998.39	0.05	1.06
b2	**1291.19**	1300.42	0.72	4.83	1297.52	0.49	10.18
c2	**1715.60**	1741.55	1.51	18.32	1747.94	1.89	29.12
d2	**1856.84**	1903.15	2.49	30.64	1916.31	3.20	38.91
e2	**1919.38**	1957.80	2.00	41.60	1988.88	3.62	41.33
f2	**2230.32**	2313.08	3.71	42.80	2330.63	4.50	42.07
g2	**1152.92**	1158.21	0.46	2.20	1158.57	0.49	3.63
h2	**1575.28**	1586.24	0.70	21.20	1608.81	2.13	26.98
i2	**1919.74**	1971.27	2.68	41.10	1982.83	3.29	40.28
j2	**2247.70**	2303.67	2.49	41.93	2344.55	4.31	41.77
Avg.			**1.68**	**24.53**		**2.40**	**27.53**
Tot. Avg.			**1.44**	**19.46**		**1.86**	**23.26**

Table 3.12.: Comparison of the results of our final AVNS implementation to those obtained by our AVNS without the problem-specific methods explicitly addressing intermediate stops (AVNS-without) on the VRPIRF instances of Crevier, Cordeau, and Laporte (2007). f^a denotes the average solution quality of ten runs. The gap of the average solution found to the BKS is given by Δ^a and the average computing time in minutes by t^a. Numbers in bold indicate the best solution found.

Chapter 4

Integrated Infrastructure Planning and Routing for Electric Vehicles

4.1. Introduction

The battery swap station location-routing problem with capacitated electric vehicles (BSS-EV-LRP) has recently been introduced by Yang and Sun (2015) in the context of planning a battery swapping infrastructure for the deployment of battery electric vehicles (BEVs) by a logistics company.

In the BSS-EV-LRP, a homogeneous fleet of BEVs with limited driving range and capacity is stationed at a single depot and has to serve a set of customers with given demands. To stay operational, the BEVs can stop at battery swap stations (BSSs) to swap their depleted battery for a fully charged one. Now, the BSS-EV-LRP calls for the simultaneous determination of (i) the BSS locations to open from a set of BSS candidate locations with given construction cost, and (ii) the routes of the BEVs so as to minimize the sum of the total travel cost and the cost for constructing the BSSs. Each customer has to be visited exactly once; visits to constructed BSSs are optional. Vehicle routes are feasible if (i) at any point of the route, the load of the vehicle does not exceed its capacity, and (ii) the traveled distance between the depot and a visit to a BSS or two consecutive visits to BSSs does not exceed the driving range of the vehicle. In the paper by Yang and Sun (2015), two versions of the problem are introduced: In the basic version, the maximal number of visits to each BSS is restricted to one visit per vehicle, in the extended version, multiple visits per vehicle are allowed. In this chapter, we restrict ourselves to the extended version because the paper by Yang and Sun (2015) does not report detailed computational results for the basic model.

The BSS-EV-LRP classifies as a location-routing problem (LRP), in which decisions on the location of facilities are jointly taken with decisions on the routing of vehicles (for recent surveys on LRPs, see Lopes et al., 2013; Drexl and Schneider, 2014; Prodhon and Prins, 2014; Cuda, Guastaroba, and Speranza, 2015). Note that, in contrast to the classical capacitated LRP, which aims at determining the locations of capacitated depots at which the vehicles must start and end their routes, the goal of the BSS-EV-LRP is to locate intermediate BSSs at which vehicles may conduct optional battery swaps en route. In addition, the motivation for dealing with the BSS-EV-LRP is based on ecological aspects. A "green" LRP is also investigated by Koç et al. (2016) which does not consider BEVs and respective BSSs but aims at minimizing emissions by considering the depot location, fleet composition, and routing of vehicles. Furthermore, the BSS-EV-LRP is closely related to approaches for the planning of recharging infrastructure, i.e., the deployment of (i) recharging or battery swap stations (MirHassani and Ebrazi, 2013; Nie and Ghamami, 2013; He et al., 2013; Wen et al., 2014; Mak, Rong, and Shen, 2014; Arslan and Karaşan, 2016), and (ii) charging lanes which allow for recharging the vehicle battery while driving

(Chen, He, and Yin, 2016). Since the BSS-EV-LRP degenerates to the $\mathcal{NP}$-hard capacitated vehicle-routing problem (CVRP) if the driving range of the vehicles is sufficiently large, exact methods are not suitable to solve larger problem instances within short run-time. Yang and Sun (2015) propose two metaheuristics for the BSS-EV-LRP – a four-phase adaptive large neighborhood search (ALNS) heuristic, called SIGALNS, and an iterated modified Clarke and Wright savings (MCWS) heuristic including a tabu search (TS) phase, called TS-MCWS. The authors assess the performance of these algorithms on modified benchmark instances from the literature (see Section 4.3.1 for a detailed description of BSS-EV-LRP benchmarks).

In this chapter, we show how algorithms for variants of the VRP with intermediate stops (VRPIS), including VRPs with intermediate replenishment facilities (also known as multidepot VRPs with interdepot routes, see Crevier, Cordeau, and Laporte, 2007; Hemmelmayr et al., 2013; Muter, Cordeau, and Laporte, 2014; Markov, Varone, and Bierlaire, 2016), and VRPs with BEVs or alternative-fuel vehicles (Erdoğan and Miller-Hooks, 2012; Adler and Mirchandani, 2014; Schneider, Stenger, and Goeke, 2014; Goeke and Schneider, 2015; Liao, Lu, and Shen, 2016; Desaulniers et al., 2016; Roberti and Wen, 2016) can be extended to address the BSS-EV-LRP. We explain the procedure using the example of an adaptive variable neighborhood Search (AVNS) algorithm, originally developed for the VRPIS in Schneider, Stenger, and Hof (2015). However, the general ideas can be transferred to modify other VRPIS algorithms in the literature in a similar fashion. In the VRPIS, vehicles may stop at different types of intermediate facilities to replenish the goods to be delivered, to refuel or to unload collected goods or waste. From an algorithmic viewpoint, BSSs can be modeled as intermediate facilities in the VRPIS. However, the major difference between the two problems is that, in the VRPIS, the number and locations of intermediate facilities are given and thus no decisions on the locations of facilities are made. To incorporate this type of decision, we add the construction cost of an intermediate facility, i.e., of a BSS, to the objective function value if the intermediate facility is visited at least once in the vehicle routes.

In the numerical studies, we investigate the performance of our extended AVNS and its components on the BSS-EV-LRP benchmark instances proposed by Yang and Sun (2015). The AVNS is able to strongly improve the results of Yang and Sun (2015) and provides new best solutions for the large majority of instances, namely for 23 out of 24 instances, with an average improvement of nearly 10%. Moreover, we significantly reduce the number of utilized BSSs on numerous instances. Therefore, we generate an additional instance set for the BSS-EV-LRP, which is more meaningful with regard to the necessity of utilizing BSSs, and we use this benchmark set to investigate the influence of varying construction cost on the location decision.

The chapter is organized as follows. Section 4.2 describes how to extend the AVNS for VRPIS to address the BSS-EV-LRP and Section 4.3 presents the numerical experiments to assess the performance of the extended AVNS. The main findings are briefly summarized in Section 4.4.

4.2. AVNS for the BSS-EV-LRP

In this section, we illustrate how to extend our AVNS for the VRPIS to be able to solve the BSS-EV-LRP. To keep the chapter concise, our description focuses on the main algorithmic elements and the extensions compared to the original algorithm for solving the VRPIS. For more details on the algorithmic components, the reader is referred to Schneider, Stenger, and Hof (2015).

AVNS enhances the VNS paradigm originally introduced by Mladenović and Hansen (1997) by a guided shaking step which relies on problem-specific selection methods for the routes and vertices involved in the shaking. The probability for choosing a selection method is adapted based on its performance in previous iterations. AVNS has been successfully applied to several VRP variants (Stenger, Schneider, and Goeke, 2013; Stenger et al., 2013; Schneider, Stenger, and Hof, 2015). Note that, in Todosijević et al. (2016), the term AVNS has been used to describe a method with a different adaptive component: Here, the order of the neighborhoods in the local search phase is adaptively selected.

A pseudocode overview of the extended AVNS is given in Figure 4.1. First, the set of neighborhood structures $\{\mathcal{N}_\kappa \mid \kappa = 1, ..., \kappa^{max}\}$ used in the shaking step is defined. In the original version of the algorithm, visits to intermediate facilities could only be rearranged in a stepwise fashion by means of the local search, i.e., a visit is removed from a route, relocated within the route, or replaced by a visit to a different facility. In the BSS-EV-LRP, in which the utilization of a BSS incurs a construction cost, this is no longer sufficient to escape from local optima concerning the BSS configuration. Therefore, we introduce two additional neighborhood structures, called *facility removal* and *facility replacement*, for the shaking step of the extended AVNS. Facility removal completely removes a random BSS from the current solution by removing all visits to this BSS. Facility replacement removes all visits to a randomly selected BSS from the current solution and replaces them with visits to a different BSS, which is arbitrarily chosen among the $\lceil \tau \, |\mathcal{J}| \rceil$ closest BSSs. Here, $\mathcal{J}$ corresponds to the set of all candidate BSSs and τ denotes the percentage of BSS candidate sites considered.

```
Define the neighborhood structures N_κ with κ = 1, ..., κ^max
Generate initial solution S
S ← localSearch(S)
Initialize best solution S* ← S
while number of iterations without improvement not reached do
  κ ← 1
  repeat
    Select route and vertex selection method and generate S' ∈ N_κ(S)
    S'' ← localSearch(S')
    if accept(S'', S) then
      S ← S''
      κ ← 1
      if S'' improves on S* then
        S* ← S''
      end if
    else
      κ ← κ + 1
    end if
    Update weights of route and vertex selection methods
    Update penalty factors
  until κ = κ^max + 1
end while
```

Figure 4.1.: Pseudocode of the extended AVNS for solving BSS-EV-LRP.

Table 4.1 shows the neighborhood structures used in the shaking step: six sequence relocation, 18 cyclic exchange (Thompson and Orlin, 1989), four facility removal, and four facility replacement operators are implemented. Each neighborhood structure is characterized by two parameters: the number of routes involved Ω and the maximum number of vertices to be relocated Γ^{max}. In case of the facility removal and facility replacement neighborhoods, Ω corresponds to the number of routes that contain at least one visit to a BSS (indicated by a dash in the table), and Γ^{max} is the maximum number of distinct BSSs to be entirely, i.e., including all associated visits, removed or replaced.

κ	Type	Ω	Γ^{max}	κ	Type	Ω	Γ^{max}
1	sequence relocation	2	1	17	cyclic exchange	3	1
2	sequence relocation	2	2	18	cyclic exchange	3	2
3	sequence relocation	2	3	19	cyclic exchange	3	3
4	facility removal	–	1	20	facility removal	–	3
5	sequence relocation	2	4	21	cyclic exchange	3	4
6	sequence relocation	2	5	22	cyclic exchange	3	5
7	sequence relocation	2	6	23	cyclic exchange	3	6
8	facility replacement	–	1	24	facility replacement	–	3
9	cyclic exchange	2	1	25	cyclic exchange	4	1
10	cyclic exchange	2	2	26	cyclic exchange	4	2
11	cyclic exchange	2	3	27	cyclic exchange	4	3
12	facility removal	–	2	28	facility removal	–	4
13	cyclic exchange	2	4	29	cyclic exchange	4	4
14	cyclic exchange	2	5	30	cyclic exchange	4	5
15	cyclic exchange	2	6	31	cyclic exchange	4	6
16	facility replacement	–	2	32	facility replacement	–	4

Table 4.1.: Neighborhood structures used within the shaking step of the extended AVNS. For the facility removal and replacement neighborhoods, the dash indicates that all routes containing at least one visit to a BSS are considered in the shaking.

During the search, infeasible solutions, i.e., solutions that do not respect the load or battery capacity, are allowed and handled by means of a dynamic penalty mechanism. More precisely, constraint violations are transformed into penalty costs by multiplying with a dedicated dynamic penalty factor for each constraint. The penalty factors are dynamically updated depending on how many iterations the corresponding constraint has been continuously satisfied or violated. The initial solution $\mathcal{S}$ is constructed by means of the modified savings algorithm of Schneider, Stenger, and Hof (2015), which is extended to the insertion of BSSs by considering the respective construction cost. The initial solution is directly improved by a greedy local search procedure that uses the classical operators 2-opt (Lin, 1965) and Or-opt (Or, 1976) in intra-route fashion, relocate (Savelsbergh, 1992) in intra- and inter-route fashion, and inter-route exchange (Savelsbergh, 1992). In addition, problem-specific operators to insert, rearrange, and remove visits to BSSs, which are adapted from Schneider, Stenger, and Hof (2015) by integrating the construction cost of BSSs, are applied. All operators are applied in random order following a first improvement strategy. An operator is reused as long as it is able to achieve an improvement of the current solution.

The following main phase of our algorithm is repeated until ω iterations without improvement are reached. In the shaking step, a random solution $\mathcal{S}'$ within the κ-th neighborhood of $\mathcal{S}$ is generated. Instead of a random selection, the adaptive mechanism uses specific selection methods for the routes and vertices to be involved. Here, we adopt the set of methods presented in Schneider, Stenger, and Hof (2015), which consists of (i) selection methods with a general validity for VRPs, and (ii) selection methods which take into account the characteristics of intermediate facilities (BSSs) like necessary detours and the density of BSSs in routes. Each of the selection methods is chosen according to a roulette-wheel selection procedure as proposed by Pisinger and Ropke (2007), and the selection probabilities are dynamically updated based on success. Each method k is initially assigned the same weight w_k. Given h different selection methods, the probability of selecting method k is calculated as $w_k / \sum_{i=1}^{h} w_i$. After γ AVNS iterations, the weight of each method is updated based on its performance during this period. The performance of a method is measured using a scoring system. We add a score of nine to the total score of a method whenever it achieved a new overall best solution, a score of three

if the new solution improves on the current and a score of one if the current solution is deteriorated but the new solution is accepted according to an acceptance mechanism inspired by simulated annealing (SA). Let ϕ_k denote the current score of method k and χ_k the application frequency of the method since the previous weight update, then the new weight is calculated as $w_k = w_k(1-\rho) + \rho\phi_k/\chi_k$. The reaction factor $\rho \in [0,1]$ allows to balance the influence of the past weight value and the score-based increment on the resulting new weight. The values ϕ_k and χ_k are reset to zero after each update.

Subsequently, the local search step described above is applied to obtain the local optimum $\mathcal{S}''$. In contrast to the initialization phase, we perform local search only on the routes involved in the preceding shaking step. If $\mathcal{S}''$ is accepted based on the SA acceptance criterion, it replaces $\mathcal{S}$ and κ is reset to one. Otherwise, $\mathcal{S}''$ is discarded and the next neighborhood is selected. The temperature parameter of the SA is initially set to a value such that a relative deterioration of the current solution $\mathcal{S}$ by Δ^{SA} is accepted with a probability of 50%. The temperature is decreased after every AVNS iteration such that the acceptance probability of a deterioration of Δ^{SA} is equal to 0.001% after θ iterations.

We reset $\mathcal{S}$ to the overall best solution $\mathcal{S}^*$ after μ main iterations without improvement of the current best solution. For the purpose of diversification, we reset the temperature parameter to its initial value after ϵ solution resets.

Summarizing, the differences of the presented extended AVNS to the original algorithm described in Schneider, Stenger, and Hof (2015) are (i) the new facility-related neighborhood structures in the shaking step, and (ii) the modified SA acceptance mechanism, which calculates the initial and minimal temperature values based on a relative solution deterioration parameter and associated acceptance probabilities instead of using predetermined values. As will be shown in Section 4.3, the introduced modifications are necessary to obtain high-quality solutions for the BSS-EV-LRP.

4.3. Computational Studies

This section presents the computational studies to assess the performance of the extended AVNS for the BSS-EV-LRP. Section 4.3.1 describes the benchmark instances used in our experiments. The parameter setting of our algorithm is presented in Section 4.3.2. In Section 4.3.3, we analyze the influence of specific algorithmic components on the performance of our algorithm. We report detailed results on the benchmark instances for the BSS-EV-LRP in Section 4.3.4. Finally, we investigate the influence of different construction costs on the location of BSSs in Section 4.3.5.

4.3.1. Benchmark Instances

Yang and Sun (2015) modify several CVRP benchmark instances from the literature (available at http://neo.lcc.uma.es/vrp/vrp-instances) to generate BSS-EV-LRP instances. A first set of small-sized instances contains ten instances of Augerat et al. (1995), which feature 16 to 70 customers and two to ten vehicles. A second set of medium-sized instances consists of twelve instances of Rochat and Taillard (1995) with 75, 100, or 150 customers. A third set of large-sized instances includes two instances of Golden et al. (1998) with 255 and 480 customers, respectively. A fourth set of very small-sized instances, which is used for a comparison of their algorithms with the commercial solver CPLEX, contains six instances. Three of them are derived from the instance P-n16-k8 of Augerat et al. (1995), the other three are created based on instance RY-att48 of Rinaldi and Yarrow (1985), which originally

contains 48 customers. For all instances, each customer location represents a candidate site for a BSS. The battery capacity B is set to $\lceil 1.2\, d_{max} \rceil$, where d_{max} corresponds to the maximal distance between any two vertices in the instance, and the construction cost of all BSSs is set to $\lceil 0.5\, B \rceil$.

In order to ensure the comparability of our results to those obtained by Yang and Sun (2015), the following information, which is either missing or incorrectly described in their paper, needed to be obtained via personal communication with the authors:

- In the instances derived from those of Augerat et al. (1995) and in the very small-sized instances based on the instance of Rinaldi and Yarrow (1985), the authors do not use the first vertex as depot, as intended, but locate the depot at coordinates $(1, -1)$.
- For comparison with CPLEX, the very small-sized instances based on instance P-n16-k8 of Augerat et al. (1995) are generated by keeping the last $n = 6, 7, 8$ customers of the instance and removing the remaining ones. In contrast to the statement in Yang and Sun (2015), the instances based on instance RY-att48 of Rinaldi and Yarrow (1985), however, are created by selecting the first $n = 12, 15, 20$ customers of the instance. Moreover, the authors neglect to state that the load capacity is assumed to be 40 for the very small-sized Augerat et al. (1995) instances, and 8 for the Rinaldi and Yarrow (1985) instances. For all other instances, the load capacity value given in the respective instance file is used.

As will be shown in Section 4.3.4, our algorithm is able to strongly reduce the number of used BSSs compared to the results reported by Yang and Sun (2015). For numerous instances, we find solutions in which zero or only a very small number of BSSs need to be constructed. Moreover, because the candidate sites for constructing BSSs are located at the customer locations, there is no need to adequately incorporate detours to BSSs into the planning process. Therefore, we introduce more meaningful instances with respect to the necessity of using BSSs. We generate 34 new instances for the BSS-EV-LRP by modifying the instances for the electric VRP with recharging facilities (EVRPRF) proposed in Schneider, Stenger, and Hof (2015). The EVRPRF instances are derived from CVRP instances by Christofides and Eilon (1969) and Golden et al. (1998). In order to generate new BSS-EV-LRP instances, we keep the following properties of the EVRPRF instances:

- the battery capacity B of each vehicle corresponds to the amount of electrical energy required to travel 60% of the average route length of a high-quality solution of the respective CVRP instance
- each original CVRP instance is extended by ten potential BSS locations and a swapping possibility at the depot.

The remaining EVRPRF characteristics, i.e., maximum route duration, customer service times, docking times, fuel-level-dependent refueling times, and vehicle deployment costs are omitted. We assume BSS construction cost equal to the objective function value of the high-quality CVRP solution associated with the respective EVRPRF instance rounded up to the nearest integer. In Section 4.3.5, we investigate the influence of this parameter on the location of BSSs.

Each new BSS-EV-LRP instance is denoted by the prefix BSS followed by the identifier of the underlying CVRP instance (CE or G, plus instance number) and the number of customers (#C) in the instance. We refer to the set of new benchmark instances as HSG (Hof, Schneider, Goeke) set.

4.3.2. Computational Environment and Parameter Setting

All experiments were conducted on a desktop computer with an Intel Core i7 processor at 2.8 GHz and 8 GB of RAM, running Windows 7 Professional. In all tests, we performed five runs on each instance.

During the development of our algorithm, we identified a subset of key parameters with a stronger influence on solution quality and run-time compared to the remaining parameters. Starting from a well-performing base setting determined during our testing activities, we consecutively refine the value of each key parameter. To this end, we examine three values for each parameter. We keep the best value as the final setting for the respective parameter and proceed with tuning the next parameter.

More precisely, we use the small and medium-sized instances introduced in Yang and Sun (2015) (see Section 4.3.1 for a detailed description), to successively tune the following parameters: the relative solution deterioration Δ^{SA}, which determines the initial and minimal temperature as well as the temperature reduction factor of the SA acceptance mechanism, the cooling period θ, the reaction factor ρ used in the adaptive mechanism, the number of iterations γ after which the weight updates are performed, and the percentage of the closest facilities τ considered in the context of the facility replacement neighborhood. The results of the parameter tuning are presented in Table 4.2. For each parameter, we provide the average of the gaps of the best solutions found to the respective best-known solutions (BKS) from the literature in percent (Δ^b). The final value for each parameter is marked in bold. For the relative solution deterioration Δ^{SA}, we additionally report the average of the average run-times over all tested instances (t^a) in seconds due to the large impact of this parameter on the run-time of our algorithm. Here, we select the second value because of the disproportionate growth of run-times for higher parameter values.

SA			
Δ^{SA}	0.005	**0.01**	0.03
Avg. Δ^b(%)	-8.73	-8.92	-8.93
Avg. t^a(s)	15.67	18.94	39.34
θ	50	**200**	500
Avg. Δ^b(%)	-8.80	-8.92	-8.83
AVNS			
ρ	0.4	**0.5**	0.6
Avg. Δ^b(%)	-8.89	-8.92	-8.83
γ	10	20	**30**
Avg. Δ^b(%)	-8.86	-8.92	-8.98
τ	0.1	**0.2**	0.3
Avg. Δ^b(%)	-8.71	-8.98	-8.72

Table 4.2.: Results of different parameter settings on the small and medium-sized instances introduced in Yang and Sun (2015). The best setting for each parameter is marked in bold and used as the final setting. For each parameter, we provide the average gap of the best solutions found to the BKS from the literature (Δ^b) in percent. For the relative solution deterioration Δ^{SA}, we additionally provide the average of the average run-times (t^a) across all tested instances in seconds. We consider the second value as the best trade-off between solution quality and run-time for this parameter.

For the remaining parameters, we choose the same setting as in the original AVNS for the VRPIS. More precisely, we set (i) the initial penalty factor $\delta^0 = 1000$, the minimal penalty factor $\delta^{min} = 10$, the maximal penalty factor $\delta^{max} = 10000$, the penalty update factor $\delta^{update} = 1.5$, the number of

iterations after which the penalty factors are decreased $\eta^- = 2$ and increased $\eta^+ = 3$, and (ii) the number of solution resets after which the SA temperature is reset $\epsilon = 4$.

Moreover, in order to achieve reasonable run-times, we set the number of non-improving iterations used as stopping criterion to $\omega = 50$ for the largest of the newly generated BSS-EV-LRP instances (the ones based on the CVRP instances of Golden et al. (1998), see Section 4.3.1 for a detailed description) and $\omega = 200$ for the remaining instances. The number of iterations after which the current solution is reset to the overall best solution μ is set to 20.

4.3.3. Influence of Algorithmic Components

This section investigates the effect of the different components of our algorithm on the solution quality and run-time. To this end, we consecutively deactivate (i) the facility removal operator (Configuration I), (ii) the facility replacement operator (Configuration II), and (iii) the adaptive mechanism in the shaking step (Configuration III). Moreover, we analyze the effect of using a more thorough local search implementation by incorporating a variable neighborhood descent (VND) approach (Configuration IV). This transforms our algorithm into an adaptive general VNS (AGVNS) as defined in Hansen et al. (2010). Instead of randomly selecting the next local search operator in each iteration, we apply all operators in a predefined order while maintaining a first improvement strategy.

Table 4.3 shows the order in which the operators are applied for each route involved in the preceding shaking step. The treatment of potential battery charge violations by applying the facility insertion operator is followed by increasingly complex intra-route operators aiming at rearranging customer and facility visits. The intra-route block is concluded by removing redundant facilities. Finally, customer exchange and relocation operators are applied between the current route and the remaining routes. Whenever an operator achieves an improvement of the current solution, we reset κ to 1, i.e., apply the first operator to the current route again. Otherwise, we increase κ by 1, and the next operator is selected. After all operators have been applied to the current route, we proceed with the next route.

κ	Type	Scope
1	facility insertion	intra-route
2	customer relocation	intra-route
3	facility relocation	intra-route
4	facility replacement	intra-route
5	Or-Opt	intra-route
6	2-Opt	intra-route
7	facility removal	intra-route
8	customer exchange	inter-route
9	customer relocation	inter-route

Table 4.3.: Neighborhood structures used within the VND step of the AGVNS.

In Table 4.4, we compare Configurations I-IV to our extended AVNS (marked in bold). More precisely, we report the average gap of the best solutions found to the BKS (Δ^b) in percent, the total number of used BSSs ($\sum \ell$) and the average of the average run-times (Avg. t^a) in seconds for each configuration and set of benchmark instances.

Starting from our extended AVNS, the deactivation of each component (Configurations I-III) leads to a significant decrease in solution quality on each instance set. The computation time can be improved by omitting the facility-related neighborhood structures (Configurations I and II). Configuration II,

which does not make use of the facility replacement neighborhood structure, yields by far the worst solution quality among all investigated configurations. Disabling the adaptive selection of the route and vertex selection methods in the shaking step (Configuration III) also results in a decrease in computation time. Interestingly, Configuration III is able to obtain the smallest number of constructed BSSs on the instance set by Yang and Sun (2015). The decreased solution quality in comparison to the extended AVNS, however, hints at a better capability of the AVNS to identify the appropriate BSS construction sites. While the AGVNS variant (Configuration IV) is able to improve the solution quality of the extended AVNS on the instance set by Yang and Sun (2015), it shows a significantly worse performance on the set HSG. This may be explained by the fact that the increased diversification effect introduced by the random order of neighborhood structures in our original local search implementation has a beneficial impact on the search compared to the rigid structure of the VND component, which restarts the search in the smallest neighborhood after each improvement. The VND implementation thus shows a stronger intensification behavior which seems to be disadvantageous on the newly generated instances. In addition, the results indicate a clearly increased time consumption of the VND component. On the benchmark instances by Yang and Sun (2015), AGVNS on average takes almost three times as long as our extended AVNS.

Summarizing, the presented results suggest that the extended AVNS provides the best trade-off between solution quality and run-time among the investigated configurations.

	Components				
facility removal	✓	✗	✓	✓	✓
facility replacement	✓	✓	✗	✓	✓
adaptive mechanism	✓	✓	✓	✗	✓
VND	✗	✗	✗	✗	✓
algorithm	**AVNS**	I	II	III (VNS)	IV (AGVNS)
Yang and Sun (2015)					
Avg. Δ^b(%)	**-9.81**	-9.74	-9.06	-9.71	-9.89
$\sum \ell$	**19**	20	18	17	18
Avg. t^a(s)	**73.00**	48.40	49.26	63.61	197.25
HSG					
Avg. Δ^b(%)	**0.64**	0.75	6.39	0.85	2.64
$\sum \ell$	**157**	157	164	157	161
Avg. t^a(s)	**1539.86**	1353.53	718.40	1246.62	2070.50
Total					
Avg. Δ^b(%)	**-3.68**	-3.59	0.00	-3.52	-2.55
$\sum \ell$	**176**	177	182	174	179
Avg. t^a(s)	**932.88**	813.47	441.52	757.10	1295.36

Table 4.4.: Comparison of algorithmic configurations.

4.3.4. Detailed Results on Established and New Benchmark Instances

We first compare the results of the extended AVNS on the set of very-small-sized instances to those of CPLEX (with a time limit of 10800 seconds as reported by Yang and Sun, 2015), and to those of the SIGALNS of Yang and Sun (2015). Note that in the entire discussion of results, we restrict the direct comparison to the SIGALNS because it provides clearly better solution quality than TS-MCWS for the large majority of test instances (20 out of 24), using only a fraction of the run-time.

Table 4.5 reports, for each instance, the name, the number of customers to be served (n), and the best-known solution (BKS) as determined by CPLEX and the algorithms TS-MCWS and SIGALNS of Yang and Sun (2015). For CPLEX, the best upper bound (UB), the gap to the BKS (Δ^{UB}), and the run-time (t) in seconds are reported. Solutions found by CPLEX within the given time limit are optimal. For SIGALNS and AVNS, the best solution found in five runs (f^b), the gap of the best solution to the BKS (Δ^b), and the average computing time (t^a) in seconds are reported. Finally, averages of the run-times and of the gaps to the BKS over the complete instance set are given at the end of the table. CPLEX is only able to solve three of the six instances to optimality. SIGALNS and AVNS provide identical results, which match or improve the results obtained by CPLEX. Both algorithms show similar run-times with a slight advantage of the AVNS. We are aware that a direct comparison of the run-times on different computers is never accurate, however, as all algorithms were tested on modern desktop computers, a rough comparison seems possible. Yang and Sun (2015) used a Dell PC Inspiron 545S with an Intel Core Duo, 2.93 GHz processor and 2 GB of RAM for their testing.

			CPLEX			SIGALNS			**AVNS**		
Inst.	n	BKS	UB	Δ^{UB}(%)	t(s)	f^b	Δ^b(%)	t^a(s)	f^b	Δ^b(%)	t^a(s)
P-n6-k2	6	**426.86**	**426.86**	0.00	10.51	**426.86**	0.00	1.73	**426.86**	0.00	0.67
P-n7-k3	7	**428.60**	**428.60**	0.00	489.69	**428.60**	0.00	1.93	**428.60**	0.00	0.38
P-n8-k3	8	**597.16**	**597.16**	0.00	1853.04	**597.16**	0.00	2.00	**597.16**	0.00	0.47
RY-n12-k2	12	**52792.61**	53117.67	0.62	10800	**52792.61**	0.00	1.91	**52792.61**	0.00	1.20
RY-n15-k3	15	**52985.62**	53901.78	1.73	10800	**52985.62**	0.00	1.86	**52985.62**	0.00	2.24
RY-n20-k4	20	**63919.20**	-	-	10800	**63919.20**	0.00	2.22	**63919.20**	0.00	5.52
Avg.				**0.47**	**4790.65**		**0.00**	**1.94**		**0.00**	**1.74**

Table 4.5.: Comparison of AVNS results on the very small-sized instances of Yang and Sun (2015) to those of CPLEX and SIGALNS, where n denotes the number of customers, BKS the previous best-known solution of the respective instance, UB the best upper bound obtained by CPLEX, Δ^{UB} the gap of UB to the BKS in percent, t the associated total run-time in seconds, f^b the best solution found in five runs, Δ^b the gap of f^b to the BKS in percent, and t^a the average computing time in seconds. Numbers in bold indicate the best solution found for each instance. Averages of the run-times and the gaps to the BKS over the complete set of instances are given at the end of the table.

Next, we compare the performance of AVNS to that of SIGALNS on the sets of small, medium, and large-sized instances proposed by Yang and Sun (2015). Table 4.6 reports the same measures as Table 4.5, but additionally, the number of vehicles used in the best solution (m), and the number of BSSs used in the best solution (ℓ) are reported for both algorithms. Moreover, in column $\overline{\text{AVNS}}$, we report the gaps Δ of the best solutions encountered during our entire testing activities to the BKS from the literature in percent for each instance. Values in bold indicate the best solution found for each instance.

Concerning solution quality, our extended AVNS is able to significantly improve on the results of SIGALNS: We provide new best solutions for 23 out of 24 instances and improve on the previous BKS by 9.81% on average. Considering all solutions ever encountered, we observe an average improvement of 10.25%. With regard to run-times, AVNS is roughly as fast as SIGALNS on the small instances, and, on average, notably faster on the remaining medium-sized and large instances. This hints at a better scalability of our approach, but a clear statement is difficult due to the small number of large instances considered by Yang and Sun (2015). Finally, the robustness of our AVNS is demonstrated by the small deviations of the average objective function values from the best ones ($< 1\%$ on average).

			SIGALNS					**AVNS**						$\overline{\textbf{AVNS}}$
Inst.	n	BKS	m	ℓ	f^b	Δ^b(%)	t^a(s)	m	ℓ	f^b	Δ^b(%)	Δ^a(%)	t^a(s)	Δ(%)
P-n16-k8	16	**1281.95**	8	1	**1281.95**	0.00	2.58	8	1	1282.38	0.03	2.40	1.66	0.03
P-n19-k2	19	471.39	2	1	471.39	0.00	2.78	2	1	**468.08**	-0.70	-0.70	1.98	**-0.70**
P-n21-k2	21	478.64	2	1	478.64	0.00	3.13	2	1	**472.74**	-1.23	-1.13	1.93	**-1.23**
P-n23-k8	23	1360.51	8	1	1360.51	0.00	3.14	8	1	1347.04	-0.99	-0.53	4.15	**-1.19**
P-n40-k5	40	893.23	5	1	893.23	0.00	6.18	5	1	**857.70**	-3.98	-3.88	8.42	**-3.98**
P-n45-k5	45	939.63	5	2	939.63	0.00	7.69	5	1	872.23	-7.17	-6.06	12.12	**-7.25**
P-n50-k7	50	1196.48	7	2	1196.48	0.00	8.52	7	2	**1130.44**	-5.52	-3.74	15.18	**-5.52**
P-n55-k8	55	1247.10	7	2	1247.10	0.00	20.13	7	2	1170.04	-6.18	-5.12	16.25	**-6.19**
P-n60-k10	60	1684.24	10	3	1684.24	0.00	24.50	10	2	1556.38	-7.59	-6.63	33.57	**-8.25**
P-n70-k10	70	1738.98	10	3	1738.98	0.00	35.93	10	2	1632.87	-6.10	-5.19	21.30	**-7.11**
Avg. P						**0.00**	**11.46**				**-3.94**	**-3.06**	**11.66**	**-4.14**
tai75a	75	1924.32	10	4	1924.32	0.00	53.69	10	0	**1664.08**	-13.52	-13.43	8.84	**-13.52**
tai75b	75	1607.22	10	3	1607.22	0.00	76.73	10	1	1471.57	-8.44	-7.80	12.10	**-8.66**
tai75c	75	1602.15	9	4	1602.15	0.00	75.11	9	0	**1381.20**	-13.79	-12.24	18.11	**-13.79**
tai75d	75	1643.63	9	3	1643.63	0.00	51.06	9	0	1405.45	-14.49	-14.38	5.44	**-14.88**
tai100a	100	2467.90	12	4	2467.90	0.00	118.80	12	0	2179.28	-11.70	-10.47	19.79	**-11.78**
tai100b	100	2393.34	12	5	2393.34	0.00	134.42	11	0	1948.73	-18.58	-18.33	10.46	**-18.92**
tai100c	100	1683.69	11	4	1783.45	5.93	123.29	11	1	1598.72	-5.05	-5.02	27.76	**-5.54**
tai100d	100	1918.81	12	4	1926.96	0.42	188.95	11	0	1609.06	-16.14	-14.63	20.80	**-16.36**
tai150a	150	3620.34	15	4	3620.34	0.00	329.67	15	0	3194.41	-11.76	-11.59	58.89	**-13.53**
tai150b	150	3354.00	14	7	3354.00	0.00	367.24	14	0	2815.80	-16.05	-16.01	45.23	**-16.84**
tai150c	150	2879.32	15	5	2879.32	0.00	337.44	15	0	2403.41	-16.53	-16.05	43.44	**-16.78**
tai150d	150	3121.36	15	5	3121.36	0.00	492.06	14	0	2744.54	-12.07	-11.26	84.91	**-12.60**
Avg. tai						**0.53**	**195.71**				**-13.18**	**-12.60**	**29.65**	**-13.60**
GWKC_09	255	787.52	14	5	790.99	0.44	1798.23	14	3	666.05	-15.42	-14.13	1017.08	**-17.53**
GWKC_16	480	2182.47	38	14	2359.08	8.09	10695.97	37	0	1693.34	-22.41	-21.85	262.64	**-24.02**
Avg. GWKC						**4.27**	**6247.10**				**-18.92**	**-17.99**	**639.86**	**-20.77**
Avg. tot.						**0.62**	**623.22**				**-9.81**	**-9.07**	**73.00**	**-10.25**
$\sum$ tot.			**88**					**19**						

Table 4.6.: Results of our extended AVNS in comparison to SIGALNS on the small, medium, and large-sized instances introduced in Yang and Sun (2015). We report the number of customers (n), the previously best-known solution (BKS) for each instance, the number of vehicles used in the best solution (m), the number of BSSs located in the best solution (ℓ), the best solution found in five runs (f^b), the gap of f^b to the BKS (Δ^b) in percent, and the average computing time (t^a) in seconds for each solution method. In addition, for AVNS, the gap of the average solution quality to the BKS (Δ^a) in percent is given. Column $\overline{\text{AVNS}}$ contains the gaps of the best solutions encountered during our entire testing activities to the BKS (Δ) in percent. Values in bold indicate the best solution found for each instance. Averages of the run-times, the gaps to the BKS, and the total numbers of deployed BSSs over the complete set of instances are given at the end of the table.

Concerning the structure of the identified solutions, we note that the extended AVNS is able to reduce the number of employed vehicles for four instances. More importantly, we are able to strongly reduce the overall number of BSSs that are constructed from 88 to 19 BSSs. This result shows that our approach is able to successfully capture the locational aspect of the BSS-EV-LRP, and is therefore likely able to also produce high quality solutions in settings where the construction cost is higher compared to the routing cost.

In Table 4.7, we compare the results obtained by our extended AVNS on the newly generated BSS-EV-LRP instance set HSG to the best solutions encountered during the entire testing ($\overline{\text{AVNS}}$). We provide the same measures as in the previous tables. In addition, f corresponds to the best solution ever obtained during our experiments.

On the new instances, AVNS also shows a reasonable scaling behavior. The increased deviations between the best and average solution values, however, suggest an increased difficulty of the new benchmarks in comparison with the instances from the literature. Mainly, the results in Table 4.7 are provided as comparison for future methods that address the BSS-EV-LRP.

4.3.5. Influence of BSS Construction Costs on the Location Decision

Finally, we use the newly generated BSS-EV-LRP instances to analyze the influence of varying construction costs on the location of BSSs. Table 4.8 shows a comparison of the total number of constructed BSSs ($\sum \ell$) for different cost configurations. The construction costs are given as decreasing fractions of the associated high-quality CVRP objective function value of each original CVRP instance. We additionally investigate the case where no construction costs for BSSs are incurred, i.e., $c^{BSS} = 0$.

In total, we can observe a high utilization of BSSs across all investigated cost configurations. As expected, the number of constructed BSSs increases with decreasing construction costs. In the case of construction costs equal to zero, BSSs are constructed on roughly 83% of all available candidate sites. The newly proposed instances thus properly capture the requirement of incorporating BSSs into the planning process. Finally, in order to demonstrate the importance of explicitly taking the locational aspect of the BSS-EV-LRP into account, we generate solutions using the AVNS version as described in Schneider, Stenger, and Hof (2015), i.e., without the facility-related neighborhood structures in the shaking step and without considering BSS construction cost. Next, we transform the resulting objective function values into BSS-EV-LRP solution values by adding the number of used BSSs multiplied with the different construction costs. The average deviation of the best BSS-EV-LRP solution values to the associated transformed solution values (Δ^{trans}) in percent is additionally reported for each cost value in Table 4.8. The results show that the transformed solutions are always significantly inferior to the solutions obtained when the construction costs and the characteristics of the BSS-EV-LRP are explicitly taken into account. Even if the construction costs are equal to zero, we can observe a notable average gap. This indicates the usefulness of the innovations introduced to the original algorithm designed for VRPIS.

4.4. Conclusion

This chapter showed how to successfully extend a solution method for vehicle-routing problems with intermediate stops (VRPIS), namely an adaptive variable neighborhood search (AVNS), to address the

		$\overline{\text{AVNS}}$			AVNS					
Inst.	c^{BSS}	m	ℓ	f	m	ℓ	f^b	$\Delta^b(\%)$	$\Delta^a(\%)$	t^a(s)
BSS-CE-01-050C	525	5	3	2251.28	5	3	**2251.28**	0.00	1.88	21.27
BSS-CE-02-075C	836	11	5	5246.86	10	5	5289.42	0.81	1.96	52.53
BSS-CE-03-100C	827	8	5	5041.80	8	5	5082.19	0.80	1.65	74.12
BSS-CE-04-150C	1029	12	6	7562.31	12	6	7585.37	0.30	3.87	305.95
BSS-CE-05-199C	1296	17	7	10707.74	18	7	10857.38	1.40	2.20	1077.33
BSS-CE-06-050C	556	5	4	2858.60	5	4	**2858.60**	0.00	1.28	25.39
BSS-CE-07-075C	910	10	5	5623.78	11	5	5631.51	0.14	0.86	41.14
BSS-CE-08-100C	866	8	5	5313.16	8	5	5316.05	0.05	0.86	91.96
BSS-CE-09-150C	1163	12	6	8299.77	12	6	8349.88	0.60	4.04	244.02
BSS-CE-10-199C	1403	17	7	11492.39	18	7	11535.24	0.37	0.97	1033.48
BSS-CE-11-120C	1043	7	3	4282.24	7	3	4336.92	1.28	4.14	125.57
BSS-CE-12-100C	820	10	6	5985.95	10	6	6030.06	0.74	4.24	56.60
BSS-CE-13-120C	1550	7	4	7431.28	7	4	**7431.28**	0.00	2.20	310.90
BSS-CE-14-100C	867	10	7	7188.93	10	7	7259.73	0.98	4.10	79.38
Avg. CE								**0.53**	**2.45**	**252.83**
$\sum$ CE			**73**			**73**				
BSS-G-01-240C	5628	9	1	20092.03	9	1	20473.11	1.90	16.29	386.10
BSS-G-02-320C	8448	10	1	27672.59	10	1	**27672.59**	0.00	18.86	2185.04
BSS-G-03-400C	11037	9	1	24505.46	9	1	24597.82	0.38	10.57	10166.99
BSS-G-04-480C	13625	10	1	29154.81	10	1	29834.44	2.33	3.36	2285.80
BSS-G-05-200C	6461	5	1	13399.61	5	1	13730.16	2.47	6.71	94.92
BSS-G-06-280C	8413	7	1	17753.50	7	1	18372.15	3.48	5.93	203.62
BSS-G-07-360C	10182	9	1	21923.28	9	1	22025.98	0.47	3.23	470.95
BSS-G-08-440C	11664	10	1	25886.27	10	1	**25886.27**	0.00	0.65	16505.95
BSS-G-09-255C	584	14	2	1905.73	14	2	**1905.73**	0.00	2.09	918.13
BSS-G-10-323C	742	16	3	3164.25	17	3	3173.22	0.28	0.95	1605.55
BSS-G-11-399C	919	18	3	3894.70	18	3	3931.24	0.94	1.10	1699.24
BSS-G-12-483C	1108	20	3	4686.10	20	3	4710.75	0.53	1.30	4705.93
BSS-G-13-252C	860	27	7	7187.78	27	7	**7187.78**	0.00	0.28	253.71
BSS-G-14-320C	1082	31	9	11145.76	31	9	**11145.76**	0.00	0.41	340.49
BSS-G-15-396C	1346	34	8	12540.50	35	8	12606.99	0.53	0.81	695.11
BSS-G-16-480C	1623	38	8	15304.30	38	8	**15304.30**	0.00	0.26	1368.95
BSS-G-17-240C	708	23	7	5900.81	24	7	5919.87	0.32	1.42	97.90
BSS-G-18-300C	998	28	7	8410.97	28	7	**8410.97**	0.00	0.74	187.71
BSS-G-19-360C	1367	35	9	14159.06	34	9	14238.38	0.56	1.09	693.10
BSS-G-20-420C	1821	39	10	20756.73	39	10	20794.43	0.18	0.66	3950.33
Avg. G								**0.72**	**3.84**	**2440.78**
$\sum$ G			**84**			**84**				

Table 4.7.: Results of our AVNS algorithm on the newly introduced BSS-EV-LRP instance set HSG in comparison with the results obtained during our entire testing activities ($\overline{\text{AVNS}}$). The BSS construction costs associated with each instance are denoted by c^{BSS}, m denotes the number of vehicles used in the best solution, ℓ the number of BSSs located in the best solution, f the best solution ever obtained, f^b the best solution found in five runs, Δ^b the gap of f^b to f in percent, Δ^a the gap of the average solution quality to f in percent, and t^a the average computing time in seconds.

$c^{BSS}\times$	**1**	**1/2**	**1/4**	**1/8**	**1/16**	**0**
$\sum \ell$	157	161	162	168	186	281
Avg. $\Delta^{trans}(\%)$	-34.87	-30.11	-23.79	-15.91	-9.22	-2.27

Table 4.8.: Analysis of (i) the effect of different construction costs on the total number of located BSSs and (ii) the average deviation in solution quality compared to the transformed solution values obtained by the original AVNS version described in Schneider, Stenger, and Hof (2015).

battery swap station location-routing problem with capacitated electric vehicles (BSS-EV-LRP). On benchmark instances from the literature, the extended AVNS proves able to significantly improve the previously known best solutions – provided by the SIGALNS and TS-MCWS of Yang and Sun (2015) – for the large majority of instances. Moreover, the AVNS strongly reduces the number of constructed BSSs in the solutions compared to SIGALNS. We additionally provide new BSS-EV-LRP instances which are more meaningful concerning the necessity of using BSSs. We use the new instances to show that (i) decreasing construction costs lead to the expected increase in the number of located BSSs, and (ii) the proposed methodological innovations related to BSSs and the explicit consideration of the BSS construction costs have a beneficial impact on the solution quality.

Chapter 5

Utilizing Mobile Depots in Urban Logistics

5.1. Introduction

There are numerous practical vehicle-routing applications, in which vehicles are employed as mobile depots to support another fleet of vehicles that operate to perform certain tasks. The mobile depots supply the task vehicles with resources required to fulfill the associated tasks. Corresponding applications are especially found in the city logistics context. In two-echelon distribution systems, for instance, small vehicles are used to navigate narrow streets and to deliver/collect goods or to collect waste, and larger vehicles serve as mobile depots to replenish the goods to be delivered or to receive collected goods or waste at accessible intermediate facilities (Cuda, Guastaroba, and Speranza, 2015; Zhou et al., 2018). Access in urban centers may not only be restricted with respect to the dimensions of the vehicles but also due to regulations on emissions, which makes some areas only accessible for environmentally-friendly vehicles such as, e.g., battery electric vehicles (Breunig et al., 2019). An interesting future application may thus be the support of a company's fleet of battery electric vehicles by means of mobile recharging stations or mobile battery swapping stations. More generally, mobile refueling stations may be reasonably employed if alternative-fuel vehicles are utilized in a region where the respective infrastructure is sparse. Future delivery concepts may also involve the use of unmanned aerial vehicles (drones) to resupply vehicles with the goods to be delivered (Dayarian, Savelsbergh, and Clarke, 2018). A further application area in the urban context is snow ploughing. Large volumes of snow plowed from roads and walkways may exceed the available space along roadways and walkways for snow storage, and therefore require disposal by some means. Most commonly, snow is loaded into trucks using some kind of snow loaders and then transported to disposal sites (Perrier, Langevin, and Campbell, 2007). Similar use cases occur in street sweeping and bitumen delivery (Rivers, 2002). Finally, road painting represents an application in a non-urban context, where mobile depots are used to refill the paint tanks of the painting vehicles (Amaya, Langevin, and Trépanier, 2010; Salazar-Aguilar, Langevin, and Laporte, 2013).

To capture the routing decisions of the described applications in a generalized fashion, we introduce the vehicle-routing problem with time windows and mobile depots (VRPTWMD) in this chapter. The VRPTWMD is characterized by a fleet of delivery vehicles (DVs) and a fleet of support vehicles (SVs). While the DVs operate to satisfy the delivery demands of a set of customers, each SV can serve as mobile depot to restore either the load capacity for the goods to be delivered or the fuel capacity of the DVs. Note that we assume the "or" to be exclusive and that all SVs restore the same capacity type. To serve in one of the described ways, an SV must meet with a DV at a location, and both vehicles must stay at that location until the capacity transfer terminates. The possible meeting points may

encompass certain customer locations and dedicated transfer locations. In the former case, a transfer may only take place when the DV serves the respective customer, i.e., each customer is visited exactly once. The order in which a DV performs the service at a customer and the transfer with the SV is a decision variable, but may in some cases be imposed by the customer time window, the maximal route duration, or a lack of capacity to meet the customer demand.

Figure 5.1 shows an example solution of a VRPTWMD instance with three DVs and two SVs. Five transfers are performed, three of which take place when serving customers 1, 5, and 9. In addition, two dedicated transfer vertices are used in the example. The superscripts s and t at the transfer customers indicate the respective task orders of the DVs, denoting service first and transfer first, respectively. Consequently, at customers 1 and 9, the service is performed before the transfer whereas at customer 5, the transfer operation is performed first.

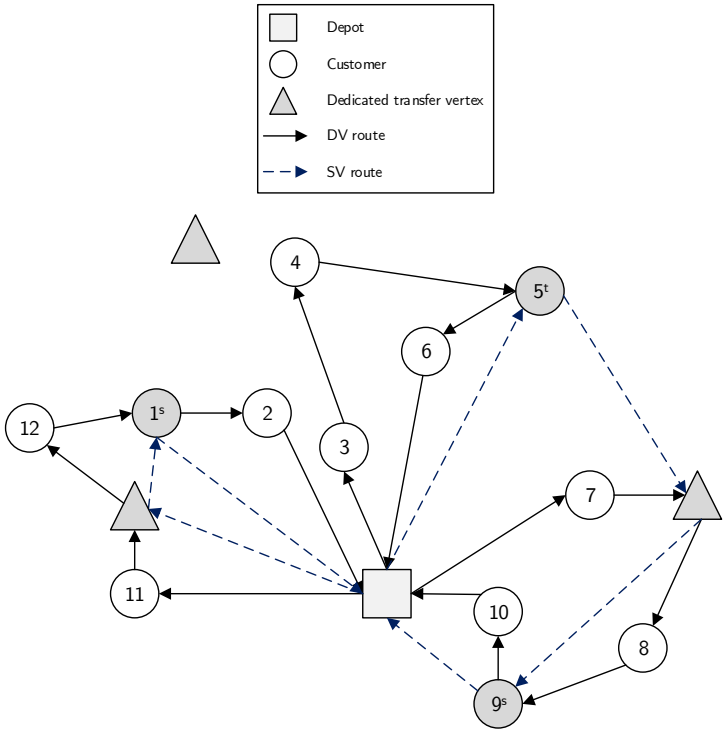


Figure 5.1.: Example solution to the VRPTWMD employing three DVs and two SVs that perform five transfers.

The contributions of the chapter are the following: We formally describe the VRPTWMD as a mixed-integer program using a compact formulation with arc flow variables and present optimal solutions on small problem instances obtained by the commercial solver CPLEX. In addition, we develop an effective and efficient adaptive large neighborhood search (ALNS) that is complemented by a path relinking (PR) approach, called ALNS-PR, to provide solutions for large problem instances. Besides components with general validity for VRPs, our ALNS-PR makes use of techniques that explicitly take transfers into account. Another important algorithmic contribution consists in move evaluation procedures enabling our algorithm to evaluate fuel, load, and time window violations in constant time.

In extensive numerical studies, we first investigate the usefulness of the problem-specific ALNS components and of our PR implementation. In addition, we evaluate the competitiveness of our algorithm on benchmark instances of the two-echelon multiple-trip VRP with satellite synchronization (2E-MTVRP-SS), which has been proposed by Grangier et al. (2016). The 2E-MTVRP-SS can be viewed as a variant of the VRPTWMD where first-level vehicles corresponding to SVs have to meet with

second-level vehicles (DVs) to enable them to fulfill the customer requests. Our ALNS-PR is able to significantly improve the large majority of previous best solutions with clearly shorter runtimes compared to the ALNS of Grangier et al. (2016). Finally, we use newly generated VRPTWMD instances to analyze the influence of certain problem characteristics, like deployment cost and load capacity of the SVs, on the structure of the resulting solutions.

We define the VRPTWMD in Section 5.2. Our ALNS-PR algorithm is presented in Section 5.3, and the numerical studies are described in Section 5.4. Finally, we summarize and conclude the chapter in Section 5.5.

5.2. Problem Definition

In this section, we present a mixed-integer program of the VRPTWMD using the following notation: Let $\mathcal{C}$ denote the set of n customers and 0 and $n+1$ instances of the same depot representing the start and end of each vehicle route, respectively. Transfers are possible at dedicated transfer vertices contained in set $\mathcal{D}$ and at specific customers contained in set $\mathcal{C}^t \subseteq \mathcal{C}$. To account for the possibility of several visits of each vehicle to a dedicated transfer vertex $i \in \mathcal{D}$, we introduce a set of dummy vertices $\mathcal{D}'$. Then, $\mathcal{T}$ denotes the set of all possible transfer locations, i.e., $\mathcal{T} = \mathcal{C}^t \cup \mathcal{D}'$. For the synchronization of both vehicle types, we need to ensure unique arrival times at depot instance $n+1$ in each vehicle route. Therefore, we introduce additional dummy visits to vertex $n+1$, which are contained in set $\mathcal{V}_e$. We further denote $\mathcal{V}'$ as the set of all vertices and visits to dedicated transfer vertices. We use indices to indicate which depot instances are considered in a vertex set, i.e., $\mathcal{V}'_0 = \mathcal{C} \cup \mathcal{D}' \cup \{0\}$, $\mathcal{V}'_{n+1} = \mathcal{C} \cup \mathcal{D}' \cup \mathcal{V}_e$, and $\mathcal{V}'_{0,n+1} = \mathcal{C} \cup \mathcal{D}' \cup \{0\} \cup \mathcal{V}_e$.

The VRPTWMD can be defined on a complete directed graph $\mathcal{G} = (\mathcal{V}'_{0,n+1}, \mathcal{A})$ with the set of arcs $\mathcal{A} = \{(i,j) : i \in \mathcal{V}'_0, j \in \mathcal{V}'_{n+1}, i \neq j\}$. Each arc $(i,j) \in \mathcal{A}$ is associated with a travel cost c_{ij}, a travel time t_{ij}, and a fuel consumption f_{ij}. Each customer $i \in \mathcal{C}$ is associated with a nonnegative demand u_i, a nonnegative service time s_i, and a hard time window $[e_i, l_i]$ within which the service of this customer must start. A homogeneous fleet of DVs with fixed deployment cost c^D, maximum route duration T^D, load capacity L^D, and fuel capacity F is based at the depot.

Depending on the application context, SVs may either replenish the load or the fuel capacity of the DVs. The fleet of SVs is also homogeneous and stationed at the depot. Each SV is characterized by a fixed cost c^S, a maximum route duration T^S, and a load capacity L^S for the DV resource that may be replenished. To perform a transfer, an SV and a DV must meet at a transfer location $i \in \mathcal{T}$ and stay at this location until the transfer, for which a fixed transfer time τ is incurred, terminates. Each transfer thus calls for exact simultaneous operation synchronization of the associated DV and SV (Drexl, 2012). Note that a transfer performed at a customer location $i \in \mathcal{C}^t$ does not have to take place within the time window of the respective customer and at any transfer location, both vehicles may wait until the other one arrives. We refer to the resource capacity that may be replenished as R and assume that R is always fully restored at each transfer.

We use the following decision variables in our model: a_i^D (a_i^S) specifies the arrival time of a DV (an SV) at vertex i. Binary variables p_i take value 1 if a transfer takes place at vertex i and 0 otherwise. If a transfer is scheduled at a customer vertex $i \in \mathcal{C}^t$, the order in which the DV performs the service and is replenished by the SV is defined using the binary decision variables o_i (transfer first: $o_i = 1$,

service first: $o_i = 0$). We further introduce variables q_i, b_i, and Ω_i to specify the load level of a DV, the fuel level of a DV, and the load level of an SV upon arriving at vertex i, respectively. Finally, binary decision variables x_{ij} (y_{ij}) take value 1 if arc (i, j) is traversed by a DV (an SV) and 0 otherwise. We summarize the notation in Table 5.1.

5.2.1. Base Model of the VRPTWMD

We first present a base model with general validity for all resources that captures the routing and synchronization requirements of both vehicle types. In the base model, each resource reduces depending on the respective consumption type but upon meeting of a DV and an SV, no actual resource replenishment takes place. Subsequently, we introduce the necessary additional constraints and modifications to the base model to obtain the model variants VRPTWMD-F and VRPTWMD-L, which consider the possibility to replenish the DVs' fuel (Section 5.2.2) and load capacity (Section 5.2.3), respectively.

The mixed-integer program of the base model is as follows:

$$min \sum_{i \in \mathcal{V}'_0} \sum_{j \in \mathcal{V}'_{n+1}} c_{ij}(x_{ij} + y_{ij}) + \sum_{j \in \mathcal{C} \cup \mathcal{D}'} (c^D x_{0j} + c^S y_{0j}) \tag{5.1}$$

$$\sum_{i \in \mathcal{V}'_0 \setminus \{j\}} x_{ij} = 1 \qquad \forall j \in \mathcal{C} \tag{5.2}$$

$$\sum_{i \in \mathcal{V}'_0 \setminus \{j\}} x_{ij} \leqslant 1 \qquad \forall j \in \mathcal{D}' \tag{5.3}$$

$$\sum_{i \in \mathcal{V}'_0 \setminus \{j\}} y_{ij} \leqslant 1 \qquad \forall j \in \mathcal{T} \tag{5.4}$$

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

$$\sum_{i \in \mathcal{V}'_0 \setminus \{j\}} y_{ij} - \sum_{i \in \mathcal{V}'_{n+1} \setminus \{j\}} y_{ji} = 0 \qquad \forall j \in \mathcal{T} \tag{5.6}$$

$$a_i^D + (t_{ij} + s_i)x_{ij} + \tau p_i - T^D(1 - x_{ij}) \leqslant a_j^D \qquad \forall i \in \mathcal{V}'_0, j \in \mathcal{V}'_{n+1}, i \neq j \tag{5.7}$$

$$a_i^S + (t_{ij} + \tau)y_{ij} - T^S(1 - y_{ij}) \leqslant a_j^S \qquad \forall i \in \mathcal{V}'_0, j \in \mathcal{V}'_{n+1}, i \neq j \tag{5.8}$$

$$0 \leqslant a_i^D \leqslant T^D \qquad \forall i \in \mathcal{V}'_{0,n+1} \tag{5.9}$$

$$0 \leqslant a_i^S \leqslant T^S \qquad \forall i \in \mathcal{V}'_{0,n+1} \tag{5.10}$$

$$e_i(1 - o_i) \leqslant a_i^D + \tau o_i \leqslant l_i \qquad \forall i \in \mathcal{V}'_{0,n+1} \tag{5.11}$$

$$e_i + (s_i + t_{ij})x_{ij} - T^D(2 - o_i - x_{ij}) \leqslant a_j^D \qquad \forall i \in \mathcal{V}'_0, j \in \mathcal{V}'_{n+1}, i \neq j \tag{5.12}$$

$$a_k^S - t_{ik} \leqslant a_j^D - t_{ij} - s_i o_i + T^D(2 - x_{ij} - y_{ik}) \qquad \forall i \in \mathcal{T}, j, k \in \mathcal{V}'_{n+1}, i \neq j, i \neq k \tag{5.13}$$

$$a_i^D + \tau + s_i(1 - o_i) - T^S(1 - y_{ik}) \leqslant a_k^S - t_{ik} \qquad \forall i \in \mathcal{T}, k \in \mathcal{V}'_{n+1}, i \neq k \tag{5.14}$$

$$p_j = \sum_{i \in \mathcal{V}'_0, i \neq j} y_{ij} \qquad \forall j \in \mathcal{T} \tag{5.15}$$

$$o_i \leqslant p_i \qquad \forall i \in \mathcal{C} \tag{5.16}$$

$$b_j \leqslant b_i - f_{ij}x_{ij} + F(1 - x_{ij}) \qquad \forall i \in \mathcal{V}'_0, j \in \mathcal{V}'_{n+1}, i \neq j \tag{5.17}$$

$$0 \leqslant b_i \leqslant F \qquad \forall i \in \mathcal{V}'_{0,n+1} \tag{5.18}$$

Sets	
$0, n+1$	instances of the depot
$\mathcal{C}$	set of customers
$\mathcal{D}'$	set of visits to dedicated transfer vertices
$\mathcal{C}^t$	subset of customers at which transfers are possible
$\mathcal{T}$	set of possible transfer locations, $\mathcal{T} = \mathcal{C}^t \cup \mathcal{D}'$
$\mathcal{V}_e$	set of visits to depot instance $n+1$
$\mathcal{V}'_0$	set of customers, visits to dedicated transfer vertices, and depot instance 0, $\mathcal{V}'_0 = \mathcal{C} \cup \mathcal{D}' \cup \{0\}$
$\mathcal{V}'_{n+1}$	set of customers, visits to dedicated transfer vertices, and visits to depot instance $n+1$, $\mathcal{V}'_{n+1} = \mathcal{C} \cup \mathcal{D}' \cup \mathcal{V}_e$
$\mathcal{V}'_{0,n+1}$	set of all vertices and visits, $\mathcal{V}'_{0,n+1} = \mathcal{C} \cup \mathcal{D}' \cup \{0\} \cup \mathcal{V}_e$
$\mathcal{A}$	set of arcs, $\mathcal{A} = \{(i,j) : i \in \mathcal{V}'_0, j \in \mathcal{V}'_{n+1}, i \neq j\}$
Parameters	
c_{ij}	travel cost on arc $(i,j) \in \mathcal{A}$
c^D	fixed cost DV
c^S	fixed cost SV
u_i	demand of vertex i ($u_i = 0$ if $i \notin \mathcal{C}$)
L^D	load capacity DV
L^S	load capacity SV
f_{ij}	fuel consumption on arc $(i,j) \in \mathcal{A}$
F	fuel capacity DV
t_{ij}	travel time on arc $(i,j) \in \mathcal{A}$
s_i	service time at vertex i ($s_i = 0$ if $i \notin \mathcal{C}$)
τ	transfer time
T^D	maximum route duration DV
T^S	maximum route duration SV
e_i	earliest start of service at vertex i
l_i	latest start of service at vertex i
Decision variables	
a_i^D	decision variable specifying the arrival time of a DV at vertex i
a_i^S	decision variable specifying the arrival time of an SV at vertex i
p_i	binary decision variable indicating if a transfer takes place at vertex i
o_i	binary decision variable specifying the precedence of transfer and service at vertex i (if $o_i = 1$, the transfer is executed before the service)
b_i	decision variable specifying the fuel level of a DV upon arriving at vertex i
q_i	decision variable specifying the load level of a DV upon arriving at vertex i
Ω_i	decision variable specifying the load level of an SV upon arriving at vertex i
x_{ij}	binary decision variable indicating if arc (i,j) is traversed by a DV
y_{ij}	binary decision variable indicating if arc (i,j) is traversed by an SV

Table 5.1.: Sets, parameters, and decision variables used in the VRPTWMD model.

$$q_j \leqslant q_i - u_i + L^D(1 - x_{ij}) \qquad \forall i \in \mathcal{V}'_0, j \in \mathcal{V}'_{n+1}, i \neq j \qquad (5.19)$$

$$0 \leqslant q_i \leqslant L^D \qquad \forall i \in \mathcal{V}'_{0,n+1} \qquad (5.20)$$

$$0 \leqslant \Omega_i \leqslant L^S \qquad \forall i \in \mathcal{V}'_{0,n+1} \qquad (5.21)$$

$$x_{ij}, y_{ij} \in \{0, 1\} \qquad \forall i \in \mathcal{V}'_0, j \in \mathcal{V}'_{n+1}, i \neq j \qquad (5.22)$$

$$p_i, o_i \in \{0, 1\} \qquad \forall i \in \mathcal{T} \qquad (5.23)$$

The goal of the VRPTWMD is to minimize the sum of the total travel cost and the fixed vehicle cost, expressed by the objective function (5.1).

Constraints (5.2) guarantee that each customer must be visited by a DV. Constraints (5.3) and (5.4) ensure optional visits to dedicated transfer vertices for DVs and to all possible transfer locations for SVs, respectively. Flow conservation for DVs and SVs is given by Constraints (5.5) and Constraints (5.6), respectively. Constraints (5.7) guarantee time feasibility for arcs traversed by DVs and Constraints (5.8) for arcs traversed by SVs. Constraints (5.9) and (5.10) limit the arrival times of DVs and SVs at each vertex to the respective maximum route durations. Constraints (5.11) and (5.12) ensure that service starts within the time window of each customer taking into account the task order at the respective vertex. Constraints (5.13) and (5.14) ensure the synchronization of DVs and SVs by equating the DV's transfer end time and the SV's departure time.

Constraints (5.15) guarantee that a transfer location is visited by an SV if a transfer is scheduled at the respective vertex. Constraints (5.16) ensure that the task order at customer vertices becomes relevant only if a transfer is scheduled. Constraints (5.17) and (5.18) guarantee fuel, and Constraints (5.19) and (5.20) load feasibility for arcs traveled by DVs without considering the possibility of replenishing the respective capacities. The range of the SVs' load is defined in Constraints (5.21). Finally, the binary decision variables are defined in Constraints (5.22) and (5.23).

5.2.2. Model VRPTWMD-F with Fuel Replenishment

In case of possible replenishments of the DVs' fuel capacity, i.e., $R = F$, Constraints (5.17) need to be replaced by the following constraints to obtain model VRPTWMD-F:

$$b_j \leqslant b_i - b'_i - f_{ij}x_{ij} + F(p_i + (1 - x_{ij})) \qquad \forall i \in \mathcal{V}'_0, j \in \mathcal{V}'_{n+1}, i \neq j \qquad (5.24)$$

$$b'_i \leqslant Fp_i \qquad \forall i \in \mathcal{V}'_0 \qquad (5.25)$$

$$b'_i \leqslant b_i \qquad \forall i \in \mathcal{V}'_0 \qquad (5.26)$$

$$b_i - F(1 - p_i) \leqslant b'_i \qquad \forall i \in \mathcal{V}'_0 \qquad (5.27)$$

$$\Omega_j \leqslant \Omega_i - (F - b_i) + L^S(1 - y_{ij}) \qquad \forall i \in \mathcal{V}'_0, j \in \mathcal{V}'_{n+1}, i \neq j \qquad (5.28)$$

Constraints (5.24) link the fuel levels at vertex i and its successor j. If no fuel transfer takes place at vertex i, the fuel level upon arrival at vertex j is calculated similarly to Constraints (5.17) in the base model, i.e., as the fuel level at vertex i reduced by the amount of fuel consumed on the arc between both vertices. Otherwise, the fuel level at vertex i is replenished to its full capacity before traveling to vertex j. Thus, in the second case, b_i has to be omitted in Constraints (5.24), which can be expressed as $b_i(1 - p_i)$. To avoid quadratic terms, we introduce the additional variable b'_i. Constraints (5.25)–(5.27) ensure that b_i is eliminated in Constraints (5.24), i.e., $b'_i = b_i$ if $p_i = 1$. Otherwise, $p_i = 0$ leads to the omission of F in the inequality and to $b'_i = 0$.

Constraints (5.28) ensure the load feasibility for SVs which leave a transfer location depending on the amount of fuel transferred to the DV.

5.2.3. Model VRPTWMD-L with Load Replenishment

If the load capacity of DVs can be replenished, i.e., $R = L^D$, we need to replace Constraints (5.19) in the base model by the following constraints to obtain model variant VRPTWMD-L:

$$q_i^s \leqslant q_i - \phi_i + L^D o_i - u_i \qquad \forall i \in \mathcal{V}_0' \qquad (5.29)$$
$$\phi_i \leqslant L^D o_i \qquad \forall i \in \mathcal{V}_0' \qquad (5.30)$$
$$\phi_i \leqslant q_i \qquad \forall i \in \mathcal{V}_0' \qquad (5.31)$$
$$q_i - L^D(1 - o_i) \leqslant \phi_i \qquad \forall i \in \mathcal{V}_0' \qquad (5.32)$$
$$q_j \leqslant q_i^s - \varsigma_i' + \varsigma_i'' + L^D(p_i - \xi_i) + L^D(1 - x_{ij}) \qquad \forall i \in \mathcal{V}_0', j \in \mathcal{V}_{n+1}', i \neq j \qquad (5.33)$$
$$\varsigma_i' \leqslant L^D p_i \qquad \forall i \in \mathcal{V}_0' \qquad (5.34)$$
$$\varsigma_i' \leqslant q_i^s \qquad \forall i \in \mathcal{V}_0' \qquad (5.35)$$
$$q_i^s - L^D(1 - p_i) \leqslant \varsigma_i' \qquad \forall i \in \mathcal{V}_0' \qquad (5.36)$$
$$\varsigma_i'' \leqslant L^D o_i \qquad \forall i \in \mathcal{V}_0' \qquad (5.37)$$
$$\varsigma_i'' \leqslant q_i^s \qquad \forall i \in \mathcal{V}_0' \qquad (5.38)$$
$$q_i^s - L^D(1 - o_i) \leqslant \varsigma_i'' \qquad \forall i \in \mathcal{V}_0' \qquad (5.39)$$
$$\xi_i \leqslant o_i \qquad \forall i \in \mathcal{V}_0' \qquad (5.40)$$
$$\xi_i \leqslant p_i \qquad \forall i \in \mathcal{V}_0' \qquad (5.41)$$
$$p_i - (1 - o_i) \leqslant \xi_i \qquad \forall i \in \mathcal{V}_0' \qquad (5.42)$$
$$\Omega_j \leqslant \Omega_i - (L^D - \phi_i - q_i^s + \varsigma_i'') + L^S(1 - y_{ij}) \qquad \forall i \in \mathcal{V}_0', j \in \mathcal{V}_{n+1}', i \neq j \qquad (5.43)$$

While in the base model, the load level of a DV monotonically decreases along a route depending on the delivered customer demands, we now need to additionally consider scheduled transfer operations that restore the current load level of a DV to L^D. In contrast to fuel, load is not consumed along arcs but upon service at customers. The evaluation of the load feasibility thus depends on the order of tasks executed at a customer. Therefore, we introduce variable q_i^s to specify the load level of a DV after service at vertex i.

Constraints (5.29) define q_i^s depending on the load level upon arrival q_i and the order of tasks o_i at vertex i. More precisely, if no load transfer takes place or the service is performed before the transfer ($o_i = 0$), the load level upon arrival q_i is reduced by the demand u_i of customer i to obtain q_i^s. Otherwise, if $o_i = 1$, the DV's load level is fully replenished before the customer is served. Thus, in this case, the load level after service corresponds to the load capacity L^D reduced by the demand and q_i has to be omitted in Constraints (5.29) which can be expressed as $q_i(1 - o_i)$. To linearize this expression, we introduce variable ϕ_i. Depending on o_i, Constraints (5.30)–(5.32) control the consideration of ϕ_i in Constraints (5.29).

Constraints (5.33) then link the load level after service at the current vertex i to the load level upon arrival at its successor j. Here, we need to consider the task order at vertex i again. If no transfer takes place or the service is performed after the transfer ($p_i = 0 \vee (p_i = 1 \wedge o_i = 1)$), the load level

upon arrival at vertex j is equal to the load level after service q_i^s. Thus, we need to link q_i^s to p_i and o_i which may be expressed as $q_i^s(1 - p_i + o_i)$. We again linearize the resulting quadratic terms by introducing the additional auxiliary variables ς_i' and ς_i'', which are defined in Constraints (5.34)–(5.39). Otherwise, if the service is performed before the transfer ($p_i = 1 \wedge o_i = 0$), q_j corresponds to the fully replenished load capacity L^D. To linearize the quadratic expression $L^D p_i(1-o_i)$, we introduce variable ξ_i which is defined in Constraints (5.40)–(5.42).

Finally, Constraints (5.43) ensure the load feasibility of SVs depending on the task order at transfer vertex i. More precisely, either the load level upon arrival ($o_i = 1$) or the load level after service ($o_i = 0$) is considered to determine the amount of load transferred to the DV. The additional variables used in the VRPTWMD-L model are summarized in Table 5.2.

Decision variables	
q_i^s	decision variable specifying the load level of a DV after serving vertex i
ϕ_i	auxiliary variable used to linearize the link between q_i and o_i
ς_i'	auxiliary variable used to linearize the link between q_i^s and p_i
ς_i''	auxiliary variable used to linearize the link between q_i^s and o_i
ξ_i	auxiliary variable used to linearize the link between p_i and o_i

Table 5.2.: Additional decision variables used in model variant VRPTWMD-L.

5.3. Adaptive Large Neighborhood Search with Path Relinking for the VRPTWMD

In this section, we describe our hybrid solution method of ALNS and PR for the VRPTWMD. A pseudocode overview of ALNS-PR is given in Figure 5.2.

```
S ← generateInitialSolution()
Initialize best solution S* ← S
Initialize set of elite solutions E ← ∅
while number of iterations without improvement ⩽ ω do
  if S already visited or set of elite solutions E not completely filled then
    S' ← performALNS(S)
  else
    S' ← performPathRelinking(S, E)
  end if
  S' ← performLocalSearch(S')
  E ← evaluateInclusion(S')
  if acceptSA(S', S) then
    S ← S'
    if S' improves on S* then
      S* ← S'
    end if
  end if
  updatePenaltyFactors(S)
end while
```

Figure 5.2.: Pseudocode of the ALNS-PR algorithm.

During the search, we allow infeasible solutions, i.e., solutions not respecting all constraints, but handle them by means of a dynamic penalty mechanism. More precisely, we transform constraint violations into penalty costs by multiplying the respective value with a dedicated penalty factor for

each constraint that is dynamically updated during the search. The penalty mechanism and the determination of constraint violations are described in Section 5.3.1.

To generate an initial solution, we employ a construction heuristic that is able to handle violations of R by inserting transfers and constructing initial SV routes (Section 5.3.2). To avoid the generation of infeasible precedence relations when inserting a transfer, we make use of a cycle-detection mechanism which is detailed in Section 5.3.3. We subsequently try to improve the resulting solution $\mathcal{S}$ by applying the local search implementation described in Section 5.3.4.

The following improvement phase is repeated until a maximum number ω of iterations without improvement of the current best solution $\mathcal{S}^*$ is reached. In each iteration, we decide whether to apply the ALNS or the PR component to the current solution $\mathcal{S}$. Our PR component relies on a dynamic set $\mathcal{E}$ of elite solutions that are recombined with new solutions found during the search. Therefore, if the current solution $\mathcal{S}$ has already been encountered before in the search, or the elite set is not completely filled, i.e., $|\mathcal{E}|$ is smaller than the maximum size λ of the elite set, we apply our ALNS component to $\mathcal{S}$ (Section 5.3.5). If the ALNS component has been applied in the current iteration, we update the selection probabilities of its components according to the adaptive mechanism described in Section 5.3.5. Otherwise, if $\mathcal{S}$ represents a new solution and the elite set $\mathcal{E}$ is filled, PR is applied between $\mathcal{S}$ and the solutions contained in $\mathcal{E}$ (Section 5.3.6).

In each case, we subsequently aim at improving the solution returned by the selected component via local search. The resulting solution $\mathcal{S}'$ is then evaluated regarding its inclusion in the set of elite solutions $\mathcal{E}$ with the goal of balancing quality and diversity among the elite solutions. As long as $\mathcal{E}$ is not completely filled, any feasible solution is added. Otherwise, $\mathcal{S}'$ is included and replaces the worst solution contained in $\mathcal{E}$ if it is feasible, and

- improves on the best solution in $\mathcal{E}$, or
- an SA-based comparison with the worst solution in $\mathcal{E}$ similar to the mechanism described in Section 5.3.7 turns out in favor of the candidate solution $\mathcal{S}'$, and the inclusion of $\mathcal{S}'$ does not deteriorate the average diversity among all elite solutions. We measure the diversity between solutions in terms of the number of arcs exclusively contained in one of both solutions (see Section 5.3.6).

Our SA acceptance mechanism then decides if $\mathcal{S}'$ replaces $\mathcal{S}$ as the current solution for the subsequent iteration (Section 5.3.7). After μ iterations without improvement, we reset $\mathcal{S}$ to $\mathcal{S}^*$. For the purpose of diversification, we set the SA temperature back to its initial value and $\mathcal{S}$ to a solution randomly chosen from the elite set after ϵ solution resets. The probability of an elite solution to be selected is proportional to the diversity between this solution and $\mathcal{S}^*$.

Finally, the penalty factors are updated depending on the number of consecutive ALNS-PR iterations during which the respective constraint has been satisfied or violated in solution $\mathcal{S}'$ (Section 5.3.1).

5.3.1. Solution Evaluation and Penalty Mechanism

For tightly constrained problems like the VRPTWMD, it is beneficial to temporarily tolerate constraint violations to be able to explore the solution space more flexibly. Therefore, we allow infeasible solutions during the search but impose dynamic penalty costs for violating the respective constraints. A solution

$\mathcal{S}$ is then evaluated according to a generalized cost function $f_{gen}(\mathcal{S})$:

$$f_{gen}(\mathcal{S}) = f(\mathcal{S}) + \delta^{LD}\upsilon^{LD}(\mathcal{S}) + \delta^{LS}\upsilon^{LS}(\mathcal{S}) + \delta^{F}\upsilon^{F}(\mathcal{S}) + \delta^{TW}\upsilon^{TW}(\mathcal{S}),$$

where $f(\mathcal{S})$ denotes the objective function value of solution $\mathcal{S}$, δ^{LD}, δ^{LS}, δ^{F}, and δ^{TW} the penalty factors for DV load, SV load, fuel, and time window violations, and $\upsilon^{LD}(\mathcal{S})$, $\upsilon^{LS}(\mathcal{S})$, $\upsilon^{F}(\mathcal{S})$, and $\upsilon^{TW}(\mathcal{S})$ the current DV load, SV load, fuel, and time window violations in solution $\mathcal{S}$, respectively.

Starting from the initial value δ^0, each penalty factor is dynamically varied during the search within the interval $[\delta^{min}, \delta^{max}]$. After η^+ consecutive ALNS-PR iterations with $\mathcal{S}'$ violating a certain constraint, the associated penalty factor is increased by multiplying by factor δ^{update}. Analogously, after η^- iterations without violating a certain constraint, the respective penalty factor is decreased by dividing by δ^{update}.

To efficiently evaluate potential moves with respect to the associated changes in constraint violations, solution methods for VRPs rely on the quick retrieval of the required solution information. This is particularly important for the VRPTWMD, where the interdependence of routes due to transfers may otherwise necessitate the recalculation of the entire solution. We store vertex-specific information that allows us to evaluate changes in resource and time window violations in $\mathcal{O}(1)$.

Before we detail the evaluation of constraint violations, we first introduce some necessary notation: Let a sequence of vertices $\langle 0, i, ..., n+1 \rangle$ define a route, where 0 and $n+1$ correspond to instances of the same depot representing the start and end of the route, respectively. We denote the direct predecessor of vertex i as i^- and the direct successor of i as i^+. Moreover, let $\mathcal{V}_r$ denote the set of vertices that are visited by route r. In addition, we introduce the set $\mathcal{D}^D$ of visits to dedicated transfer vertices contained in DV routes, the set $\mathcal{C}^+ \subseteq \mathcal{C}^t$ of customers at which a transfer is currently performed, the set $\mathcal{C}^- = \mathcal{C} \backslash \mathcal{C}^+$ of customers at which no transfer is currently performed, and the set $\mathcal{T}^S$ of SV visits. A solution $\mathcal{S}$ is defined as the union of the set of DV routes $\mathcal{S}^D$ and the set of SV routes $\mathcal{S}^S$, i.e., $\mathcal{S} = \mathcal{S}^D \cup \mathcal{S}^S$. Finally, in our algorithm, we use the following operators to modify a given solution:

- Insertion of a customer $i \in \mathcal{C}^-$
- Insertion of a dedicated transfer, i.e., insertion of the associated DV visit $i \in \mathcal{D}^D$ and SV visit $i' \in \mathcal{T}^S$
- Insertion of a customer $i \in \mathcal{C}^+$, i.e., insertion of i and the associated SV visit $i' \in \mathcal{T}^S$
- Assignment of a transfer status to a customer $i \in \mathcal{C}^t \backslash \mathcal{C}^+$, i.e., $\mathcal{C}^+ \leftarrow \mathcal{C}^+ \cup \{i\} \wedge \mathcal{C}^- \leftarrow \mathcal{C}^- \backslash \{i\}$ and insertion of the associated SV visit $i' \in \mathcal{T}^S$
- Removal of a customer $i \in \mathcal{C}^-$
- Removal of a dedicated transfer, i.e., removal of the associated DV visit $i \in \mathcal{D}^D$ and SV visit $i' \in \mathcal{T}^S$
- Removal of a customer $i \in \mathcal{C}^+$, i.e., removal of i and the associated SV visit $i' \in \mathcal{T}^S$
- Removal of the transfer status of a customer $i \in \mathcal{C}^+$, i.e., $\mathcal{C}^- \leftarrow \mathcal{C}^- \cup \{i\} \wedge \mathcal{C}^+ \leftarrow \mathcal{C}^+ \backslash \{i\}$ and removal of the associated SV visit $i' \in \mathcal{T}^S$

In the following descriptions of the evaluation of constraint violations, we always refer to the general case of performing transfers at customer locations. However, note that transfers at dedicated transfer vertices may be evaluated in similar fashion considering that $s_i = 0$ and $u_i = 0$ for each dedicated transfer visit $i \in \mathcal{D}^D$ thus rendering the task order irrelevant.

5.3.1.1. Calculation of Fuel Violations

To efficiently evaluate fuel violations in DV routes, we adopt the idea introduced in Schneider, Stenger, and Goeke (2014) for the E-VRPTW and define the following two variables for each vertex i: $f_i^{\rightarrow}$ represents the amount of fuel consumed since the last fuel transfer or the depot (in case no fuel transfer is scheduled or $R \neq F$) and $f_i^{\leftarrow}$ is the amount of fuel required to reach the next fuel transfer or the depot. Both variables are calculated as follows:

$$f_i^{\rightarrow} = \begin{cases} 0 & \text{if } i = 0, \\ f_{i^- i} & \text{else if } i^- = 0 \vee (p_{i^-} = 1 \wedge R = F), \\ f_{i^-}^{\rightarrow} + f_{i^- i} & \text{else.} \end{cases}$$

$$f_i^{\leftarrow} = \begin{cases} 0 & \text{if } i = n+1, \\ f_{i i^+} & \text{else if } i^+ = n+1 \vee (p_{i^+} = 1 \wedge R = F), \\ f_{i^+}^{\leftarrow} + f_{i i^+} & \text{else.} \end{cases}$$

The fuel violation of a solution $\mathcal{S}$ can then be determined as the sum of the individual violations for each fuel segment, i.e., at each fuel transfer and upon returning to the depot:

$$\upsilon^F(\mathcal{S}) = \sum_{r \in \mathcal{S}^D} \left(\sum_{i \in \mathcal{V}_r : p_i = 1 \wedge R = F} \max(f_i^{\rightarrow} - F, 0) + \max(f_{n+1}^{\rightarrow} - F, 0) \right).$$

Figure 5.3 shows an example route containing six vertices (ignoring customer 7 for now) to illustrate the calculation of the variables $f_i^{\rightarrow}$ and $f_i^{\leftarrow}$ assuming that $R = F$. Fuel consumptions between vertices are given on the arcs. Moreover, we assume $F = 10$, and at each transfer and the end depot, we indicate if the fuel capacity is satisfied (green) or exceeded (red) in the respective segment.

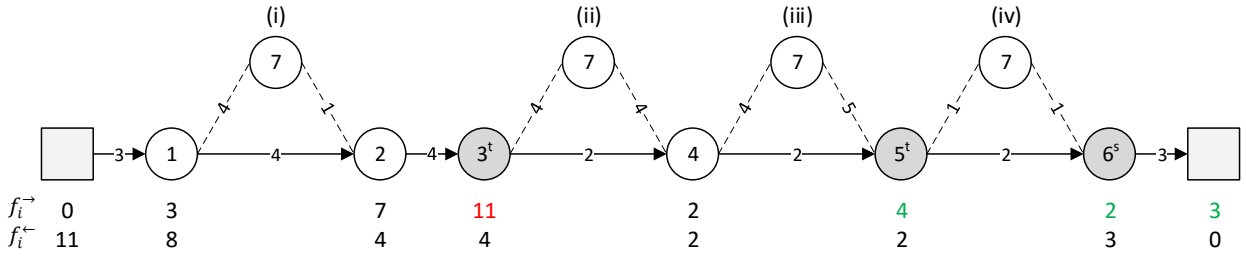


Figure 5.3.: Example to illustrate the fuel evaluation for different cases of vertex insertions.

The calculation of the new fuel violation for a route segment resulting from the insertion ν_j^+ of a vertex j between vertices i and k requires to differentiate between the insertion of a customer ($j \in \mathcal{C}^-$) and the insertion ($j \in \mathcal{D}^D \vee j \in \mathcal{C}^+$) or assignment ($j \in \mathcal{C}^t \backslash \mathcal{C}^+$) of a transfer if $R = F$. If $j \in \mathcal{C}^-$, depending on whether fuel transfers are scheduled at the potential new predecessor i or the potential new successor k of customer j to be inserted, the new fuel violation of the corresponding route segment

is defined as:

$$\upsilon^F(\nu_j^+) = \begin{cases} \max(f_i^{\rightarrow} + f_k^{\leftarrow} + f_{ij} + f_{jk} - F, 0) & \text{if } (p_i = 0 \wedge p_k = 0) \vee R \neq F, \\ \max(f_k^{\leftarrow} + f_{ij} + f_{jk} - F, 0) & \text{else if } p_i = 1 \wedge p_k = 0, \\ \max(f_i^{\rightarrow} + f_{ij} + f_{jk} - F, 0) & \text{else if } p_i = 0 \wedge p_k = 1, \\ \max(f_{ij} + f_{jk} - F, 0) & \text{else if } p_i = 1 \wedge p_k = 1. \end{cases}$$

We illustrate the cases above by evaluating the insertion of customer 7 for different insertion positions as shown in Figure 5.3. More precisely, we evaluate the insertion of customer 7 between (i) customers 1 and 2, (ii) transfer vertex 3^t and customer 4, (iii) customer 4 and transfer vertex 5^t, and (iv) transfer vertices 5^t and 6^t. Assuming that $R = F$, the corresponding violations are then calculated as follows:

$$\upsilon^F(\nu_7^+) = \begin{cases} \text{(i):} & \max(f_1^{\rightarrow} + f_2^{\leftarrow} + f_{17} + f_{72} - 10, 0) = \max(3 + 4 + 4 + 1 - 10, 0) = 2, \\ \text{(ii):} & \max(f_4^{\leftarrow} + f_{37} + f_{74} - 10, 0) = \max(2 + 4 + 4 - 10, 0) = 0, \\ \text{(iii):} & \max(f_4^{\rightarrow} + f_{47} + f_{75} - 10, 0) = \max(2 + 4 + 5 - 10, 0) = 1, \\ \text{(iv):} & \max(f_{57} + f_{76} - 10, 0) = \max(1 + 1 - 10, 0) = 0. \end{cases}$$

If $R \neq F$, the new fuel violation caused by the insertion of a transfer (of the other resource) is evaluated similarly as in the customer case. Otherwise, if $R = F$, the insertion ($j \in \mathcal{D}^D \vee j \in \mathcal{C}^+$) or assignment ($j \in \mathcal{C}^t \backslash \mathcal{C}^+$) of a transfer splits the respective route segment into two segments and the associated fuel consumptions need to be compared separately to the fuel capacity F. Then, we need to distinguish the following four cases with respect to the existence of fuel transfers at vertices i and k:

$$\upsilon^F(\nu_j^+) = \begin{cases} \max(f_i^{\rightarrow} + f_k^{\leftarrow} + f_{ij} + f_{jk} - F, 0) & \text{if } R \neq F, \\ \max(f_i^{\rightarrow} + f_{ij} - F, 0) + \max(f_{jk} + f_k^{\leftarrow} - F, 0) & \text{else if } p_i = 0 \wedge p_k = 0, \\ \max(f_{ij} - F, 0) + \max(f_{jk} + f_k^{\leftarrow} - F, 0) & \text{else if } p_i = 1 \wedge p_k = 0, \\ \max(f_i^{\rightarrow} + f_{ij} - F, 0) + \max(f_{jk} - F, 0) & \text{else if } p_i = 0 \wedge p_k = 1, \\ \max(f_{ij} - F, 0) + \max(f_{jk} - F, 0) & \text{else if } p_i = 1 \wedge p_k = 1. \end{cases}$$

We use Figure 5.3 to illustrate the calculation for each case, now assuming that a fuel transfer is scheduled at customer 7. Note that in case of fuel, the task order at a customer does not affect the calculation of the fuel violation (indicated with $7^{s/t}$). The resulting fuel violations are given as follows:

$$\upsilon^F(\nu_{7^{s/t}}^+) = \begin{cases} \text{(i):} & \max(f_1^{\rightarrow} + f_{17} - 10, 0) + \max(f_{72} + f_2^{\leftarrow} - 10, 0) \\ & = \max(3 + 4 - 10, 0) + \max(1 + 4 - 10, 0) = 0, \\ \text{(ii):} & \max(f_{37} - 10, 0) + \max(f_{74} + f_4^{\leftarrow} - 10, 0) \\ & = \max(4 - 10, 0) + \max(4 + 2 - 10, 0) = 0, \\ \text{(iii):} & \max(f_4^{\rightarrow} + f_{47} - 10, 0) + \max(f_{75} - 10, 0) \\ & = \max(2 + 4 - 10, 0) + \max(5 - 10, 0) = 0, \\ \text{(iv):} & \max(f_{57} - 10, 0) + \max(f_{76} - 10, 0) \\ & = \max(1 - 10, 0) + \max(1 - 10, 0) = 0. \end{cases}$$

To determine the new fuel violation for a route segment after removing vertex i, we need to distinguish

between the actual removal of a vertex and the conversion of a transfer vertex into a regular customer, i.e., only the removal of its transfer status. Both cases differ with respect to the fuel consumption required to connect the two partial routes adjacent to the route position being modified. We define variable f_i^* that specifies the respective fuel consumption to consider:

$$f_i^* = \begin{cases} f_{i^-i^+} & \text{if actual vertex removal,} \\ f_{i^-i} + f_{ii^+} & \text{if status conversion.} \end{cases}$$

This value can now be used in the calculation of the new fuel violation in a segment caused by removal operation ν_i^- which additional needs to distinguish four cases depending on whether fuel transfers are scheduled at the predecessor and successor of vertex i:

$$\upsilon^F(\nu_i^-) = \begin{cases} \max(f_{i^-}^{\rightarrow} + f_{i^+}^{\leftarrow} + f_i^* - F, 0) & \text{if } (p_{i^-} = 0 \wedge p_{i^+} = 0) \vee R \neq F, \\ \max(f_{i^+}^{\leftarrow} + f_i^* - F, 0) & \text{else if } p_{i^-} = 1 \wedge p_{i^+} = 0, \\ \max(f_{i^-}^{\rightarrow} + f_i^* - F, 0) & \text{else if } p_{i^-} = 0 \wedge p_{i^+} = 1, \\ \max(f_i^* - F, 0) & \text{else if } p_{i^-} = 1 \wedge p_{i^+} = 1. \end{cases}$$

For the purpose of illustration, Figure 5.4 shows an extended example route for which we want to evaluate the removal of (i) customer 7 between customers 1 and 2, (ii) customer 8 between transfer vertex 3^t and customer 4, (iii) customer 9 between customer 4 and transfer vertex 5^t, and (iv) customer 10 between transfer vertices 5^t and 6^s. The corresponding fuel violations are calculated as follows:

$$\upsilon^F(\nu_i^-) = \begin{cases} \text{(i) i = 7:} & \max(f_1^{\rightarrow} + f_2^{\leftarrow} + f_{12} - 10, 0) = \max(3 + 4 + 4 - 10, 0) = 1, \\ \text{(ii) i = 8:} & \max(f_4^{\leftarrow} + f_{34} - 10, 0) = \max(9 + 2 - 10, 0) = 1, \\ \text{(iii) i = 9:} & \max(f_4^{\rightarrow} + f_{45} - 10, 0) = \max(8 + 2 - 10, 0) = 0, \\ \text{(iv) i = 10:} & \max(f_{56} - 10, 0) = \max(2 - 10, 0) = 0. \end{cases}$$

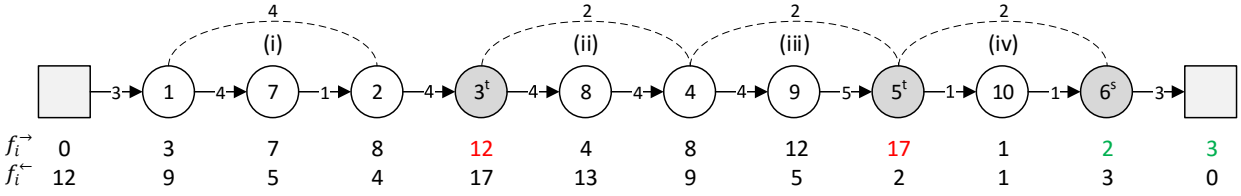


Figure 5.4.: Example to illustrate the fuel evaluation for different cases of vertex removals.

For any operation, the change in fuel violation is then calculated as the difference between the fuel violation caused by the operation and the current fuel violation in the respective fuel segment.

5.3.1.2. Calculation of Load Violations for DVs

To evaluate load violations in DV routes, we define $l_i^{\rightarrow}$ as the cumulated demand delivered since the last load transfer or the depot and $l_i^{\leftarrow}$ as the cumulated demand to be delivered until the next load transfer or the depot for each vertex i. While fuel is only consumed along arcs, the load capacity L^D reduces upon performing service at a customer. Thus, the calculation of both load variables requires the consideration of task orders at the current vertex and at its predecessor and successor if load

transfers are scheduled to correctly assign the respective demands to the adjacent route segments:

$$l_i^{\rightarrow} = \begin{cases} 0 & \text{if } i = 0, \\ l_{i^-}^{\rightarrow} + u_i & \text{else if } (p_{i^-} = 0 \wedge p_i = 0) \vee R \neq L^D, \\ l_{i^-}^{\rightarrow} + u_i(1 - o_i) & \text{else if } p_{i^-} = 0, \\ u_{i^-}o_{i^-} + u_i(1 - o_i) & \text{else.} \end{cases}$$

$$l_i^{\leftarrow} = \begin{cases} 0 & \text{if } i = n + 1, \\ l_{i^+}^{\leftarrow} + u_i & \text{else if } (p_{i^+} = 0 \wedge p_i = 0) \vee R \neq L^D, \\ l_{i^+}^{\leftarrow} + u_i o_i & \text{else if } p_{i^+} = 0 \wedge p_i = 1, \\ u_{i^+}(1 - o_{i^+}) + u_i & \text{else if } p_{i^+} = 1 \wedge p_i = 0, \\ u_{i^+}(1 - o_{i^+}) + u_i o_i & \text{else if } p_{i^+} = 1 \wedge p_i = 1. \end{cases}$$

The total load violation in the DV routes of a solution $\mathcal{S}$ can then be calculated by summing up the individual violations for each load segment, i.e., at every load transfer and upon returning to the depot:

$$\upsilon^{LD}(\mathcal{S}) = \sum_{r \in \mathcal{S}^D} \left(\sum_{i \in \mathcal{V}_r : p_i = 1 \wedge R = L^D} \max(l_i^{\rightarrow} - L^D, 0) + \max(l_{n+1}^{\rightarrow} - L^D, 0) \right).$$

Figure 5.5 shows an example to illustrate the calculation of $l_i^{\rightarrow}$ and $l_i^{\leftarrow}$ based on a load capacity $L^D = 10$ and the assumption of $R = L^D$. In the displayed situation, all segments are load-feasible. However, note that switching the task order at vertex 3^t would lead to a violation of the load capacity due to the additional consideration of its demand in the preceding load segment instead of the succeeding (less utilized) one.

The calculation of the new load violation for a load segment after inserting a vertex also requires the consideration of task orders if load transfers take place at the potential new predecessor and successor. Analogous to the fuel evaluation, we additionally need to distinguish between the insertion of a customer ($j \in \mathcal{C}^-$) and the insertion ($j \in \mathcal{D}^D \vee j \in \mathcal{C}^+$) or assignment ($j \in \mathcal{C}^t \backslash \mathcal{C}^+$) of a transfer in case of $R = L^D$. The new load violation for a load segment caused by inserting a customer $j \in \mathcal{C}^-$ between vertices i and k is defined as:

$$\upsilon^{LD}(\nu_j^+) = \begin{cases} \max(l_i^{\rightarrow} + l_k^{\leftarrow} + u_j - L^D, 0) & \text{if } (p_i = 0 \wedge p_k = 0) \vee R \neq L^D, \\ \max(u_i o_i + l_k^{\leftarrow} + u_j - L^D, 0) & \text{else if } p_i = 1 \wedge p_k = 0, \\ \max(l_i^{\rightarrow} + u_k(1 - o_k) + u_j - L^D, 0) & \text{else if } p_i = 0 \wedge p_k = 1, \\ \max(u_i o_i + u_k(1 - o_k) + u_j - L^D, 0) & \text{else if } p_i = 1 \wedge p_k = 1. \end{cases}$$

Using Figure 5.5a, we evaluate the insertion of customer 7 between (i) customers 1 and 2, (ii) transfer vertex 3^t and customer 4, (iii) customer 4 and transfer vertex 5^s, and (iv) transfer vertices 5^s and 6^t,

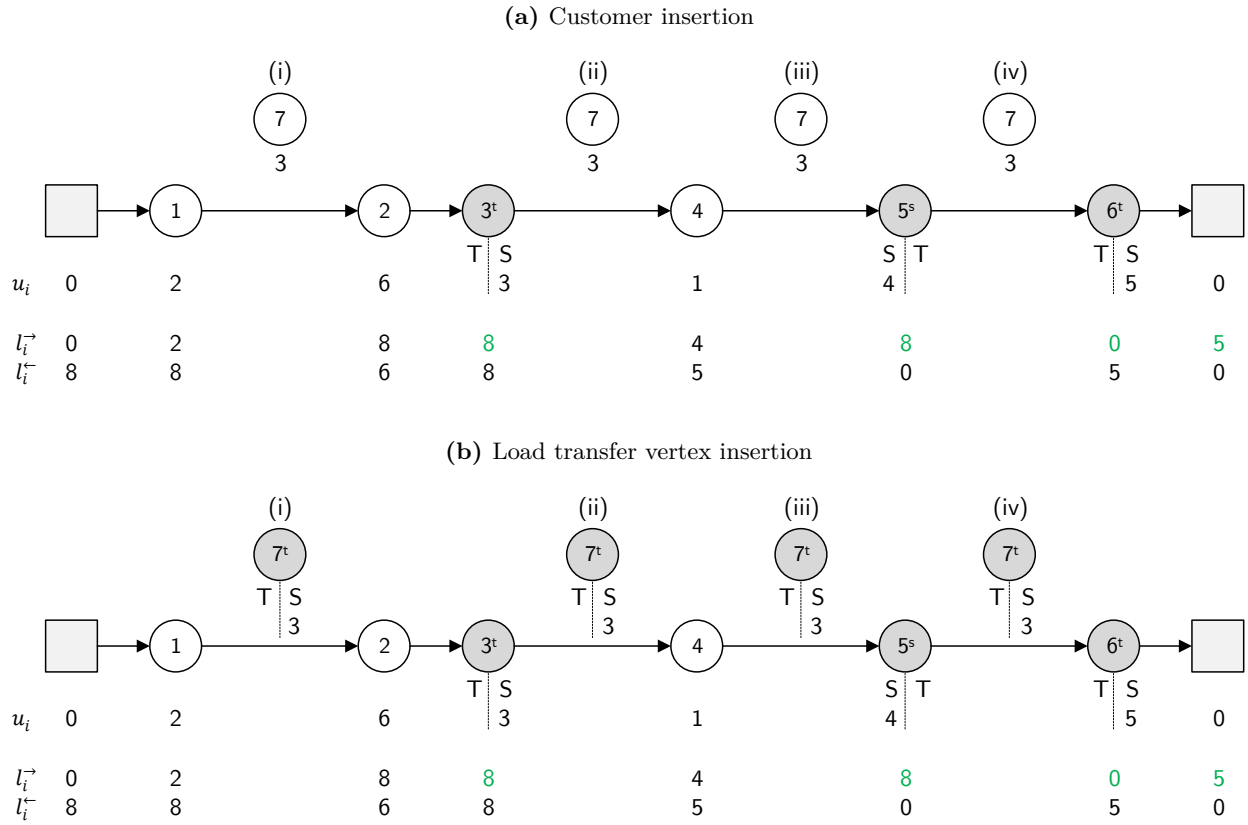


Figure 5.5.: Example to illustrate the load evaluation for different cases of vertex insertions.

this time with respect to potential load violations:

$$\upsilon^{LD}(\nu_7^+) = \begin{cases} \text{(i):} & \max(l_1^{\rightarrow} + l_2^{\leftarrow} + u_7 - 10, 0) = \max(2 + 6 + 3 - 10, 0) = 1, \\ \text{(ii):} & \max(u_3 o_3 + l_4^{\leftarrow} + u_7 - 10, 0) = \max(3 + 5 + 3 - 10, 0) = 1, \\ \text{(iii):} & \max(l_4^{\rightarrow} + u_5(1 - o_5) + u_7 - 10, 0) = \max(4 + 4 + 3 - 10, 0) = 1, \\ \text{(iv):} & \max(u_5 o_5 + u_6(1 - o_6) + u_7 - 10, 0) = \max(0 + 0 + 3 - 10, 0) = 0. \end{cases}$$

If $R \neq L^D$, the load violation for both customer and transfer insertion/assignment is evaluated in identical fashion. Otherwise, if $R = L^D$, the insertion ($j \in \mathcal{D}^D \vee j \in \mathcal{C}^+$) or assignment ($j \in \mathcal{C}^t \backslash \mathcal{C}^+$) of a transfer splits the respective load segment in two segments which have to be evaluated separately with respect to L^D. The order of tasks at the vertex j to be inserted determines which of both load segments its demand is assigned to:

$$\upsilon^{LD}(\nu_j^+) = \begin{cases} \max(l_i^{\rightarrow} + l_k^{\leftarrow} + u_j - L^D, 0) & \text{if } R \neq L^D, \\ \max(l_i^{\rightarrow} + u_j(1 - o_j) - L^D, 0) + \max(u_j o_j + l_k^{\leftarrow} - L^D, 0) & \text{else if } p_i = 0 \wedge p_k = 0, \\ \max(u_i o_i + u_j(1 - o_j) - L^D, 0) + \max(u_j o_j + l_k^{\leftarrow} - L^D, 0) & \text{else if } p_i = 1 \wedge p_k = 0, \\ \max(l_i^{\rightarrow} + u_j(1 - o_j) - L^D, 0) + \max(u_j o_j + u_k(1 - o_k) - L^D, 0) & \text{else if } p_i = 0 \wedge p_k = 1, \\ \max(u_i o_i + u_j(1 - o_j) - L^D, 0) + \max(u_j o_j + u_k(1 - o_k) - L^D, 0) & \text{else if } p_i = 1 \wedge p_k = 1. \end{cases}$$

Now, assume that a load transfer is scheduled at customer 7 which is performed before the service ($o_7 = 1$) as shown in Figure 5.5b. The resulting load violations for the different insertion positions are

then calculated as:

$$\upsilon^{LD}(\nu_{7^t}^{+}) = \begin{cases} \text{(i):} & \max(l_1^{\rightarrow} + u_7(1-o_7) - 10, 0) + \max(u_7 o_7 + l_2^{\leftarrow} - 10, 0) \\ & = \max(2 + 0 - 10, 0) + \max(3 + 6 - 10, 0) = 0, \\ \text{(ii):} & \max(u_3 o_3 + u_7(1-o_7) - 10, 0) + \max(u_7 o_7 + l_4^{\leftarrow} - 10, 0) \\ & = \max(3 + 0 - 10, 0) + \max(3 + 5 - 10, 0) = 0, \\ \text{(iii):} & \max(l_4^{\rightarrow} + u_7(1-o_7) - 10, 0) + \max(u_7 o_7 + u_5(1-o_5) - 10, 0) \\ & = \max(4 + 0 - 10, 0) + \max(3 + 4 - 10, 0) = 0, \\ \text{(iv):} & \max(u_5 o_5 + u_7(1-o_7) - 10, 0) + \max(u_7 o_7 + u_6(1-o_6) - 10, 0) \\ & = \max(0 - 10, 0) + \max(3 + 0 - 10, 0) = 0. \end{cases}$$

When removing a vertex, we need to distinguish between the actual removal of a vertex and the removal of its transfer status analogous to the fuel evaluation. Therefore, we define variable l_i^* to specify the additional load to consider to connect both partial routes:

$$l_i^* = \begin{cases} 0 & \text{if actual vertex removal,} \\ u_i & \text{if status conversion.} \end{cases}$$

The calculation of the new load violation for a load segment after the removal ν_i^- of a vertex i then also requires the consideration of task orders if load transfers take place at the adjacent vertices:

$$\upsilon^{LD}(\nu_i^-) = \begin{cases} \max(l_{i^-}^{\rightarrow} + l_{i^+}^{\leftarrow} + l_i^* - L^D, 0) & \text{if } (p_{i^-} = 0 \wedge p_{i^+} = 0) \vee R \neq L^D, \\ \max(u_{i^-} o_{i^-} + l_{i^+}^{\leftarrow} + l_i^* - L^D, 0) & \text{else if } p_{i^-} = 1 \wedge p_{i^+} = 0, \\ \max(l_{i^-}^{\rightarrow} + u_{i^+}(1-o_{i^+}) + l_i^* - L^D, 0) & \text{else if } p_{i^-} = 0 \wedge p_{i^+} = 1, \\ \max(u_{i^-} o_{i^-} + u_{i^+}(1-o_{i^+}) + l_i^* - L^D, 0) & \text{else if } p_{i^-} = 1 \wedge p_{i^+} = 1. \end{cases}$$

We use Figure 5.6 to illustrate the load evaluation for the removal of (i) customer 7 between customers 1 and 2, (ii) customer 8 between transfer vertex 3^t and customer 4, (iii) customer 9 between customer 4 and transfer vertex 5^s, and (iv) customer 10 between transfer vertices 5^s and 6^t, assuming that $R = L^D$:

$$\upsilon^{LD}(\nu_i^-) = \begin{cases} \text{(i) i = 7:} & \max(l_1^{\rightarrow} + l_2^{\leftarrow} + 0 - 10, 0) = \max(2 + 6 + 0 - 10, 0) = 0, \\ \text{(ii) i = 8:} & \max(u_3 o_3 + l_4^{\leftarrow} + 0 - 10, 0) = \max(3 + 8 + 0 - 10, 0) = 1, \\ \text{(iii) i = 9:} & \max(l_4^{\rightarrow} + u_5(1-o_5) + 0 - 10, 0) = \max(7 + 4 + 0 - 10, 0) = 1, \\ \text{(iv) i = 10:} & \max(u_5 o_5 + u_6(1-o_6) + 0 - 10, 0) = \max(0 + 0 + 0 - 10, 0) = 0. \end{cases}$$

For any operation, the change in load violation is then calculated as the difference between the load violation caused by the operation and the current load violation in the respective load segment.

5.3.1.3. Calculation of Load Violations for SVs

We evaluate load violations in SV routes similar to load violations in DV routes. However, while the customer demands to be delivered by the DVs are fixed and given by the problem instance, the demands that need to be satisfied by the SVs are dynamically created depending on the DVs'

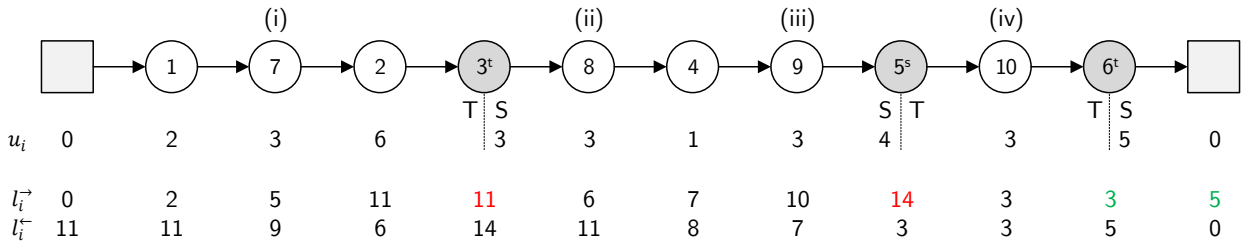


Figure 5.6.: Example to illustrate the load evaluation for different cases of vertex removals.

consumption of the replenishable resource R. The amount $u_{i'}^S$ to be replenished by an SV at visit $i' \in \mathcal{T}^S$ corresponds to the resource amount consumed by the DV at the connected visit $i \in \mathcal{D}^D$ or $i \in \mathcal{C}^+$ since the last transfer or the depot, limited to the respective capacity:

$$u_{i'}^S = \begin{cases} \min(f_i^{\rightarrow}, F) & \text{if } R = F, \\ \min(l_i^{\rightarrow}, L^D) & \text{if } R = L^D. \end{cases}$$

Based on these prerequisites, we define $l_{i'}^{S\rightarrow}$ as the cumulated resource amount replenished since the depot and $l_{i'}^{S\leftarrow}$ as the cumulated resource amount to be replenished until the depot for each SV visit i':

$$l_{i'}^{S\rightarrow} = \begin{cases} 0 & \text{if } i' = 0, \\ l_{i'-}^{S\rightarrow} + u_{i'}^S & \text{else.} \end{cases}$$

$$l_{i'}^{S\leftarrow} = \begin{cases} 0 & \text{if } i' = n+1, \\ l_{i'+}^{S\leftarrow} + u_{i'}^S & \text{else.} \end{cases}$$

The SV load violation of a solution $\mathcal{S}$ can then be calculated as the sum of the violations upon returning to the depot for all SV routes:

$$\upsilon^{LS}(\mathcal{S}) = \sum_{r \in \mathcal{S}^S} \max(l_{n+1}^{S\rightarrow} - L^S, 0).$$

Figure 5.7 illustrates the calculation of $u_{i'}^S$, $l_{i'}^{S\rightarrow}$, and $l_{i'}^{S\leftarrow}$ for an example SV route that is connected to a DV route via three transfers. In the example, we assume $R = F$ and $L^S = 20$. The displayed SV route is thus feasible with respect to load capacity L^S.

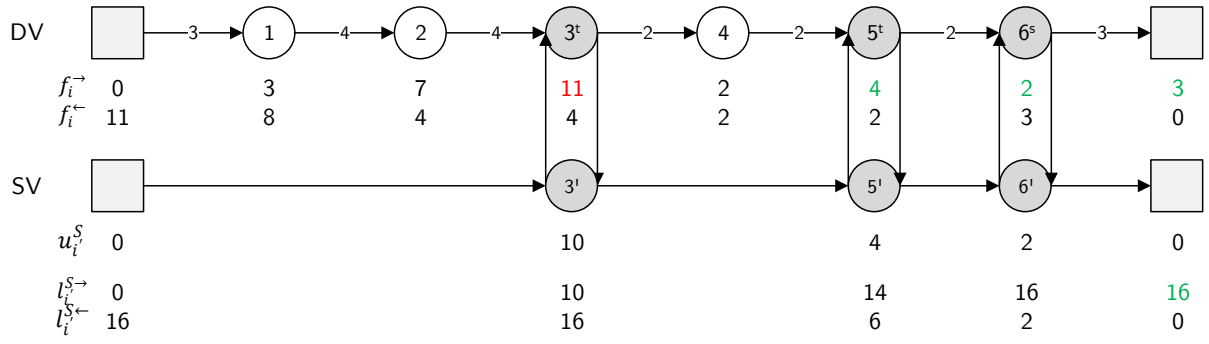


Figure 5.7.: Example to illustrate the load evaluation for SVs.

Finally, the new load violation resulting from inserting a visit j' into an SV route between visits i' and k' is calculated as:

$$v^{LS}(\nu^{+}_{j'}) = \max(l^{S\rightarrow}_{i'} + l^{S\leftarrow}_{k'} + u^{S}_{j'} - L^{S}, 0),$$

and from removing a visit i' from an SV route as:

$$v^{LS}(\nu^{-}_{i'}) = \max(l^{S\rightarrow}_{i'-} + l^{S\leftarrow}_{i'+} - L^{S}, 0).$$

For any operation, the change in SV load violation is then calculated as the difference between the load violation caused by the operation and the current load violation in the regarded SV route.

5.3.1.4. Calculation of Time Window Violations

To efficiently evaluate time window violations, we make use of the approach that has been proposed by Nagata, Bräysy, and Dullaert (2010) and enhanced by Schneider, Sand, and Stenger (2013). Here, a time window violation is considered only once at the vertex where it occurred. Then, based on the notion of time travel, it is assumed that the vehicle performs the service at this customer at the latest feasible moment, i.e., at the end of the time window of the customer. In this way, violations are not accumulated along succeeding vertex sequences which, on their own, are feasible with respect to the associated time windows. In the classical VRPTW, this assumption allows to compute changes in time window violation for conventional inter-route moves in $\mathcal{O}(1)$. We adapt this approach to the VRPTWMD by taking into account the synchronization of the DVs and SVs.

Following Nagata, Bräysy, and Dullaert (2010), we define extended earliest and latest time variables for each vertex. The term "extended" indicates the possibility of time travels.

Calculation of earliest times We start with introducing the extended earliest time variables, i.e., extended arrival time a^{D}_{i} of a DV, arrival time $a^{S}_{i'}$ of an SV, departure time d^{D}_{i} of a DV, departure time $d^{S}_{i'}$ of an SV, and service start time a'_{i} at vertex i. In addition, $a^{T}_{i'}$ denotes the transfer start time at vertex i and the associated support visit $i' \in \mathcal{T}^{S}$. Starting at the beginning of the planning horizon, i.e., $a^{D}_{0} = a^{S}_{0} = d^{D}_{0} = d^{S}_{0} = a'_{0} = a^{T}_{0} = e_0$, the extended earliest times are recursively calculated from the depot of each route in forward direction. Figure 5.8a exemplifies the calculation of the extended earliest times. Here, the value on each arc represents the travel time t_{ij} between the associated vertices i and j. In addition, we assume $\tau = 5$ and $s_i = 5\ \forall i \in \mathcal{C}$.

For any vertex succeeding the depot, the extended earliest arrival time a^{D}_{i} of a DV is calculated as the sum of the extended earliest departure time d^{D}_{i-} at its predecessor and the travel time t_{i-i} between the two vertices:

$$a^{D}_{i} = d^{D}_{i-} + t_{i-i}.$$

The extended earliest departure time d^{D}_{i} depends on the last task performed at vertex i:

$$d^{D}_{i} = \begin{cases} a'_{i} + s_i & \text{if } p_i = 0 \vee (p_i = 1 \wedge o_i = 1), \\ a^{T}_{i'} + \tau & \text{if } p_i = 1 \wedge o_i = 0. \end{cases}$$

If no transfer is scheduled or the service is performed after the transfer, the DV may leave as soon as the service terminates, i.e., at the point in time corresponding to the extended earliest service start time a'_i plus the service time s_i. Otherwise, if the transfer is scheduled after the service, the earliest departure time of the DV coincides with the earliest end of the transfer operation calculated as earliest transfer start time $a^T_{i'}$ plus transfer time τ.

The extended earliest service start time a'_i also depends on the task order at vertex i:

$$a'_i = \begin{cases} \min(\max(a^D_i, e_i), l_i) & \text{if } o_i = 0, \\ \min(\max(a^T_{i'} + \tau, e_i), l_i) & \text{if } o_i = 1. \end{cases}$$

More precisely, in case no transfer takes place at customer i or the service is scheduled before the transfer, the service can start as soon as the vehicle has arrived but not before the earliest start of service e_i. In case a transfer is performed first, the service may start as soon as the transfer is finished and the time window is open. In both cases, the vehicle has to travel back in time if the resulting time exceeds the latest start of service l_i.

The extended earliest arrival time $a^S_{i'}$ of an SV at vertex i' is analogously defined based on the extended earliest departure time $d^S_{i'-}$ at its predecessor:

$$a^S_{i'} = d^S_{i'-} + t_{i'-i'}.$$

An SV can leave as soon as the transfer terminates. Thus, the extended earliest departure time $d^S_{i'}$ of an SV at vertex i' is given as:

$$d^S_{i'} = a^T_{i'} + \tau.$$

Finally, the extended earliest transfer start time $a^T_{i'}$ is determined as follows:

$$a^T_{i'} = \begin{cases} \max(a^S_{i'}, a'_i + s_i) & \text{if } o_i = 0, \\ \max(a^S_{i'}, a^D_i) & \text{if } o_i = 1. \end{cases}$$

If the customer service is executed first, the transfer may start as soon as the service is finished and the SV has arrived. Otherwise, the transfer begins as soon as both vehicles are present at the transfer location.

To illustrate the synchronization of both vehicles, first consider vertex 3^t in Figure 5.8a. Here, the transfer operation is scheduled before the service and may thus start as soon as both vehicles have arrived:

$$a^T_{3'} = \max(a^S_{3'}, a^D_3) = \max(10, 22) = 22.$$

The service can then start when the transfer terminates:

$$a'_3 = \min(\max(a^T_{3'} + \tau, e_3), l_3) = \min(\max(22 + 5, 20), 30) = 27.$$

The DV can leave after the service:

$$d^D_3 = a'_3 + s_3 = 27 + 5 = 32,$$

and the SV after the transfer:

$$d^S_{3'} = a^T_{3'} + 5 = 22 + 5 = 27.$$

At vertex 5^s, the service is performed first. Because the DV arrives earlier than the earliest start of service, it has to wait until the time window is open:

$$a'_5 = \min(\max(a^D_5, e_5), l_5) = \min(\max(8, 10), 15) = 10.$$

The transfer can then begin as soon as the service is finished and the SV has arrived:

$$a^T_{5'} = \max(a^S_{5'}, a'_5 + s_5) = \max(32, 10 + 5) = 32.$$

Finally, both vehicles can leave after the transfer terminates, i.e., $d^D_5 = d^S_{5'} = a^T_{5'} + \tau = 32 + 5 = 37$.

Calculation of latest times Moreover, for each vertex i and the possibly associated support visit $i' \in \mathcal{T}^S$, we define extended latest times, i.e., the extended latest arrival time z^D_i of a DV, latest arrival time $z^S_{i'}$ of an SV, latest departure time w^D_i of a DV, latest departure time $w^S_{i'}$ of an SV, latest service start time z'_i, and latest transfer start time $z^T_{i'}$. The extended latest times are recursively calculated backwards from the depot of each route based on the end of the planning horizon, i.e., $z^D_{n+1} = z^S_{n+1} = w^D_{n+1} = w^S_{n+1} = z'_{n+1} = z^T_{n+1} = l_{n+1}$. Thus, each extended latest time represents the latest point in time at which the corresponding event has to take place such that no (additional) time window violation occurs at succeeding vertices. The example in Figure 5.8a additionally illustrates the calculation of the extended latest times.

For any preceding vertex i, the extended latest departure time w^D_i of a DV is calculated based on the extended latest arrival time $z^D_{i^+}$ at its successor:

$$w^D_i = z^D_{i^+} - t_{ii^+}.$$

The DV has to arrive at a vertex before the first task starts. Thus, depending on the task order at vertex i, z^D_i corresponds to the extended latest service start time z'_i or the extended latest transfer start time $z^T_{i'}$. However, in case of performing the transfer first, we also need to consider the actual arrival time $a^S_{i'}$ of the associated SV at vertex i'. If the SV is currently arriving later than $z^T_{i'}$, no benefits result from an arrival of the DV earlier than $a^S_{i'}$. The time window violation caused by the late arrival of the SV is not affected. Thus, in this case, we only need to consider the additional violation that occurs if the DV's time of arrival exceeds $a^S_{i'}$. Integrating this rationale, z^D_i is then determined as:

$$z^D_i = \begin{cases} z'_i & \text{if } o_i = 0, \\ \max(z^T_{i'}, a^S_{i'}) & \text{if } o_i = 1. \end{cases}$$

For calculating the extended latest service start time z'_i, we need to distinguish two cases again with respect to the task order at vertex i. If no transfer takes place or the service is scheduled after the transfer, the DV may leave after the service terminates. Thus, the service must end before the extended latest departure time w^D_i. Otherwise, if the service is executed first, it must be finished before the extended latest transfer start time $z^T_{i'}$. Analogous to the calculation of z^D_i, however, the

end of service may be postponed until the arrival $a^S_{i'}$ of the SV associated with the transfer if it exceeds $z^T_{i'}$.

In both cases, we subtract the service time s_i from the respective reference time to obtain the extended latest service start time z'_i. In addition, the service must start within the respective time window which leads to:

$$z'_i = \begin{cases} \max(\min(w^D_i - s_i, l_i), e_i) & \text{if } p_i = 0 \vee (p_i = 1 \wedge o_i = 1), \\ \max\Big(\min\big(\max(z^T_{i'}, a^S_{i'}) - s_i, l_i\big), e_i\Big) & \text{if } p_i = 1 \wedge o_i = 0. \end{cases}$$

The extended latest departure time of an SV at vertex i' is analogously defined as for a DV based on the extended latest arrival time $z^S_{i'^+}$ at its successor:

$$w^S_{i'} = z^S_{i'^+} - t_{i'i'^+}.$$

The extended latest arrival time $z^S_{i'}$ of an SV at vertex i' depends on the extended latest transfer start time $z^T_{i'}$ and, analogous to the DV-related calculations, on the actual time the corresponding DV is ready to transfer. An SV may arrive later than $z^T_{i'}$ if the arrival time ($o_i = 1$) or the service end time ($o_i = 0$) of the associated DV already violates this threshold. The extended latest arrival time $z^S_{i'}$ is then defined as:

$$z^S_{i'} = \begin{cases} \max(z^T_{i'}, a^D_i) & \text{if } o_i = 1, \\ \max(z^T_{i'}, a'_i + s_i) & \text{if } o_i = 0. \end{cases}$$

Finally, the extended latest transfer start time $z^T_{i'}$ is calculated as follows:

$$z^T_{i'} = \begin{cases} \min\big(w^S_{i'} - \tau, w^D_i - \tau\big) & \text{if } o_i = 0, \\ \min\big(w^S_{i'} - \tau, z'_i - \tau\big) & \text{if } o_i = 1. \end{cases}$$

If the transfer is performed last, it has to be finished before the latest departure time of both vehicles. Otherwise, if the transfer is performed before the service, the latest transfer start time must be set such that the latest departure time of the SV and the latest service start time can be met.

Consider for example transfer vertex 5^s in Figure 5.8a again. Because the transfer is performed after the service, both vehicles may leave after the transfer terminates. The latest transfer start time $z^T_{5'}$ is thus calculated as:

$$z^T_{5'} = \min\Big(w^S_{5'} - \tau, w^D_5 - \tau\Big) = \min(42 - 5, 30 - 5) = 25.$$

Based on $z^T_{5'}$, the latest start of service z'_5 can then be determined taking into account the actual arrival time $a^S_{5'}$ of the SV:

$$z'_5 = \max\Big(\min\big(\max(z^T_{5'}, a^S_{5'}) - s_5, l_5\big), e_5\Big) = \max\Big(\min\big(\max(25, 32) - 5, 15\big), 10\Big) = 15.$$

Thus, the DV must not arrive later than the latest start of service, i.e., $z^D_5 = 15$ and the SV not later than the latest start of transfer:

$$z^S_{5'} = \max(z^T_{5'}, a'_5 + s_5) = \max(25, 15 + 5) = 25.$$

At transfer vertex 3^t, the service is performed last. The latest service start time z'_3 is thus calculated based on the latest departure time w_3^D of the DV:

$$z'_3 = \max(\min(w_3^D - s_3, l_3), e_3) = \max(\min(33 - 5, 30), 20) = 28.$$

The latest transfer start time $z_{3'}^T$ is then given as:

$$z_{3'}^T = \min(w_{3'}^S - \tau, z'_3 - \tau) = \min(20 - 5, 28 - 5) = 15.$$

Finally, the DV must have arrived at:

$$z_3^D = \max(z_{3'}^T, a_{3'}^S) = \max(15, 10) = 15,$$

and the SV at:

$$z_{3'}^S = \max(z_{3'}^T, a_3^D) = \max(15, 22) = 22.$$

The arrival of the SV could thus be postponed beyond the latest transfer start time because of the DV's late arrival.

Based on these prerequisites, we are able to evaluate changes in time window violation caused by the operations described in Section 5.3.1 in $\mathcal{O}(1)$. In contrast to the classical VRPTW, however, a time window violation does not necessarily occur in the route that is currently modified if transfers exist. This is why in the VRPTWMD, a time window violation cannot be assigned to a specific route in straightforward fashion. Therefore, we estimate the change in time window violation caused by an insertion or removal operation ν using a surrogate $\Delta\tilde{v}^{TW}$, which is calculated as the difference between the new time window violation $\tilde{v}^{TW}(\nu)$ caused by the respective operation and the current time window violation $\tilde{v}_i^{TW}$ originating at the route position i that is being modified, i.e., $\Delta\tilde{v}^{TW} = \tilde{v}^{TW}(\nu) - \tilde{v}_i^{TW}$. This procedure is detailed in the following.

Evaluation of insertions To evaluate insertions, we need to distinguish again between the actual insertion of a vertex and the assignment of a transfer to a customer that is already part of the solution. In the former case of inserting a vertex j between vertices i and k in a DV route, we define the current time window violation $\tilde{v}_k^{TW}$ as the time window violation originating at the potential new successor k of the vertex j to be inserted:

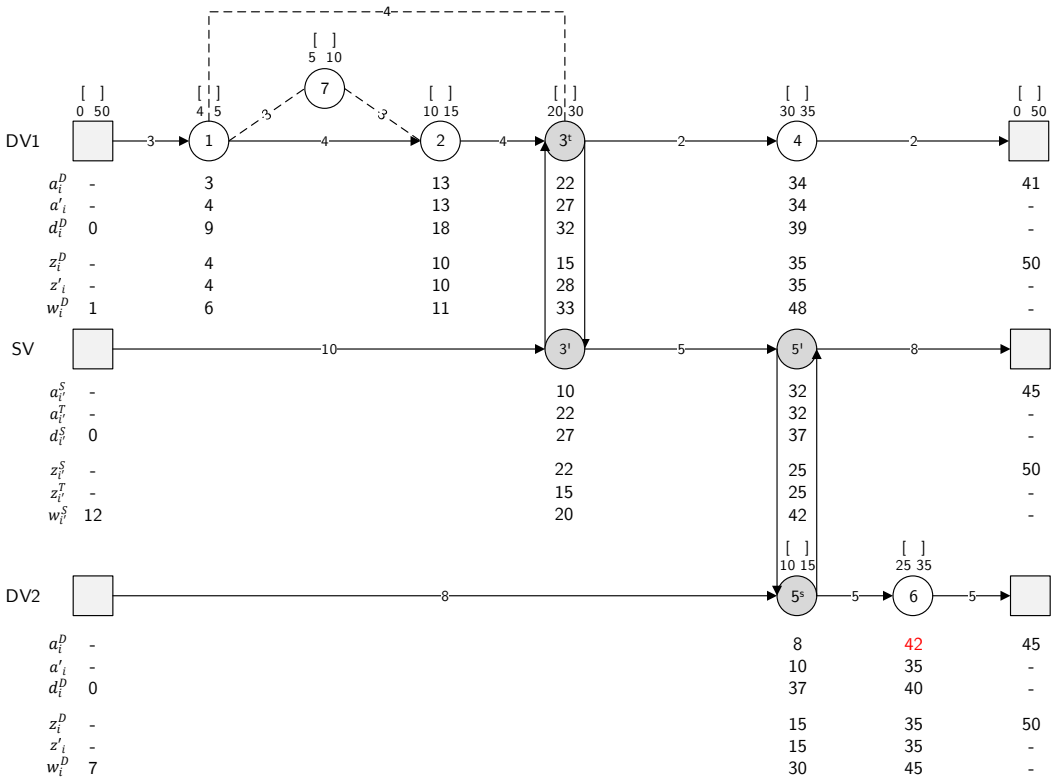
$$\tilde{v}_k^{TW} = \max(a_k^D - z_k^D, 0).$$

If the insertion operation corresponds to the assignment of a transfer to a customer $j \in \mathcal{C}^t \backslash \mathcal{C}^+$, the current time window violation represents the time window violation occurring at j plus succeeding violations based on the departure time at j:

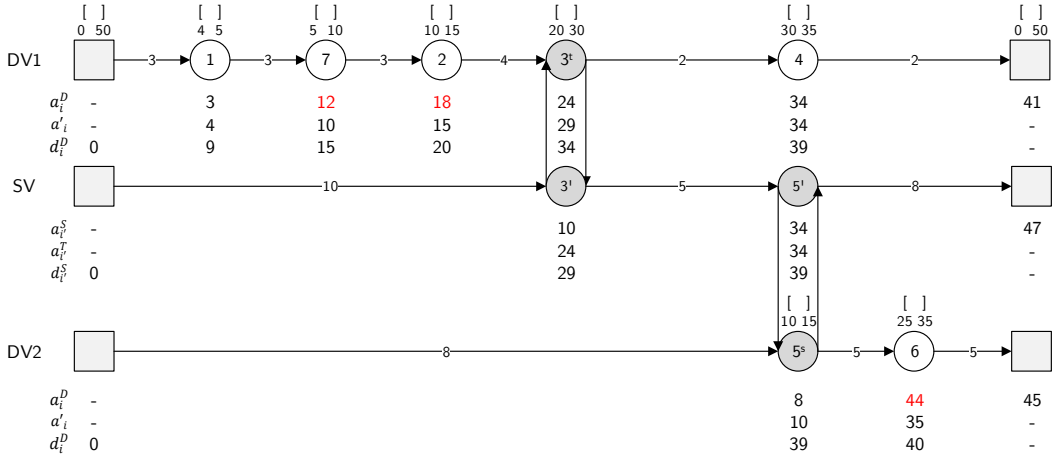
$$\tilde{v}_j^{TW} = \max(a_j^D - l_j, 0) + \max(d_j^D - w_j^D, 0).$$

In the case of a transfer insertion/assignment, we additionally need to consider the change in time window violation resulting from inserting the associated SV visit $j' \in \mathcal{T}^S$ between visits i' and k'. We

(a) Initial situation



(b) Extended earliest times after insertion of customer 7



(c) Extended earliest times after removal of customer 2

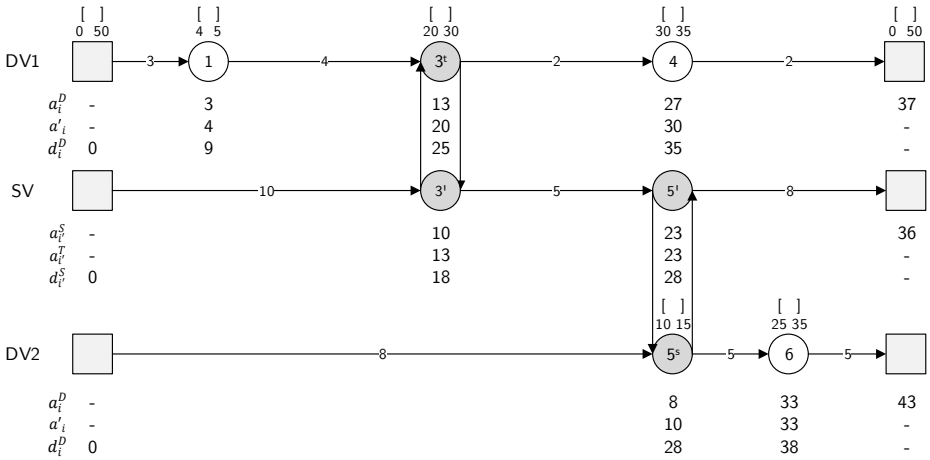


Figure 5.8.: Example to illustrate the evaluation of time window violations.

define the current time window violation originating at k' as:

$$\tilde{v}_{k'}^{TW} = \max(a_{k'}^S - z_{k'}^S, 0).$$

To determine the time window violation caused by insertion ν_j^+, we first calculate the resulting arrival time $a_j(\nu^+)$ at the vertex j to be inserted. In case of a DV route, the new arrival time is calculated as follows:

$$a_j^D(\nu_j^+) = d_i^D + t_{ij}.$$

Then, the time window violation caused by the insertion of a customer $j \in \mathcal{C}^-$ is defined as the violation occurring at j itself plus succeeding violations resulting from a late arrival at its new successor k:

$$\tilde{v}^{TW}(\nu_j^+) = \max(a_j^D(\nu_j^+) - l_j, 0) + \max\Big(\min\big(\max(a_j^D(\nu_j^+), e_j), l_j\big) + s_j + t_{jk} - z_k^D, 0\Big).$$

Consider for example the insertion of customer 7 in Figure 5.8a. The resulting situation is depicted in Figure 5.8b. The current time window violation originating at customer 2, the potential new successor of customer 7 amounts to:

$$\tilde{v}_2^{TW} = \max(a_2^D - z_2^D, 0) = \max(13 - 10, 0) = 3.$$

The new arrival time at customer 7 is calculated as:

$$a_7^D(\nu_7^+) = d_1^D + t_{17} = 9 + 3 = 12.$$

Using $a_7^D(\nu_7^+)$, the new time window violation caused by the insertion can be determined as:

$$\begin{aligned}\tilde{v}^{TW}(\nu_7^+) &= \max(a_7^D(\nu_7^+) - l_7, 0) + \max\Big(\min\big(\max(a_7^D(\nu_7^+), e_7), l_7\big) + s_7 + t_{72} - z_2^D, 0\Big) \\ &= \max(12 - 10, 0) + \max\Big(\min\big(\max(12, 5), 10\big) + 5 + 3 - 10, 0\Big) = 2 + 8 = 10.\end{aligned}$$

Thus, the insertion would cause an increase in time window violation of $\Delta\tilde{v}^{TW} = 10 - 3 = 7$ (compare the total time window violations in Figures 5.8a and 5.8b).

In case of inserting or assigning a transfer, we additionally need to consider the associated support visit $j' \in \mathcal{T}^S$ to be inserted between visits i' and k'. The new arrival time at j' is given as:

$$a_{j'}^S(\nu_{j'}^+) = d_{i'}^S + t_{i'j'}.$$

The time window violation caused by the insertion of the DV visit $j \in \mathcal{D}^D$ or the customer $j \in \mathcal{C}^+$ associated with the transfer, or the assignment of a transfer status to customer $j \in \mathcal{C}^t \backslash \mathcal{C}^+$ can then be determined depending on the task order at j:

$$\tilde{v}^{TW}(\nu_j^+) = \begin{cases} \max(a_j^D(\nu_j^+) - l_j, 0) + \max(a_{j'}^T(\nu_j^+) + \tau + t_{jk} - z_k^D, 0) & \text{if } o_j = 0, \\ \max(a_{j'}^T(\nu_j^+) + \tau - l_j, 0) + \max(a_j'(\nu_j^+) + s_j + t_{jk} - z_k^D, 0) & \text{if } o_j = 1. \end{cases}$$

More precisely, if the service is performed first, the evaluation of the potential service-related time window violation at j is based on the new arrival time $a_j^D(\nu_j^+)$ at this visit. The DV may leave after

the transfer terminates. Thus, potential succeeding violations are calculated based on the new transfer start time $a^T_{j'}(\nu^+_j)$. Otherwise, if the transfer is performed first, a time window violation occurs at j if the new transfer end time exceeds the latest start of service l_j. Potential succeeding violations now depend on the new service start time $a'_j(\nu^+_j)$. Analogously to the calculation of the earliest times, if $o_j = 0$, the service may start as soon as the DV has arrived, or if $o_j = 1$, as soon as the transfer ends and the time window is open:

$$a'_j(\nu^+_j) = \begin{cases} \min\big(\max(a^D_j(\nu^+_j), e_j), l_j\big) & \text{if } o_j = 0, \\ \min\big(\max(a^T_{j'}(\nu^+_j) + \tau, e_j), l_j\big) & \text{if } o_j = 1. \end{cases}$$

The transfer in turn may begin as soon as both vehicles have arrived at the transfer location and, if $o_j = 0$, the service is finished:

$$a^T_{j'}(\nu^+_j) = \begin{cases} \max(a^S_{j'}(\nu^+_j), a'_j(\nu^+_j) + s_j) & \text{if } o_j = 0, \\ \max(a^S_{j'}(\nu^+_j), a^D_j(\nu^+_j)) & \text{if } o_j = 1. \end{cases}$$

The insertion of support visit j' associated with the transfer causes time window violations if the new transfer start time exceeds the end of the planning horizon l_{n+1} or the new arrival time at its successor the associated threshold:

$$\tilde{v}^{TW}(\nu^+_{j'}) = \max(a^T_{j'}(\nu^+_{j'}) - l_{n+1}, 0) + \max(a^T_{j'}(\nu^+_{j'}) + \tau + t_{j'k'} - z^S_{k'}, 0).$$

Finally, to calculate the change in time window violation resulting from the insertion or assignment of a transfer, we need to take into account both the DV and the SV values:

$$\Delta\tilde{v}^{TW} = \tilde{v}^{TW}(\nu^+_j) + \tilde{v}^{TW}(\nu^+_{j'}) - \tilde{v}^{TW}_k - \tilde{v}^{TW}_{k'}.$$

Evaluation of removals We define the current time window violation for the removal of a vertex as the time window violation occurring at the vertex i to be removed plus succeeding violations based on the departure time at i. In case of removing a vertex from a DV route, the current time window violation is calculated as follows:

$$\tilde{v}^{TW}_i = \max(a^D_i - l_i, 0) + \max(d^D_i - w^D_i, 0),$$

and in case of removing an SV visit $i' \in \mathcal{T}^S$ as:

$$\tilde{v}^{TW}_{i'} = \max(a^S_{i'} - l_{i'}, 0) + \max(d^S_{i'} - w^S_{i'}, 0).$$

To obtain the new time window violation caused by the removal ν^-_i of a vertex, we need to distinguish between the removal of a customer $i \in \mathcal{C}^-$ and the removal of a transfer $i \in \mathcal{D}^D$ or $i \in \mathcal{C}^+$. For the customer case, the time window violation resulting from concatenating both adjacent partial routes is given as:

$$\tilde{v}^{TW}(\nu^-_i) = \max(d^D_{i^-} + t_{i^-i^+} - z^D_{i^+}, 0).$$

More precisely, a time window violation occurs if the new arrival time at the successor exceeds the associated latest arrival time.

To illustrate these calculations, consider customer 2 to be removed in Figure 5.8a. The resulting situation is shown in Figure 5.8c. The current time window violation originating at customer 2 is given as:

$$\tilde{v}_2^{TW} = \max(a_2^D - l_2, 0) + \max(d_2^D - w_2^D, 0) = \max(13 - 15, 0) + \max(18 - 11, 0) = 7.$$

The new time window violation after the removal is calculated as:

$$\tilde{v}^{TW}(\nu_2^-) = \max(d_1^D + t_{13} - z_3^D, 0) = \max(9 + 4 - 15, 0) = 0,$$

which leads to $\Delta\tilde{v}^{TW} = 0 - 7 = -7$. Thus, as shown in Figure 5.8c, the time window of customer 6 is not violated anymore after the removal of customer 2.

The removal of a transfer requires the consideration of both associated visits. With respect to the DV visit removal, we need to further distinguish between the actual removal of visit $i \in \mathcal{D}^D$ or customer $i \in \mathcal{C}^+$ and only the removal of the transfer status of customer $i \in \mathcal{C}^+$. If the visit or customer is entirely removed, the resulting time window violation is calculated in identical fashion as in the case of $i \in \mathcal{C}^-$. Otherwise, if only the transfer status is removed, we need to evaluate a possibly new service start and departure time at customer $i \in \mathcal{C}^+$:

$$\tilde{v}^{TW}(\nu_i^-) = \max(a_i^D - l_i, 0) + \max\Big(\min\big(\max(a_i^D, e_i), l_i\big) + s_i - w_i^D, 0\Big).$$

Finally, the time window violation resulting from removing the SV visit i' associated with the transfer is also defined analogously to the customer case:

$$\tilde{v}^{TW}(\nu_{i'}^-) = \max(d_{i'-}^S + t_{i'-i'+} - z_{i'+}^S, 0).$$

Again, the calculation of the change in time window violation resulting from the removal of a transfer requires to take into account the influence of the removal of the DV and the SV visit:

$$\Delta\tilde{v}^{TW} = \tilde{v}^{TW}(\nu_i^-) + \tilde{v}^{TW}(\nu_{i'}^-) - \tilde{v}_i^{TW} - \tilde{v}_{i'}^{TW}.$$

5.3.2. Initialization

We generate an initial solution $\mathcal{S}$ with $m^D = \lceil \sum_{i \in \mathcal{C}} u_i / L^D \rceil$ DV routes, i.e., with the minimum number of DV routes without considering the possibility of load replenishments. We initialize each route with a seed customer. To this end, we sort all customers in ascending order of their latest start time of service l_i. The seed customers are then determined by selecting the first m^D entries. Subsequently, we successively insert the remaining customers into the routes at their cost-minimal position according to $f_{gen}(\mathcal{S})$. As soon as a route violates R, we add transfers until the route becomes R-feasible again. This transfer insertion procedure is also applied later in the search. While in the initialization phase, there is at most one route segment violating R at a time, several segments of a route may become infeasible with respect to R during the search. In the following, we therefore describe the general procedure for

inserting transfers that is applicable to an arbitrary number of R-infeasible route segments. Figure 5.9 shows the respective pseudocode.

while route violates R **do**
  $\nu^+_{best} \leftarrow \varnothing$
  for each R-infeasible route segment **do**
    $\iota \leftarrow$ *getPositionOfLastFeasibleCustomer*()
    $\kappa \leftarrow$ *getPositionOfPrecedingTransfer*(ι)
    $\nu^+ \leftarrow$ *getBestTransferInsertionOnPath*($\kappa + 1, \iota + 1, \mathcal{D}^D \cup \mathcal{C}^t \backslash \mathcal{C}^+$)
    if $\nu^+_{best} = \varnothing \vee \nu^+$ improves on ν^+_{best} **then**
      $\nu^+_{best} \leftarrow \nu^+$
    end if
  end for
  $\mathcal{S} \leftarrow$ *performTransferInsertion*(ν^+_{best})
end while

Figure 5.9.: Pseudocode of the transfer insertion procedure.

For each R-infeasible route segment, we can identify the last customer that can still be served without violating R. Let ι denote the position of this customer and κ the position of the last transfer or the depot preceding this customer. Now, we determine the best transfer insertion on the path from position $\kappa+1$ to position $\iota+1$ for each infeasible route segment. For each dedicated transfer vertex, we need to evaluate each possible combination of insertion position on the respective path and SV route insertion position. In addition, we need to evaluate the assignment of a transfer to each customer $i \in \mathcal{C}^t$ currently served on the path for both possible orders of tasks. Note that whenever we insert a transfer, we need to ensure that no infeasible precedence relations, i.e., precedence relations between a vertex and itself are created. To this end, we utilize the cycle-detection mechanism described in Section 5.3.3 to detect invalid transfer insertions. The overall best insertion according to f_{gen} among all allowed insertions is then performed.

We subsequently try to further improve the initial solution by applying a local search step (see Section 5.3.4).

5.3.3. Cycle Detection

As mentioned above, if multiple transfers can be performed in a route as in the VRPTWMD, the creation of infeasible precedence relations, i.e., cycles, has to be avoided. A cycle implies a precedence relation between a vertex and itself (Masson, Lehuédé, and Péton, 2013).

Figure 5.10 shows an example insertion of an additional transfer into a DV route and an SV route that are already connected by a transfer. More precisely, DV visit 5^t is inserted between customers 3 and 1, and the corresponding support visit $5'$ between $4'$ and the depot. As shown in Figure 5.10b, after the insertion, there is a path from each vertex on the cycle to itself, i.e., each vertex on the cycle is its own predecessor and successor, which is clearly inconsistent.

To efficiently detect cycles when inserting a transfer, especially if the interdependence between routes is more complex than depicted in the example, we adapt the approach described by Grangier et al. (2016). In detail, we define a set Γ^+_k for each vertex k containing all (direct and indirect) successors of k. Now, let i denote a transfer visit to be inserted into a DV route between i^- and i^+, and j' the associated visit to be inserted into an SV route between j'^- and j'^+. Then, the insertion of the regarded transfer would introduce a cycle if $j'^- \in \Gamma^+_{i^+}$ or $i^- \in \Gamma^+_{j'^+}$.

The displayed insertion thus creates a cycle because the new predecessor $4'$ of $5'$ is a successor of customer 1, the new direct successor of 5^t, i.e., $4' \in \Gamma_1^+ = \{4^s, 4', 2\}$ (with respect to Figure 5.10a).

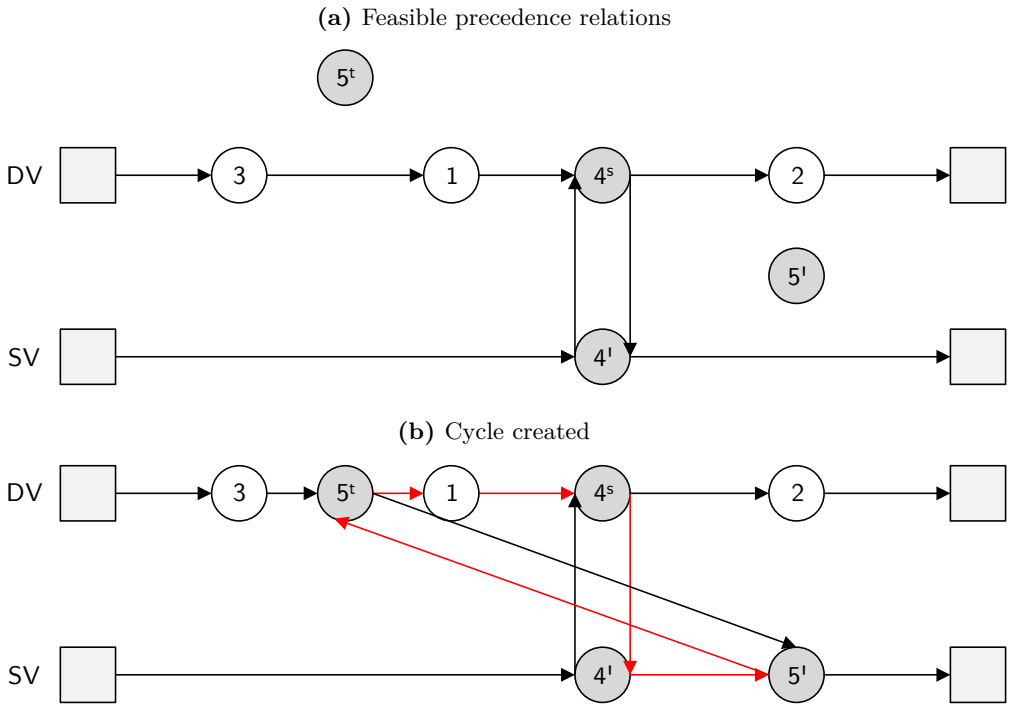


Figure 5.10.: Introduction of a cycle due to the invalid insertion of an additional transfer.

5.3.4. Local Search

After the generation of the initial solution (and after every application of the ALNS or the PR component), we aim at improving the current solution by means of a local search procedure. Our local search follows a first-improvement strategy and is stopped when no further improvement of the current solution can be identified.

We first handle violations of R by inserting (additional) transfers according to the procedure described in Section 5.3.2. Next, we store each customer $i \in \mathcal{C}^-$ and transfer $i \in \mathcal{D}^D \cup \mathcal{C}^+$ currently performed in the solution in a list B and perturb its order. Now, we iteratively relocate each vertex in B with the goal of improving the current solution. More precisely, if the currently considered vertex corresponds to a customer $i \in \mathcal{C}^-$, we remove this customer and reinsert it at the best position. If the current vertex corresponds to a transfer $i \in \mathcal{D}^D \cup \mathcal{C}^+$, we first remove both associated visits from the corresponding DV and SV routes. We subsequently evaluate all possible insertions for this transfer, i.e., each possible combination of (i) DV route insertion position, and (ii) SV route insertion position, which does not introduce a cycle, and if $i \in \mathcal{C}^+$, (iii) task order. The best insertion among all possibilities is then performed. If the resulting solution improves on the previous one, we restart the local search at the first element of B. Otherwise, we proceed with the next vertex. The main phase of the local search stops when all vertices in B have been considered and no further improvement could be achieved.

Finally, we remove unnecessary transfers from the solution. More precisely, while there are transfers which can be removed without increasing the violation of R, we perform the removal associated with the greatest cost reduction.

5.3.5. The Adaptive Large Neighborhood Search Component

The LNS paradigm follows the idea of iteratively destroying and repairing solutions by removing and reinserting relatively large numbers of customers (Shaw, 1998). ALNS extends this approach by deploying several competing removal and insertion operators which are chosen at each iteration depending on their previous performance in the search. To this end, each operator is assigned a selection probability, which is dynamically updated during the search (Ropke and Pisinger, 2006b).

Highly complex problems like the VRPTWMD call for the utilization of solution methods that offer strong diversification capabilities. While classical local-search-based algorithms, like, e.g., tabu search, often get stuck in local optima from which they are not able to escape, ALNS allows to modify large parts of a solution at once and thus to overcome local traps more easily. This has been demonstrated by ALNS algorithms for classical VRP variants (see, e.g., Ropke and Pisinger, 2006b; Ropke and Pisinger, 2006a; Pisinger and Ropke, 2007; Hemmelmayr, Cordeau, and Crainic, 2012) as well as for VRPs with multiple synchronization constraints (see, e.g., Masson, Lehuédé, and Péton, 2012; Grangier et al., 2016). Figure 5.11 shows our ALNS implementation in pseudocode.

{Randomly choose option 1 or option 2, i.e., route removal (1) or any removal operator (2)}
if option 1 selected $\vee$ (option 2 $\wedge$ route-based removal operator based on probabilities $\boldsymbol{\pi}^-$ selected **then**
  {Apply randomly selected route-based removal operator}
  $\mathcal{S}' \leftarrow removeRouteSegment(\mathcal{S})$
else if option 2 $\wedge$ vertex-based removal operator based on probabilities $\boldsymbol{\pi}^-$ selected **then**
  {*Determine number of vertices to remove depending on scope of operator and probabilities* $\boldsymbol{\pi}^{||}$}
  $n^- \leftarrow drawNumberOfVerticesToRemove(\boldsymbol{\pi}^{||})$
  {*Apply randomly selected vertex-based removal operator*}
  $\mathcal{S}' \leftarrow removeVertices(\mathcal{S}, n^-)$
end if
if customers previously removed **then**
  {*Apply randomly selected customer insertion operator based on probabilities* $\boldsymbol{\pi}^+$}
  $\mathcal{S}' \leftarrow insertCustomers(\mathcal{S}', \boldsymbol{\pi}^+)$
end if
if transfers previously removed **then**
  {*Apply randomly selected transfer insertion operator based on probabilities* $\boldsymbol{\pi}^+$}
  $\mathcal{S}' \leftarrow insertTransfers(\mathcal{S}', \boldsymbol{\pi}^+)$
end if

Figure 5.11.: Pseudocode of our ALNS component.

To destroy the current solution, we use two types of removal operators (see Section 5.3.5.1): The first type of operators, called *route-based* removal operators, removes an entire structure, i.e., route or route segment from a solution at each iteration. The second type of operators is able to remove a certain number of vertices that may belong to different routes or segments. The latter are denoted as *vertex-based* removal operators.

To stronger influence the reduction of the number of employed vehicles and thus the vehicle deployment cost, we randomly decide between two options at the beginning of each ALNS iteration, i.e., whether to apply (i) a route removal operator, or (ii) any of the implemented removal operators to the current solution $\mathcal{S}$. Each option is selected with a probability of 50%. In the first case, we randomly select a route removal operator out of four implemented variants, which entirely removes a DV or an SV route from $\mathcal{S}$. The choice of the operator is performed according to a roulette-wheel selection procedure based on the probability vector $\boldsymbol{\pi}^-$, which is dynamically updated during the search according to the adaptive mechanism described in Section 5.3.5.3.

If the second option is selected, we first select a removal operator from the set of all removal operators, i.e., all route-based and vertex-based removal operators based on probabilities $\boldsymbol{\pi}^-$. If a vertex-based removal operator has been selected, we subsequently determine the number of vertices that are removed from $\mathcal{S}$. While ALNS implementations for classical VRP variants are usually only concerned with the removal and reinsertion of customers, the VRPTWMD offers a variety of possibilities in this regard. Therefore, we implement several variants for each vertex-based removal operator with respect to the scope of the associated removal operation:

Customers exclusively removes customers ($i \in \mathcal{C}^-$) from the current solution.

Transfers exclusively removes transfers, i.e., the associated DV ($i \in \mathcal{C}^+ \cup \mathcal{D}^D$) and SV visits ($i' \in \mathcal{T}^S$) from a solution. Respective operator variants can only be selected if transfers exist in the current solution.

All vertices may remove all types of vertices.

Transfer states turns transfer vertices in DV routes ($i \in \mathcal{C}^+$) into regular customers and removes the corresponding SV visits ($i' \in \mathcal{T}^S$) from the solution. The associated operator variants can only be selected if transfers are scheduled at customers in the current solution.

We additionally experimented with rearranging partial transfers, i.e., only one of both associated visits at each iteration while the respective counterpart stays in the current solution. However, as shown in Section 5.4.3, it proved to be of vital importance to simultaneously consider the DV and the SV level to escape from local optima. The upper part of Table 5.3 gives an overview of the vertex-based removal operator variants with respect to the type of the permitted removal operation.

	Scope			
	Customers	Transfers	All vertices	Transfer states
Removal operators				
Random	✓	✓	✓	✓
Cluster	✓		✓	
Worst	✓	✓	✓	✓
Relatedness	✓	✓		
Neighbor graph	✓	✓	✓	
Synchronization offset		✓		✓
Route selection policies				
Cost	✓	✓	✓	✓
Distance	✓	✓	✓	✓
Efficiency	✓	✓	✓	✓
Synchronization offset	✓	✓	✓	✓
Transfer density		✓		✓

Table 5.3.: Vertex-based removal operator variants and compatibility of route selection policies with respect to the type of the removal operation.

In the literature, the number of customers to be removed from a solution is usually determined by randomly selecting the percentage of customers to remove from a relatively large interval $\Psi^{||} = [\varsigma^{||}_{min}, \varsigma^{||}_{max}]$ (Ropke and Pisinger, 2006b; Hemmelmayr, Cordeau, and Crainic, 2012). However, we observed that ideal removal percentages are highly instance-dependent. Therefore, we split $\Psi^{||}$ into several sub-intervals similar to Goeke and Schneider (2015). Then, we select a sub-interval based on dynamic probabilities $\boldsymbol{\pi}^{||}$ and randomly determine the number n^- of vertices to remove from $\mathcal{S}$ within this interval. Note that the maximum number of vertices than can be removed is defined by the scope of the selected removal operator. For example, if the currently selected operator aims at removing customers, the selected removal percentage is applied to value $|\mathcal{C}^-|$.

After the removal step, both regular customers and transfers may have to be reinserted. Because transfer insertion consists of two insertions, one on the DV and one on the SV level, we separately insert customers and transfers using different insertion operators (Section 5.3.5.1) to allow a fair comparison of insertion operations with respect to insertion cost. The selection of both insertion operators is performed according to probabilities $\boldsymbol{\pi}^+$.

5.3.5.1. Removal and Insertion Operators

We use the following route-based removal operators:

Route removal DV removes all visits associated with a DV route and is implemented in two variants where the route to be removed (i) is randomly selected, or (ii) corresponds to the route with the smallest cumulated demand in the current solution.

Route removal SV is defined analogously to route removal DV but for SV routes.

Segment removal removes all vertices on a segment of a DV route starting with a transfer and ending with the predecessor of the next transfer or of the depot. Segment removal is implemented in two variants where the transfer defining the segment to be removed is (i) randomly selected, or (ii) corresponds to the transfer, which shows the largest synchronization offset (see Grangier et al., 2016, for a similar approach). We define the synchronization offset Δ_i^{sync} for a transfer $i \in \mathcal{C}^+ \cup \mathcal{D}^D$ as the absolute deviation between the time the DV would be ready to transfer and the arrival time of the SV:

$$\Delta_i^{sync} = \begin{cases} |a'_i + s_i - a_i^S| & \text{if } o_i = 0, \\ |a_i^D - a_i^S| & \text{if } o_i = 1. \end{cases}$$

In the following, we describe the vertex-based removal operators. We assume that an operator iteratively removes n^- vertices from the current solution independent of the type(s) of vertices it may remove. Note that the removal of a transfer vertex corresponds to the removal of both associated visits from the corresponding DV and SV routes. The removal of a transfer state, in which only the respective SV visit is removed while the associated customer stays in the solution, also counts as one removed vertex.

Random removal randomly removes vertices from $\mathcal{S}$ until n^- vertices are removed.

Cluster removal was first introduced by Ropke and Pisinger (2006b) and aims at removing vertices that are located close to each other. Initially, we select a route and the first vertex to be removed from this route at random. Subsequently, the following steps are repeated until n^- vertices are removed from the current solution: We randomly choose a vertex among the already removed ones and identify the route r which yields the smallest average distance of its vertices to the selected vertex. Next, we apply Kruskal's algorithm (Kruskal, 1956) to solve the minimum-spanning-tree (MST) problem to the sub-graph composed of the vertices of route r. Let n_r denote the number of vertices served by route r. We abort the execution of the MST algorithm as soon as the number of generated edges is equal to $n_r - 2$, i.e., two sub-trees remain from which one is randomly chosen for removal. If the size of the selected cluster exceeds the number of remaining vertices to be removed, we randomly remove vertices from the cluster until n^- vertices are removed.

Relatedness removal is based on the idea that vertices that are in some sense similar are likely to be interchangeable in the subsequent insertion step (Shaw, 1997). The relatedness of two vertices

i and j is measured in terms of the travel cost c_{ij} between them, the difference in their demands $|u_i - u_j|$, and the difference between the earliest start times of their time windows $|e_i - e_j|$. The relatedness measure $R_{i,j}$ of two vertices i and j is thus calculated as follows:

$$R_{i,j} = \chi^c \frac{c_{ij}}{\max\limits_{i,j \in \mathcal{C} \cup \mathcal{D}^D}(c_{ij})} + \chi^d \frac{|u_i - u_j|}{\max\limits_{i \in \mathcal{C}}(u_i) - \min\limits_{i \in \mathcal{C}}(u_i)} + \chi^e \frac{|e_i - e_j|}{\max\limits_{i \in \mathcal{C}}(e_i) - \min\limits_{i \in \mathcal{C}}(e_i)}.$$

Each component is weighted with a parameter χ and normalized using the respective extreme values across the set of all customers and dedicated transfers $\mathcal{C} \cup \mathcal{D}^D$ given by solution $\mathcal{S}$.

Initially, we select a route and the first vertex to be removed from this route at random. Then, at each iteration, we first randomly select a vertex from the already removed ones. Next, all remaining vertices are stored in a list B in ascending order of their relatedness value with respect to the selected vertex. From this list, we draw the vertex at position $\lfloor |B| \zeta^{\chi^{rel}} \rfloor$, where ζ is a random number $\in [0, 1)$ and χ^{rel} a parameter $\geqslant 1$, which allows to balance the influence of randomness and relatedness on the selection. We repeat this procedure until n^- vertices are removed.

Worst removal, as originally introduced by Ropke and Pisinger (2006b), aims at removing vertices, which appear to be unfavorably positioned in the current solution with respect to the additional routing cost caused by visiting them. All vertices are stored in a list B and sorted in descending order of the cost reduction resulting from their removal. At each iteration, we choose the vertex at position $\lfloor |B| \zeta^{\chi^{worst}} \rfloor$. Again, ζ is a random number $\in [0, 1)$, and $\chi^{worst} \geqslant 1$ allows to control the randomness of the selection.

Neighbor graph removal, proposed in Ropke and Pisinger (2006a), is based on exploiting information about promising visiting orders gathered in the course of the search. More precisely, a complete directed and weighted auxiliary graph, called the neighbor graph, is introduced, whose vertices correspond to the vertices in the problem instance. Each arc (i, j) is weighted with the objective function value of the best solution found so far in which vertex j is visited directly after vertex i. Initially, the weight of each arc is set to positive infinity and dynamically updated during the search.

All vertices in $\mathcal{S}$ are stored in a list B and sorted in descending order of a score which is based on the present visiting orders. More precisely, the score for each visit to a vertex i is calculated by summing up the weights of the arcs (i^-, i) and (i, i^+) in the neighbor graph, where i^- and i^+ correspond to the predecessor and successor of i, respectively. At each iteration, the operator removes the vertex at position $\lfloor |B| \zeta^{\chi^{nb}} \rfloor$ with random number $\zeta \in [0, 1)$ and $\chi^{nb} \geqslant 1$. After the removal of vertex i, the neighbor scores of vertices i^- and i^+ are updated accordingly.

Synchronization offset removal aims at removing poorly synchronized transfers. At each iteration, this operator removes the transfer which shows the maximum synchronization offset until n^- transfers are removed. The synchronization offset Δ_i^{sync} for a transfer $i \in \mathcal{C}^+ \cup \mathcal{D}^D$ is defined analogously to the respective segment removal variant.

To reinsert the previously removed vertices, we use several insertion operators. For each operator variant, the selection of the insertion to be performed at each iteration is conducted independently of the scope of the operator but the number of insertions to consider differs depending on the type of vertices to reinsert. More precisely, for each customer $i \in \mathcal{C}^-$, each DV route insertion position has to be evaluated. For each transfer $i \in \mathcal{C}^+ \cup \mathcal{D}^D$ to be inserted, we need to evaluate each possible

combination of (i) DV route insertion position, and (ii) SV route insertion position, which does not introduce a cycle, and if $i \in \mathcal{C}^+$, (iii) task order. Moreover, in case of a previous removal of transfer states, we subsequently assign transfers to n^- customers ($i \in \mathcal{C}^t \backslash \mathcal{C}^+$) that are served in the current solution, i.e., for which the DV route position is given but for the associated SV visit, each possible SV route insertion position has to be considered. We use the following insertion operators:

Greedy insertion determines the best insertion for each vertex at each iteration. The vertex associated with the overall best insertion is inserted accordingly.

GRASP insertion is inspired by the metaheuristic introduced in Feo and Resende (1989). The best insertion is determined for each remaining vertex and stored in a list B which is sorted in ascending order of cost increase. At each iteration, the next vertex to be inserted is randomly selected among the best $\lfloor \chi^{GRASP} |B| \rfloor$ remaining insertions. Here, χ^{GRASP} corresponds to a number $\in (0, 1)$.

Regret insertion was proposed by Ropke and Pisinger (2006b) and tries to overcome the myopic behavior of greedy insertion by implementing a look-ahead strategy. More specifically, a regret-k value is calculated for each vertex as the difference between the cost increase resulting from the cheapest insertion of this vertex into the k-best DV route and its optimal insertion into the best DV route. At each iteration, we perform the best insertion of the vertex associated with the largest regret value. We implement regret insertion for $k = 2, 3, 4$. Note that we only use this operator for the insertion of customers and transfers but not for the assignment of transfers to customers that are already served in the current solution.

5.3.5.2. Route Selection and Evaluation of Removal and Insertion

Besides removal operators that rely on removing vertices without considering the properties of the routes they are currently assigned to, we implement operators which aim at removing vertices from routes with certain characteristics. More precisely, several route selection policies can be applied by route and segment removal (random variants) and by random, cluster, and relatedness removal. Besides policies with general validity for VRPs, we use new ones specifically designed for the VRP-TWMD.

In addition to a purely random selection, we implement the following policies that determine routes according to a roulette-wheel selection mechanism based on specific criteria:

Cost selects a route with a probability proportional to the cost of the route including penalty costs.

Distance is based on the length of a route. The selection probability of a route increases with the associated traveled distance.

Efficiency aims at identifying inefficient routes, i.e., routes that are characterized by large detours to cover a relatively small quantity of customer or transfer demands. The probability of a route to be selected is proportional to the ratio of the associated traveled distance and the total demand in this route.

Synchronization offset favors routes containing possibly poorly synchronized transfers. To this end, this policy selects a route with a probability proportional the cumulated synchronization offset over all transfers performed in a route that is calculated similarly as in the synchronization offset removal operator.

Transfer density aims at realizing cost savings by removing unnecessary transfers. The selection

probability of a route is proportional to the ratio of the number of transfers and the total number of vertices in this route.

With respect to the route-based removal operators, route removal DV and segment removal are able to use all selection policies, whereas transfer density is not defined for route removal SV. Moreover, the lower part of Table 5.3 shows the compatibility of the route selection policies and the vertex-based removal operators random, cluster, and relatedness removal with respect to the scope of the removal operation. Once a compatible removal operator has been selected, we subsequently select a policy based on probabilities $\boldsymbol{\pi}^r$. The selected route selection policy is used by the current removal operator for the entire iteration.

Moreover, for the purpose of guiding the search towards still unexplored areas, we create two additional variants for each scope-related variant of worst removal and for greedy insertion, GRASP insertion, and regret insertion that differ in how the corresponding operation is evaluated. Besides the basic operator variants that evaluate the associated operation based on the change in routing cost according to the generalized cost function (Δf_{gen}), we implement variants that employ the evaluation measures:

1. *diversification* Δf_{div}, which augments the change in routing cost by a diversification penalty based on past arc occurrence frequencies:

$$\Delta f_{div}(\nu, \mathcal{S}) = \Delta f_{gen}(\nu, \mathcal{S}) + \kappa \frac{f_{gen}(\mathcal{S})}{(n + n^t)} \sqrt{\sum_{(i,j) \in \mathcal{A}_\nu^+} h(i,j)}.$$

   Here, $\Delta f_{gen}(\nu, \mathcal{S})$ represents the change of the objective function value of solution $\mathcal{S}$ caused by removal or insertion operation ν, κ a real-valued parameter to control the amount of diversification, $f_{gen}(\mathcal{S})$ the current objective function value of solution $\mathcal{S}$, n the number of customers given by the problem instance, n^t the number of transfers currently scheduled in solution $\mathcal{S}$, $h(i,j)$ the occurrence frequency of arc (i,j) in previously generated solutions, and $\mathcal{A}_\nu^+$ the set of arcs generated by operation ν, and
2. *noise* Δf_{noise}, which multiplies the change in routing cost due to operation ν by a random number ζ drawn from the interval $[\zeta_{min}, \zeta_{max}]$ (see Ropke and Pisinger, 2006b; Hemmelmayr, Cordeau, and Crainic, 2012, for a similar approach):

$$\Delta f_{noise}(\nu, \mathcal{S}) = \Delta f_{gen}(\nu, \mathcal{S}) \zeta.$$

5.3.5.3. Adaptive Mechanism

In each ALNS iteration, we perform the choice of removal interval, removal operator, route selection policy (if applicable), and insertion operator(s) according to a roulette-wheel selection procedure as proposed in Ropke and Pisinger (2006b) based on the probability vectors $\boldsymbol{\pi}^{||}$, $\boldsymbol{\pi}^-$, $\boldsymbol{\pi}^r$, and $\boldsymbol{\pi}^+$, respectively. More precisely, given a set of adaptive components denoted as $\mathcal{X} \in \{||, -, r, +\}$, the selection probability of component $i \in \mathcal{X}$ is calculated as $\pi_i = w_i / \sum_{j \in \mathcal{X}} w_j$, where w_i corresponds to the weight of component i. All components of a set $\mathcal{X}$ are initially assigned the same weight and dynamically updated depending on their past search performance. We measure the performance of an adaptive component by means of a scoring system. We add a score of σ^{best} to the current score of a component whenever it contributed to improve the current best solution $\mathcal{S}^*$, a score of σ^{imp} if the new solution $\mathcal{S}'$ improves on the current one $\mathcal{S}$ and has never been encountered before, and a score

of σ^{acc} if $\mathcal{S}'$ is worse than $\mathcal{S}$ but accepted according to our SA acceptance mechanism and has never been generated before in the search. We keep track of the solutions already visited in the course of the search by means of a solution memory.

The weight of each component is updated every γ ALNS iterations based on the performance of the component during this interval. Let $\varnothing_i$ denote the current score of component i and β_i the number of applications of the component since the last weight update. Then, the new weight is calculated as $w_i = w_i(1-\alpha) + \alpha\varnothing_i/\beta_i$. The reaction factor $\alpha \in [0,1]$ allows to control the sensitivity of the weight adjustment with respect to the performance of the component. We reset the values of $\varnothing_i$ and β_i to zero after each update.

5.3.6. The Path Relinking Component

The PR paradigm as originally proposed by Glover (1997) represents an intensification strategy that exploits the observation that good solutions often share common characteristics. To this end, PR investigates the trajectories between elite solutions obtained during the search with the goal of discovering improving solutions in the process. More precisely, PR consists in iteratively transforming an initial solution into a guiding solution, thus creating a path of solutions between them. This is realized by successively incorporating characteristics of the guiding solution into the initial solution. The diversity between both solutions is thus stepwise decreased.

PR reasonably complements our ALNS component, which mainly serves the purpose of diversifying the search. We aim at discovering improving solutions that are not reachable via local search by investigating the trajectories between promising solutions located in distant regions of the search space. The successful hybridization of well-known metaheuristics for VRP variants with PR approaches is also demonstrated by, e.g., Ho and Gendreau (2006), Villegas et al. (2011), and Nguyen, Prins, and Prodhon (2012). Figure 5.12 shows our PR implementation in pseudocode.

$\mathcal{S}' \leftarrow \varnothing$
for each solution $\mathcal{S}_e \in \mathcal{E}$ **do**
  {*Initialize current and guiding solution*}
  $\mathcal{S}_c \leftarrow \mathcal{S}$
  $\mathcal{S}_g \leftarrow \mathcal{S}_e$
  $B \leftarrow initVertexList()$
  {*Determine arcs exclusively contained in guiding solution and initialize path length*}
  $\mathcal{A}^{\neq} \leftarrow getDifferingArcs(\mathcal{S}, \mathcal{S}_g)$
  $p \leftarrow |\mathcal{A}^{\neq}|$
  for each vertex $i \in B$ **do**
    {*Adjust transfer status of vertex if necessary*}
    $\mathcal{S}_c \leftarrow equalizeVertex(i, \mathcal{S}_g)$
    $\mathcal{S}_c \leftarrow performRelocation(i, \mathcal{S}_c, \mathcal{A}^{\neq})$
    if $\mathcal{S}_c \neq \mathcal{S} \wedge \mathcal{S}_c \neq \mathcal{S}_g \wedge (\mathcal{S}' = \varnothing \vee \mathcal{S}_c$ improves on $\mathcal{S}')$ **then**
      $\mathcal{S}' \leftarrow \mathcal{S}_c$
    end if
    $\mathcal{A}^{\neq} \leftarrow updateDifferingArcs(\mathcal{S}_c)$
    if $|\mathcal{A}^{\neq}| < \lfloor(1-\rho)\cdot p\rfloor$ **then**
      break
    end if
  end for
end for

Figure 5.12.: Pseudocode of our PR component.

Whenever we find a solution $\mathcal{S}$ that has never been encountered before in the search and the elite set $\mathcal{E}$ is filled, we apply our PR procedure between $\mathcal{S}$ and all solutions contained in $\mathcal{E}$. To this end, we set for each solution $\mathcal{S}_e$ in $\mathcal{E}$, the current solution on the solution path $\mathcal{S}_c$ to the initial solution $\mathcal{S}$ and the guiding solution $\mathcal{S}_g$ to $\mathcal{S}_e$. If $\mathcal{S}_g$ utilizes a higher number of vehicles, we need to equalize both vehicle numbers by adding empty routes of the respective type(s) to $\mathcal{S}_c$ accordingly.

We explain the following steps of our PR implementation using Figure 5.13, which illustrates the transformation of an initial solution into a guiding solution for an example instance composed of $n = 10$ customers. The guiding solution deploys five and the initial solution four DVs. Thus, we need to add an empty DV route to $\mathcal{S}_c$. In both solutions, $n^t = 3$ transfers are scheduled that are served by one SV. All routes start and end with the depot vertices 0 and $n + 1 = 11$, respectively.

Initially, we determine those arcs that are contained in $\mathcal{S}_g$ but not in the initial solution S and store them in the set of differing arcs $\mathcal{A}^{\neq}$. In our example, there are seven arcs which are exclusively contained in $\mathcal{S}_g$ (marked as ✗). Our PR procedure now aims at iteratively incorporating the arcs in $\mathcal{A}^{\neq}$ into the current solution, thus stepwise approaching the guiding solution. However, because the guiding solution is locally optimal (with respect to the penalty costs valid at the time of its discovery), we deem it unlikely to find improving solutions in its immediate proximity. Therefore, we only investigate a fraction ρ of the solution path. To this end, we use a modified version of our local search described in Section 5.3.4 which is stopped as soon as the number of remaining differing arcs is smaller than the fraction $(1 - \rho)$ of the initial cardinality of $\mathcal{A}^{\neq}$. However, for the purpose of illustration, Figure 5.13 shows the creation of the entire solution path, leading to the rediscovery of the guiding solution in iteration 3.

As in the local search, we first store all vertices contained in $\mathcal{S}_c$ in a list B with random order and iteratively try to relocate each vertex. It is important to note that, if the number of visits to dedicated transfer vertices in $\mathcal{S}_c$ is smaller than the corresponding number in $\mathcal{S}_g$, we initially create the missing visits to the respective transfer locations and add them to list B.

If the current vertex is a customer, we first check if its transfer state complies with that of its counterpart in the guiding solution. More precisely, if a transfer is scheduled at a customer i in $\mathcal{S}_c$ but not in $\mathcal{S}_g$, we remove the transfer status of customer i and thus the associated SV visit from $\mathcal{S}_c$. On the other hand, if no transfer is scheduled at customer i in $\mathcal{S}_c$ but it is in $\mathcal{S}_g$, we assign a transfer to i with the respective task order as executed in $\mathcal{S}_g$. We perform the corresponding support visit insertion that maximizes the number of created differing arcs. Ties are broken based on the change in routing cost according to f_{gen}. In our example, the transfer state needs to be removed from customer 4 and assigned to customer 2.

Subsequently, we evaluate the relocation of the regarded vertex. In contrast to the local search, we only allow the removal of a vertex if at least one of the arcs to be removed is not contained in the guiding solution. In case the relocation is permitted, we subsequently perform the reinsertion that maximizes the number of created differing arcs with ties again broken via f_{gen}.

For example, assume that customer 5 is selected first. It may not be relocated from route DV1 because both arcs $(10, 5)$ and $(5, 11)$ are matching arcs, i.e., part of both solutions. Instead, we skip this customer and move to customer 9 whose relocation is permitted because only arc $(0, 9)$ is contained in the guiding solution. The removal of customer 9 leads to the creation of the differing arc $(0, 8)$. Then, we reinsert customer 9 into the empty route DV5. In this way, two differing arcs

are created, namely $(0, 9)$, which has previously been removed, and $(9, 11)$. Set $\mathcal{A}^{\neq}$ is dynamically updated accordingly. We set the best solution $\mathcal{S}'$ to $\mathcal{S}_c$ if (i) $\mathcal{S}_c$ differs from $\mathcal{S}$ and $\mathcal{S}_g$, and (ii) $\mathcal{S}'$ has not been initialized yet, or $\mathcal{S}_c$ improves on $\mathcal{S}'$. Now, assume that customer 4^s is the next vertex to consider. In the guiding solution, no transfer takes place at customer 4. Therefore, we first remove the transfer status from customer 4 and the corresponding support visit $4'$ from $\mathcal{S}_c$. Then, customer 4 is relocated within the same route, introducing arcs $(2, 1)$, $(1, 4)$, and $(4, 11)$. Finally, a transfer is scheduled at customer 2 in $\mathcal{S}_g$ but not in $\mathcal{S}_c$. Thus, we need to assign a transfer to customer 2 and insert the corresponding support visit $2'$ between the support visits $10'$ and $3'$, which introduces the remaining differing arcs $(10, 2)$ and $(2, 3)$.

Iteration	Route	$\mathcal{S}_c$	$\mathcal{S}_g$	Arcs created in $\mathcal{S}_c$
0	DV1	0 - 10^t - 5 - 11	0 - 10^t - 5 - 11	
	DV2	0 - 6 - 3^s - 11	0 - 6 - 3^s - 11	
	DV3	0 - 2 - 4^s - 1 - 11	0 - 2^s ✗ 1 ✗ 4 ✗ 11	
	DV4	0 - 9 - 8 - 7 - 11	0 ✗ 8 - 7 - 11	
	DV5	0 - 11	0 - 9 ✗ 11	
	SV1	0 - $10'$ - $4'$ - $3'$ - 11	0 - $10'$ ✗ $2'$ ✗ $3'$ - 11	
1	DV1	0 - 10^t - 5 - 11	0 - 10^t - 5 - 11	
	DV2	0 - 6 - 3^s - 11	0 - 6 - 3^s - 11	
	DV3	0 - 2 - 4^s - 1 - 11	0 - 2^s ✗ 1 ✗ 4 ✗ 11	
	DV4	0 - 8 - 7 - 11	0 - 8 - 7 - 11	(0,8)
	DV5	0 - 9 - 11	0 - 9 - 11	(9,11)
	SV1	0 - $10'$ - $4'$ - $3'$ - 11	0 - $10'$ ✗ $2'$ ✗ $3'$ - 11	
2	DV1	0 - 10^t - 5 - 11	0 - 10^t - 5 - 11	
	DV2	0 - 6 - 3^s - 11	0 - 6 - 3^s - 11	
	DV3	0 - 2 - 1 - 4 - 11	0 - 2^s - 1 - 4 - 11	(2,1), (1,4), (4,11)
	DV4	0 - 8 - 7 - 11	0 - 8 - 7 - 11	
	DV5	0 - 9 - 11	0 - 9 - 11	
	SV1	0 - $10'$ - $3'$ - 11	0 - $10'$ ✗ $2'$ ✗ $3'$ - 11	
3	DV1	0 - 10^t - 5 - 11	0 - 10^t - 5 - 11	
	DV2	0 - 6 - 3^s - 11	0 - 6 - 3^s - 11	
	DV3	0 - 2^s - 1 - 4 - 11	0 - 2^s - 1 - 4 - 11	
	DV4	0 - 8 - 7 - 11	0 - 8 - 7 - 11	
	DV5	0 - 9 - 11	0 - 9 - 11	
	SV1	0 - $10'$ - $2'$ - $3'$ - 11	0 - $10'$ - $2'$ - $3'$ - 11	(10,2), (2,3)

Figure 5.13.: Example application of our PR procedure.

5.3.7. Acceptance Decision

As an additional way of promoting diversification, we employ a mechanism inspired by SA to decide if solution $\mathcal{S}'$ should replace $\mathcal{S}$ as starting point for the next iteration.

While solutions that improve $\mathcal{S}$ are always accepted, we accept worse solutions with probability $e^{-(f_{gen}(\mathcal{S}')-f_{gen}(\mathcal{S}))/\vartheta}$. Initially, we set the temperature parameter ϑ to a value such that a deterioration of the current solution $\mathcal{S}$ by Δ^{SA} is accepted with a probability of 50%. We decrease the temperature after every ALNS-PR iteration such that the acceptance probability of a relative deterioration of Δ^{SA} is equal to 0.1% after a cooling period of θ iterations.

5.4. Computational Studies

In this section, we present the numerical studies to evaluate the performance of our ALNS-PR. We begin with the description of the benchmark instances that we use to test our algorithm in Section 5.4.1. Subsequently, we present the computational environment and the parameter setting for our solution method (Section 5.4.2). In Section 5.4.3, we investigate the contribution of specific components on the performance of our algorithm. In Section 5.4.4, we assess the competitiveness of our ALNS-PR on the 2E-MTVRP-SS benchmark from the literature. Finally, we present our detailed results on the new VRPTWMD instances in Section 5.4.5 and analyze the effect of different problem characteristics on the solution structure in Section 5.4.6.

5.4.1. Benchmark Instances

In this section, we present the benchmark instances used in our computational experiments. We first describe the 2E-MTVRP-SS instances from the literature that we use to assess the competitiveness of our algorithm (Section 5.4.1.1) and subsequently present the generation scheme to create instances for the VRPTWMD (Section 5.4.1.2).

5.4.1.1. 2E-MTVRP-SS Instances

The 2E-MTVRP-SS proposed by Grangier et al. (2016) is characterized by two fleets of vehicles: First-level vehicles carry requests from a city distribution center (CDC) to satellite facilities and second-level vehicles transport these requests from the satellites to the customers. Each second-level vehicle may perform multiple trips starting from different satellites. The 2E-MTVRP-SS can thus be interpreted as a variant of the VRPTWMD where the second-level vehicles corresponding to DVs have to meet with first-level vehicles that act as SVs at dedicated transfer vertices, the satellites, to be enabled to fulfill the customer requests. Thus, $\mathcal{C}^t = \varnothing$ applies in the 2E-MTVRP-SS. In addition, note that each DV starts empty and its load is not necessarily replenished to its full capacity at a transfer but only to the amount required to serve the customers contained in the succeeding trip to the next transfer or the depot. Grangier et al. (2016) further assume a hierarchical objective function minimizing the number of SVs first, the number of DVs second, and the travel cost third. To achieve this, we set $c^S = 1000$ and $c^D = 250$ for these instances.

The 2E-MTVRP-SS instances are derived from the 56 VRPTW instances with 100 customers proposed by Solomon (1987). In these instances, the customers are clustered (prefix c), randomly distributed (prefix r), or a mixture of both (prefix rc). In addition, each instance is either characterized by narrow time windows and small vehicle capacities (prefix c1/r1/rc1) or large time windows and large vehicle capacities (prefix c2/r2/rc2).

To obtain 2E-MTVRP-SS instances, the authors extend each original VRPTW instance by a CDC and eight satellite facilities. The depot in the original instance corresponds to the depot of the DVs. Because the CDC is located farther away from the customers than the DV depot, the authors additionally extend the boundaries of each original customer time window by a certain offset to guarantee the feasibility of each request. No transfer time is incurred upon meeting of a DV and an SV. Finally, let L denote the capacity given in the original instance. Then, the authors set $L^D = 0.5L$ and $L^S = 4L$ for the instances of type 1 and $L^D = 0.25L$ and $L^S = 2L$ for the instances of type 2.

5.4.1.2. Generation of VRPTWMD Instances

To create VRPTWMD instances, we also make use of the well-known VRPTW instances by Solomon (1987). We assume that transfers are only possible at the customer locations, i.e., $\mathcal{C}^t = \mathcal{C}$ and $\mathcal{D} = \varnothing$. Thus, no additional vertices need to be located.

For each application context, we create 56 VRPTWMD instances, by setting R to a certain percentage of the corresponding average resource utilization in a high-quality solution of the respective VRPTW instance. More precisely, if (i) $R = F$, we set F to 70% of the average route length in the high-quality solution of the respective original VRPTW instance, or (ii) $R = L^D$, we set L^D to 70% of the average cumulated demand per route in the high-quality solution of the respective original VRPTW instance with the lower bound being the maximum demand given in the instance.

Each resulting capacity value is rounded down to the next multiple of ten and the remaining capacities are either taken from the original instance or set to a sufficiently large value if not provided in the instance. The VRPTW solution values are taken from https://www.sintef.no/projectweb/top/vrptw/solomon-benchmark/100-customers/.

The load capacity L^S of the SVs is set to $4R$. Moreover, to keep the time consumption due to transfers within the same order of magnitude as the time values defined in the original instance, we set the transfer time τ to the service duration given in the instance. To account for the additional time consumption, we increase the latest start of service l_i by $0.05l_0$ for each vertex i. The fixed costs c^D per used DV are calculated by dividing the objective function value of the respective high-quality VRPTW solution by the number of vehicles employed in this solution (rounded up to the next multiple of 10). We assume the deployment of SVs to be twice as expensive, i.e., we set $c^S = 2c^D$.

In Section 5.4.6, we analyze the influence of (i) the capacity of the replenishable resource R, (ii) the ratio of L^S and R, (iii) the ratio of the SV and the DV deployment cost, and (iv) the transfer time on the solution structure.

In addition, we generate a set of small VRPTWMD instances as follows: For the first VRPTW instance of each group (c1/r1/rc1/c2/r2/rc2), we create three small VRPTW instances by drawing five, ten, and 15 customers of the original instance at random and removing the remaining ones. In addition, we divide the vehicle capacity of the original instance by four. We subsequently solve the resulting 18 small VRPTW instances using our ALNS-PR without the consideration of transfers. Finally, we derive 36 small VRPTWMD instances similar to the procedure described above to obtain the large VRPTWMD instances. More precisely, we use the solutions obtained on the small VRPTW instances to calculate the vehicle capacities and costs for each VRPTWMD variant.

All instances are denoted by the prefix VRPTWMD, followed by the capacity type that may be replenished via transfers, the identifier of the underlying VRPTW instance, and the number of customers in the instance. For example, VRPTWMD-F-c101-5 denotes the VRPTWMD instance which allows the replenishment of the DVs fuel capacity, is obtained from the VRPTW instance c101, and contains five customers.

5.4.2. Computational Environment and Parameter Setting

Our ALNS-PR is implemented as single-thread code in Java. All tests were performed on a Windows 10 Professional desktop computer with an Intel Core i5-6600 processor running at 3.30 GHz and 16

GB RAM. In all experiments, we performed ten runs on each instance.

During the development of our solution method, we identified a subset of parameters with a stronger impact on the solution quality than the remaining parameters. We adopt the procedure proposed in Ropke and Pisinger (2006b) to refine the setting of these key parameters. More precisely, starting from a reasonably well-performing parameter setting obtained during the development of our algorithm, we successively refine the setting of each key parameter. To this end, we evaluate three values for each parameter on a subset of ten randomly selected VRPTWMD-F instances. The best value is kept as the final setting for the respective parameter and we subsequently proceed with tuning the next one. In Table 5.4, we report the results of the parameter tuning.

ALNS			
$\Psi^{\vert\vert}$	**[0.01, 0.1]**	[0.01, 0.25]	[0.01, 0.4]
Avg. Δ^b(%)	0.16	1.03	0.57
α	0.6	0.7	**0.8**
Avg. Δ^b(%)	0.58	0.16	0.04
χ^{worst}	1	**2**	3
Avg. Δ^b(%)	0.57	0.04	0.65
χ^{rel}	1	2	**3**
Avg. Δ^b(%)	2.00	0.04	0.00
χ^c, χ^u, χ^e	**1, 2, 4**	2, 1, 2	4, 2, 1
Avg. Δ^b(%)	0.00	0.29	0.68
χ^{nb}	1	**2**	3
Avg. Δ^b(%)	1.57	0.00	0.46
χ^{GRASP}	**0.05**	0.1	0.15
Avg. Δ^b(%)	0.00	0.59	0.59
κ	0.5	**0.7**	0.9
Avg. Δ^b(%)	0.29	0.00	0.46
$[\zeta_{min}, \zeta_{max}]$	[0.95, 1.05]	**[0.9, 1.1]**	[0.85, 1.15]
Avg. Δ^b(%)	0.66	0.00	0.08
PR			
λ	5	**10**	15
Avg. Δ^b(%)	0.62	0.00	2.39
ρ	0.5	**0.7**	0.9
Avg. Δ^b(%)	0.26	0.00	0.19
Penalties			
δ^{update}	**1.1**	1.5	2
Avg. Δ^b(%)	0.00	0.19	0.98
SA			
Δ^{SA}	0.005	**0.01**	0.015
Avg. Δ^b(%)	0.34	0.00	0.60
θ	50	**100**	200
Avg. Δ^b(%)	0.30	0.00	0.55

Table 5.4.: Results of different parameter settings on a randomly drawn subset of ten VRPTWMD-F instances structured according to the associated algorithmic components. For each parameter, we examine three values and provide for each value the respective average gap of the best solutions found to the BKS as determined during our entire testing activities (Δ^b) in percent. The best setting for each parameter is marked in bold and used as the final setting.

For the ALNS, we refine the boundaries of the removal interval ($\Psi^{\vert\vert}$), the reaction factor for the weight

adjustment of the adaptive components (α), the parameters to control the degree of randomness in specific removal operators (χ^{worst}, χ^{rel}, and χ^{nb}), the weights used to calculate the relatedness measure between two customers (χ^{c}, χ^{u}, and χ^{e}), the percentage of best insertions considered by the GRASP insertion operator (χ^{GRASP}), the factor used to calculate the diversification penalty (κ), and the interval from which a random number is drawn to augment the change in routing cost by noise ($[\zeta_{min}, \zeta_{max}]$).

For the PR component, we investigate the influence of the size of the elite set (λ) and the percentage of the solution path which we investigate in each PR execution (ρ). Finally, we tune the penalty update factor (δ^{update}) and the relative solution deterioration (Δ^{SA}) used to determine the initial and minimal temperature and the cooling period (θ) of our SA-inspired acceptance mechanism. For each parameter, we provide the average of the gaps of the best solutions found to the respective best-known solution (BKS) as determined during our entire testing activities in percent (Δ^{b}). The best and thus final value for each parameter is marked in bold.

The complete parameter setting is shown in Table 5.5. We additionally report the general ALNS-PR parameters maximum number of iterations without improvement (ω) and number of non-improving iterations after which the current solution is reset to the current best (μ). For the experiments on the new VRPTWMD instances, we set $\omega = 2000$ and for the 2E-MTVRP-SS instances, we set $\omega = 500$ to obtain competitive computation times.

For our ALNS component, we additionally provide the scores to measure the success of the adaptive components (σ^{best}, σ^{imp}, and σ^{acc}) and the number of iterations after which the weights of the adaptive components are updated (γ). We further report the initial (δ^{0}), minimum (δ^{min}), and maximum (δ^{max}) penalty factors and the numbers of ALNS-PR iterations after which the penalty factors are increased (η^{+}) as well as decreased (η^{-}). Finally, the parameters for the SA-inspired acceptance mechanism are complemented by the number of solution resets after which the temperature is reset to its initial value and the current solution is set to a solution randomly drawn from the elite set (ϵ).

ALNS-PR		**ALNS**		**PR**		**Penalties**		**SA**	
ω	2000/500	$\Psi^{\vert\vert}$	$[0.01, 0.1]$	λ	10	δ^{0}	10	Δ^{SA}	0.01
μ	200	σ^{best}, σ^{imp}, σ^{acc}	$6, 9, 3$	ρ	0.7	δ^{min}	0.1	θ	100
		α	0.8			δ^{max}	10 000	ϵ	2
		γ	20			δ^{update}	1.1		
		χ^{nb}	2			η^{+}	2		
		χ^{worst}	2			η^{-}	2		
		χ^{rel}	3						
		χ^{c}, χ^{u}, χ^{e}	$1, 2, 4$						
		χ^{GRASP}	0.05						
		κ	0.7						
		$[\zeta_{min}, \zeta_{max}]$	$[0.9, 1.1]$						

Table 5.5.: Final parameter setting of our ALNS-PR structured according to algorithmic components.

5.4.3. Influence of Algorithmic Components

In this section, we investigate the contribution of specific components of our ALNS-PR algorithm. To this end, we again use ten randomly selected VRPTWMD-F instances.

With respect to the ALNS, we analyze the effect of the problem-specific route selection policies (synchronization offset and transfer density) and removal operators (synchronization offset and segment

removal). In addition, we are interested in the impact of rearranging partial transfers, i.e., only one of both visits associated with a transfer instead of the entire transfer. To this end, we implement an algorithmic variant that separately relocates DV or SV visits at each ALNS iteration where the respective counterparts stay in the current solution. Moreover, we assess the effect of our PR approach on the solution quality.

To measure the individual contribution of each component, we successively disable each component and keep the remaining components enabled. We then compare the results obtained with the respective configuration to those of our full ALNS-PR where all components are active. In Table 5.6, we summarize the results of the component study. In particular, we compare the average percentage gaps of the best solutions found to the BKS (Avg. Δ^b) of each algorithmic configuration. The configuration representing our full ALNS-PR is shown in bold.

	Components							
Route selection								
Synchronization offset	✓	✗	✓	✓	✓	✓	✓	✓
Transfer density	✓	✓	✗	✓	✓	✓	✓	✓
ALNS operators								
Synchronization offset removal	✓	✓	✓	✗	✓	✓	✓	✓
Segment removal random	✓	✓	✓	✓	✗	✓	✓	✓
Segment removal synchronization	✓	✓	✓	✓	✓	✗	✓	✓
Consider only entire transfers	✓	✓	✓	✓	✓	✓	✗	✓
Path relinking	✓	✓	✓	✓	✓	✓	✓	✗
Avg. Δ^b(%)	**0.00**	0.18	0.72	0.17	0.52	0.59	1.53	0.11

Table 5.6.: Comparison of the performance of different algorithmic configurations.

We can observe a notable decrease in solution quality for each reduced algorithmic configuration ranging from 0.11% to 1.53% compared to the full algorithm. The problem-specific route selection and removal operators used in the ALNS step appear to adequately address the existence of transfers and the associated synchronization requirements. On the contrary, the consideration of partial transfers in the ALNS step seems to considerably impede the search and shows by far the most significant deviation. This hints at the importance of jointly considering both the DV and the SV level to overcome local optima in the VRPTWMD. Finally, the results show that our ALNS is reasonably complemented by the proposed PR approach.

5.4.4. Results on 2E-MTVRP-SS Instances

In Table 5.7, we present our results on the 2E-MTVRP-SS instances proposed by Grangier, Gendreau, Lehuédé, and Rousseau (2016) in comparison to the results obtained by the same authors on these instances (GGLR).

For each instance, we report the number of SVs (m^S), the number of DVs (m^D), and the objective function value (f) associated with the BKS as reported by Grangier et al. (2016). For GGLR and ALNS-PR, we report the absolute deviation of the number of employed SVs (Δ^{m^S}) and of the number of employed DVs (Δ^{m^D}) in the best solution found in ten runs to the respective best-known vehicle numbers, the percentage gap of the best solution quality of ten runs to the BKS (Δ^b), the percentage gap of the average solution quality of ten runs to the BKS (Δ^a), and the average run-time (t^a) in minutes. For ALNS-PR, we additionally report the objective function value of the best solution of ten

runs (f^b). Moreover, averages of the run-times, the gaps to the BKS, and the total absolute deviations of the number of employed vehicles to the BKS over the complete set of instances are given at the end of the table. In addition, we provide the average of the gaps of the best solution quality to the BKS across all instances where we obtain the previous best-known vehicle numbers (Avg. ($\Delta^m = 0$)). Finally, to favor a fair run-time comparison, we translate the run-times of both methods into a common time measure that takes into account the processors used. To this end, we relate the Passmark score (see www.passmark.com) of the processor used by GGLR to the score of our i5-6600. The resulting times in minutes are given as t^c.

The results obtained by ALNS-PR are quite impressive. We are able to improve the previous BKS for 49 out of 56 instances. In detail, we reduce the number of employed DVs by 11 across the entire set of instances. With respect to the instances where we achieve the best-known vehicle numbers, we are able to improve the previous BKS by -3.21% on average. Even if we calculate the average over the entire set of instances, we observe a notable average improvement of -2.67%. In addition, the speed of ALNS-PR is remarkable. ALNS-PR only spends roughly 38% of the time required by GGLR. Moreover, ALNS-PR proves to be quite robust showing an average deviation of the average solution quality to the BKS of less than one percent. Considering that GGLR only allow feasible solutions during the search, the presented results might hint at the benefits of temporarily allowing infeasible solutions in tightly constrained problems like the 2E-MTVRP-SS and the VRPTWMD.

5.4.5. Results on VRPTWMD Instances

In this section, we present our results on the small VRPTWMD instances in comparison with the commercial solver CPLEX. As comparison for future methods that address the VRPTWMD, we provide our detailed results on the large VRPTWMD benchmark in Appendix A.

Table 5.8 shows the results of ALNS-PR and those of CPLEX on the small VRPTWMD-F and VRP-TWMD-L instances. For CPLEX, we report for each instance, the number of SVs (m^S), the number of DVs (m^D), the best upper bound (UB), and the run-time (t) in seconds. CPLEX is given a time limit of 7200 seconds and we allow a maximum of ten visits to the end depot $n+1$. Solutions found by CPLEX within the given time limit are optimal. For ALNS-PR, we provide the absolute deviation of the number of employed SVs (Δ^{m^S}) and of the number of employed DVs (Δ^{m^D}) in the best solution to the respective vehicle numbers found by CPLEX, the best solution quality of ten runs (f^b), the percentage gap of the best solution quality of tens runs to UB (Δ^b), the percentage gap of the average solution quality of ten runs to UB (Δ^a), and the average run-time (t^a) in seconds. In addition, averages of the run-times, the gaps to UB, and the total absolute deviations of deployed SVs and DVs are given for each set of instances.

CPLEX is able to solve 24 out of 36 instances, i.e., all instances with five and ten customers to optimality. With respect to the best solution quality, the results of ALNS-PR match or improve the quality of all optimal CPLEX solutions and all CPLEX upper bounds while showing significantly lower average run-times of only a few seconds. With respect to the structure of the identified solutions, we are able to reduce the number of employed DVs for two VRPTWMD-F instances. Finally, the robustness of our ALNS-PR is again indicated by the small deviations of the average solution quality to the best solution quality.

	BKS			GGLR					**ALNS-PR**					
Instance	m^S	m^D	f	Δ^{m^S}	Δ^{m^D}	Δ^b(%)	Δ^a(%)	t^a(min)	Δ^{m^S}	Δ^{m^D}	f^b	Δ^b(%)	Δ^a(%)	t^a(min)
c101	3	11	2022.4	0	0	0.00	1.73	52.4	0	0	1936.05	**-4.27**	-2.35	16.52
c102	3	10	**1947.6**	0	0	**0.00**	1.40	88.5	0	0	1966.35	0.96	3.35	18.38
c103	3	9	1880.7	0	0	0.00	3.51	56.7	0	0	1868.38	**-0.65**	2.45	22.00
c104	3	9	1811.1	0	0	0.00	2.55	58.9	0	0	1777.13	**-1.88**	2.54	27.21
c105	3	10	1934.0	0	0	0.00	1.30	61.7	0	0	1807.66	**-6.53**	-1.79	15.22
c106	3	**10**	1945.0	0	**0**	0.00	1.52	71.9	0	1	1838.05	-5.50	-0.11	15.31
c107	3	10	1888.9	0	0	0.00	0.74	43.7	0	0	1837.02	**-2.75**	0.28	15.96
c108	3	10	1875.3	0	0	0.00	1.88	47.4	0	0	1813.15	**-3.31**	0.42	16.77
c109	3	9	1863.1	0	0	0.00	3.12	69.8	0	0	1817.21	**-2.46**	2.46	18.42
c201	2	3	1389.3	0	0	0.00	1.79	65.0	0	0	1275.00	**-8.23**	-4.19	8.80
c202	2	3	1305.0	0	0	0.00	0.99	51.7	0	0	1289.12	**-1.22**	4.34	15.40
c203	2	3	1272.7	0	0	0.00	0.78	61.7	0	0	1269.69	**-0.24**	5.29	24.27
c204	2	3	1237.9	0	0	0.00	1.89	64.0	0	0	1181.24	**-4.58**	5.94	18.94
c205	2	3	1312.1	0	0	0.00	0.82	39.8	0	0	1247.37	**-4.93**	0.46	8.38
c206	2	3	1312.6	0	0	0.00	1.22	40.5	0	0	1217.65	**-7.23**	-0.30	13.93
c207	2	3	1280.4	0	0	0.00	2.05	42.1	0	0	1217.36	**-4.92**	2.14	13.53
c208	2	3	1278.3	0	0	0.00	1.70	41.5	0	0	1231.19	**-3.69**	3.05	15.47
r101	2	19	2333.5	0	0	0.00	0.80	48.2	0	0	2192.80	**-6.03**	-3.74	7.50
r102	2	18	2136.8	0	0	0.00	0.75	56.1	0	**-1**	1995.05	**-6.63**	-6.08	8.64
r103	2	13	1942.7	0	0	0.00	0.65	70.7	0	0	1860.60	**-4.23**	-7.11	13.87
r104	2	10	1777.2	0	0	0.00	1.55	98.9	0	**-1**	1603.66	**-9.76**	-7.84	20.68
r105	2	14	2096.8	0	0	0.00	1.99	44.9	0	0	1984.69	**-5.35**	-5.45	10.38
r106	2	12	1992.4	0	0	0.00	1.79	62.7	0	0	1821.93	**-8.56**	-5.61	15.70
r107	2	11	1779.2	0	0	0.00	2.12	67.3	0	**-1**	1675.62	**-5.82**	-1.56	17.67
r108	2	10	1654.3	0	0	0.00	2.90	60.4	0	**-1**	1546.95	**-6.49**	-1.94	18.06
r109	2	12	1925.9	0	0	0.00	2.95	60.4	0	0	1756.52	**-8.79**	-3.84	18.17
r110	2	12	1833.6	0	0	0.00	3.02	70.5	0	**-1**	1754.27	**-4.33**	-0.36	14.33
r111	2	12	1770.8	0	0	0.00	2.35	76.2	0	**-1**	1684.81	**-4.86**	0.85	20.81
r112	2	11	1746.0	0	0	0.00	3.14	78.3	0	**-1**	1664.30	**-4.68**	-2.34	21.66
r201	1	4	1587.8	0	0	0.00	1.68	33.7	0	0	1537.15	**-3.19**	0.89	11.57
r202	1	3	1530.8	0	0	0.00	1.14	72.7	0	0	1481.01	**-3.25**	-0.78	13.31
r203	1	3	1255.1	0	0	0.00	1.85	56.1	0	0	1213.16	**-3.34**	0.64	22.52
r204	1	2	1191.7	0	0	0.00	0.70	122.7	0	0	1110.16	**-6.84**	-2.83	16.17
r205	1	3	1319.1	0	0	0.00	2.95	36.5	0	0	1312.70	**-0.49**	4.15	10.97
r206	1	3	1228.3	0	0	0.00	1.89	52.1	0	0	1219.57	**-0.71**	2.23	21.62
r207	1	3	1140.2	0	0	0.00	1.53	65.0	0	**-1**	1186.66	4.07	6.68	24.72
r208	1	2	1050.2	0	0	0.00	3.08	59.8	0	0	1010.05	**-3.82**	3.73	15.78
r209	1	3	1258.7	0	0	0.00	1.34	46.3	0	0	1235.64	**-1.83**	2.08	12.11
r210	1	3	1279.8	0	0	0.00	1.61	53.8	0	0	1264.88	**-1.17**	2.65	15.40
r211	1	3	1118.2	0	0	0.00	2.75	65.7	0	**-1**	1278.68	14.35	4.27	16.16
rc101	3	16	2577.0	0	0	0.00	1.41	46.8	0	0	2511.19	**-2.55**	0.15	12.78
rc102	3	14	**2407.1**	0	0	**0.00**	2.22	66.2	0	0	2423.26	0.67	2.63	16.02
rc103	3	11	2476.9	0	0	0.00	2.62	121.8	0	0	2128.90	**-14.05**	-9.68	18.48
rc104	3	11	2125.9	0	0	0.00	1.77	82.6	0	**-1**	2003.53	**-5.76**	-2.05	22.06
rc105	3	15	**2542.6**	0	0	**0.00**	2.35	61.4	0	0	2564.89	0.88	-0.44	14.20
rc106	3	13	2494.9	0	0	0.00	3.60	74.4	0	0	2299.56	**-7.83**	-4.33	19.09
rc107	3	13	2271.1	0	0	0.00	1.76	80.5	0	**-1**	2362.09	4.01	4.09	21.17
rc108	3	12	2202.9	0	0	0.00	1.21	84.7	0	0	2181.81	**-0.96**	2.60	19.74
rc201	1	4	**1787.6**	0	0	**0.00**	2.62	36.3	0	0	1796.32	0.49	5.03	9.18
rc202	1	4	1513.8	0	0	0.00	2.01	102.5	0	**-1**	1811.46	19.66	10.89	14.32
rc203	1	3	1416.2	0	0	0.00	2.44	53.7	0	0	1408.91	**-0.51**	4.43	13.71
rc204	1	3	1188.2	0	0	0.00	1.99	60.1	0	0	1184.06	**-0.35**	4.53	19.18
rc205	1	4	1693.7	0	0	0.00	1.24	61.7	0	0	1635.92	**-3.41**	1.05	16.32
rc206	1	3	1583.1	0	0	0.00	2.78	76.2	0	0	1538.72	**-2.80**	2.72	12.51
rc207	1	3	**1449.8**	0	0	**0.00**	4.10	60.7	0	0	1495.71	3.17	8.78	11.07
rc208	1	3	**1257.3**	0	0	**0.00**	3.51	66.3	0	0	1292.04	2.76	5.70	13.70
Avg.						**0.00**	**1.98**	**63.4**				**-2.67**	**0.69**	**16.18**
Avg. ($\Delta^m = 0$)												**-3.21**		
$\sum$				**0**	**0**				**0**	**-11**				
Processor type				Xeon X5675					i5-6600					
Processor speed				3.07 GHz					3.30 GHz					
Passmark score				1402					2098					
t^c(min)				**42.37 × 10**					**16.18 × 10**					

Table 5.7.: Results of ALNS-PR and GGLR on the 2E-MTVRP-SS instances proposed by Grangier et al. (2016).

	CPLEX				ALNS-PR					
Instance	m^S	m^D	UB	t(s)	Δ^{m^S}	Δ^{m^D}	f^b	Δ^b(%)	Δ^a(%)	t^a(s)
VRPTWMD-F-c101-5	1	3	819.87	0.39	0	0	819.87	0.00	0.00	0.44
VRPTWMD-F-c101-10	1	4	990.61	434.49	0	0	990.61	0.00	0.00	4.80
VRPTWMD-F-c101-15	1	7	1613.79	7200	0	-1	1514.52	-6.15	-5.19	23.61
VRPTWMD-F-c201-5	0	2	443.03	0.92	0	0	443.03	0.00	0.00	0.16
VRPTWMD-F-c201-10	1	2	1012.56	1074.61	0	0	1012.56	0.00	0.00	0.81
VRPTWMD-F-c201-15	1	3	1228.47	7200	0	-1	1208.47	-1.63	-0.98	6.28
VRPTWMD-F-r101-5	0	4	674.21	1.61	0	0	674.21	0.00	0.00	0.13
VRPTWMD-F-r101-10	1	5	802.05	264.86	0	0	802.05	0.00	0.00	5.11
VRPTWMD-F-r101-15	2	5	1313.71	7200	0	0	1313.71	0.00	1.57	15.49
VRPTWMD-F-r201-5	1	2	716.59	7.80	0	0	716.59	0.00	0.00	0.26
VRPTWMD-F-r201-10	1	2	758.34	4944.37	0	0	758.34	0.00	0.60	1.71
VRPTWMD-F-r201-15	1	2	962.91	7200	0	0	962.91	0.00	0.00	7.87
VRPTWMD-F-rc101-5	1	2	683.53	1.19	0	0	683.53	0.00	0.00	0.29
VRPTWMD-F-rc101-10	1	5	1387.00	227.68	0	0	1387.00	0.00	0.00	1.71
VRPTWMD-F-rc101-15	2	6	1789.29	7200	0	0	1789.29	0.00	0.67	25.90
VRPTWMD-F-rc201-5	1	1	708.31	5.79	0	0	708.31	0.00	0.15	0.65
VRPTWMD-F-rc201-10	0	4	1111.38	3941.19	0	0	1111.38	0.00	0.00	0.27
VRPTWMD-F-rc201-15	1	2	951.13	7200	0	0	941.78	-0.98	-0.69	5.43
Avg.				**3182.62**				**-0.49**	**-0.21**	**5.61**
$\sum$					**0**	**-2**				
VRPTWMD-L-c101-5	0	3	517.64	4.65	0	0	517.64	0.00	0.00	0.17
VRPTWMD-L-c101-10	0	5	720.95	83.98	0	0	720.95	0.00	0.00	0.91
VRPTWMD-L-c101-15	1	5	1299.41	7200	0	0	1288.34	-0.85	-0.85	10.39
VRPTWMD-L-c201-5	0	2	421.81	9.67	0	0	421.81	0.00	0.00	0.15
VRPTWMD-L-c201-10	0	3	848.09	439.17	0	0	848.09	0.00	0.00	0.45
VRPTWMD-L-c201-15	0	4	999.33	7200	0	0	999.33	0.00	0.00	0.96
VRPTWMD-L-r101-5	0	3	550.34	2.23	0	0	550.34	0.00	0.00	0.14
VRPTWMD-L-r101-10	1	4	740.32	1080.77	0	0	740.32	0.00	0.23	4.75
VRPTWMD-L-r101-15	1	5	1064.15	7200	0	0	1064.15	0.00	0.22	6.04
VRPTWMD-L-r201-5	0	3	576.98	18.97	0	0	576.98	0.00	0.00	0.13
VRPTWMD-L-r201-10	0	4	680.99	2444.95	0	0	680.99	0.00	0.00	0.42
VRPTWMD-L-r201-15	0	4	882.60	7200	0	0	882.60	0.00	0.00	2.34
VRPTWMD-L-rc101-5	0	3	580.23	3.68	0	0	580.23	0.00	0.00	0.14
VRPTWMD-L-rc101-10	0	5	1062.09	173.80	0	0	1062.09	0.00	0.00	0.31
VRPTWMD-L-rc101-15	1	5	1256.62	7200	0	0	1256.62	0.00	0.28	2.19
VRPTWMD-L-rc201-5	0	3	558.51	24.98	0	0	558.51	0.00	0.00	0.15
VRPTWMD-L-rc201-10	0	3	927.02	1087.07	0	0	927.02	0.00	0.00	0.80
VRPTWMD-L-rc201-15	1	1	841.77	7200	0	0	841.77	0.00	0.00	6.85
Avg.				**2857.02**				**-0.05**	**-0.01**	**2.07**
$\sum$					**0**	**0**				

Table 5.8.: Comparison of the results of ALNS-PR and CPLEX on the small VRPTWMD-F and VRPTWMD-L instances.

5.4.6. Effect of Different Problem Characteristics

In this section, we investigate the effect of (i) the DVs' capacity of the replenishable resource, (ii) the ratio of the SVs' load capacity and the capacity of the replenishable resource, (iii) the ratio of the SV and the DV deployment cost, and (iv) the time incurred for each transfer operation on the structure of the VRPTWMD solutions. To this end, we first draw ten VRPTWMD instances for each application context at random. Based on these instances, we generate modified sets of instances for each problem characteristic by varying the respective value. The results on these instances are then compared to those obtained on the original ones.

In addition, we analyze the savings that can be achieved by treating the task order at a customer as a decision variable. To this end, we solve each original VRPTWMD-F instance where we scheduled at least one transfer again under the more rigid assumption of always executing the transfer first. We then compare the results obtained in this way with the original ones.

Vehicle capacities In Figure 5.14, we present the total number of deployed SVs ($\sum m^S$), the total number of deployed DVs ($\sum m^D$), the total number of performed transfers $\sum n^t$, and the average travel distance (*Avg.* TD) across all considered instances for different capacities of the replenishable resource R for each application context. More precisely, we vary the percentage that we use to obtain R from the respective resource consumption occurring in the high-quality solutions to the original VRPTW instances (see Section 5.4.1.2). In particular, we generate modified instances for 60%, 80%, and 90%. The value used to create the original VRPTWMD instances (70%) is marked in bold. Note that for VRPTWMD-F, we are not able to obtain a feasible solution for each considered instance with respect to the 60% capacity configuration. Therefore, we do not report results for this scenario and configuration.

Unsurprisingly, the number of employed SVs strongly correlates with the number of performed transfers. Both measures monotonically decrease with increasing resource capacity in both application scenarios. However, in case of $R = L^D$, no transfers seem to be necessary for a capacity percentage of 80% and higher, whereas fuel transfers are scheduled even for the least restrictive capacity configuration of 90%. Moreover, we observe a slight decrease in the number of employed DVs and of the average travel distance with decreasing capacity restriction.

Figure 5.15 shows the effect of different ratios of the SVs' load capacity L^S and R, namely 1, 2, 3, and 4, on the same measures. With respect to $R = F$, the number of necessary resource transfers only slightly fluctuates around the mean value of 48. As a consequence, an increasing number of SVs have to be utilized with decreasing load capacity L^S to meet this demand. However, this has no significant effect on the number of employed DVs or the traveled distance.

Although the effect described for the fuel scenario can also be observed for the load scenario initially, no transfers are performed at all for a capacity ratio of two and lower. While the depletion of the fuel capacity prevents a DV from moving in space, it is still able to reach every vertex with empty load capacity. Thus it seems reasonable to exchange SVs for a higher number of less expensive DVs as the difference between their capacities decreases. Here, the traveled distance is also barely affected. This is basically true for the large majority of investigated configurations because all vehicles are based at the same depot and no detours to dedicated transfer locations are necessary.

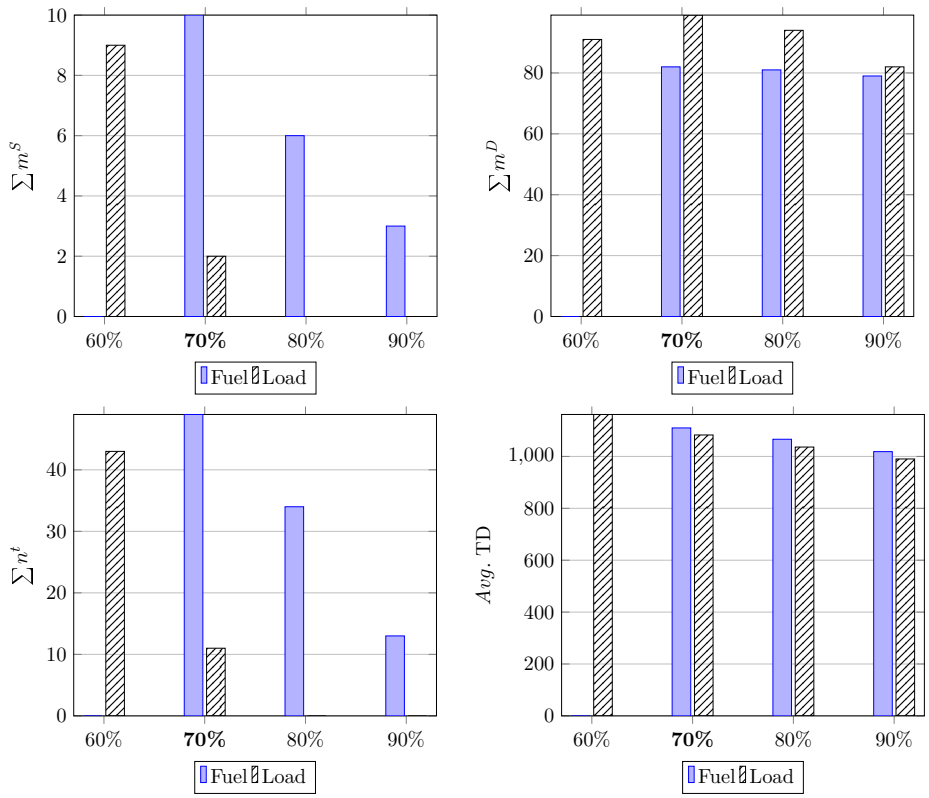


Figure 5.14.: Effect of different capacities of the replenishable resource R.

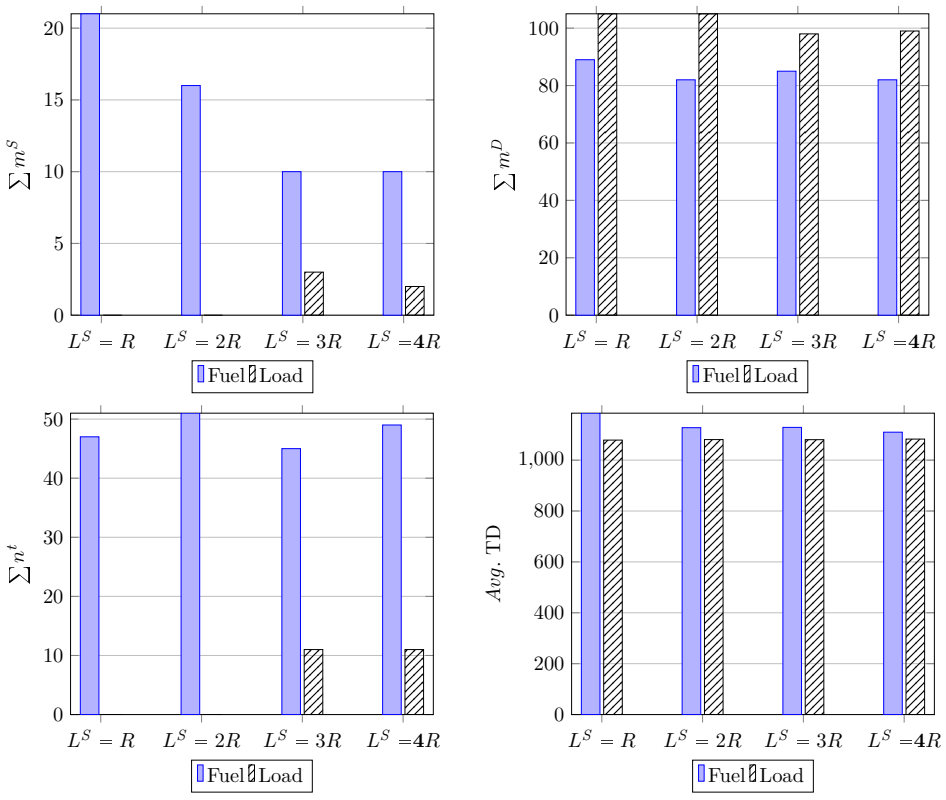


Figure 5.15.: Effect of different ratios of load capacity L^S and replenishable resource R.

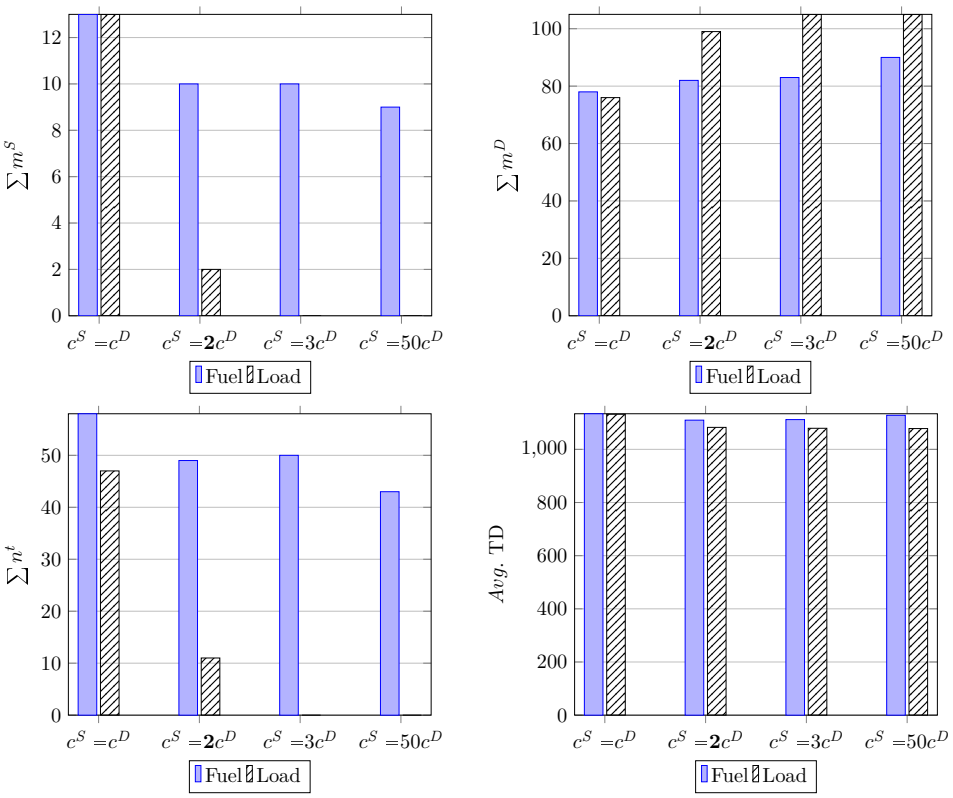


Figure 5.16.: Effect of different ratios of the DV and SV deployment cost.

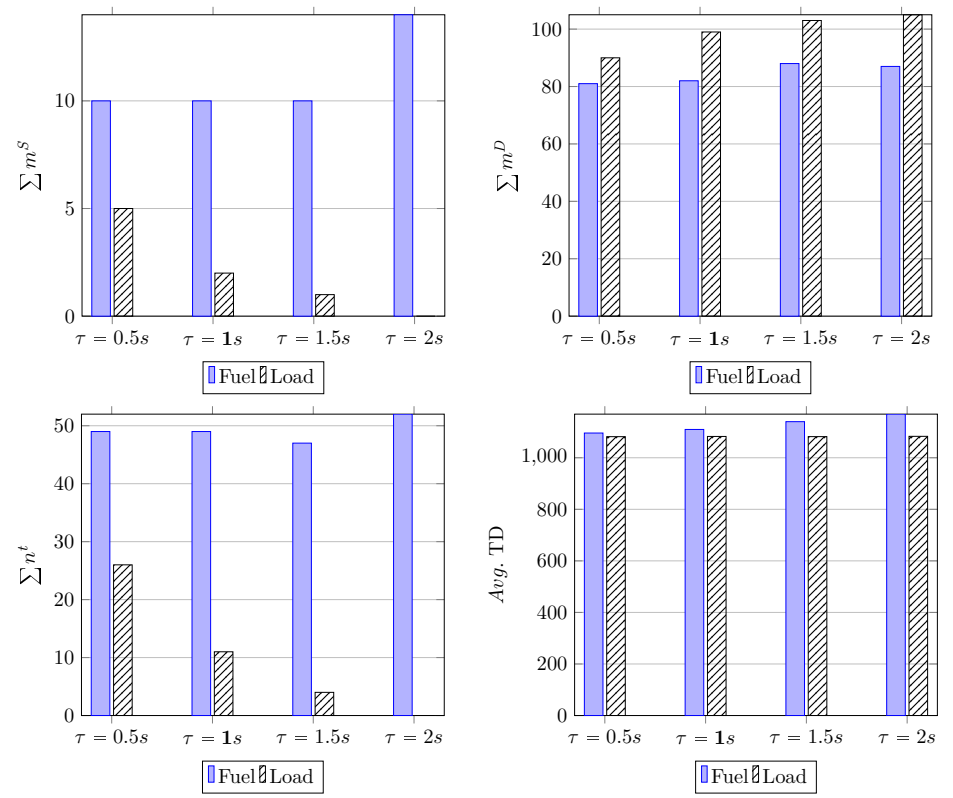


Figure 5.17.: Effect of different transfer times.

Vehicle cost Figure 5.16 displays the effect of different ratios of the SV and DV deployment cost. We investigate the values 1, 2, 3, and 50. The latter value is used to model the hierarchical minimization of vehicles. With respect to fuel, an increase of the SV deployment cost only leads to a slight decrease in the number of performed transfers and thus employed SVs. In turn, the number of employed DVs slightly increases. This hints again at the necessity of fuel replenishments to achieve feasibility of deliveries.

In the load scenario, the benefits of considering transfers vanish quickly as the utilization of SVs becomes more expensive. At a cost ratio of 3 already, we do not observe the use of transfers anymore.

Transfer time In Figure 5.17, we investigate the effect of different transfer times based on varying percentages of the service times given in the original VRPTW instances. In detail, we investigate 50%, 100%, 150%, and 200%.

Interestingly, the number of employed fuel SVs remain constant for the first three values and suddenly increase significantly in the case of 200%. As the time consumption associated with transfers rises, some time windows seem to become impossible to satisfy with the initial vehicle configuration at this point. Thus, a higher parallelization of tasks needs to be achieved by utilizing additional SVs and DVs.

For the load scenario, we again observe a clear trend of discarding the use of transfers as the problem setting becomes increasingly constrained.

Task order Finally, in Table 5.9, we show an aggregated comparison of the results obtained under the assumption of always executing the transfer first ($o_i = 1$) to those obtained when the task order is treated as a decision variable, i.e., $o_i \in \{0, 1\}$.

$o_i = 1$			$\boldsymbol{o_i \in \{0,1\}}$			
m^S	m^D	n^t	Δ^{m^S}	Δ^{m^D}	Δ^{n^t}	$\Delta^b(\%)$
58	377	266	-1	-7	-9	-1.08

Table 5.9.: Impact of treating the task order at a customer as a decision variable.

The results show that the flexibility of deciding which task is performed first allows a cost saving of -1.08%, on average. Moreover, in total, we are able to save one SV–as a consequence of requiring nine transfers less–and seven DVs.

5.5. Conclusion

In this chapter, we introduce the generic vehicle-routing problem with time windows and mobile depots (VRPTWMD) that is characterized by a fleet of support vehicles (SVs) and a fleet of delivery vehicles (DVs). Depending on the application scenario, the SVs serve as mobile depots to replenish either the fuel or the load capacity of the DVs.

We present a mixed-integer program for the VRPTWMD and optimal solutions on small problem instances for each application context. In addition, we develop an adaptive large neighborhood search

algorithm, complemented by a path relinking approach (ALNS-PR) to address large instances for the VRPTWMD.

In extensive numerical studies, we first show that the omission of the problem-specific components and of our PR implementation leads to a notable decrease in solution quality. Moreover, we demonstrate the competitiveness of our approach on benchmark instances for the related two-echelon multiple-trip VRP with satellite synchronization. We are able to significantly improve the large majority of the previous best-known solutions, namely 49 out of 56, while spending remarkably less computation time in comparison to the state-of-the-art algorithm from the literature.

Finally, on newly generated VRPTWMD instances, we analyze the effect of different problem characteristics, namely (i) the DVs' capacity of the resource that may be replenished, (ii) the ratio of the SVs' load capacity and the DVs' capacity of the replenishable resource, (iii) the ratio of the SV and the DV deployment cost, and (iv) the time incurred for each transfer operation on the solution structure. We observe that fuel transfers are essential in each problem configuration to be able to serve customers that are located far away from the depot. In contrast, we note a clear trend of discarding the use of load transfers as the problem setting becomes increasingly constrained with respect to transfer operations, i.e., due to a decreasing load capacity of the SVs, rising cost for the utilization of SVs, and an increasing time consumption associated with transfers. In addition, we show that treating the task order at a customer as a decision variable yields significant savings compared to a more rigid assumption of, e.g., always performing the transfer operation first.

Chapter 6

Reverse Logistics: Simultaneous Pickup and Delivery

6.1. Introduction

In this chapter, we study a class of VRPs with simultaneous pickup and delivery (VRPSPDs), in which customers may require (i) delivery service of goods originating at the depot, and (ii) pickup service for goods which need to be returned to the depot. VRPSPDs arise, for instance, in the context of reverse logistics. Reverse logistics is defined as the process of planning, implementing, and controlling the flow of, e.g., raw materials or finished goods from the point of consumption to the point of origin for the purpose of recapturing value or proper disposal by The Council of Logistics Management (Stock, 1992).

In particular, this chapter is motivated by a practical application occurring at DHL Freight in Sweden. There, the logistics company operates about 1400 vehicles which are based at 25 depots. Up to 50 000 customers requiring bulky goods and parcels to be delivered and to be picked up need to be served on a daily basis amounting to approximately 18 million shipments per year. This scenario can be modeled as a multi-depot VRPSPD whose scale calls for its decomposition into several VRPSPDs which need to be solved efficiently. In a first step, we aim at developing a solution method that is flexible enough to address the VRPSPD and its variants from the literature, and that provides competitive results on these problems.

In addition to the standard VRPSPD, we investigate (i) the VRPSPD with time limit (VRPSPDTL), which imposes a maximum duration on the vehicle routes, (ii) the VRPSPD with time windows (VRPSPDTW), where customers are associated with time intervals in which their service must start, (iii) the VRP with divisible deliveries and pickups (VRPDDP), which allows to satisfy a customer's pickup and delivery requests in two separate visits, (iv) the previously unstudied VRP with restricted mixing of divisible deliveries and pickups (VRPRMDDP), which additionally requires that a certain percentage of the vehicle capacity must remain free when both types of demand are loaded at the same time, and (v) the previously unstudied VRPDDP with time windows (VRPDDPTW).

Because the standard VRPSPD and its variants extend the $\mathcal{NP}$-hard capacitated VRP (CVRP), heuristic solution methods are a suitable choice for solving larger problem instances of practical relevance in short computation time.

For the VRPSPD, the most successful heuristics are the parallel iterated local search algorithm by Subramanian et al. (2010), that makes use of the variable neighborhood descent paradigm with random neighborhood ordering and is embedded in a multi-start framework, the approach of Zachariadis,

Tarantilis, and Kiranoudis (2010), which is characterized by a memory structure to store and recombine promising vertex sequences, the iterated local search of Subramanian, Uchoa, and Ochi (2013), that is combined with a set partitioning approach, the unified hybrid genetic search by Vidal et al. (2014a), and the variable neighborhood search (VNS) algorithm of Polat et al. (2015) that is extended by a perturbation mechanism.

For the VRPSPDTL, the approaches of Subramanian, Uchoa, and Ochi (2013), Polat et al. (2015), and that of Kalayci and Kaya (2016), which combines an ant colony system with a VNS algorithm, are the best-performing heuristics.

For the VRPSPDTW, Wang and Chen (2012) propose a co-evolution genetic algorithm and respective benchmark instances that are derived from instances for the VRP with time windows (VRPTW). Wang et al. (2015) are able to improve on numerous results reported by Wang and Chen (2012) using a parallel simulated annealing (SA) algorithm.

The VRPDDP is introduced by Nagy et al. (2015). The authors use a reactive tabu search algorithm to investigate the conditions under which the division of customer demands is beneficial. Recently, Polat (2017) presented cooperative VNS—a parallel approach based on the VNS paradigm—for the VRPDDP, which is able to obtain new best solutions for the large majority of the instances proposed by Nagy et al. (2015).

The VRPRMDDP is inspired by the investigations of Nagy, Wassan, and Salhi (2013) and Hoff et al. (2009). Nagy, Wassan, and Salhi (2013) introduce the VRP with restricted mixing of deliveries and pickups (VRPRMDP), which accounts for the difficulty of rearranging the vehicle load by requiring that a certain percentage of the vehicle capacity must remain unoccupied when both pickup and delivery demands are simultaneously loaded. In the VRPRMDP, each customer may demand pickup or delivery service but not both. The VRPRMDDP generalizes the VRPRMDP by assuming that customers can be associated with both types of demand that may be satisfied in two separate visits as in the VRPDDP. A related problem is investigated by Hoff et al. (2009). The authors aim at generating so-called "lasso" routes for the VRPDDP by exclusively satisfying delivery requests at the beginning of each route. Then, as soon as a certain percentage of the vehicle capacity has been freed, the remaining customers of the route may be visited for pickup and delivery. Similar to the motivation of the VRPRM(D)DP, ensuring a minimum percentage of free vehicle capacity before allowing pickups aims at improving the accessibility of the items still to be delivered. Finally, the initial delivery path is visited in reverse order to fulfill the remaining pickup requests before returning to the depot. The VRPRMDDP thus differs from the problem studied by Hoff et al. (2009) in that the given percentage of vehicle capacity must be free as long as the vehicle load consists of a mixture of both types of demand and not only before entering the pickup-and-delivery part of the route. In addition, no particular shape is imposed on the vehicle routes in the VRPRMDDP.

Ropke and Pisinger (2006a) proposed an adaptive large neighborhood search (ALNS) algorithm for a class of VRPs with backhauls that was also applied to a small subset of VRPSPD instances but could not compete with the state-of-the-art approaches specifically tailored to the VRPSPD. The contribution of this chapter is to propose a more effective ALNS heuristic combined with a path relinking (PR) approach, called ALNS-PR, to address the VRPSPD and its variants. Besides established ALNS components with general validity for VRPs, we introduce an innovative removal operator and route selection policies that explicitly take the load characteristics of VRPSPDs into account.

In extensive computational studies, we show that the algorithmic components, especially the hybridization with PR and the newly proposed ALNS operator, positively impact the quality of the obtained solutions and significantly accelerate the convergence rate of the search. Moreover, we demonstrate the competitiveness of our approach on established benchmark instances for VRPSPD variants proposed in the literature. We improve numerous previous best-known solutions (BKS), especially on VRPSPDTL, VRPSPDTW, and VRPDDP instances. Moreover, we introduce a new VRPDDPTW benchmark and report results on these instances. In addition, we analyze the savings achievable by dividing customer demands by comparing the VRPDDPTW results to VRPSPDTW results obtained on the new instances. To demonstrate the suitability of our solution method for an even larger variety of problems, we perform additional experiments on instances for (i) the VRP with mixed deliveries and pickups (VRPMDP), which represents a special case of the VRPSPD(TL) in which each customer is associated with only one type of demand, and (ii) the VRPRMDP, the special case of the VRPRMDDP. Here, our approach also achieves convincing results and is able to obtain numerous new best solutions.

The remainder of this chapter is structured as follows: Our ALNS-PR algorithm is explained in detail in Section 6.2. We describe the numerical studies to evaluate the performance of the proposed approach in Section 6.3. The chapter is summarized and concluded in Section 6.4.

6.2. Adaptive Large Neighborhood Search with Path Relinking for VRPSPDs

In this section, we describe our solution method that combines an ALNS algorithm with a PR approach. A pseudocode overview of the ALNS-PR is given in Figure 6.1.

$\mathcal{S} \leftarrow generateInitialSolution()$
Initialize best solution $\mathcal{S}^* \leftarrow \mathcal{S}$
Initialize set of elite solutions $\mathcal{E} \leftarrow \varnothing$
while number of iterations without improvement not reached **do**
  if $\mathcal{S}$ already visited or set of elite solutions $\mathcal{E}$ not completely filled **then**
    {*Perform single ALNS iteration and local search on subset $\mathcal{K}$ of routes drawn from $\mathcal{S}$*}
    $\mathcal{S}' \leftarrow performALNS(\mathcal{S}, \mathcal{K})$
    $\mathcal{S}' \leftarrow performLocalSearch(\mathcal{S}', \mathcal{K})$
  else
    $\mathcal{S}' \leftarrow performPathRelinking(\mathcal{S}, \mathcal{E})$
    $\mathcal{S}' \leftarrow performLocalSearch(\mathcal{S}')$
  end if
  $\mathcal{E} \leftarrow evaluateInclusion(\mathcal{S}')$
  if $accept(\mathcal{S}', \mathcal{S})$ **then**
    $\mathcal{S} \leftarrow \mathcal{S}'$
    if $\mathcal{S}'$ improves on $\mathcal{S}^*$ **then**
      $\mathcal{S}^* \leftarrow \mathcal{S}'$
    end if
  end if
  $updatePenaltyFactors(\mathcal{S})$
end while

Figure 6.1.: Pseudocode of the ALNS-PR algorithm for solving VRPSPDs.

The initial solution $\mathcal{S}$ is generated by means of the savings algorithm introduced by Clarke and Wright (1964) (Section 6.2.2) and improved by a local search procedure (Section 6.2.5).

The following main phase of our algorithm is then repeated until a maximum number of iterations ω without improvement of the current best solution $\mathcal{S}^*$ is reached. In each iteration, we decide whether to apply the ALNS or the PR component to the current solution $\mathcal{S}$. Our PR component relies on a dynamic set $\mathcal{E}$ of elite solutions that are recombined with new solutions found during the search. If the current solution $\mathcal{S}$ has already been visited before, or the elite set is not completely filled, i.e., $|\mathcal{E}|$ is smaller than the maximum size λ of the elite set, a single iteration of our ALNS component is performed. We apply ALNS and the following local search step only to a subset $\mathcal{K}$ of routes drawn from the current solution $\mathcal{S}$ that are closely located to each other (Section 6.2.3), resulting in solution $\mathcal{S}'$. We observed a beneficial impact on solution quality and run-time compared to the application of both components to the entire solution.

Otherwise, if $\mathcal{S}$ represents a new solution and the elite set $\mathcal{E}$ is filled, PR is applied between $\mathcal{S}$ and the solutions contained in $\mathcal{E}$ (Section 6.2.4). Subsequently, we perform local search on the best solution $\mathcal{S}'$ returned by the PR routine.

For the resulting solution $\mathcal{S}'$, we evaluate if it should be included in the set of elite solutions $\mathcal{E}$. As long as $\mathcal{E}$ is not completely filled, any feasible solution is added. Otherwise, $\mathcal{S}'$ is included and replaces the worst solution contained in $\mathcal{E}$ if it is feasible, and

- improves on the best solution in $\mathcal{E}$, or
- an SA-based comparison with the worst solution in $\mathcal{E}$ similar to the mechanism described in Section 6.2.6 turns out in favor of the candidate solution $\mathcal{S}'$, and the inclusion of $\mathcal{S}'$ does not decrease the average diversity among all elite solutions. We measure the diversity between solutions in terms of the number of arcs exclusively contained in one of both solutions (see Section 6.2.4).

This procedure aims at balancing quality and diversity of the elite solutions.

Next, the SA acceptance mechanism decides if $\mathcal{S}'$ replaces $\mathcal{S}$ as the current solution for the subsequent iteration (Section 6.2.6). After μ iterations without improvement, we reset $\mathcal{S}$ to $\mathcal{S}^*$. For the purpose of diversification, we set the SA temperature back to its initial value and $\mathcal{S}$ to a solution randomly chosen from the elite set after ϵ solution resets. The probability of an elite solution to be selected is proportional to the diversity between this solution and $\mathcal{S}^*$.

Infeasible solutions, i.e., solutions not respecting all constraints are allowed and handled by means of a dynamic penalty mechanism. More precisely, we transform constraint violations into penalty costs by multiplying the respective value with a dedicated penalty factor for each constraint. The penalty factors of solution $\mathcal{S}$ are dynamically updated during the search depending on the number of consecutive ALNS-PR iterations during which the respective constraint has been satisfied or violated (Section 6.2.1). Moreover, if the ALNS component has been applied in the current iteration, we update the selection probabilities of its components according to the adaptive mechanism described in Section 6.2.3.

Handling time windows and divisible demands Heuristic optimization for the classical VRPTW is usually conducted under the assumption of a hierarchical objective function with the main goal of minimizing the number of employed vehicles (Bräysy and Gendreau, 2005a). This is also true for the investigated problem variants VRPSPDTW and VRPDDPTW. If customer time windows are present, we enable a vehicle minimization phase if the current solution $\mathcal{S}$ is feasible and the number of unsuccessful vehicle reduction attempts is smaller than a nonnegative integer ι^{rem}. Then, our ALNS tries

to remove a route from the current solution, and in our PR component, we ignore elite solutions that utilize a higher number of vehicles than $\mathcal{S}$ for relinking. On the other hand, if $\mathcal{S}$ has been infeasible for ι^{add} iterations, we add an empty route to $\mathcal{S}$. In the following description of our solution method, we explicitly point out the additional modifications that become necessary when time windows need to be taken into account.

Finally, to consider the possibility of satisfying a customer's pickup and delivery requests via two separate visits as in the VRPDDP(TW) and the VRPRMDDP, we extend the previously described algorithm by an additional phase. Any VRPDDP can be transformed into a VRPMDP with twice as many customers where each (partial) customer either requires pickup or delivery service but not both. The same is true for the VRPRMDDP which can be transformed into a VRPRMDP of double size. To this end, each customer of an original VRPDDP(TW) or VRPRMDDP instance is replaced by (i) a duplicate of that customer having a delivery demand equal to the delivery demand of the original customer and a pickup demand equal to zero, and (ii) a duplicate having a delivery demand equal to zero and a pickup demand equal to the pickup demand of the original customer. Naturally, doubling the instance size significantly increases the computational effort required by any solution method. Therefore, we refrain from duplicating each customer at the beginning of the search. Instead, we first apply the main phase of our algorithm as previously described under the assumption of indivisible customer demands. After ω iterations without improvement of the current best VRPSPD(TW) solution $\mathcal{S}^*$, we duplicate each customer in $\mathcal{S}^*$ and in each current elite solution. We subsequently repeat the main phase of our algorithm using half of the initial value of ω as stopping condition with the goal of obtaining an improved VRPDDP(TW) or VRPRMDDP solution.

6.2.1. Solution Evaluation and Penalty Determination

In order to increase the flexibility in exploring the solution space, our ALNS-PR temporarily tolerates constraint violations by imposing dynamic penalty costs on infeasible solutions. The objective function value of a solution $\mathcal{S}$ is then determined using a generalized cost function $f_{gen}(\mathcal{S})$ which includes penalty costs for violating the capacity and time window constraints:

$$f_{gen}(\mathcal{S}) = f_{dist}(\mathcal{S}) + \delta^C \upsilon^C(\mathcal{S}) + \delta^{TW} \upsilon^{TW}(\mathcal{S}).$$

Here, $f_{dist}(\mathcal{S})$ denotes the total traveled distance of solution $\mathcal{S}$. The penalty factor for capacity (time window) violation is denoted as δ^C (δ^{TW}), and the current capacity (time window) violation in solution $\mathcal{S}$ is given by $\upsilon^C(\mathcal{S})$ ($\upsilon^{TW}(\mathcal{S})$).

All penalty factors are initially set to δ^0 and dynamically varied within the interval $[\delta^{min}, \delta^{max}]$. After η^+ consecutive ALNS-PR iterations during which $\mathcal{S}$ has been infeasible with respect to a certain constraint, the associated penalty factor is increased by factor δ^{update}. Analogously, after η^- iterations without violating a certain constraint, the respective penalty factor is divided by the factor δ^{update}.

In contrast to the classical CVRP, the vehicle load in the VRPSPD and its extensions does not monotonically decrease or increase along a route but rather fluctuates. Therefore, for a solution to the VRPSPD, feasibility with respect to the vehicle capacity is given if for each vertex i of each route, the total demand picked up until vertex i plus the total demand still to be delivered at vertex i does not exceed the vehicle capacity. Thus, even for intra-route moves, capacity evaluations cannot be performed in constant time without introducing additional data structures. We efficiently evaluate

capacity violations by implementing similar data structures as described in Zachariadis, Tarantilis, and Kiranoudis (2010), which rely on storing forward and backward demand and load quantities for each vertex in a route. Given a route r and let $\mathcal{V}_r$ denote the set of vertices visited by r, Q the vehicle capacity, and l_i the vehicle load at vertex i, we define the capacity violation of route r as the maximum violation of the vehicle capacity encountered along all vertices of r:

$$\upsilon^C(r) = \max_{i \in \mathcal{V}_r}(\max(l_i - Q, 0)).$$

Time window violations are calculated based on the principle of time travel as described in Nagata, Bräysy, and Dullaert (2010) and Schneider, Sand, and Stenger (2013). After considering a time window violation only once at the vertex of occurrence, it is assumed that the vehicle is allowed to perform service at the latest feasible moment, i.e., at the end of the time window of the regarded customer. This procedure ensures that violations are not accumulated along actually feasible succeeding vertex sequences. Using this approach, the computation of changes in time window violation for conventional inter-route moves can be conducted in $\mathcal{O}(1)$.

As stated in the previous section, in the case of VRPSPD variants with customer time windows, we are faced with a hierarchical objective function with the main goal of minimizing the number of employed vehicles. Then, f_{gen} becomes the secondary objective function that needs to be evaluated if two solutions utilize the same number of vehicles.

6.2.2. Initialization with Savings Algorithm

In order to quickly generate an initial solution, we implement an adaption of the savings algorithm introduced by Clarke and Wright (1964) consisting of the following steps:

1. Generate back-and-forth tours for all customers.
2. Calculate the cost saving for each pair of customers resulting from merging the associated routes as $s = c_{0i} + c_{0j} - c_{ij}$, where c_{ij} corresponds to the distance between two vertices i and j, and sort the savings in decreasing order.
3. While there are positive cost savings, merge the associated routes if this does not result in any constraint violation. Note that, if both routes either start or end with the customers corresponding to the current saving value, we need to reverse the order of customer visits in one route before the merging can be performed. However, this is usually undesirable in settings where tight time windows need to be considered. In addition, due to the resulting shift of pickup and delivery demands, a reversal of the route direction might render an initially feasible route infeasible with respect to the capacity constraint. Thus, if the route reversal would lead to any constraint violation, the merging is not performed.

After the merging procedure, the resulting number of routes may exceed the number of available vehicles. In this case, we identify the route in which the maximum load encountered along all customer visits is minimum. This route is then dissolved, and each customer inserted at the cheapest position in the remaining routes. Capacity and time window violations are handled by imposing penalty costs according to the generalized cost function (see Section 6.2.1). We repeat the process of dissolving routes until the number of routes complies with the number of available vehicles.

The obtained solution is subsequently improved by a local search step (see Section 6.2.5).

6.2.3. The Adaptive Large Neighborhood Search Component

The LNS paradigm, originally introduced by Shaw (1998), consists in iteratively destroying and subsequently repairing solutions by removing and reinserting relatively large numbers of customers. ALNS extends this approach by deploying several competing removal and insertion operators which are chosen at each iteration depending on their previous search performance. To this end, each operator is assigned a selection probability which is dynamically updated during the search (Ropke and Pisinger, 2006b). ALNS has been successfully applied to several VRP variants (see, e.g., Ropke and Pisinger, 2006b; Ropke and Pisinger, 2006a; Pisinger and Ropke, 2007; Hemmelmayr, Cordeau, and Crainic, 2012).

In our solution approach, the ALNS component mainly serves the purpose of diversifying the search. Figure 6.2 shows a single iteration of our ALNS implementation in pseudocode.

$\mathcal{K} \leftarrow getNearestRoutes(\mathcal{S}, \boldsymbol{\pi}^d)$
if vehicle minimization enabled **then**
  $\mathcal{S}' \leftarrow removeRoute(\mathcal{S}, \mathcal{K})$
else
  $n^- \leftarrow$ drawNumberOfCustomersToRemove($\boldsymbol{\pi}^{||}$, $\mathcal{K}$)
  {*Apply randomly selected removal operator based on probabilities* $\boldsymbol{\pi}^-$}
  $\mathcal{S}' \leftarrow removeCustomers(\mathcal{S}, \mathcal{K}, n^-, \boldsymbol{\pi}^-)$
end if
{*Apply randomly selected insertion operator based on probabilities* $\boldsymbol{\pi}^+$}
$S' \leftarrow insertCustomers(\mathcal{S}', \mathcal{K}, \boldsymbol{\pi}^+)$

Figure 6.2.: Pseudocode of our ALNS component.

First, we determine the subset of routes $\mathcal{K}$ from which customers are removed and subsequently reinserted. To this end, a seed route is randomly selected. The remaining routes to be contained in $\mathcal{K}$ are identified by repeating the following procedure: For each route not yet in $\mathcal{K}$, we calculate the average distance of its customers to all customers already contained in $\mathcal{K}$. If this value is smaller than $\varsigma^d \max_{i,j \in \mathcal{V}}(c_{ij})$, where $\varsigma^d \in (0, 1]$ is called distance threshold factor, and $\max_{i,j \in \mathcal{V}}(c_{ij})$ corresponds to the maximum distance between any two vertices in the problem instance, the respective route is added to the subset. To increase the flexibility of the subset generation, at each ALNS iteration, ς^d is randomly selected from list $\Psi^d = (\varsigma^d_{min}, \varsigma^d_{min} + 0.05, ..., \varsigma^d_{max} - 0.05, \varsigma^d_{max})$ with the minimum and maximum values ς^d_{min} and ς^d_{max} being multiples of 0.05. Each value in Ψ^d is assigned a probability from vector $\boldsymbol{\pi}^d$, which is dynamically updated during the search according to the mechanism described in Section 6.2.3.3.

If the vehicle minimization phase is currently enabled, we use the route removal operator (Section 6.2.3.1) to remove a route contained in $\mathcal{K}$ from the current solution. Otherwise, we first select the number of customers that are removed from $\mathcal{K}$. In most LNS implementations, this is done by randomly selecting the percentage of customers to remove from a relatively large interval $\Psi^{||} = [\varsigma^{||}_{min}, \varsigma^{||}_{max}]$. However, to account for the observation that ideal removal percentages are highly instance-dependent, we split $\Psi^{||}$ into five sub-intervals (Goeke and Schneider, 2015). At each ALNS iteration, we select a sub-interval based on dynamic probabilities $\boldsymbol{\pi}^{||}$. The number n^- of customers to remove from $\mathcal{K}$ is then randomly determined within the selected interval. The customer removal is subsequently performed by means of a removal operator that is selected based on probabilities $\boldsymbol{\pi}^-$.

Finally, we select an insertion operator (Section 6.2.3.1) according to probabilities $\boldsymbol{\pi}^+$ and successively reinsert the previously removed customers into the routes of $\mathcal{K}$.

6.2.3.1. Removal and Insertion Operators

To remove customers from the previously determined subset of routes $\mathcal{K}$, we use the following operators:

Random removal randomly removes customers from $\mathcal{K}$ until n^- customers are removed.

Cluster removal was first introduced by Ropke and Pisinger (2006b) and aims at removing customers that are located close to each other. First, we select a route and the first customer to be removed from this route at random. Subsequently, the following steps are repeated until n^- customers are removed from the current solution: We randomly choose a customer among the already removed customers and identify the route r which yields the smallest average distance of its customers to the selected customer. Next, we apply Kruskal's algorithm (Kruskal, 1956) for solving minimum-spanning-tree (MST) problems to the sub-graph composed of the customers of route r. Let n_r denote the number of customers served by route r. We abort the execution of the MST algorithm as soon as the number of generated edges is equal to $n_r - 2$, i.e., two sub-trees remain from which one is randomly chosen for removal. If the size of the selected cluster exceeds the number of remaining customers to be removed, we randomly remove customers from the cluster until n^- customers are removed.

Relatedness removal follows the idea of removing customers that are considered similar and thus likely to be interchangeable (Shaw, 1997). The relatedness of two customers i and j is measured in terms of the distance c_{ij} between them, the difference in their delivery as well as pickup demands $|d_i^d - d_j^d|$ and $|d_i^p - d_j^p|$, respectively, and the difference between the earliest start times of their time windows $|e_i - e_j|$. Each partial relatedness measure is weighted with a parameter χ and normalized using the respective extreme values across the set of all customers $\mathcal{C}$ given by the problem instance. The relatedness measure $R_{i,j}$ of two customers i and j is thus calculated as follows:

$$R_{i,j} = \chi^c \frac{c_{ij}}{\max\limits_{i,j \in \mathcal{C}}(c_{ij})} + \chi^d \frac{|d_i^d - d_j^d|}{\max\limits_{i \in \mathcal{C}}(d_i^d) - \min\limits_{i \in \mathcal{C}}(d_i^d)} + \chi^p \frac{|d_i^p - d_j^p|}{\max\limits_{i \in \mathcal{C}}(d_i^p) - \min\limits_{i \in \mathcal{C}}(d_i^p)} + \chi^e \frac{|e_i - e_j|}{\max\limits_{i \in \mathcal{C}}(e_i) - \min\limits_{i \in \mathcal{C}}(e_i)}.$$

Initially, we select a route and the first customer to be removed from this route at random. Then, we first randomly select a customer from the already removed customers. Next, all remaining customers in $\mathcal{K}$ are stored in a list of size L in ascending order of their relatedness value with respect to the selected customer. From this list, we draw the customer at position $\lfloor L\zeta^{\chi^{rel}} \rfloor$, where ζ is a random number $\in [0, 1)$ and χ^{rel} a parameter $\geqslant 1$, which allows to balance the influence of randomness and relatedness on the selection. We repeat this procedure until n^- customers are removed.

Worst removal as originally introduced by Ropke and Pisinger (2006b) aims at removing customers which appear to be unfavorably positioned in the current solution with respect to the additional routing cost caused by serving them. All customers contained in the current subset $\mathcal{K}$ are stored in a list of size L and sorted in descending order of the cost reduction resulting when removing them from the current solution. At each iteration, we choose the customer at position $\lfloor L\zeta^{\chi^{worst}} \rfloor$. Again, ζ is a random number $\in [0, 1)$, and $\chi^{worst} \geqslant 1$ allows to control the randomness of the selection.

Neighbor graph removal, proposed in Ropke and Pisinger (2006a), is based on exploiting information about promising orders of customer visits gathered in the course of the search. More precisely, a complete directed and weighted auxiliary graph, called the neighbor graph, is intro-

duced, whose vertices correspond to the customers contained in the current subset of routes $\mathcal{K}$. Each arc (i, j) is weighted with the objective function value of the best solution found so far in which customer j is visited directly after customer i. Initially, the weight of each arc is set to positive infinity and dynamically updated during the search.

All customers in $\mathcal{K}$ are stored in a list of size L and sorted in descending order of a score which is based on the current visiting orders in the routes of $\mathcal{K}$. More precisely, the score for each customer i is calculated by summing up the the weights of the arcs (i^-, i) and (i, i^+) in the neighbor graph, where i^- and i^+ correspond to the predecessor and successor of i, respectively. At each iteration, the operator removes the customer at position $\lfloor L\zeta^{\chi^{nb}} \rfloor$ with random number $\zeta \in [0, 1)$ and $\chi^{nb} \geqslant 1$. After the removal of customer i, the neighbor scores of customers i^- and i^+ are updated accordingly.

Route removal removes all customers from the route in $\mathcal{K}$ with the smallest maximum load encountered along all customers. As stated before, we exclusively apply this removal operator if the vehicle minimization phase is currently enabled.

Load balance removal aims at specifically taking the load characteristics of the VRPSPD and its extensions into account. Intuitively, customers with high pickup demands should be visited late while customers with high delivery demands should be served early in a route. We try to identify and rearrange customer visits which seem to be unfavorably positioned with respect to the associated demand quantities and the capacity utilization in the corresponding route. To this end, we first select a route r from $\mathcal{K}$ at random. Next, we calculate a position score $\varnothing_i^{pos}$ for each customer i in r:

$$\varnothing_i^{pos} = p_i \frac{d_i^p}{\sum_{j \in \mathcal{V}_r} d_j^p} + (|\mathcal{V}_r| - 1 - p_i) \frac{d_i^d}{\sum_{j \in \mathcal{V}_r} d_j^d},$$

where p_i corresponds to the current insertion position of customer i. All customers contained in route r are stored in a list of size L in ascending order of their position score. We subsequently draw the customer from position $\lfloor L\zeta^{\chi^{load}} \rfloor$. Analogous to the previous operators, ζ corresponds to a random number $\in [0, 1)$, and $\chi^{load} \geqslant 1$ controls the influence of the position score on the customer selection.

To reinsert the previously removed customers into the routes of subset $\mathcal{K}$, we use one of the following insertion operators:

Greedy insertion is implemented in two variants:

1. At each iteration, the best insertion position is determined for each of the remaining customers. The customer associated with the smallest cost increase is inserted accordingly.
2. The procedure is similar to the first variant except that we do not allow the reinsertion of a customer into the route it had been removed from in the previous removal step. A comparable approach called greedy insertion forbidden is used by Hemmelmayr, Cordeau, and Crainic (2012).

GRASP insertion is inspired by the metaheuristic introduced in Feo and Resende (1989). The best insertion is determined for each remaining customer and stored in a list of size L which is sorted in ascending order of cost increase. At each iteration, the next customer to be inserted is randomly selected among the best $\lfloor \chi^{GRASP} L \rfloor$ remaining insertions. Here, χ^{GRASP} corresponds to a number $\in (0, 1)$.

Regret insertion was also proposed by Ropke and Pisinger (2006b) and tries to overcome the myopic

behavior of greedy insertion by implementing a look-ahead strategy. More specifically, a regret-k value is calculated for each customer as the difference between the cost increase resulting from the cheapest insertion of this customer into the k-best route and its optimal insertion into the best route from $\mathcal{K}$. At each iteration, we perform the best insertion of the customer associated with the largest regret value. We implement regret insertion for $k = 2, 3, 4$.

Random insertion aims at solution diversification by inserting each customer at a random position of a randomly selected route from $\mathcal{K}$.

6.2.3.2. Route Selection and Evaluation of Removal and Insertion

While some of the previously described removal operators rely on removing customers without considering the properties of the routes they belong to, we additionally use operators which aim at removing customers from routes with certain characteristics. To this end, we use various route selection policies which are combined with the corresponding removal operators (see Schneider, Stenger, and Hof, 2015, for a similar approach).

Besides a purely random selection, we implement the following policies that determine routes according to a roulette wheel selection mechanism based on specific criteria:

Cost selects a route with a probability proportional to the cost of the route including penalty costs.

Distance is based on the length of a route. The selection probability of a route increases with the associated traveled distance.

Efficiency aims at identifying inefficient routes, i.e., routes that are characterized by large detours to cover a relatively small quantity of customer demands. The probability of a route to be selected is proportional to the ratio of the associated traveled distance and the average capacity utilization in this route.

High average utilization selects a route with a probability proportional to the average capacity utilization in the route.

Low average utilization determines a selection probability for each route that is inversely proportional to the respective average capacity utilization.

High maximum load performs the route selection according to a probability that increases with the maximum load encountered along all customers served in a route.

Low maximum load selects each route based on a probability inversely proportional to the maximum load encountered along all customers of the route.

The last four selection policies focus on the utilization of the vehicle capacity or the fluctuating nature of the vehicle load along routes arising in the investigated problem variants. Obviously, both variants of the average utilization and maximum load policies pursue opposing strategies. However, the adaptive mechanism of our ALNS component is able to successfully identify the appropriate strategies for the problem instance at hand, see Section 6.3.3.

With respect to compatibility, random removal, cluster removal, and relatedness removal are able to make use of each selection policy. The load balance removal operator exclusively employs the high average utilization policy. Once a compatible removal operator has been selected, we subsequently select a policy based on probabilities $\boldsymbol{\pi}^r$. The selected route selection policy is used by the current removal operator for the entire iteration.

Moreover, to guide the search towards yet unexplored areas, we implement two additional variants of worst removal, greedy insertion, GRASP insertion, and regret insertion that differ in how the corresponding operation is evaluated. Besides the basic operator variants that assess the associated operation based on the change in routing cost according to the generalized cost function (Δf_{gen}), we implement variants that employ the evaluation measures:

1. diversification Δf_{div}, which additionally considers a diversification penalty based on historic arc occurrence frequencies:

$$\Delta f_{div}(o, \mathcal{S}) = \Delta f_{gen}(o, \mathcal{S}) + \kappa \frac{f_{gen}(\mathcal{S})}{n} \sqrt{\sum_{(i,j) \in \mathcal{A}_o^+} h(i,j)},$$

   where $\Delta f_{gen}(o, \mathcal{S})$ corresponds to the change of the objective function value of solution $\mathcal{S}$ caused by removal or insertion operation o, κ to a real-valued parameter to control the amount of diversification, $f_{gen}(\mathcal{S})$ to the current objective function value of solution $\mathcal{S}$, n to the number of customers given by the problem instance, $h(i,j)$ represents the occurrence frequency of arc (i,j) in previously generated solutions, and $\mathcal{A}_o^+$ the set of arcs generated by operation o, and
2. noise Δf_{noise}, which multiplies the change in routing cost caused by operation o by a random number ζ drawn from the interval $[\zeta_{min}, \zeta_{max}]$ (see Ropke and Pisinger, 2006b; Hemmelmayr, Cordeau, and Crainic, 2012, for a similar approach):

$$\Delta f_{noise}(o, \mathcal{S}) = \Delta f_{gen}(o, \mathcal{S}) \zeta.$$

6.2.3.3. Adaptive Mechanism

At each ALNS iteration, the choice of distance threshold factor, removal interval, removal operator, route selection policy (if applicable), and insertion operator is performed according to a roulette wheel selection procedure as proposed in Ropke and Pisinger (2006b) based on the probability vectors $\boldsymbol{\pi}^d$, $\boldsymbol{\pi}^{||}$, $\boldsymbol{\pi}^-$, $\boldsymbol{\pi}^r$, and $\boldsymbol{\pi}^+$, respectively. More precisely, given a set of adaptive components denoted as $\mathcal{X} \in \{d, ||, -, r, +\}$, the selection probability of component $i \in \mathcal{X}$ is calculated as $\pi_i = w_i / \sum_{j \in \mathcal{X}} w_j$, where w_i corresponds to the weight of component i. All components of a set $\mathcal{X}$ are initially assigned the same weight and dynamically updated during the search depending on their performance. The performance of an adaptive component is measured in terms of a scoring system. A score of σ^{best} is added to the current score of a component whenever a new overall best solution is found, a score of σ^{imp} if the new solution $\mathcal{S}'$ improves on the current one $\mathcal{S}$ and has never been encountered before, and a score of σ^{acc} if $\mathcal{S}'$ is worse than $\mathcal{S}$ but accepted according to the SA acceptance mechanism and has never been encountered before in the search.

We maintain a solution memory to keep track of already obtained solutions and allow our algorithm to fill this memory for 100 ALNS-PR iterations before any weight update takes place. The weight of each component is then updated every γ ALNS iterations based on its performance during this period. If $\varnothing_i$ denotes the current score of component i and β_i the number of applications of the component since the last weight update, then the new weight is determined as $w_i = w_i(1 - \alpha) + \alpha \varnothing_i / \beta_i$. The factor $\alpha \in [0, 1]$ allows to control the reaction speed of the weight adjustment to the performance of the component. The values of $\varnothing_i$ and β_i are reset to zero after each update.

6.2.4. The Path Relinking Component

PR represents an intensification strategy which was originally proposed by Glover (1997) and aims at discovering promising solutions on the trajectories between elite solutions obtained during the search. Based on the assumption that good solutions are likely to share common characteristics, PR consists in creating a path of solutions between an initial and a guiding solution in the hope of obtaining improving solutions in the process. The initial solution is transformed into the guiding solution by successively incorporating characteristics of the guiding solution, thus stepwise decreasing the diversity between them.

The PR component complements our ALNS, which primarily focuses on diversifying the search. By connecting promising solutions located in distant regions of the search space, we aim at discovering improving solutions that are not reachable via local search. Ho and Gendreau (2006), Fallahi, Prins, and Calvo (2008), and Nguyen, Prins, and Prodhon (2012), for instance, show how to successfully hybridize well-known metaheuristics for routing problems with PR approaches. Figure 6.3 shows our PR implementation in pseudocode.

$\mathcal{S}' \leftarrow \varnothing$
for each solution $\mathcal{S}_e \in \mathcal{E}$ **do**
  {*Initialize current and guiding solution*}
  $\mathcal{S}_c \leftarrow \mathcal{S}$
  $\mathcal{S}_g \leftarrow \mathcal{S}_e$
  {*Determine arcs exclusively contained in guiding solution and initialize path length*}
  $\mathcal{A}^{\neq} \leftarrow$ getDifferingArcs($\mathcal{S}$, $\mathcal{S}_g$)
  $p \leftarrow |\mathcal{A}^{\neq}|$
  while $|\mathcal{A}^{\neq}| \geqslant \lfloor(1-\rho)\cdot p\rfloor$ **do**
    $m_{best} \leftarrow \varnothing$
    for each arc $(i,j) \in \mathcal{A}^{\neq}$ **do**
      {*Determine move which creates arc yielding the least cost change*}
      $m \leftarrow$ getBestMoveToCreateArc(i,j)
      if $m_{best} = \varnothing \vee m$ improves on m_{best} **then**
        $m_{best} \leftarrow m$
      end if
    end for
    $\mathcal{S}_c \leftarrow$ performMove($\mathcal{S}_c$, m_{best})
    if $\mathcal{S}_c \neq \mathcal{S} \wedge \mathcal{S}_c \neq \mathcal{S}_g \wedge (\mathcal{S}' = \varnothing \vee \mathcal{S}_c$ improves on $\mathcal{S}')$ **then**
      $\mathcal{S}' \leftarrow \mathcal{S}_c$
    end if
    {*Remove arcs created by previously performed move from set of differing arcs*}
    $\mathcal{A}^{\neq} \leftarrow \mathcal{A}^{\neq} \backslash \mathcal{A}^{+}_{m_{best}}$
  end while
end for

Figure 6.3.: Pseudocode of our PR component.

Whenever a solution $\mathcal{S}$ has been encountered for the first time in the search, it is passed to our PR algorithm. For each solution $\mathcal{S}_e$ contained in the elite set $\mathcal{E}$, we set the current solution on the solution path $\mathcal{S}_c$ to the initial solution $\mathcal{S}$ and the guiding solution $\mathcal{S}_g$ to $\mathcal{S}_e$. In general, we perform PR between $\mathcal{S}$ and all elite solutions in $\mathcal{E}$. However, if we are currently trying to reduce the number of vehicles, only solutions in the elite set with a number of routes less than or equal to the number of routes in $\mathcal{S}$ are considered for relinking. Otherwise, if $\mathcal{S}_g$ utilizes a higher number of vehicles, we need to equalize both vehicle numbers by adding empty routes to $\mathcal{S}$ accordingly.

We explain the following steps of our PR implementation using Figure 6.4, which illustrates the

transformation of an initial solution into a guiding solution for an example instance composed of eleven customers. Both solutions consist of five routes each starting and ending with the depot vertex 0. Initially, we determine those arcs that are contained in $\mathcal{S}_g$ but not in the initial solution S and store them in set $\mathcal{A}^{\neq}$. The goal of our PR procedure is now to iteratively incorporate the arcs in $\mathcal{A}^{\neq}$ into the current solution. In our example, there are ten such differing arcs which are exclusively contained in $\mathcal{S}_g$ (marked as ✗).

Iteration	Route	$\mathcal{S}_c$	$\mathcal{S}_g$	Arcs created in $\mathcal{S}_c$
0	A	0 - 3 - 11 - 0	0 - 3 - 11 - 0	
	B	0 - 5 - 0	0 - 5 - 0	
	C	0 - 2 - 10 - 0	0 ✗ 7 ✗ 10 ✗ 4 ✗ 0	
	D	0 - 1 - 9 - 8 - 0	0 - 1 ✗ 8 ✗ 2 ✗ 0	
	E	0 - 4 - 7 - 6 - 0	0 ✗ 6 ✗ 9 ✗ 0	
1	A	0 - 3 - 11 - 0	0 - 3 - 11 - 0	
	B	0 - 5 - 0	0 - 5 - 0	
	C	0 - 2 - 7 - 10 - 0	0 ✗ 7 - 10 ✗ 4 ✗ 0	$(7, 10)$
	D	0 - 1 - 9 - 8 - 0	0 - 1 ✗ 8 ✗ 2 ✗ 0	
	E	0 - 4 - 6 - 0	0 ✗ 6 ✗ 9 ✗ 0	
2	A	0 - 3 - 11 - 0	0 - 3 - 11 - 0	
	B	0 - 5 - 0	0 - 5 - 0	
	C	0 - 2 - 7 - 10 - 0	0 ✗ 7 - 10 ✗ 4 ✗ 0	
	D	0 - 1 - 8 - 0	0 - 1 - 8 ✗ 2 ✗ 0	$(1, 8)$
	E	0 - 4 - 6 - 9 - 0	0 ✗ 6 - 9 - 0	$(6, 9), (9, 0)$
3	A	0 - 3 - 11 - 0	0 - 3 - 11 - 0	
	B	0 - 5 - 0	0 - 5 - 0	
	C	0 - 7 - 10 - 0	0 - 7 - 10 ✗ 4 ✗ 0	$(0, 7)$
	D	0 - 1 - 8 - 2 - 0	0 - 1 - 8 - 2 - 0	$(8, 2), (2, 0)$
	E	0 - 4 - 6 - 9 - 0	0 ✗ 6 - 9 - 0	
4	A	0 - 3 - 11 - 0	0 - 3 - 11 - 0	
	B	0 - 5 - 0	0 - 5 - 0	
	C	0 - 7 - 10 - 4 - 0	0 - 7 - 10 - 4 - 0	$(10, 4), (4, 0)$
	D	0 - 1 - 8 - 2 - 0	0 - 1 - 8 - 2 - 0	
	E	0 - 6 - 9 - 0	0 - 6 - 9 - 0	$(0, 6)$

Figure 6.4.: Example application of our PR procedure.

During the execution of PR, we avoid destroying matching arcs, i.e., arcs contained in both solutions. To this end, we connect the vertices incident to matching arcs to fixed vertex sequences that are treated as atomic units and not allowed to be split. Initially, there are two fixed sequences for each customer i, $\Gamma_i^- = (y, ..., i)$ and $\Gamma_i^+ = (i, ..., z)$, corresponding to the fixed vertex sequence ending with customer i and the fixed vertex sequence starting with customer i, respectively. Both sequences associated with a customer only contain this customer at the beginning. Consequently, for customer 3 that is served by route A in Figure 6.4, $\Gamma_3^- = \Gamma_3^+ = (3)$ initially holds.

Before applying the main phase of our PR approach, we fix all arcs that are already contained in both solutions. An arc (i, j), where i and j correspond to customers, is fixed by merging the vertex sequences $\Gamma_i^- = (y, ..., i)$ and $\Gamma_j^+ = (j, ..., z)$. The result of the merging is stored in Γ_y^+ and Γ_z^-, i.e., $\Gamma_y^+ = \Gamma_z^- = (y, ..., i, j, ..., z)$. We then remove Γ_i^- and Γ_j^+ from the set of fixed vertex sequences. Consider arc $(3, 11)$, that is contained in both example solutions to be fixed first. This requires merging of Γ_3^- and Γ_{11}^+, leading to $\Gamma_3^+ = \Gamma_{11}^- = (3, 11)$. From now on, arc $(3, 11)$ is not allowed to be removed from $\mathcal{S}_c$ anymore. In other words, Γ_3^- and Γ_{11}^+ may not be removed from the fixed sequence they are now part of and are therefore removed from the sequence set. In case the regarded arc starts (ends) with

the depot vertex 0, the corresponding depot visit is inserted at the beginning (end) of $\Gamma_j^+ = (j, ..., z)$ ($\Gamma_i^- = (y, ..., i)$), leading to the modified sequence $\Gamma_z^- = (0, j, ..., z)$ ($\Gamma_y^+ = (y, ..., i, 0)$) and discarding Γ_j^+ (Γ_i^-). For example, fixing arc $(0, 3)$ in Figure 6.4 after arc $(3, 11)$ results in $\Gamma_{11}^- = (0, 3, 11)$ and the removal of Γ_3^+.

Next, we successively incorporate the arcs from $\mathcal{A}^{\neq}$ into the current solution, thus advancing on the solution path between $\mathcal{S}$ and $\mathcal{S}_g$. Because the guiding solution is close to a local optimum, it is unlikely to discover improving solutions in its immediate proximity. Therefore, we refrain from exploring the entire path between both solutions and restrict the search to a fraction of ρ of the solution path. More precisely, we repeat the following steps until the number of remaining differing arcs is smaller than the fraction $(1 - \rho)$ of the initial cardinality of $\mathcal{A}^{\neq}$:

For each arc in $\mathcal{A}^{\neq}$, we determine the best move that inserts this arc into the current solution. Consider arc (i, j) to be created in $\mathcal{S}_c$. We distinguish five cases with respect to the creation of (i, j):

1. If Γ_i^- and Γ_j^+ do not contain the depot, two moves are possible: (i) Sequence Γ_i^- may be moved before Γ_j^+, or (ii) sequence Γ_j^+ can be relocated after sequence Γ_i^-. Assume arc $(7, 10)$ to be created in $\mathcal{S}_c$ in the first iteration of our example application. Then, we may relocate customer 7 from route E to route C, i.e., Γ_7^- before Γ_{10}^+. Alternatively, it is allowed to move customer 10 from route C to route E, i.e., Γ_{10}^+ after Γ_7^-.
2. If one of both sequences starts or ends with the depot, we only allow the relocation of the vertex sequence that is not connected to the depot. Sequence $\Gamma_1^- = (0, 1)$ in Figure 6.4, for instance, may not be relocated to create arc $(1, 8)$.
3. If Γ_i^- starts and Γ_j^+ ends with the depot, and both sequences are part of the same route, arc (i, j) can only be implicitly created by relocating all vertices lying between i and j.
4. If Γ_i^- starts and Γ_j^+ ends with the depot, and both sequences are served in different routes, the *2-opt** operator which is described in Section 6.2.5 is applied to merge the two route segments.
5. Finally, if the arc to be created starts (ends) with the depot, we determine the best relocation for Γ_j^+ (Γ_i^-) among the routes where the start (end) depot visit is not part of a fixed vertex sequence.

The overall best move out of the set of best moves per arc is subsequently applied to the current solution $\mathcal{S}_c$. We set the best solution $\mathcal{S}'$ to $\mathcal{S}_c$ if (i) $\mathcal{S}_c$ does not correspond to the initial solution $\mathcal{S}$ or the guiding solution $\mathcal{S}_g$, and (ii) $\mathcal{S}'$ has not been initialized yet, or $\mathcal{S}_c$ improves on $\mathcal{S}'$.

Moreover, we remove the matching arcs created by the previously performed move from $\mathcal{A}^{\neq}$ and update the set of fixed vertex sequences accordingly. Note that each move might introduce more differing arcs than the currently considered arc into $\mathcal{S}_c$. Take for example iteration 2 in Figure 6.4. Here, we identify the relocation of sequence $\Gamma_9^- = (9)$ from its former position in route D to the last position in route E as the best move to introduce arc $(9, 0)$ into $\mathcal{S}_c$. In doing so, three differing arcs, namely $(1, 8)$, $(6, 9)$, and the regarded arc $(9, 0)$ are created simultaneously and subsequently removed from $\mathcal{A}^{\neq}$. Next, the associated fixed vertex sequences need to be updated. Assume this is done according to the previously mentioned order of arcs. Then, we merge $\Gamma_1^- = (0, 1)$ and $\Gamma_8^+ = (8)$ into $\Gamma_8^- = (0, 1, 8)$, $\Gamma_6^- = (6)$ and $\Gamma_9^+ = (9)$ into $\Gamma_9^- = \Gamma_6^+ = (6, 9)$, and attach the depot to Γ_9^-, leading to $\Gamma_6^+ = (6, 9, 0)$. Finally, Γ_1^-, Γ_8^+, Γ_6^-, Γ_9^+, and Γ_9^- are removed from the sequence set.

For the purpose of illustration, Figure 6.4 shows the creation of the entire solution path, leading to the equalization of both solutions in iteration 4.

6.2.5. Local Search

In the initialization phase and after each application of our ALNS and PR components, a local search procedure is applied.

Our local search implements a best-improvement strategy and is stopped when no further improving move can be identified. We use a composite neighborhood composed of the following operators: A *relocate* operator, originally introduced in Savelsbergh (1992) is implemented which moves a customer from its position to a different position in the same or a different route. Moreover, we use an *exchange* operator (Savelsbergh, 1992) which is able to swap customers between and within routes. The *2-opt* operator replaces two arcs of a single route by two new ones reversing the order of customers between the vertices incident to the removed arcs (Lin, 1965). Finally, we implement a *2-opt** operator as an inter-route modification of the 2-opt operator which removes one arc from each route and reconnects the first part of the first route with the second part of the second route and vice versa (Potvin and Rousseau, 1995). A reversal of the route orientation, which is usually undesirable in problems with time windows, is hereby avoided.

In order to speed up the search, we maintain a set of modified routes, which is initially filled with the routes that have been altered by the preceding algorithmic component. We restrict the evaluation and thus application to those moves that involve at least one route from this set. The set of modified routes is dynamically updated during the execution of the local search, i.e., a previously unchanged route is added to the set after being modified by a local search move.

6.2.6. Acceptance Decision

To decide if the solution $\mathcal{S}'$ returned by the local search should replace $\mathcal{S}$, we use an acceptance mechanism based on SA.

While solutions that improve $\mathcal{S}$ are always accepted, we accept worse solutions with probability

$$e^{\frac{-(f_{gen}(\mathcal{S}')-f_{gen}(\mathcal{S}))}{\vartheta}}.$$

The temperature parameter ϑ is initially set to a value such that a deterioration of the current solution $\mathcal{S}$ by Δ^{SA} is accepted with a probability of 50%. The temperature is decreased after every ALNS-PR iteration such that the acceptance probability of a relative deterioration of Δ^{SA} is equal to 1% after a cooling period of θ iterations.

6.3. Computational Studies

In this section, we present the computational studies conducted to evaluate the performance of our ALNS-PR. The benchmark instances used for testing our algorithm are presented in Section 6.3.1. Section 6.3.2 describes the experimental environment and the setting of the algorithmic parameters. In Section 6.3.3, we analyze the influence of specific components of our algorithm on solution quality and computation time. Finally, in Section 6.3.4, we discuss the competitiveness of our approach on VRPSPD, VRPSPDTL, VRPMDP, VRPSPDTW, VRPDDP, and VRPRMDP benchmark instances from the literature and present first results for VRPRMDDP and VRPDDPTW instances.

6.3.1. Benchmark Instances

In the following, we describe the benchmark instances that we use in our computational studies structured according to problem type.

6.3.1.1. VRPSPD Instances

To assess the performance of ALNS-PR for the VRPSPD, we use the instances proposed by Salhi and Nagy (1999), from now on referred to as set Salhi-VRPSPD, Dethloff (2001), denoted as set Dethloff, and Montané and Galvão (2006), for which the subset of instances with up to 200 customers is called Montané-Medium and the whole set Montané-All.

The set Salhi-VRPSPD is based on a CVRP benchmark introduced in Christofides, Mingozzi, and Toth (1979), that contains 14 instances with 50 to 199 customers which are randomly distributed in ten instances and clustered in the remaining ones. Salhi and Nagy (1999) use seven of the CVRP instances that do not contain a maximum route duration and customer service times to generate seven VRPSPD instances by splitting the original customer demands into delivery and pickup demands based on a ratio of the customer coordinates. This subset of instances is referred to as X set. Another subset of seven instances (Y set) is generated from the X-series of instances by swapping the delivery and pickup demands of every other customer. Unfortunately, these instances are not consistently treated in the literature. Some authors swap the delivery and pickup demands of every customer instead of every other customer or round the distances and demand quantities. We do not round, but we exchange the demands of every customer in set X. Note that the latter assumption results in identical optimal solutions for both variants of each instance where the routes of one solution correspond to the reversed routes of the other. However, to the best of our knowledge, all authors presenting the best-performing heuristics for the VRPSPD have generated these instances according to this assumption. Thus, to be able to compare our results to those of the state-of-the-art heuristics for the VRPSPD, we had to interpret these instances in identical fashion.

Set Dethloff contains 40 VRPSPD instances with 50 customers and three or eight vehicles. Half of the instances, denoted as SCA, are characterized by a uniform distribution of the customer coordinates. In the other 20 instances, denoted as CON, half of the customers are uniformly distributed, and the remaining customers located in clusters.

The benchmark Montané-All contains 18 instances with $100, 200$, and 400 customers that are derived from a respective subset of the VRPTW instances proposed in Solomon (1987) and Gehring and Homberger (1999). The customers are clustered (prefix C), randomly distributed (prefix R), or a mixture of both (prefix RC). The authors create VRPSPD instances by omitting the time window constraints and assigning a discrete pickup demand to each customer that is randomly drawn from the interval used to generate the delivery demands in the original instances.

6.3.1.2. VRPSPDTL Instances

VRPSPDTL instances are proposed by Salhi and Nagy (1999), called Salhi-VRPSPDTL, and Polat et al. (2015), which we from now on refer to as Polat-VRPSPDTL.

Set Salhi-VRPSPDTL is composed of 14 instances which are generated according to the same procedure as described in Section 6.3.1.1 to generate VRPSPD instances using the remaining seven CVRP

instances of Christofides, Mingozzi, and Toth (1979) which contain a maximum route duration and customer service times.

The Polat-VRPSPDTL instances are created by first converting the CVRP instances of Christofides and Eilon (1969) to VRPSPD instances according to the approach described in Nagy et al. (2015) and then adding time limits. The instances do not contain customer service times.

6.3.1.3. VRPMDP Instances

To demonstrate the competitiveness of our approach on an even larger variety of problems, we additionally perform experiments on instances for the VRPMDP that represents a special case of the VRPSPD(TL). The VRPMDP instances have been introduced by Salhi and Nagy (1999) and are from now on referred to as Salhi-VRPMDP.

The set Salhi-VRPMDP is created by generating three VRPMDP instances based on each original instance of the CVRP benchmark of Christofides, Mingozzi, and Toth (1979) by assigning every second (H), fourth (Q), or tenth (T) customer a pickup demand equal to the original delivery demand and a delivery demand equal to zero.

6.3.1.4. VRPSPDTW Instances

Two sets of VRPSPDTW instances are provided by Wang and Chen (2012), which we denote as Wang-Medium, and Wang et al. (2015), from now on referred to as Wang-Large.

The instances of set Wang-Medium are created by adding pickup demands to the 56 VRPTW instances with 100 customers proposed by Solomon (1987). In addition to the customer distribution described in Section 6.3.1.1, each instance is either characterized by narrow time windows and small vehicle capacities (prefix C1/R1/RC1) or large time windows and large vehicle capacities (prefix C2/R2/RC2).

Wang-Large contains 30 large-scale instances with $200, 400, 600, 800$, and 1000 customers which are derived from a subset of the VRPTW instances introduced in Gehring and Homberger (1999) by adding pickup demands.

6.3.1.5. VRPDDP Instances

For the VRPDDP, we use three instance sets of Nagy et al. (2015), which we label as Nagy-VRPDDP1, Nagy-VRPDDP2, and Nagy-VRPDDP3 and two sets of instances proposed by Polat (2017), which we denote as Polat-VRPDDP1 and Polat-VRPDDP2. All VRPDDP instances are based on VRPSPD and VRPSPDTL instances and solved under the assumption of divisible customer demands.

The sets Nagy-VRPDDP1, Nagy-VRPDDP2, and Nagy-VRPDDP3 are based on the 28 instances of Salhi-VRPSPD and Salhi-VRPSPDTL with distances and demands rounded to the nearest integer. The set Nagy-VRPDDP1 corresponds to the unchanged original instances. Taking the original set as a basis, the delivery and pickup demands of each customer are increased by multiplying each value by four and adding ten percent of the vehicle capacity in order to obtain instance set Nagy-VRPDDP2. The instances for benchmark Nagy-VRPDDP3 are created by adding 75% of the vehicle capacity to the delivery demand and 20% of the vehicle capacity to the pickup demand of every odd customer

and adding 20% of the vehicle capacity to the delivery demand and 75% of the vehicle capacity to the pickup demand of every even customer. Again, the resulting demand values are rounded to the nearest integer. Moreover, for the instances with a maximum duration imposed on the vehicle routes, we assume that the service time associated with a customer is incurred twice if it is served via two separate visits. Polat (2017) interprets these instances differently and considers only half of the original service time for each visit of a divided customer.

The instances of set Polat-VRPDDP1 are based on the VRPSPD benchmark Dethloff. Polat-VRP-DDP2 corresponds to the instances of Montané-Medium.

6.3.1.6. VRPRMDDP Instances

For the VRPRMDDP, we use the VRPDDP instances proposed by Hoff et al. (2009), denoted as Hoff.

These instances are based on 35 CVRP instances of VRPLIB (http://www.or.deis.unibo.it/research_pages/ORinstances/VRPLIB/VRPLIB.html) containing seven to 484 vertices and two to 25 vehicles. The authors first create VRPSPD instances by setting the delivery demand of each customer i to the demand d_i given in the original instance and the pickup demand of each even customer to $\lfloor 0.8\, d_i \rfloor$ and of each odd customer to $\lfloor 1.2\, d_i \rfloor$. The demands of each customer are considered to be divisible and the original vehicle capacities remain unchanged.

To obtain VRPRMDDP instances, we solve these VRPSPD instances also under the assumption of divisible customer demands and subject to varying percentages of unavailable vehicle capacity in case both pickup and delivery demands are part of the vehicle load. The percentages used correspond to those suggested by Hoff et al. (2009).

6.3.1.7. VRPRMDP Instances

We additionally assess the performance of our algorithm on three sets of instances for the VRPRMDP which represents a special case of the VRPRMDDP. These instances have been proposed by Nagy, Wassan, and Salhi (2013) and are from now on referred to as Nagy-VRPRMDP1, Nagy-VRPRMDP2, and Nagy-VRPRMDP3.

The three sets are derived from the instances for the VRP with backhauls introduced by Goetschalckx and Jacobs-Blecha (1989) and Toth and Vigo (1999), and the instances of set Salhi-VRPMDP, respectively, by assuming different percentages of the vehicle capacity that must remain unoccupied when both demand types are simultaneously loaded.

6.3.1.8. VRPDDPTW Instances

Finally, we generate a new set of VRPDDPTW instances, denoted as HS, by modifying the VRP-SPDTW instances contained in Wang-Medium similar to the approach described in Section 6.3.1.5.

More precisely, we add 20% of the vehicle capacity to the delivery demand and 5% of the vehicle capacity to the pickup demand of every odd customer and 5% of the vehicle capacity to the delivery demand and 20% of the vehicle capacity to the pickup demand of every even customer. Note that we do not round and that the resulting demand quantities are limited to the vehicle capacity. Again, the whole service time is incurred for each partial visit of a divided customer.

6.3.2. Computational Environment and Parameter Setting

Our ALNS-PR is implemented as single-thread code in Java. All tests were performed on a Windows 10 Professional desktop computer with an Intel Core i5-6600 processor running at 3.30 GHz and 16 GB RAM. In all experiments, we performed ten runs on each instance.

In the course of the development of our algorithm, we observed that some parameters have a stronger impact on solution quality compared to the remaining parameters. Starting from a reasonably well-performing parameter setting found during the development of our method, we successively refine the setting of each such key parameter (Ropke and Pisinger, 2006b). To this end, we use ten instances randomly drawn from Salhi-VRPSPD and Salhi-VRPSPDTL and evaluate three values for each key parameter. The best value is kept as the final setting for the respective parameter, and we subsequently proceed with tuning the next one. The results of the parameter tuning are presented in Table 6.1.

In particular, we successively tune the following parameters: the factor which controls how fast the adaptive weight adjustment reacts to the performance of the adaptive ALNS components (α), the parameters to control the degree of randomness in specific removal operators (χ^{worst}, χ^{load}, χ^{rel}, and χ^{nb}), the percentage of best insertions considered by the GRASP insertion operator (χ^{GRASP}), the factor to control the amount of diversification in operation evaluation measure Δf_{div} (κ), the interval from which a random number is drawn in the context of evaluation measure Δf_{noise} ($[\zeta_{min}, \zeta_{max}]$), the size of the elite set (λ), the percentage of the solution path which we investigate in each PR execution (ρ), the minimum penalty factor (δ^{min}), the penalty update factor (δ^{update}), the relative solution deterioration (Δ^{SA}) used to determine the initial and minimal SA temperature, and the cooling period (θ) of our SA-inspired acceptance mechanism.

For each parameter, we provide the average of the gaps of the best solutions found to the respective BKS from the literature in percent (Δ^b). The final value for each parameter is marked in bold.

The complete parameter setting is shown in Table 6.2. With respect to the main ALNS-PR routine, we additionally report the maximum number of iterations without improvement (ω), the number of non-improving iterations after which the current solution is reset to the current best (μ), the maximum number of unsuccessful vehicle reduction attempts (ι^{rem}), and the number of iterations during which the feasibility of the current solution could not be restored and whereupon an empty route is added (ι^{add}). To achieve a competitive solution quality and computation times on the different sets of benchmark instances, we set $\omega = 4000$ for all VRPSPD benchmarks and the sets Salhi-VRPSPDTL, Salhi-VRPMDP, Polat-VRPDDP1, and Polat-VRPDDP2. For the remaining benchmarks, we set $\omega = 500$.

For our ALNS component, we additionally provide the boundaries of the removal interval ($\Psi^{||}$) and the distance threshold interval (Ψ^d), the scores to evaluate the performance of the adaptive components (σ^{best}, σ^{imp}, and σ^{acc}), the number of iterations after which the weights of the adaptive components are updated (γ), and the weights used to calculate the relatedness measure between two customers (χ^c, χ^d, χ^p, and χ^e).

We further report the initial (δ^0) and maximum (δ^{max}) penalty factors, and the numbers of ALNS-PR iterations after which the penalty factors are increased (η^+) as well as decreased (η^-).

Finally, we provide the number of solution resets after which the SA temperature is reset to its initial value, and the current solution is set to a solution randomly drawn from the elite set (ϵ).

ALNS			
α	0.1	**0.2**	0.3
Avg. Δ^b(%)	0.02	0.00	0.01
χ^{worst}	3	**4**	5
Avg. Δ^b(%)	0.01	0.00	0.02
χ^{load}	3	4	**5**
Avg. Δ^b(%)	0.02	0.03	0.00
χ^{rel}	3	4	**5**
Avg. Δ^b(%)	0.01	0.02	0.00
χ^{nb}	3	4	**5**
Avg. Δ^b(%)	0.01	0.03	0.00
χ^{GRASP}	0.2	**0.3**	0.4
Avg. Δ^b(%)	0.02	0.00	0.02
κ	0.5	**0.7**	0.9
Avg. Δ^b(%)	0.02	0.00	0.02
$[\zeta_{min}, \zeta_{max}]$	$[0.95, 1.05]$	$\mathbf{[0.9, 1.1]}$	$[0.85, 1.15]$
Avg. Δ^b(%)	0.03	0.00	0.02
PR			
λ	5	**10**	15
Avg. Δ^b(%)	0.01	0.00	0.01
ρ	0.5	**0.7**	0.9
Avg. Δ^b(%)	0.01	0.00	0.01
Penalties			
δ^{min}	0.01	**0.1**	1
Avg. Δ^b(%)	0.02	0.00	0.02
δ^{update}	**1.1**	1.5	2
Avg. Δ^b(%)	0.00	0.01	0.02
SA			
Δ^{SA}	0.001	**0.01**	0.05
Avg. Δ^b(%)	0.01	0.00	0.01
θ	**100**	200	300
Avg. Δ^b(%)	0.00	0.02	0.02

Table 6.1.: Results of different parameter settings on a randomly drawn subset of VRPSPD and VRPSPDTL instances structured according to the associated algorithmic components. For each parameter, we examine three values and provide for each value the respective average gap of the best solutions found to the BKS from the literature (Δ^b) in percent. The best setting for each parameter is marked in bold and used as the final setting.

ALNS-PR		**ALNS**		**PR**		**Penalties**		**SA**	
ω	4000/500	$\Psi^{\|\|}$	$[0.01, 0.25]$	λ	10	δ^0	10	Δ^{SA}	0.01
μ	500	Ψ^d	$(0.2, ..., 0.5)$	ρ	0.7	δ^{min}	0.1	θ	100
ι^{rem}	200	σ^{best}, σ^{imp}, σ^{acc}	$6, 9, 3$			δ^{max}	10 000	ϵ	3
ι^{add}	100	α	0.2			δ^{update}	1.1		
		γ	20			η^+	2		
		χ^{load}	5			η^-	2		
		χ^{nb}	5						
		χ^{worst}	4						
		χ^{rel}	5						
		χ^c, χ^d, χ^p, χ^e	$2, 1, 1, 1$						
		χ^{GRASP}	0.3						
		κ	0.7						
		$[\zeta_{min}, \zeta_{max}]$	$[0.9, 1.1]$						

Table 6.2.: Final parameter setting of our ALNS-PR structured according to algorithmic components. Each column provides the parameters deployed in the respective component of our algorithm and their setting used for the remaining experiments.

We are aware that the use of the different algorithmic components, which have a significant positive effect on the performance of our method (see Section 6.3.3) is associated with a relatively high total number of parameters. However, as shown in Table 6.1, ALNS-PR is very robust with respect to changes in the parameter values provided that the values stay within the same orders of magnitude as the reported values thus somewhat mitigating this drawback.

6.3.3. Influence of Algorithmic Components

In this section, we analyze the effect of specific components of our ALNS-PR algorithm on solution quality and computation time. For this purpose, we again use ten instances randomly drawn from the instance sets Salhi-VRPSPD and Salhi-VRPSPDTL.

We investigate the contribution of

1. specific ALNS components, namely the decomposition of the original problem into subsets of closely located routes, the route selection policies, the evaluation measures Δf_{div} and Δf_{noise}, the newly introduced load balance removal operator, and the random insertion operator, and
2. our PR implementation.

To this end, we consecutively disable each component while keeping the remaining components enabled. The corresponding results are compared to those obtained by our full ALNS-PR algorithm. More precisely, in Table 6.3, we contrast the average percentage gaps of the best solutions found to the BKS from the literature (Avg. Δ^b) of each algorithmic configuration. Furthermore, Δ^t denotes the percentage deviation of the average of the average run-times per instance of each configuration to the respective value of our full ALNS-PR (shown in bold).

For each reduced algorithmic configuration, we observe a decrease in solution quality compared to our complete approach with a maximum deviation of the best solution quality of 0.09%. Moreover, the large majority of variants also shows a significant increase in computation time if the respective component is omitted. This indicates that the associated components accelerate the convergence rate of the search by helping to discover promising solutions early.

If the current solution is not decomposed in the ALNS step, we observe a run-time increase of roughly 30%. Moreover, omitting the introduced route selection policies slows down computation by roughly

	Components													
Decomposition	✓	✗	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
Route selection														
Cost	✓	✓	✗	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
Distance	✓	✓	✓	✗	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
Efficiency	✓	✓	✓	✓	✗	✓	✓	✓	✓	✓	✓	✓	✓	✓
High avg. util.	✓	✓	✓	✓	✓	✗	✓	✓	✓	✓	✓	✓	✓	✓
Low avg. util.	✓	✓	✓	✓	✓	✓	✗	✓	✓	✓	✓	✓	✓	✓
High max. load	✓	✓	✓	✓	✓	✓	✓	✗	✓	✓	✓	✓	✓	✓
Low max. load	✓	✓	✓	✓	✓	✓	✓	✓	✗	✓	✓	✓	✓	✓
Evaluation														
Δf_{noise}	✓	✓	✓	✓	✓	✓	✓	✓	✓	✗	✓	✓	✓	✓
Δf_{div}	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✗	✓	✓	✓
ALNS operators														
Load balance removal	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✗	✓	✓
Random insertion	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✗	✓
Path relinking	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✗
Avg. Δ^b(%)	**0.01**	0.03	0.05	0.05	0.10	0.03	0.03	0.02	0.03	0.02	0.02	0.03	0.03	0.04
Δ^t(%)		30.09	16.58	21.04	11.28	-0.32	25.85	13.18	15.60	36.75	-2.26	32.47	7.74	110.46

Table 6.3.: Comparison of the performance of different algorithmic configurations.

15%, on average. A negligible speed-up with a decrease in solution quality can be obtained when the high average utilization selection policy is not employed. Deactivating evaluation measure Δf_{noise} in the ALNS step seems to impede the search in finding good solutions early and thus to decelerate its convergence rate by over 36%. On the other hand, the omission of Δf_{div} results in a slight speed-up of roughly 2%. However, we prefer the increased solution quality by utilizing this component over the small decrease in computation time that can be observed if it is omitted. Furthermore, the newly introduced load balance removal operator seems to properly capture the characteristics of the VRPSPD. By deactivating the new operator, the resulting algorithmic configuration additionally takes roughly a third of the time spent by our full ALNS-PR algorithm. In addition, the results hint at an appropriate diversification behavior of the random insertion operator. Besides the increase in solution quality, its utilization results in a noticeable acceleration of the search.

Finally, the hybridization of our ALNS with the described PR implementation yields the most significant increase in convergence speed. Using only our ALNS algorithm more than doubles the computation time required compared to the proposed hybrid approach.

6.3.4. Computational Results on Benchmark Instances

This section presents the results obtained by our ALNS-PR algorithm on the benchmark instances from the literature for the VRPSPD, the VRPSPDTL, the VRPMDP (Section 6.3.4.1), the VRPSPDTW (Section 6.3.4.2), the VRPDDP (Section 6.3.4.3), the VRPRMDDP and the VRPRMDP (Section 6.3.4.4), and on the newly proposed VRPDDPTW instances (Section 6.3.4.5).

With respect to the benchmarks from the literature, we present aggregate views on the performance of all relevant heuristics. The corresponding detailed results can be found in Appendix B.

6.3.4.1. Results on VRPSPD, VRPSPDTL, and VRPMDP Instances

In Table 6.4, we present the summarized results of all relevant heuristics on the VRPSPD (Salhi-VRPSPD, Dethloff, Montané-Medium, and Montané-All) and VRPSPDTL (Salhi-VRPSPDTL and Polat-VRPSPDTL) benchmark instances. The upper part of the table shows the state-of-the-art heuristics that have been applied to the VRPSPD but not the VRPSPDTL; the lower part of the table contains the most successful approaches that provide solutions for both problem types, including our ALNS-PR, which is the only method applied to each set of instances.

For each benchmark, we report the average percentage gap of the best solution quality based on several runs to the BKS (Avg. Δ^b) for each method that has been tested on the benchmark. The average percentage gap of the average solution quality to the BKS (Avg. Δ^a) is provided for those instance sets for which this measure is available for the large majority of comparison algorithms. The BKS for the benchmarks Salhi-VRPSPD and Dethloff are taken from Subramanian et al. (2010), for the benchmarks Montané-Medium, Montané-All, and Salhi-VRPSPDTL from Subramanian, Uchoa, and Ochi (2013), and for the benchmark Polat-VRPSPDTL from Polat et al. (2015).

Moreover, we translate the run-times of all methods into a common time measure that takes into account the processors used. To this end, we relate the Passmark scores (see www.passmark.com) of the processors used in the computational studies of the papers to the score of our i5-6600. Each Passmark score is referring to the performance of a single core of the respective processor. In case of parallel solution approaches, we multiply the translated run-times by the number of utilized threads as reported in the corresponding paper. In addition, the run-times are multiplied by the number of runs performed by the respective algorithm. The resulting times in seconds are given as t^c. We are aware that, due to the use of different operating systems and programming languages, an exact run-time comparison is never possible. However, this procedure is the closest we can get to a fair comparison.

In the following, we list the algorithms that are compared in Table 6.4 and explain how to interpret the corresponding results and run-times:

- For **ZTK** (Zachariadis, Tarantilis, and Kiranoudis, 2010) the number of runs performed to obtain the best solution reported is unknown (marked as ? in the table) and the run-time is based on the time elapsed when the best solution was found.
- The results of **SDBOF** (Subramanian, Drummond, Bentes, Ochi, and Farias, 2010) are based on 50 runs performed by 256 parallel threads and the average time per run.
- For **GKA** (Goksal, Karaoglan, and Altiparmak, 2013), **VCGP** (Vidal, Crainic, Gendreau, and Prins, 2014a), **SUO** (Subramanian, Uchoa, and Ochi, 2013), and **KK** (Kalayci and Kaya, 2016), the table provides results based on ten runs and the average time per run.
- **P** (Polat, 2017) uses six parallel threads and performs ten runs. The run-time is based on the average time required to obtain the best solution reported.
- For **PKKG** (Polat, Kalayci, Kulak, and Günther, 2015), the results are based on ten runs and the time of the best run for benchmarks Salhi-VRPSPD and Salhi-VRPSPDTL, and on the average time per run for benchmark Polat-VRPSPDTL.
- The results reported for ALNS-PR are based on ten runs and the average time per run.

	ZTK		SDBOF			GKA		VCGP			P	
Benchmark	Avg. Δ^b(%)	t^c(s)	Avg. Δ^b(%)	Avg. Δ^a(%)	t^c(s)	Avg. Δ^b(%)	t^c(s)	Avg. Δ^b(%)	Avg. Δ^a(%)	t^c(s)	Avg. Δ^b(%)	t^c(s)
Salhi-VRPSPD	0.11	5.13 × ?	0.00	-	134 656	0.10	1294.87	0.00	-	518.60	-	-
Dethloff	0.00	0.91 × ?	0.00	-	18 048	0.00	19.60	-	-	-	0.00	189.00
Montané-Medium	0.15	7.48 × ?	0.02	0.10	232 064	-	-	0.03	0.10	827.30	0.04	8110.20
Montané-All	0.47	26.79 × ?	0.17	0.30	1 156 352	-	-	0.07	0.20	2226.90	-	-

	SUO			PKKG			KK			**ALNS-PR**		
Benchmark	Avg. Δ^b(%)	Avg. Δ^a(%)	t^c(s)	Avg. Δ^b(%)	Avg. Δ^a(%)	t^c(s)	Avg. Δ^b(%)	Avg. Δ^a(%)	t^c(s)	Avg. Δ^b(%)	Avg. Δ^a(%)	t^c(s)
Salhi-VRPSPD	0.21	-	1710.40	0.02	-	337.50	0.02	-	639.80	0.00	-	1308.50
Dethloff	-	-	-	-	-	-	0.00	-	38.20	0.00	-	105.60
Montané-Medium	0.00	0.03	4943.60	-	-	-	-	-	-	0.03	0.23	2246.30
Montané-All	0.00	0.08	22 585.90	-	-	-	-	-	-	0.09	0.37	9872.40
Salhi-VRPSPDTL	0.00	0.14	629.30	0.18	0.19	1155.10	0.00	0.03	1574.90	0.00	0.21	891.50
Polat-VRPSPDTL	-	-	-	0.00	0.14	1044.40	-	-	-	-0.15	0.16	52.10

Table 6.4.: Comparison of the results of ALNS-PR to those of the state-of-the-art VRPSPD and VRPSPDTL heuristics from the literature on the Salhi-VRPSPD, Dethloff, Montané-Medium, Montané-All, Salhi-VRPSPDTL, and Polat-VRPSPDTL benchmark sets.

Our ALNS-PR belongs to the best-performing approaches showing an average gap of the best solutions to the BKS of 0.00% on the sets Salhi-VRPSPD, Dethloff, and Salhi-VRPSPDTL, and competitive run-times with regard to the most successful heuristics that allow a fair comparison. On set Montané-All, SUO is the only approach able to obtain the BKS for each instance while also spending by far the most computation time. The slightly worse solution quality achieved by ALNS-PR is accompanied by a run-time advantage of roughly 56% compared to SUO. The best trade-off between solution quality and run-time on these instances can be achieved by VCGP. On set Polat-VRPSPDTL, we improve three out of seven previous BKS (see Table B.5 in Appendix B) and note an average improvement of -0.15% while spending only a fraction of the run-time required by PKKG. It is further noteworthy, that we are able to improve two previous BKS on set Salhi-VRPSPDTL during our entire testing activities (see Table B.4 in Appendix B). The robustness of our ALNS-PR is indicated by the small deviations of the average solution quality to the best solution quality.

In Table 6.5, we additionally provide an aggregated comparison of our results to those of SUO and VCGP on instance set Salhi-VRPMDP. Again, our ALNS-PR shows a competitive solution quality while improving two previous BKS (see Appendix Table B.6). The BKS are taken from Vidal et al. (2014b).

	SUO			VCGP			**ALNS-PR**		
Benchmark	Avg. Δ^b(%)	Avg. Δ^a(%)	t^c(s)	Avg. Δ^b(%)	Avg. Δ^a(%)	t^c(s)	Avg. Δ^b(%)	Avg. Δ^a(%)	t^c(s)
Salhi-VRPMDP	0.00	0.09	1101.20	0.00	0.06	457.20	0.00	0.18	1172.72

Table 6.5.: Comparison of the results of ALNS-PR to those of SUO and VCGP on the VRPMDP benchmark Salhi-VRPMDP.

6.3.4.2. Results on VRPSPDTW Instances

Table 6.6 shows an aggregated view on the performance of ALNS-PR and **WMZS** (Wang, Mu, Zhao, and Sutherland, 2015) on the VRPSPDTW benchmark sets Wang-Medium and Wang-Large, based on the BKS reported by Wang and Chen (2012) and Wang et al. (2015). For WMZS, the results are based on 66 threads and the time of the best run. The total number of runs performed is unknown. In addition to the previously reported measures, we provide for each method and set of instances, the total absolute deviation of the number of employed vehicles in the best solution to the best-known vehicle number ($\sum \Delta^m$).

	WMZS			**ALNS-PR**		
Benchmark	$\sum \Delta^m$	Avg. Δ^b(%)	t^c(s)	$\sum \Delta^m$	Avg. Δ^b(%)	t^c(s)
Wang-Medium	0	1.12	3961.32 × ?	-17	-2.32	421.20
Wang-Large	0	0.00	170326.86 × ?	-102	-16.93	19076.30

Table 6.6.: Comparison of the results of ALNS-PR to those of WMZS on the VRPSPDTW benchmark sets Wang-Medium and Wang-Large.

On benchmark Wang-Medium, our ALNS-PR is able to reduce the number of employed vehicles for 17 out of 56 instances. On each of the remaining instances, we match the best-known vehicle number and significantly reduce the associated traveled distance for 31 instances (see Table B.7 in Appendix B). The average gaps of the best solution quality to the BKS are calculated across all instances where the

reported vehicle number is equal to the previous best-known number of vehicles. For ALNS-PR, we note an average improvement of the previous BKS of −2.32%.

On set Wang-Large, we reduce the number of vehicles for 22 out of 30 instances by an average count of five vehicles. For the remaining instances, we match the previous best-known vehicle number and significantly reduce the associated traveled distance. In contrast to Wang-Medium, we now consider all instances for calculating the average gap of the best solution quality to the BKS, i.e., even instances for which we obtain a smaller number of vehicles. In general, a reduction of the number of employed vehicles may increase the traveled distance. However, as shown in Table B.8 in Appendix B, for each except one instance where we employ fewer vehicles, we also achieve a notable reduction of the previous best traveled distance as reported by WMZS. On average, we observe a gap of the best solution quality to the BKS of −16.93%.

Finally, it is remarkable that on both benchmark sets, ALNS-PR spends in total only a fraction of the time required by a single run of algorithm WMZS.

6.3.4.3. Results on VRPDDP Instances

In this section, we analyze the performance of our ALNS-PR on the VRPDDP benchmarks Nagy-VRPDDP1, Nagy-VRPDDP2, Nagy-VRPDDP3, Polat-VRPDDP1, and Polat-VRPDDP2. To this end, we compare our results on these instances to those of the algorithms **NWSA** (Nagy, Wassan, Speranza, and Archetti, 2015) and P (cp. Section 6.3.4.1).

With respect to NWSA, the authors initially conducted experiments using a reactive tabu search algorithm that first determines a VRPSPD solution, which is then transformed into a VRPMDP solution with twice as many customers (cp. Section 6.2). Subsequently, the authors develop several algorithmic variants that duplicate not all but only those customers identified to be promising for division according to the insights gained from the previous experiments. The following comparison with NWSA is based on the algorithmic version called DVA, which additionally makes use of an operator that allows for dividing customers during the execution of the algorithm and showed the best performance on average. The results reported for NWSA are based on a single run and the associated run-time. Unfortunately, we could not obtain a Passmark score for the processor used by NWSA (UltraSPARC-IIIi). Following P, we use an equivalent Intel Pentium 4 processor running at 1.90 GHz (Passmark score: 209) to translate the run-times reported by NWSA.

Table 6.7 shows the aggregated results of NWSA, P, and ALNS-PR on the different VRPDDP instance sets. P and ALNS-PR provide solutions for each set of VRPDDP benchmark instances; NWSA has only been applied to the sets Nagy-VRPDDP1 and Nagy-VRPDDP2. Therefore, we do not report results for NWSA on set Nagy-VRPDDP3. For ALNS-PR, we additionally report the total number of customers that have been divided in the best solutions on each benchmark ($\sum n^d$). Unfortunately, this measure is neither reported for NWSA nor P.

On the instances of sets Nagy-VRPDDP1, Nagy-VRPDDP2, and Nagy-VRPDDP3, algorithm P is able to match or significantly improve each previous BKS reported by the authors NWSA. Our ALNS-PR further improves the results of P by −1.42% (Nagy-VRPDDP1), −0.40% (Nagy-VRPDDP2), and −0.61% (Nagy-VRPDDP3) on average while showing notably faster computation times. More precisely, the total time spent by ALNS-PR is significantly lower than the time required to obtain the best solution by a single run of method P on each of the three benchmarks. This is even more

	NWSA		P		**ALNS-PR**		
Benchmark	Avg. Δ^b(%)	t^c(s)	Avg. Δ^b(%)	t^c(s)	$\sum n^d$	Avg. Δ^b(%)	t^c(s)
Nagy-VRPDDP1	3.98	11.99	0.00	12 484.00	36	-1.42	586.50
Nagy-VRPDDP2	2.86	10.25	0.00	25 001.00	865	-0.40	490.40
Nagy-VRPDDP3	-	-	0.00	28 480.00	1743	-0.61	351.30
Polat-VRPDDP1	-	-	0.00	735.00	23	-0.06	348.20
Polat-VRPDDP2	-	-	0.01	30 379.80	12	-0.03	5378.30

Table 6.7.: Comparison of the results of ALNS-PR to those of the heuristics of NWSA and P on the VRPDDP benchmark sets Nagy-VRPDDP1, Nagy-VRPDDP2, Nagy-VRPDDP3, Polat-VRPDDP1, and Polat-VRPDDP2.

remarkable taking into account our more restrictive interpretation of service times in case of divided customers on these instances. As shown in Table B.9 in Appendix B, our ALNS-PR is able to improve the previous BKS for 20 instances in Nagy-VRPDDP1, for 20 instances in Nagy-VRPDDP2, and for 26 instances in Nagy-VRPDDP3 out of 28 instances each. Moreover, only roughly 1% of all customers are divided on set Nagy-VRPDDP1, 27% on Nagy-VRPDDP2 and roughly 55% on Nagy-VRPDDP3. Interestingly, NWSA report a total of 61 divided customers on Nagy-VRPDDP1 for their basic approach while only 36 customers are identified by ALNS-PR to be divided. This might hint at a better capability of our ALNS-PR to identify and reassemble partial customers that have been unnecessarily divided in the course of the search.

Our ALNS-PR is superior to P also on the recently introduced VRPDDP benchmarks Polat-VRPDDP1 and Polat-VRPDDP2. We improve the previous best results by -0.06% and -0.03% on average, and obtain seven and two new BKS, respectively (see Tables B.10 and B.11 in Appendix B). For Polat-VRPDDP1, the BKS are taken from Polat (2017) and for Polat-VRPDDP2 from Subramanian, Uchoa, and Ochi (2013) and Polat (2017). Moreover, ALNS-PR is roughly twice as fast as P on set Polat-VRPDDP1 and nearly six times as fast on set Polat-VRPDDP2.

6.3.4.4. Results on VRPRMDDP and VRPRMDP Instances

In this section, we first investigate the VRPRMDDP on the benchmark Hoff. As explained in Section 6.1, Hoff et al. (2009) study a problem different to the VRPRMDDP. Therefore, the results obtained by Hoff et al. (2009) on these instances are not comparable to our VRPRMDDP results, and we do not report them. In Table 6.8, we present our results for 100%, 80%, 60%, 40%, 20%, and 10% of unavailable vehicle capacity when both demand types are simultaneously loaded as comparison for future solution approaches considering the VRPRMDDP. For each instance and configuration, we report the best solution found in ten runs (f^b) and the average computing time (t^a) in seconds. As expected, we observe a decrease of the average solution quality with increasing restriction of the vehicle capacity in case of mixed loads.

In Table 6.9, we provide an aggregated comparison of the results of our ALNS-PR on the VRPRMDP benchmarks Nagy-VRPRMDP1, Nagy-VRPRMDP2, and Nagy-VRPRMDP3 for 100%, 75%, 50%, 25%, and 10% of unavailable vehicle capacity in case of mixed loads to the respective BKS as reported by Nagy, Wassan, and Salhi (2013).

For each benchmark and capacity configuration, the average over the best solutions found by ALNS-PR significantly improves the previous BKS. As shown in Appendix B, our ALNS-PR is able to match

	ALNS-PR											
	100%		80%		60%		40%		20%		10%	
Inst.	f^b	t^a(s)	f^b	t^a(s)	f^b	t^a(s)	f^b	t^a(s)	f^b	t^a(s)	f^b	t^a(s)
NE-007-02	118.00	0.37	118.00	0.15	118.00	0.10	118.00	0.14	110.00	0.14	110.00	0.16
NE-011-04	3485.00	0.37	3485.00	0.23	3485.00	0.17	3464.00	0.48	3287.00	0.34	3287.00	0.62
NE-013-04	406.00	0.49	406.00	0.29	394.00	0.36	343.00	0.40	322.00	0.37	343.00	0.56
E-016-03	416.79	1.29	403.60	1.50	372.74	1.50	338.06	1.51	318.66	1.71	297.80	1.49
E-016-05	444.63	0.92	444.63	1.14	433.95	0.89	394.47	0.94	378.56	1.00	361.49	0.82
E-021-04	526.56	2.32	526.45	2.72	482.86	2.44	456.37	2.29	404.37	2.40	396.07	2.40
E-021-06	582.42	2.07	578.30	1.45	565.80	1.34	529.74	1.85	512.77	1.45	492.12	1.50
E-022-04	514.02	1.97	514.02	2.34	506.94	2.21	457.70	2.31	424.59	2.14	385.51	2.41
E-022-06	594.50	2.03	594.50	1.87	582.80	2.21	562.14	1.65	521.99	1.93	505.79	1.16
E-023-03	886.11	2.91	822.33	2.73	705.82	2.79	609.73	3.50	569.88	2.00	569.88	2.00
E-026-08	739.40	2.32	739.40	2.35	711.58	2.73	702.31	2.12	674.43	2.98	647.96	2.73
E-030-03	750.70	6.54	690.72	7.55	669.63	8.26	652.88	7.06	561.09	7.83	547.45	8.71
NE-031-07	1080.00	11.40	1097.00	4.92	1061.00	4.91	1046.00	5.60	1014.00	4.76	1046.00	5.89
E-031-09	747.02	3.82	749.31	5.08	729.07	3.80	702.84	3.89	669.22	4.81	649.37	3.77
E-033-04	1038.37	8.67	1011.15	9.65	981.20	9.29	952.63	8.99	887.39	11.64	857.23	8.12
E-036-11	841.97	5.54	841.97	5.25	820.21	5.04	796.31	6.92	755.13	5.79	756.30	4.44
E-041-14	1007.64	7.44	1009.49	7.13	996.94	4.95	960.17	5.50	939.49	6.06	923.99	6.82
E-045-04	935.77	20.14	903.00	33.62	806.48	30.33	778.91	33.49	754.94	31.21	747.37	21.46
E-048-04	54428.06	20.65	49698.22	19.08	44541.01	16.44	41873.60	18.60	40565.89	14.32	40307.62	16.27
E-051-05	840.67	33.46	793.58	30.47	734.39	26.70	649.25	26.10	593.25	38.16	564.13	24.49
E-072-04	350.11	96.02	320.61	114.85	304.89	113.68	274.82	101.81	266.88	167.47	250.24	234.72
E-076-07	1065.54	96.49	1008.57	64.73	892.80	70.54	788.39	58.76	714.72	78.09	700.12	54.61
E-076-08	1097.27	77.76	1054.82	61.59	958.90	56.54	868.75	52.79	794.19	51.96	762.46	65.96
E-076-10	1162.38	67.37	1116.48	61.93	1048.03	58.92	969.55	56.79	912.08	48.32	892.80	62.08
E-076-15	1260.44	45.89	1240.89	41.96	1191.77	39.10	1147.70	34.86	1091.64	40.94	1069.10	29.85
E-101-08	1287.94	170.87	1170.39	136.03	1048.93	154.46	952.96	161.15	870.84	144.55	842.93	90.28
E-101-10	1092.46	156.48	1066.41	117.72	1018.71	81.33	956.57	88.90	893.99	89.20	867.30	63.47
E-101-14	1421.65	172.72	1379.64	119.68	1307.14	84.41	1209.70	90.45	1160.63	99.47	1141.95	68.10
E-121-07	1342.72	488.75	1295.64	449.57	1239.97	403.34	1163.31	501.13	1104.29	454.93	1074.05	441.60
E-135-07	1579.66	722.38	1542.31	904.35	1457.63	1088.64	1388.67	1048.27	1285.91	1598.00	1256.57	1202.25
E-151-12	1483.47	650.49	1411.54	351.15	1281.87	350.67	1177.97	377.30	1096.94	331.31	1077.48	358.61
E-200-16	1813.70	1597.81	1801.92	708.79	1685.09	769.74	1580.38	911.10	1479.02	922.16	1483.75	746.02
E-200-17	1764.09	1590.04	1683.03	731.01	1560.29	641.37	1470.23	747.93	1387.36	771.41	1362.05	681.96
E-262-25	7029.52	3848.94	6897.87	1239.63	6689.25	1294.97	6426.36	1663.84	6200.79	1612.11	6061.35	1373.77
E-484-19	1639.21	14192.01	1497.65	14926.82	1367.96	14855.63	1274.26	18167.54	1199.36	19426.40	1188.81	16185.15
Avg.	**2736.39**	**688.82**	**2568.98**	**576.27**	**2364.36**	**576.85**	**2229.65**	**691.31**	**2134.95**	**742.21**	**2109.34**	**622.12**

Table 6.8.: Results on the VRPRMDDP benchmark Hoff for different percentages of unavailable vehicle capacity in case of mixed loads.

	ALNS-PR									
	100%		75%		50%		25%		10%	
Benchmark	Avg. Δ^b(%)	Avg. t^a(s)	Avg. Δ^b(%)	Avg. t^a(s)	Avg. Δ^b(%)	Avg. t^a(s)	Avg. Δ^b(%)	Avg. t^a(s)	Avg. Δ^b(%)	Avg. t^a(s)
Nagy-VRPRMDP1	-9.36	15.63	-8.09	18.37	-6.10	20.88	-4.17	21.61	-2.71	21.49
Nagy-VRPRMDP2	-8.05	9.18	-5.73	9.88	-4.03	10.27	-2.66	11.06	-1.95	10.50
Nagy-VRPRMDP3	-10.67	28.65	-7.83	27.46	-5.80	34.14	-4.62	35.58	-4.20	39.46

Table 6.9.: Comparison of the results of ALNS-PR to the BKS as reported by Nagy, Wassan, and Salhi (2013) on the VRPRMDP benchmark sets Nagy-VRPRMDP1, Nagy-VRPRMDP2, and Nagy-VRPRMDP3 for different percentages of unavailable vehicle capacity in case of mixed loads.

or improve every previous BKS of the sets Nagy-VRPRMDP1 (Table B.12) and Nagy-VRPRMDP3 (Table B.14) and the large majority of BKS of set Nagy-VRPRMDP2 (Table B.13) while using average run-times clearly below one minute. Nagy, Wassan, and Salhi (2013) do not report the computation times of their approach, and therefore no precise comparison is possible.

6.3.4.5. Results on VRPDDPTW Instances

In Table 6.10, we finally provide detailed results on the newly generated VRPDDPTW instances as comparison for future methods that address the VRPDDPTW. Moreover, in order to analyze the savings achievable by dividing customer demands, we compare the VRPDDPTW solutions to VRPSPDTW solutions obtained by our ALNS-PR on the new instances. We provide for each instance, the name and the number of customers, and for both problem types, the number of employed vehicles (m) and the best solution found in ten runs (f^b). For the VRPDDPTW, we additionally report the absolute deviation of the number of employed vehicles in the best solution to the vehicle number in the corresponding best VRPSPDTW solution, the number of divided customers in the best solution (n^d), the percentage gap of the best solution to the best VRPSPDTW solution (Δ^b), and the average computing time (t^a) in seconds.

Out of 56 instances, our ALNS-PR identifies 38 instances for which the division of demands is beneficial. In total, 652 customers are divided, allowing for a reduction of the number of employed vehicles by 23 compared to the VRPSPDTW case. On average, the traveled distance on the instances with an equal number of vehicles can be improved by -1.83% if dividing demands is permitted. Finally, our ALNS-PR shows reasonable computation times of less than one minute on average.

6.4. Conclusion

We present an adaptive large neighborhood search algorithm combined with a path relinking approach, called ALNS-PR, to address a class of VRPs with simultaneous pickup and delivery (VRPSPD).

In extensive numerical studies, we first demonstrate the usefulness of the proposed algorithmic components. We show that the omission of each component leads to a decrease in solution quality. Moreover, we find that especially the hybridization of our ALNS component with the proposed PR implementation, and the introduction of an innovative ALNS operator, which explicitly considers the load characteristics of the VRPSPD and its variants, significantly accelerate the convergence rate of the search.

The competitiveness of the proposed approach is demonstrated on benchmark instances from the literature. On established instances, ALNS-PR can compete with the state-of-the-art approaches for the corresponding problems. With respect to the more recently introduced problem variants, especially on VRPSPDTL, VRPSPDTW, and VRPDDP instances, our method proves to be superior to the majority of comparison algorithms and provides numerous new best solutions.

Our ALNS-PR is therefore suitable to tackle the practical application occurring at DHL Freight in Sweden that particularly motivated this chapter. There, up to 50 000 customers need to be served from multiple depots. The scale of the resulting multi-depot VRPSPD suggests its decomposition into several VRPSPDs that can be solved independently. Future research thus needs to address the necessary algorithmic modifications to appropriately decompose a corresponding large-scale instance.

		ALNS-PR							
		VRPSPDTW		VRPDDPTW					
Inst.	n	m	f^b	m	Δ^m	n^d	f^b	Δ^b(%)	t^a(s)
HS-cdp101	100	23	**2345.75**	23	0	0	**2345.75**	0.00	27.04
HS-cdp102	100	23	**1840.89**	23	0	0	**1840.89**	0.00	31.10
HS-cdp103	100	22	1799.17	22	0	17	**1761.04**	-2.12	30.44
HS-cdp104	100	22	1752.16	22	0	8	**1652.09**	-5.71	34.93
HS-cdp105	100	23	**2085.45**	23	0	0	**2085.45**	0.00	22.86
HS-cdp106	100	22	**2772.87**	22	0	0	**2772.87**	0.00	27.39
HS-cdp107	100	22	**2684.66**	22	0	0	**2684.66**	0.00	25.92
HS-cdp108	100	22	**2378.12**	22	0	0	**2378.12**	0.00	25.79
HS-cdp109	100	22	**1982.66**	22	0	0	**1982.66**	0.00	29.35
HS-cdp201	100	17	1611.83	17	0	7	**1486.13**	-7.80	66.93
HS-cdp202	100	17	1461.67	16	**-1**	13	1803.94	23.42	52.97
HS-cdp203	100	17	1422.22	16	**-1**	13	1411.96	-0.72	71.46
HS-cdp204	100	17	1394.54	16	**-1**	12	1385.99	-0.61	60.97
HS-cdp205	100	17	1433.86	16	**-1**	26	2202.92	53.64	73.98
HS-cdp206	100	17	1421.85	16	**-1**	24	2102.17	47.85	50.61
HS-cdp207	100	17	1409.30	17	0	6	**1396.02**	-0.94	57.86
HS-cdp208	100	17	1421.41	16	**-1**	17	2184.17	53.66	64.15
HS-rdp101	100	22	**1965.84**	22	0	0	**1965.84**	0.00	25.92
HS-rdp102	100	21	**2031.76**	21	0	0	**2031.76**	0.00	36.01
HS-rdp103	100	21	**1662.40**	21	0	0	**1662.40**	0.00	35.77
HS-rdp104	100	20	**1906.62**	20	0	0	**1906.62**	0.00	42.75
HS-rdp105	100	21	**1967.15**	21	0	0	**1967.15**	0.00	26.10
HS-rdp106	100	20	**2467.94**	20	0	0	**2467.94**	0.00	39.30
HS-rdp107	100	20	1962.77	20	0	27	**1925.47**	-1.90	38.56
HS-rdp108	100	20	1623.36	20	0	4	**1608.53**	-0.91	36.84
HS-rdp109	100	21	**1668.95**	21	0	0	**1668.95**	0.00	31.83
HS-rdp110	100	20	**2239.71**	20	0	0	**2239.71**	0.00	35.61
HS-rdp111	100	21	**1637.16**	21	0	0	**1637.16**	0.00	33.30
HS-rdp112	100	20	1657.49	20	0	0	**1656.78**	-0.04	32.43
HS-rdp201	100	15	2337.05	15	0	10	**2090.14**	-10.57	52.64
HS-rdp202	100	15	1572.11	14	**-1**	39	1936.52	23.18	105.56
HS-rdp203	100	15	1635.94	14	**-1**	28	2256.06	37.91	113.51
HS-rdp204	100	15	1314.99	14	**-1**	17	1675.77	27.44	108.30
HS-rdp205	100	15	1671.05	14	**-1**	43	2202.27	31.79	92.56
HS-rdp206	100	15	1547.44	14	**-1**	31	1854.27	19.83	94.33
HS-rdp207	100	15	1350.41	14	**-1**	34	1678.72	24.31	98.72
HS-rdp208	100	15	1333.82	14	**-1**	31	1415.20	6.10	109.50
HS-rdp209	100	15	1609.23	14	**-1**	22	2069.46	28.60	79.36
HS-rdp210	100	15	1544.21	14	**-1**	41	1965.03	27.25	116.10
HS-rdp211	100	15	1365.45	15	0	11	**1209.21**	-11.44	117.33
HS-rcdp101	100	23	2340.21	23	0	1	**2336.11**	-0.18	23.69
HS-rcdp102	100	22	2435.76	22	0	19	**2320.66**	-4.73	32.72
HS-rcdp103	100	22	2173.06	22	0	10	**2038.64**	-6.19	45.59
HS-rcdp104	100	22	2011.60	22	0	12	**1967.52**	-2.19	56.28
HS-rcdp105	100	22	**2588.82**	22	0	0	**2588.82**	0.00	22.42
HS-rcdp106	100	22	2386.96	22	0	1	**2374.35**	-0.53	28.48
HS-rcdp107	100	22	2121.41	22	0	11	**2033.68**	-4.14	36.16
HS-rcdp108	100	22	2021.30	22	0	7	**1999.16**	-1.10	37.99
HS-rcdp201	100	16	2004.33	15	**-1**	17	2859.48	42.67	72.38
HS-rcdp202	100	16	1716.64	15	**-1**	12	1672.16	-2.59	82.14
HS-rcdp203	100	16	1588.73	15	**-1**	17	1518.74	-4.41	64.54
HS-rcdp204	100	16	1584.29	15	**-1**	10	1464.00	-7.59	91.59
HS-rcdp205	100	16	1874.17	15	**-1**	8	2239.42	19.49	60.44
HS-rcdp206	100	16	1693.38	15	**-1**	16	1973.35	16.53	78.13
HS-rcdp207	100	16	1689.49	15	**-1**	17	1602.60	-5.14	80.79
HS-rcdp208	100	16	1572.66	15	**-1**	13	1445.75	-8.07	82.49
Avg.								**-1.83**	**56.28**
$\sum$					**-23**	**652**			

Table 6.10.: Results on the newly generated VRPDDPTW benchmark HS.

Chapter 7

Summary, Conclusion, and Outlook

This thesis addresses several challenges faced in the context of sustainable logistics from the perspective of operations research. We develop respective models and solution methods to provide decision support for planning operations that have to (1) integrate intermediate stops at, e.g., recharging stations into the routes of transportation vehicles, (2) jointly take decisions on the location of battery swap stations (BSSs) with decisions on the routing of battery electric vehicles (BEVs), (3) consider possible resource transfers between vehicles, and (4) simultaneously manage the flow of goods to the end consumers as well as back to the point of origin.

This chapter summarizes the contents of this thesis, lists its major contributions and provides possible avenues for future research.

7.1. Summary and Conclusion

In Chapter 2, we introduce the capacitated vehicle-routing problem (CVRP) which builds the common foundation of all problems investigated in this thesis. In addition, we describe the basic principles of the metaheuristic concepts simulated annealing (SA), variable neighborhood search (VNS), large neighborhood search (LNS), and path relinking (PR) that we utilize to develop the solution methods presented in this work.

In Chapter 3, we study the VRP with intermediate stops (VRPIS), which considers the possibility for vehicles to stop en route to replenish the goods to be delivered or to refuel at appropriate facilities. To provide solutions for the VRPIS, we develop an adaptive VNS (AVNS) heuristic. The shaking step is based on sequence relocation and cyclic exchange neighborhood structures. Our algorithm guides the shaking step by adapting the selection probabilities of the methods which determine the routes and vertices involved in the shaking based on the past performance of these methods. We implement problem-specific methods that explicitly take the characteristics of intermediate stops like necessary detours to respective facilities or the facility density in routes into account. A local search procedure that makes use of classical operators as well as operators able to insert, rearrange, and remove visits to intermediate facilities is employed to perform the local descent in each investigated neighborhood. To further diversify the search, we use an acceptance mechanism inspired by SA.

On benchmark instances from the literature for the two VRPIS special cases green VRP and VRP with intermediate replenishment facilities, our AVNS algorithm shows a convincing performance compared to the approaches from the literature and is able to obtain numerous new best solutions. Moreover, we consider an additional special case of the VRPIS, the electric VRP with recharging facilities

(EVRPRF). We design two sets of small and large EVRPRF instances based on well-known CVRP benchmarks. On the small instances, we show in comparison to the commercial solver CPLEX, that our AVNS is capable of identifying highly efficient vehicle routes making use of the available recharging stations. Finally, we perform experiments that clearly prove the usefulness of the introduced algorithmic innovations regarding the characteristics of intermediate stops.

In Chapter 4, we investigate the battery swap station location-routing problem with capacitated electric vehicles (BSS-EV-LRP). To address the BSS-EV-LRP, we extend our AVNS algorithm originally designed for VRPIS by incorporating the BSS construction cost into the objective function and introducing the new facility-related neighborhood structures facility removal and facility replacement in the shaking step. Our extended AVNS is able to significantly improve the results of the comparison algorithm SIGALNS by Yang and Sun (2015) and provides new best solutions for the large majority of instances, namely for 23 out of 24, showing an average improvement of nearly 10%. Moreover, the AVNS strongly reduces the number of constructed BSSs in the solutions compared to SIGALNS. Therefore, we generate new BSS-EV-LRP instances which are more meaningful concerning the necessity of using BSSs. Additional experiments on the new instances show that (i) decreasing construction costs lead to the expected increase in the number of located BSSs, and (ii) the proposed methodological innovations related to BSSs and the explicit consideration of the BSS construction costs have a beneficial impact on the solution quality.

In Chapter 5, we introduce the VRP with time windows and mobile depots (VRPTWMD), which is characterized by a fleet of delivery vehicles (DVs) and a fleet of support vehicles (SVs) that may replenish the fuel or load capacity of the DVs. We develop an adaptive LNS (ALNS) heuristic combined with a PR approach, called ALNS-PR, to address the VRPTWMD. At each ALNS iteration of our ALNS-PR algorithm, several sub-components like removal and insertion operators to be applied to the current solution are selected based on probabilities that are dynamically updated during the search. In addition, our ALNS-PR makes use problem-specific operators and route selection policies that aim at capturing the characteristics of transfer operations. Finally, we develop move evaluation procedures that allow us to evaluate changes in fuel, load, and time window violations in constant time. With respect to the latter, we adapt the time travel approach proposed by Nagata, Bräysy, and Dullaert (2010) to the synchronization of both vehicle types.

We generate two sets of VRPTWMD instances. On a set of small-sized instances, the comparison with the commercial solver CPLEX shows that our ALNS-PR is able to obtain high-quality solutions and thus to appropriately capture the necessity of performing transfers and the associated deployment of SVs. On large VRPSPD instances, we demonstrate the usefulness of the implemented transfer-related algorithmic components and of our PR approach. In addition, we use the large VRPTWMD instances to study the effect of different problem characteristics on the solution structure. We find that fuel transfers are essential in each problem configuration to be able to serve customers that are located far away from the depot. In contrast, the utilization of load transfers quickly diminishes as the problem setting becomes increasingly constrained with respect to transfer operations, i.e., due to a decreasing load capacity of the SVs, rising cost for the utilization of SVs, and an increasing time consumption associated with transfers. We further demonstrate that the flexibility gained from treating the task order at a customer as a decision variable results in notable cost savings compared to the assumption of always executing the transfer operation before the service. Moreover, we demonstrate the competitiveness of our approach on benchmark instances for the related two-echelon multiple-trip

VRP with satellite synchronization. We are able to significantly improve the large majority (49 out of 56) of the previous best-known solutions, while spending remarkably less computation time than the corresponding state-of-the-art algorithm from the literature.

Finally, Chapter 6 investigates a class of VRPs with simultaneous pickup and delivery (VRPSPDs). To provide solutions for the investigated VRPSPD variants, we again use a hybrid algorithm composed of ALNS and PR. We show that each component is necessary to achieve a high solution quality. Especially the hybridization of our ALNS component with the proposed PR approach, and the introduction of an innovative ALNS operator, which explicitly considers the fluctuating nature of the vehicle load arising in the VRPSPD and its variants, significantly accelerate the convergence rate of the search.

With respect to the competitiveness of the proposed method, on established instances, ALNS-PR can compete with the state-of-the-art approaches for the corresponding problems. Regarding the more recently introduced problem variants, especially on the instances for the VRPSPD with time limit, the VRPSPD with time windows, and the VRP with divisible deliveries and pickups, our method outperforms the majority of comparison algorithms. This is also true for the special cases VRP with mixed deliveries and pickups and VRP with restricted mixing of deliveries and pickups, where ALNS-PR shows a convincing performance and provides numerous new best solutions.

7.2. Outlook

With regard to the problems addressed in this thesis, several promising avenues for future research come to mind. The BSS-EV-LRP model might be reasonably augmented by the consideration of recharging stations with different recharging technologies that, depending on the construction cost, offer varying recharging speeds. The latter is crucial in problem settings where tight time windows need to be taken into account. Moreover, as described in Section 1, future recharging demand is expected to be highly localized. Thus, investors in recharging infrastructure will most likely have to face construction cost for recharging stations or BSSs that additionally depend on the attractiveness of the associated locations which should also be adequately addressed by respective planning models.

Integrating a heterogeneous fleet of SVs into the VRPTWMD model that allows to replenish both the fuel and the load capacity of the DVs could further increase its utility for practical applications. Especially the small and possibly electrically powered DVs employed in urban distribution systems rely on a frequent resupply of both resources.

Another interesting and challenging application scenario that could enrich the VRPTWMD model consists in the replenishment of the time capacity of the DVs in the form of driver exchanges. A similar problem occurring in long-distance road transport is studied by Drexl et al. (2013). There, a driver who needs to rest visits a relay station where another driver takes over the truck and continues the tour. The drivers may be transported between the relay stations using shuttle vans which thus may be viewed as SVs.

Similar to fuel, time is consumed along arcs as travel time and similar to load, at vertices in form of service, transfer, and possibly waiting times. Time is always changing. Moreover, in contrast to the resource types already considered, time changes may propagate through the entire solution as it is the case with changes in time window violation. Therefore, the evaluation of changes in time violation does not only require the consideration of the route segment currently being modified but

also the succeeding segments connected via transfers. Thus, innovative preprocessing procedures likely become necessary to quickly retrieve the required solution information and thus to efficiently evaluate the respective changes.

In addition, it seems worthwhile to consider two- or three-dimensional loading constraints and to investigate different loading patterns in the VRPSPD variants with restricted mixing of deliveries and pickups especially with respect to different types of loading access (rear or side loading) determined by the specific transportation vehicles used. While two- and three-dimensional versions of the basic VRPSPD are studied by Zachariadis, Tarantilis, and Kiranoudis (2016) and Koch, Bortfeldt, and Wäscher (2018), respectively, similar studies with regard to other VRPSPD variants are, to the best of our knowledge, still open for investigation.

Finally, practical reverse logistics applications are usually characterized by multi-echelon distribution and collection systems involving collection centers where goods to be returned are consolidated before they are transported to different rework sites like repair and refurbishing or remanufacturing centers. While this assumption is generally incorporated in respective network design problems (see, e.g., Min, Ko, and Ko, 2006; Srivastava, 2008), the vehicle-routing literature exclusively considers single-stage networks, to the best of our knowledge. Thus, a combination of VRPSPDs with the VRPIS or the VRPTWMD models appears to be another beneficial future research direction.

Bibliography

Adler, J. D. and P. B. Mirchandani (2014). Online routing and battery reservations for electric vehicles with swappable batteries. In: *Transportation Research Part B: Methodological* 70, pp. 285–302.

Amaya, A., A. Langevin, and M. Trépanier (2007). The capacitated arc routing problem with refill points. In: *Operations Research Letters* 35 (1), pp. 45–53.

Amaya, C.-A., A. Langevin, and M. Trépanier (2010). A heuristic method for the capacitated arc routing problem with refill points and multiple loads. In: *Journal of the Operational Research Society* 61 (7), pp. 1095–1103.

Angelelli, E. and R. Mansini (2002). The vehicle routing problem with time windows and simultaneous pick-up and delivery. In: *Quantitative Approaches to Distribution Logistics and Supply Chain Management*. Ed. by A. Klose, M. G. Speranza, and L. N. Van Wassenhove. Berlin, Heidelberg: Springer Berlin Heidelberg, pp. 249–267.

Appel, F. (2017). *Climate-efficient logistics is possible*. URL: https://delivering-tomorrow.com/climate-efficient-logistics-is-possible/ (visited on 10/08/2018).

Archetti, C., M. G. Speranza, and D. Vigo (2014). Chapter 10: Vehicle Routing Problems with Profits. In: *Vehicle Routing*, pp. 273–297.

Arslan, O. and O. E. Karaşan (2016). A Benders decomposition approach for the charging station location problem with plug-in hybrid electric vehicles. In: *Transportation Research Part B: Methodological* 93, Part A, pp. 670–695.

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

Bard, J., L. Huang, M. Dror, and P. Jaillet (1998). A branch and cut algorithm for the VRP with satellite facilities. In: *IIE Transactions* 30 (9), pp. 821–834.

Barnes, J., V. Wiley, J. Moore, and D. Ryer (2004). Solving the aerial fleet refueling problem using group theoretic tabu search. In: *Mathematical and Computer Modelling* 39 (6–8), pp. 617–640.

Beliën, J., L. De Boeck, and J. Van Ackere (2014). Municipal solid waste collection and management problems: a literature review. In: *Transportation Science* 48 (1), pp. 78–102.

Benjamin, A. and J. Beasley (2010). Metaheuristics for the waste collection vehicle routing problem with time windows, driver rest period and multiple disposal facilities. In: *Computers & Operations Research* 37 (12), pp. 2270–2280.

Blum, C. and A. Roli (2003). Metaheuristics in combinatorial optimization: overview and conceptual comparison. In: *ACM Comput. Surv.* 35 (3), pp. 268–308.

Bönnighausen, D. (2017). *Deutsche Post DHL hat 5.000 StreetScooter im Einsatz*. URL: https://www.electrive.net/2017/11/28/5-000-streetscooter-bei-deutsche-post-dhl-im-einsatz/ (visited on 10/08/2018).

Bräysy, O. and M. Gendreau (2005a). Vehicle routing problem with time windows, Part I: route construction and local search algorithms. In: *Transportation Science* 39 (1), pp. 104–118.

– (2005b). Vehicle routing problem with time windows, Part II: metaheuristics. In: *Transportation Science* 39 (1), pp. 119–139.

Breunig, U., R. Baldacci, R. Hartl, and T. Vidal (2019). The electric two-echelon vehicle routing problem. In: *Computers & Operations Research* 103, pp. 198–210.

Černý, V. (1985). Thermodynamical approach to the traveling salesman problem: an efficient simulation algorithm. In: *Journal of Optimization Theory and Applications* 45 (1), pp. 41–51.

Chen, Z., F. He, and Y. Yin (2016). Optimal deployment of charging lanes for electric vehicles in transportation networks. In: *Transportation Research Part B: Methodological* 91, pp. 344–365.

Christofides, N. and S. Eilon (1969). An algorithm for the vehicle-dispatching problem. In: *Operational Research Quarterly* 20 (3), pp. 309–318.

Christofides, N., A. Mingozzi, and P. Toth (1979). The vehicle routing problem. In: *Combinatorial optimization*.

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

Coene, S., A. Arnout, and F. C. R. Spieksma (2010). On a periodic vehicle routing problem. In: *Journal of the Operational Research Society* 61, pp. 1719–1728.

Coffman, E. G., M. R. Garey, and D. S. Johnson (1984). Approximation algorithms for bin-packing — an updated survey. In: *Algorithm Design for Computer System Design*. Ed. by G. Ausiello, M. Lucertini, and P. Serafini. Vienna: Springer Vienna, pp. 49–106.

Conrad, R. G. and M. A. Figliozzi (2011). The recharging vehicle routing problem. In: *Proceedings Industrial Engineering Research Conference*.

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

Crainic, T. G., N. Ricciardi, and G. Storchi (2009). Models for evaluating and planning city logistics systems. In: *Transportation Science* 43 (4), pp. 432–454.

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, pp. 756–773.

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

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

Dayarian, I., M. Savelsbergh, and J.-P. Clarke (2018). *Same-Day Delivery with Drone Resupply*. Tech. rep. Milton Stewart School of Industrial & Systems Engineering, Georgia Institute of Technology, Atlanta, USA.

Dekker, R., M. Fleischmann, K. Inderfurth, and L. N. van Wassenhove (2013). Reverse logistics: quantitative models for closed-loop supply chains. Springer Science & Business Media.

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.

Dethloff, J. (2001). Vehicle routing and reverse logistics: The vehicle routing problem with simultaneous delivery and pick-up. In: *OR Spectrum* 23 (1), pp. 79–96.

Drexl, M. (2012). Synchronization in vehicle routing – A survey of VRPs with multiple synchronization constraints. In: *Transportation Science* 46 (3), pp. 297–316.

Drexl, M., J. Rieck, T. Sigl, and B. Press (2013). Simultaneous vehicle and crew routing and scheduling for partial- and full-load long-distance road transport. In: *Business Research* 6 (2), pp. 242–264.

Drexl, M. and M. Schneider (2014). A survey of variants and extensions of the location-routing problem. In: *European Journal of Operational Research* 241 (2), pp. 283–308.

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.

Fallahi, A. E., C. Prins, and R. W. Calvo (2008). A memetic algorithm and a tabu search for the multi-compartment vehicle routing problem. In: *Computers & Operations Research* 35 (5), pp. 1725–1741.

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

Gehring, H. and J. Homberger (1999). A parallel hybrid evolutionary metaheuristic for the vehicle routing problem with time windows. In: *Proceedings of EUROGEN99*. Ed. by K. Miettinen, M. Mäkelä, and J. Toivanen. A2. Springer, Berlin, pp. 57–64.

Gendreau, M. and J.-Y. Potvin (2005). Metaheuristics in combinatorial optimization. In: *Annals of Operations Research* 140 (1), pp. 189–213.

Glover, F. (1997). Tabu search and adaptive memory programming — advances, applications and challenges. In: *Interfaces in Computer Science and Operations Research: Advances in Metaheuristics, Optimization, and Stochastic Modeling Technologies*. Boston, MA: Springer US, pp. 1–75.

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.

Goetschalckx, M. and C. Jacobs-Blecha (1989). The vehicle routing problem with backhauls. In: *European Journal of Operational Research* 42 (1), pp. 39–51.

Goksal, F. P., I. Karaoglan, and F. Altiparmak (2013). A hybrid discrete particle swarm optimization for vehicle routing problem with simultaneous pickup and delivery. In: *Computers & Industrial Engineering* 65 (1), pp. 39–53.

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

Grangier, P., M. Gendreau, F. Lehuédé, and L.-M. Rousseau (2016). An adaptive large neighborhood search for the two-echelon multiple-trip vehicle routing problem with satellite synchronization. In: *European Journal of Operational Research* 254 (1), pp. 80–91.

Hansen, P. and N. Mladenović (2001). Variable neighborhood search: principles and applications. In: *European Journal of Operational Research* 130 (3), pp. 449–467.

Hansen, P., N. Mladenović, J. Brimberg, and J. A. M. Pérez (2010). Variable neighborhood search. In: *Handbook of Metaheuristics*. Ed. by M. Gendreau and J.-Y. Potvin. Boston: Springer US, pp. 61–86.

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.

Hemmelmayr, V. C., J.-F. Cordeau, and T. G. Crainic (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.

Hemmelmayr, V. C., K. F. Doerner, and R. F. Hartl (2009). A variable neighborhood search heuristic for periodic routing problems. In: *European Journal of Operational Research* 195 (3), pp. 791–802.

Hemmelmayr, V., K. F. Doerner, R. F. Hartl, 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.

Ho, S. C. and M. Gendreau (2006). Path relinking for the vehicle routing problem. In: *Journal of Heuristics* 12 (1), pp. 55–72.

Hof, J. and M. Schneider (forthcoming). An adaptive large neighborhood search with path relinking for a class of vehicle-routing problems with simultaneous pickup and delivery. In: *Networks*.

Hof, J., 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.

Hoff, A., I. Gribkovskaia, G. Laporte, and A. Løkketangen (2009). Lasso solution strategies for the vehicle routing problem with pickups and deliveries. In: *European Journal of Operational Research* 192 (3), pp. 755–766.

International Energy Agency (2018). *CO2 Emissions from fuel combustion 2018: overview*. URL: https://webstore.iea.org/download/direct/1082?filename=co2_emissions_from_fuel_combustion_2018_overview.pdf (visited on 10/08/2018).

Intergovernmental Panel on Climate Change (IPCC) (2018). *Summary for Policymakers of IPCC Special Report on Global Warming of 1.5°C approved by governments*. URL: www.ipcc.ch/news_and_events/pr_181008_P48_spm.shtml (visited on 10/08/2018).

Irnich, S., P. Toth, and D. Vigo (2014). Chapter 1: The Family of Vehicle Routing Problems. In: *Vehicle Routing: Problems, Methods, and Applications*. Ed. by P. Toth and D. Vigo. SIAM, pp. 1–33.

Kalayci, C. B. and C. Kaya (2016). An ant colony system empowered variable neighborhood search algorithm for the vehicle routing problem with simultaneous pickup and delivery. In: *Expert Systems with Applications* 66, pp. 163–175.

Kim, B.-I., S. Kim, and S. Sahoo (2006). Waste collection vehicle routing problem with time windows. In: *Computers & Operations Research* 33, pp. 3624–3642.

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

Knupfer, S. M., R. Hensley, P. Hertzke, and P. Schaufuss (2017). *Electrifying insights: how automakers can drive electrified vehicle sales and profitability*. URL: https://www.mckinsey.com/industries/automotive-and-assembly/our-insights/electrifying-insights-how-automakers-can-drive-electrified-vehicle-sales-and-profitability (visited on 10/06/2018).

Koç, Ç., T. Bektaş, O. Jabali, and G. Laporte (2016). The impact of depot location, fleet composition and routing on emissions in city logistics. In: *Transportation Research Part B: Methodological* 84, pp. 81–102.

Koch, H., A. Bortfeldt, and G. Wäscher (2018). A hybrid algorithm for the vehicle routing problem with backhauls, time windows and three-dimensional loading constraints. In: *OR Spectrum*.

Kruskal, J. B. (1956). On the shortest spanning subtree of a graph and the traveling salesman problem. In: *Proceedings of the American Mathematical Society* 7.

Laguna, M. and R. Marti (1999). GRASP and path relinking for 2-layer straight line crossing minimization. In: *INFORMS Journal on Computing* 11 (1), pp. 44–52.

Laporte, G., S. Ropke, and T. Vidal (2014). Chapter 4: Heuristics for the Vehicle Routing Problem. In: *Vehicle Routing: Problems, Methods, and Applications*. Ed. by P. Toth and D. Vigo. SIAM, pp. 87–116.

Liao, C.-S., S.-H. Lu, and Z.-J. M. Shen (2016). The electric vehicle touring problem. In: *Transportation Research Part B: Methodological* 86, pp. 163–180.

Lin, S. (1965). Computer solutions to the traveling-salesman problem. In: *Bell System Technical Journal* 44 (10), pp. 2245–2269.

Lopes, R. B., C. Ferreira, B. S. Santos, and S. Barreto (2013). A taxonomical analysis, current methods and objectives on location-routing problems. In: *International Transactions in Operational Research* 20 (6), pp. 795–822.

Mak, H.-Y., Y. Rong, and Z.-J. M. Shen (2014). Infrastructure planning for electric vehicles with battery swapping. In: *Management Science* 59 (7), pp. 1557–1575.

Markov, I., S. Varone, and M. Bierlaire (2016). Integrating a heterogeneous fixed fleet and a flexible assignment of destination depots in the waste collection VRP with intermediate facilities. In: *Transportation Research Part B: Methodological* 84, pp. 256–273.

Masson, R., F. Lehuédé, and O. Péton (2012). An adaptive large neighborhood search for the pickup and delivery problem with transfers. In: *Transportation Science*, pp. 1–12.

– (2013). Efficient feasibility testing for request insertion in the pickup and delivery problem with transfers. In: *Operations Research Letters* 41 (3), pp. 211–215.

McDonough, W. and M. Braungart (2010). Cradle to cradle: Remaking the way we make things. North point press.

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

Metropolis, N., A. W. Rosenbluth, M. N. Rosenbluth, A. H. Teller, and E. Teller (1953). Equation of state calculations by fast computing machines. In: *The Journal of Chemical Physics* 21 (6), pp. 1087–1092.

Min, H., H. J. Ko, and C. S. Ko (2006). A genetic algorithm approach to developing the multi-echelon reverse logistics network for product returns. In: *Omega* 34 (1), pp. 56–69.

MirHassani, S. A. and R. Ebrazi (2013). A flexible reformulation of the refueling station location problem. In: *Transportation Science* 47 (4), pp. 617–628.

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

Montané, F. A. T. and R. D. Galvão (2006). A tabu search algorithm for the vehicle routing problem with simultaneous pick-up and delivery service. In: *Computers & Operations Research* 33 (3), pp. 595–619.

Muter, I., J.-F. Cordeau, and G. Laporte (2014). A branch-and-price algorithm for the multidepot vehicle routing problem with interdepot routes. In: *Transportation Science* 48 (3), pp. 425–441.

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.

Nagy, G., N. A. Wassan, M. G. Speranza, and C. Archetti (2015). The vehicle routing problem with divisible deliveries and pickups. In: *Transportation Science* 49 (2), pp. 271–294.

Nagy, G., N. A. Wassan, and S. Salhi (2013). The vehicle routing problem with restricted mixing of deliveries and pickups. In: *Journal of Scheduling* 16 (2), pp. 199–213.

Nguyen, V.-P., C. Prins, and C. Prodhon (2012). Solving the two-echelon location routing problem by a GRASP reinforced by a learning process and path relinking. In: *European Journal of Operational Research* 216 (1), pp. 113–126.

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

Nikolaev, A. G. and S. H. Jacobson (2010). Simulated annealing. In: *Handbook of Metaheuristics*. Ed. by M. Gendreau and J.-Y. Potvin. Boston, MA: Springer US, pp. 1–39.

Or, I. (1976). "Traveling salesman-type problems and their relation to the logistics of regional blood banking". PhD thesis. Department of Industrial Engineering and Management Sciences, Northwestern University, Evanston.

Osman, I. H. and G. Laporte (1996). Metaheuristics: A bibliography. In: *Annals of Operations Research* 63 (5), pp. 511–623.

Perrier, N., A. Langevin, and J. F. Campbell (2007). A survey of models and algorithms for winter road maintenance. Part IV: Vehicle routing and fleet sizing for plowing and snow disposal. In: *Computers & Operations Research* 34 (1), pp. 258–294.

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

– (2010). Large neighborhood search. In: *Handbook of Metaheuristics*. Ed. by M. Gendreau and J.-Y. Potvin. Boston, MA: Springer US, pp. 399–419.

Polat, O. (2017). A parallel variable neighborhood search for the vehicle routing problem with divisible deliveries and pickups. In: *Computers & Operations Research* 85, pp. 71–86.

Polat, O., C. B. Kalayci, O. Kulak, and H.-O. Günther (2015). A perturbation based variable neighborhood search heuristic for solving the vehicle routing problem with simultaneous pickup and delivery with time limit. In: *European Journal of Operational Research* 242 (2), pp. 369–382.

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

Prescott-Gagnon, E., G. Desaulniers, and L.-M. Rousseau (2014). Heuristics for an oil delivery vehicle routing problem. In: *Flexible Services and Manufacturing Journal* 26 (4), pp. 516–539.

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

Raviv, T. and M. Kaspi (2012). The locomotive fleet fueling problem. In: *Operations Research Letters* 40 (1), pp. 39–45.

Resende, M. G., C. C. Ribeiro, F. Glover, and R. Martí (2010). Scatter search and path-relinking: fundamentals, advances, and applications. In: *Handbook of Metaheuristics*. Ed. by M. Gendreau and J.-Y. Potvin. Boston, MA: Springer US, pp. 87–107.

Rinaldi, G. and L. Yarrow (1985). "Optimizing a 48-city traveling salesman problem: a case study in combinatorial problem solving". PhD thesis. New York University, Graduate School of Business Administration.

Rivers, C. (2002). "Coordination in vehicle routing". PhD thesis. Massey University, Palmerston North, New Zealand.

Roberti, R. and M. Wen (2016). The electric traveling salesman problem with time windows. In: *Transportation Research Part E: Logistics and Transportation Review* 89, pp. 32–52.

Rochat, Y. and E. 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.

Salazar-Aguilar, M. A., A. Langevin, and G. Laporte (2013). The synchronized arc and node routing problem: application to road marking. In: *Computers & Operations Research* 40 (7), pp. 1708–1715.

Salhi, S. and G. Nagy (1999). A cluster insertion heuristic for single and multiple depot vehicle routing problems with backhauling. In: *The Journal of the Operational Research Society* 50 (10), pp. 1034–1042.

Savelsbergh, M. W. P. (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., 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*. Technical Report. Glasgow, Scotland: PES Group, Department of Computer Science, University of Strathclyde.

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. Springer Berlin Heidelberg, pp. 417–431.

Smokers, R., L. Tavasszy, M. Chen, and E. Guis (2014). Chapter 1: Options for Competitive and Sustainable Logistics. In: *Sustainable Logistics*. Ed. by C. Macharis, S. Melo, J. Woxenius, and T. V. Lier. Emerald Group Publishing Limited, pp. 1–30.

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

Srivastava, S. K. (2008). Network design for reverse logistics. In: *Omega* 36 (4). Special Issue on Logistics: New Perspectives and Challenges, pp. 535–548.

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.

Stock, J. R. (1992). Reverse logistics: white paper. In: *Council of Logistics Management*.

Subramanian, A., L. M. A. Drummond, C. Bentes, L. S. Ochi, and R. Farias (2010). A parallel heuristic for the vehicle routing problem with simultaneous pickup and delivery. In: *Computers & Operations Research* 37 (11), pp. 1899–1911.

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.

Tarantilis, C. D., E. E. Zachariadis, and C. T. Kiranoudis (2008). A hybrid guided local search for the vehicle-routing problem with intermediate replenishment facilities. In: *INFORMS Journal on Computing* 20 (1), pp. 154–168.

Thompson, P. M. and J. B. Orlin (1989). *Theory of cyclic transfers*. Working Paper. Cambridge, USA: MIT, Operations Research Center.

Todosijević, R., M. Mladenović, S. Hanafi, N. Mladenović, and I. Crévits (2016). Adaptive general variable neighborhood search heuristics for solving the unit commitment problem. In: *International Journal of Electrical Power & Energy Systems* 78, pp. 873–883.

Toth, P. and D. Vigo (1999). A heuristic algorithm for the symmetric and asymmetric vehicle routing problems with backhauls. In: *European Journal of Operational Research* 113 (3), pp. 528–543.

United Nations (UN) (2018). *Goal 12: Ensure sustainable consumption and production patterns.* URL: https://www.un.org/sustainabledevelopment/sustainable-consumption-production/ (visited on 10/08/2018).

Vidal, T., T. G. Crainic, M. Gendreau, N. Lahrichi, and W. Rei (2012). A hybrid genetic algorithm for multidepot and periodic vehicle routing problems. In: *Operations Research* 60 (3), pp. 611–624.

Vidal, T., T. G. Crainic, M. Gendreau, and C. Prins (2014a). A unified solution framework for multi-attribute vehicle routing problems. In: *European Journal of Operational Research* 234 (3), pp. 658–673.

Vidal, T., T. Crainic, M. Gendreau, and C. Prins (2014b). *Electronic Companion of: A Unified Solution Framework for Multi-Attribute Vehicle Routing Problems.* Tech. rep.

Vigo, D. and P. Toth, eds. (2014). Vehicle Routing. Philadelphia, PA: Society for Industrial and Applied Mathematics.

Villegas, J. G., C. Prins, C. Prodhon, A. L. Medaglia, and N. Velasco (2011). A GRASP with evolutionary path relinking for the truck and trailer routing problem. In: *Computers & Operations Research* 38 (9), pp. 1319–1334.

Wang, C., D. Mu, F. Zhao, and J. W. Sutherland (2015). A parallel simulated annealing method for the vehicle routing problem with simultaneous pickup-delivery and time windows. In: *Computers & Industrial Engineering* 83 (Complete), pp. 111–122.

Wang, H.-F. and Y.-Y. Chen (2012). A genetic algorithm for the simultaneous delivery and pickup problems with time window. In: *Computers & Industrial Engineering* 62 (1), pp. 84–95.

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.

Wen, M., G. Laporte, O. B. G. Madsen, A. V. Norrelund, and A. Olsen (2014). Locating replenishment stations for electric vehicles. In: *Journal of the Operational Research Society* 65 (10), pp. 1555–1561.

Yang, J. and H. Sun (2015). Battery swap station location-routing problem with capacitated electric vehicles. In: *Computers & Operations Research* 55, pp. 217–232.

Zachariadis, E. E., C. D. Tarantilis, and C. T. Kiranoudis (2010). An adaptive memory methodology for the vehicle routing problem with simultaneous pick-ups and deliveries. In: *European Journal of Operational Research* 202 (2), pp. 401–411.

– (2016). The vehicle routing problem with simultaneous pick-ups and deliveries and two-dimensional loading constraints. In: *European Journal of Operational Research* 251 (2), pp. 369–386.

Zhou, L., R. Baldacci, D. Vigo, and X. Wang (2018). A multi-depot two-echelon vehicle routing problem with delivery options arising in the last mile distribution. In: *European Journal of Operational Research* 265 (2), pp. 765–778.

Appendices

Appendix A

Detailed Results on the Large VRPTWMD Instances

In Table A.1, we report our results on the large-sized VRPTWMD instances. For each instance, we report the number of employed SVs (m^S), the number of employed DVs (m^D), the number of performed transfers (n^t), the best solution quality of ten runs (f^b), the average solution quality of ten runs (f^a), and the average run-time (t^a) in minutes.

	ALNS-PR							ALNS-PR					
Instance	m^S	m^D	n^t	f^b	f^a	t^a(min)	Instance	m^S	m^D	n^t	f^b	f^a	t^a(min)
VRPTWMD-F-c101-100	2	11	10	2733.08	2781.55	16.51	VRPTWMD-L-c101-100	0	14	0	2457.57	2494.88	3.46
VRPTWMD-F-c102-100	2	10	9	2599.14	2674.51	17.61	VRPTWMD-L-c102-100	0	14	0	2421.27	2455.88	8.57
VRPTWMD-F-c103-100	2	10	9	2453.50	2481.25	20.96	VRPTWMD-L-c103-100	1	11	5	2306.72	2358.41	14.90
VRPTWMD-F-c104-100	2	10	9	2431.28	2437.90	17.26	VRPTWMD-L-c104-100	1	11	5	2299.81	2318.12	16.06
VRPTWMD-F-c105-100	2	10	9	2442.08	2467.52	14.12	VRPTWMD-L-c105-100	0	14	0	2478.00	2485.19	3.78
VRPTWMD-F-c106-100	2	10	8	2448.04	2453.12	15.59	VRPTWMD-L-c106-100	1	12	4	2458.14	2466.67	6.21
VRPTWMD-F-c107-100	2	10	9	2436.08	2445.15	13.61	VRPTWMD-L-c107-100	1	11	6	2353.09	2401.72	12.11
VRPTWMD-F-c108-100	2	10	9	2437.54	2454.77	18.77	VRPTWMD-L-c108-100	1	11	6	2346.27	2422.21	12.08
VRPTWMD-F-c109-100	2	10	9	2420.61	2427.12	18.38	VRPTWMD-L-c109-100	1	11	5	2302.15	2316.63	14.21
VRPTWMD-F-c201-100	1	3	3	1716.20	1716.20	7.35	VRPTWMD-L-c201-100	0	5	0	1686.39	1701.56	9.99
VRPTWMD-F-c202-100	1	3	3	1716.20	1716.20	8.30	VRPTWMD-L-c202-100	0	5	0	1679.62	1693.40	10.90
VRPTWMD-F-c203-100	1	3	3	1707.04	1707.04	9.06	VRPTWMD-L-c203-100	0	5	0	1670.82	1691.77	13.41
VRPTWMD-F-c204-100	1	3	3	1703.70	1708.85	12.65	VRPTWMD-L-c204-100	0	5	0	1665.73	1691.27	15.63
VRPTWMD-F-c205-100	1	3	3	1715.81	1715.81	7.27	VRPTWMD-L-c205-100	0	5	0	1683.43	1697.16	10.70
VRPTWMD-F-c206-100	1	3	3	1715.64	1715.64	8.94	VRPTWMD-L-c206-100	0	5	0	1680.33	1694.65	11.92
VRPTWMD-F-c207-100	1	3	3	1700.96	1700.96	7.99	VRPTWMD-L-c207-100	0	5	0	1679.30	1695.85	13.09
VRPTWMD-F-c208-100	1	3	3	1710.91	1713.96	10.89	VRPTWMD-L-c208-100	0	5	0	1683.04	1709.52	11.49
VRPTWMD-F-r101-100	2	17	11	3662.07	3864.94	28.16	VRPTWMD-L-r101-100	1	22	5	3957.22	4065.35	8.47
VRPTWMD-F-r102-100	2	15	10	3317.31	3484.08	43.55	VRPTWMD-L-r102-100	1	18	5	3372.45	3423.87	17.27
VRPTWMD-F-r103-100	2	12	9	3023.22	3161.02	20.48	VRPTWMD-L-r103-100	1	15	5	3117.50	3141.23	16.12
VRPTWMD-F-r104-100	1	11	7	2703.05	2825.35	15.97	VRPTWMD-L-r104-100	0	13	0	2635.74	2652.47	7.39
VRPTWMD-F-r105-100	2	15	11	3473.73	3599.23	32.23	VRPTWMD-L-r105-100	1	16	4	3297.58	3405.64	5.17
VRPTWMD-F-r106-100	1	14	6	3169.41	3276.13	17.05	VRPTWMD-L-r106-100	1	13	5	3052.44	3158.80	14.53
VRPTWMD-F-r107-100	1	12	6	2956.61	3091.30	14.56	VRPTWMD-L-r107-100	1	11	5	2777.37	2850.42	7.55
VRPTWMD-F-r108-100	1	11	7	2561.93	2580.84	16.94	VRPTWMD-L-r108-100	1	10	4	2434.37	2500.03	7.59
VRPTWMD-F-r109-100	1	13	6	2993.76	3134.54	15.15	VRPTWMD-L-r109-100	0	15	0	2914.72	2954.54	3.27
VRPTWMD-F-r110-100	1	12	6	2945.85	3068.54	11.69	VRPTWMD-L-r110-100	0	14	0	2844.86	2858.59	5.18
VRPTWMD-F-r111-100	1	12	6	2785.95	2887.52	19.70	VRPTWMD-L-r111-100	0	14	0	2686.75	2705.65	5.34
VRPTWMD-F-r112-100	1	11	6	2565.59	2602.94	19.60	VRPTWMD-L-r112-100	0	13	0	2503.01	2514.71	7.33
VRPTWMD-F-r201-100	0	5	0	2678.97	2931.32	8.33	VRPTWMD-L-r201-100	1	3	3	2909.69	2962.59	40.28
VRPTWMD-F-r202-100	0	4	0	2574.24	2586.86	20.99	VRPTWMD-L-r202-100	0	5	0	2955.09	2985.90	26.25
VRPTWMD-F-r203-100	0	4	0	2135.89	2278.45	13.25	VRPTWMD-L-r203-100	0	5	0	2456.58	2479.50	37.45
VRPTWMD-F-r204-100	0	3	0	2001.58	2012.67	32.30	VRPTWMD-L-r204-100	0	3	0	2000.19	2020.29	35.36
VRPTWMD-F-r205-100	0	4	0	2275.01	2387.75	11.35	VRPTWMD-L-r205-100	0	5	0	2620.83	2641.36	30.71
VRPTWMD-F-r206-100	0	4	0	2107.73	2392.88	16.84	VRPTWMD-L-r206-100	1	2	4	2335.66	2429.55	37.93
VRPTWMD-F-r207-100	0	3	0	2152.78	2161.82	17.06	VRPTWMD-L-r207-100	0	3	0	2164.07	2182.43	37.10
VRPTWMD-F-r208-100	0	3	0	1812.87	1827.24	16.30	VRPTWMD-L-r208-100	0	3	0	1814.41	1832.75	58.89
VRPTWMD-F-r209-100	0	4	0	2078.61	2151.37	6.29	VRPTWMD-L-r209-100	0	5	0	2377.01	2408.40	13.26
VRPTWMD-F-r210-100	0	5	0	2460.34	2477.76	10.62	VRPTWMD-L-r210-100	0	5	0	2476.71	2529.02	18.14
VRPTWMD-F-r211-100	0	3	0	2097.00	2122.78	17.25	VRPTWMD-L-r211-100	0	3	0	2122.25	2136.64	18.97
VRPTWMD-F-rc101-100	2	15	8	4242.55	4271.36	8.85	VRPTWMD-L-rc101-100	0	20	0	4461.93	4481.06	1.99
VRPTWMD-F-rc102-100	1	14	5	3666.29	3713.42	5.59	VRPTWMD-L-rc102-100	0	16	0	3674.26	3705.35	2.19
VRPTWMD-F-rc103-100	1	13	5	3269.30	3367.73	6.22	VRPTWMD-L-rc103-100	1	13	6	3370.14	3439.39	5.27
VRPTWMD-F-rc104-100	2	10	11	3122.02	3207.89	23.24	VRPTWMD-L-rc104-100	1	11	6	2945.61	3055.69	4.98
VRPTWMD-F-rc105-100	2	13	9	3938.66	4057.33	13.07	VRPTWMD-L-rc105-100	0	18	0	4018.99	4053.76	2.70
VRPTWMD-F-rc106-100	1	13	5	3445.26	3462.81	5.29	VRPTWMD-L-rc106-100	0	16	0	3623.63	3670.89	2.31
VRPTWMD-F-rc107-100	2	11	9	3320.32	3444.54	14.29	VRPTWMD-L-rc107-100	1	13	5	3302.09	3395.00	6.74
VRPTWMD-F-rc108-100	2	11	9	3210.36	3236.20	16.88	VRPTWMD-L-rc108-100	0	14	0	3016.02	3051.28	4.07
VRPTWMD-F-rc201-100	0	6	0	3341.83	3355.56	4.57	VRPTWMD-L-rc201-100	0	6	0	3374.71	3430.88	13.68
VRPTWMD-F-rc202-100	0	4	0	2927.56	2968.02	7.07	VRPTWMD-L-rc202-100	0	5	0	3364.69	3419.64	19.12
VRPTWMD-F-rc203-100	0	4	0	2319.15	2340.31	3.87	VRPTWMD-L-rc203-100	0	5	0	2679.15	2737.09	24.73
VRPTWMD-F-rc204-100	0	5	0	2144.81	2155.55	18.57	VRPTWMD-L-rc204-100	1	2	3	2017.89	2113.53	32.19
VRPTWMD-F-rc205-100	0	6	0	3063.61	3069.07	3.50	VRPTWMD-L-rc205-100	1	3	4	3038.58	3103.55	20.03
VRPTWMD-F-rc206-100	0	5	0	2976.26	3003.37	6.03	VRPTWMD-L-rc206-100	0	5	0	3006.75	3065.14	14.37
VRPTWMD-F-rc207-100	0	4	0	2388.17	2436.27	6.30	VRPTWMD-L-rc207-100	0	5	0	2747.22	2788.09	25.07
VRPTWMD-F-rc208-100	0	4	0	1876.40	1882.54	4.06	VRPTWMD-L-rc208-100	0	5	0	2175.51	2202.88	15.31
Avg.						**14.26**							**15.05**

Table A.1.: Results of ALNS-PR on the large-sized VRPSPD instances.

Appendix B

Detailed Results on the Benchmark Instances for the VRPSPD Variants from the Literature

In the following, we present the detailed results on the benchmark instances for the VRPSPD variants from the literature structured according to problem type.

VRPSPD Tables B.1, B.2, and B.3 show detailed comparisons of our ALNS-PR with the state-of-the-art approaches for the VRPSPD on the benchmarks Salhi-VRPSPD, Dethloff, and Montané-All (including Montané-Medium), respectively.

We report for each instance, the name, the number of customers (n), and the BKS from the literature. For each method, we provide the percentage gap of the best solution found in several runs to the BKS (Δ^b) and the average computing time (t^a), the time elapsed when the best solution was found (t^b), the average time elapsed when the best solution was found (t^{ab}), or the total time of the best run (t) in seconds (see Section 6.3.4 for details on how to interpret the results of the comparison algorithms). In Table B.3, the percentage gap of the average solution quality to the BKS (Δ^a) is additionally given. Values in bold indicate the best solution quality for each instance. Averages of the gaps to the BKS and the run-times are reported after the detailed results per instance at the end of each table.

VRPSPDTL In Tables B.4 and B.5, we present the detailed results on the VRPSPDTL benchmarks Salhi-VRPSPDTL and Polat-VRPSPDTL, respectively.

In the course of our computational experiments, we were able to obtain two new BKS on the instances of Salhi-VRPSPDTL. In addition to the previously reported measures, we therefore report in Table B.4, the best solutions encountered during our entire testing activities (f) and the corresponding percentage gaps to the previous BKS (Δ) in column $\overline{\text{ALNS-PR}}$.

On set Polat-VRPSPDTL, we improve three out of seven previous BKS based on ten runs. The corresponding absolute solution values are given in column f^b of Table B.5.

VRPMDP Table B.6 contains the detailed results on the VRPMDP benchmark Salhi-VRPMDP.

VRPSPDTW Tables B.7 and B.8 show the detailed results on the VRPSPDTW instance sets Wang-Medium and Wang-Large, respectively.

In addition to the previous measures, we report the number of employed vehicles in the best solution (m) and the absolute deviation of the number of vehicles in the best solution found to the previous best-known vehicle number (Δ^m) for each instance and solution method.

With respect to Table B.8, note that we identified the WMZS solutions on instances RC2_8_1 and RC2_10_1 to be infeasible (indicated by asterisks). The average route lengths of 1789.81 and 2119.78 calculated for both solutions exceed the latest arrival times at the depot given in the corresponding instances of 1573 and 1821, respectively. Therefore, we omit these instances when analyzing the solution quality of both comparison algorithms.

VRPDDP We report our detailed results for the VRPDDP on the benchmarks Nagy-VRPDDP1, Nagy-VRPDDP2, and Nagy-VRPDDP3 in Table B.9 and on the benchmarks Polat-VRPDDP1 and Polat-VRPDDP2 in Tables B.10 and B.11, respectively. We report the same measures as in the previous tables, and additionally for ALNS-PR, the number of customers that have been divided in the best solution found (n^d) for each instance.

VRPRMDP Finally, we provide the detailed results for the VRPRMDP on the benchmarks Nagy-VRPRMDP1, Nagy-VRPRMDP2, and Nagy-VRPRMDP3 for different percentages of unavailable vehicle capacity in case of mixed loads in Tables B.12, B.13, and B.14, respectively.

			ZTK		SDBOF		GKA		SUO		VCGP		PKKG		KK		**ALNS-PR**	
Inst.	n	BKS	Δ^b(%)	t^b(s)	Δ^b(%)	t^a(s)	Δ^b(%)	t^a(s)	Δ^b(%)	t^a(s)	Δ^b(%)	t^a(s)	Δ^b(%)	t(s)	Δ^b(%)	t^a(s)	Δ^b(%)	t^a(s)
CMT1X	50	466.77	0.65	2.10	**0.00**	2.28	**0.00**	1.30	**0.00**	2.08	**0.00**	43.20	**0.00**	16.52	**0.00**	8.50	**0.00**	13.31
CMT1Y	50	466.77	0.65	3.80	**0.00**	2.27	**0.00**	1.40	**0.00**	1.97	**0.00**	42.60	**0.00**	8.26	**0.00**	8.50	**0.00**	12.20
CMT2X	75	684.21	**0.00**	5.40	**0.00**	6.44	**0.00**	35.60	**0.00**	12.79	**0.00**	79.20	**0.00**	44.92	**0.00**	32.50	**0.00**	32.03
CMT2Y	75	684.21	**0.00**	6.80	**0.00**	6.41	**0.00**	36.80	**0.00**	10.83	**0.00**	81.00	**0.00**	46.73	**0.00**	36.50	**0.00**	34.90
CMT3X	100	721.27	**0.00**	11.90	**0.00**	12.10	**0.00**	41.70	**0.00**	17.69	**0.00**	101.40	**0.00**	52.18	**0.00**	45.20	**0.00**	72.55
CMT3Y	100	721.27	**0.00**	11.00	**0.00**	12.28	**0.00**	55.50	**0.00**	17.61	**0.00**	107.40	**0.00**	46.09	**0.00**	40.30	**0.00**	127.43
CMT12X	100	662.22	**0.00**	9.30	**0.00**	10.29	0.11	141.40	**0.00**	9.07	**0.00**	104.40	**0.00**	33.91	**0.00**	38.25	**0.00**	39.97
CMT12Y	100	662.22	**0.00**	4.80	**0.00**	10.76	0.19	105.40	**0.00**	9.34	**0.00**	101.40	**0.00**	33.34	**0.00**	41.50	**0.00**	41.82
CMT11X	120	833.92	**0.00**	21.20	**0.00**	18.87	**0.00**	244.90	1.48	51.82	**0.00**	165.00	**0.00**	33.91	**0.00**	42.45	**0.00**	99.29
CMT11Y	120	833.92	**0.00**	14.40	**0.00**	19.03	**0.00**	368.90	1.48	48.63	**0.00**	159.60	**0.00**	49.64	**0.00**	40.50	**0.00**	98.83
CMT4X	150	852.46	**0.00**	29.60	**0.00**	30.89	0.04	380.20	**0.00**	98.03	**0.00**	249.60	**0.00**	118.97	**0.00**	142.10	**0.00**	187.30
CMT4Y	150	852.46	**0.00**	27.40	**0.00**	31.61	**0.00**	414.60	**0.00**	80.63	**0.00**	237.00	**0.00**	136.37	**0.00**	136.35	**0.00**	216.23
CMT5X	199	1029.25	0.13	62.80	**0.00**	71.50	0.41	500.00	**0.00**	1786.74	**0.00**	479.40	0.13	554.39	0.13	420.15	0.06	407.47
CMT5Y	199	1029.25	0.13	47.70	**0.00**	69.58	0.66	500.00	**0.00**	1726.18	**0.00**	396.00	0.13	287.05	0.13	410.50	**0.00**	448.52
Avg.			**0.11**	**18.44**	**0.00**	**21.74**	**0.10**	**201.98**	**0.21**	**276.67**	**0.00**	**167.66**	**0.02**	**104.45**	**0.02**	**103.09**	**0.00**	**130.85**
Processor type			T5500		Xeon		Xeon		i7		Opteron 250		Core 2 Duo T5750		Xeon E5-2650		i5-6600	
Processor speed			1.66 GHz		2.66 GHz		3.16 GHz		2.93 GHz		2.4 GHz		2.00 GHz		2.00 GHz		3.30 GHz	
Passmark score			584		1015		1345		1297		649		678		1302		2098	
t^c(s)			**5.13 × ?**		**10.52 × 256 × 50**		**129.49 × 10**		**171.04 × 10**		**51.86 × 10**		**33.75 × 10**		**63.98 × 10**		**130.85 × 10**	

Table B.1.: Detailed results of ALNS-PR and the state-of-the-art heuristics for the VRPSPD on the Salhi-VRPSPD benchmark.

			ZTK		SDBOF		GKA		KK		P		**ALNS-PR**	
Inst.	n	BKS	Δ^b(%)	t^b(s)	Δ^b(%)	t^a(s)	Δ^b(%)	t^a(s)	Δ^b(%)	t^a(s)	Δ^b(%)	t^a(s)	Δ^b(%)	t^a(s)
SCA3-0	50	635.62	**0.00**	2.50	**0.00**	2.31	**0.00**	4.90	**0.00**	4.77	**0.00**	4.77	**0.00**	19.37
SCA3-1	50	697.84	**0.00**	2.50	**0.00**	2.28	**0.00**	0.80	**0.00**	5.24	**0.00**	5.76	**0.00**	14.59
SCA3-2	50	659.34	**0.00**	2.90	**0.00**	2.14	**0.00**	0.40	**0.00**	7.47	**0.00**	8.22	**0.00**	22.38
SCA3-3	50	680.04	**0.00**	2.30	**0.00**	2.49	**0.00**	1.00	**0.00**	5.20	**0.00**	6.24	**0.00**	11.74
SCA3-4	50	690.50	**0.00**	2.90	**0.00**	2.18	**0.00**	0.30	**0.00**	4.96	**0.00**	3.97	**0.00**	11.95
SCA3-5	50	659.90	**0.00**	3.00	**0.00**	2.23	**0.00**	2.00	**0.00**	5.18	**0.00**	5.70	**0.00**	16.41
SCA3-6	50	651.09	**0.00**	3.10	**0.00**	2.51	**0.00**	0.80	**0.00**	4.68	**0.00**	5.15	**0.00**	17.31
SCA3-7	50	659.17	**0.00**	2.80	**0.00**	2.49	**0.00**	1.60	**0.00**	6.06	**0.00**	6.67	**0.00**	12.38
SCA3-8	50	719.47	**0.00**	3.50	**0.00**	2.26	**0.00**	0.50	**0.00**	4.51	**0.00**	4.96	**0.00**	12.73
SCA3-9	50	681.00	**0.00**	4.70	**0.00**	1.90	**0.00**	0.80	**0.00**	7.08	**0.00**	8.50	**0.00**	15.45
SCA8-0	50	961.50	**0.00**	2.70	**0.00**	3.37	**0.00**	4.80	**0.00**	5.33	**0.00**	5.33	**0.00**	11.69
SCA8-1	50	1049.65	**0.00**	3.80	**0.00**	2.89	**0.00**	6.80	**0.00**	5.62	**0.00**	6.74	**0.00**	7.33
SCA8-2	50	1039.64	**0.00**	3.90	**0.00**	2.38	**0.00**	10.20	**0.00**	6.05	**0.00**	5.45	**0.00**	9.42
SCA8-3	50	983.34	**0.00**	2.60	**0.00**	2.98	**0.00**	13.00	**0.00**	8.39	**0.00**	8.39	**0.00**	9.24
SCA8-4	50	1065.49	**0.00**	2.40	**0.00**	2.81	**0.00**	3.00	**0.00**	6.07	**0.00**	6.07	**0.00**	7.96
SCA8-5	50	1027.08	**0.00**	3.40	**0.00**	3.31	**0.00**	4.10	**0.00**	6.96	**0.00**	5.57	**0.00**	13.51
SCA8-6	50	971.82	**0.00**	2.70	**0.00**	3.51	**0.00**	1.60	**0.00**	7.76	**0.00**	6.98	**0.00**	10.09
SCA8-7	50	1051.28	**0.00**	5.10	**0.00**	3.12	**0.00**	3.40	**0.00**	8.14	**0.00**	9.77	**0.00**	10.38
SCA8-8	50	1071.18	**0.00**	3.60	**0.00**	2.92	**0.00**	0.80	**0.00**	7.06	**0.00**	7.06	**0.00**	10.21
SCA8-9	50	1060.50	**0.00**	4.80	**0.00**	2.18	**0.00**	7.30	**0.00**	5.29	**0.00**	5.82	**0.00**	9.09
CON3-0	50	616.52	**0.00**	4.70	**0.00**	3.12	**0.00**	2.10	**0.00**	6.80	**0.00**	6.12	**0.00**	8.21
CON3-1	50	554.47	**0.00**	2.20	**0.00**	2.83	**0.00**	1.30	**0.00**	5.01	**0.00**	4.01	**0.00**	14.49
CON3-2	50	518.00	**0.00**	3.10	**0.00**	2.77	**0.00**	1.30	**0.00**	7.55	**0.00**	9.06	**0.00**	10.42
CON3-3	50	591.19	**0.00**	3.20	**0.00**	2.34	**0.00**	0.50	**0.00**	5.75	**0.00**	6.90	**0.00**	22.12
CON3-4	50	588.79	**0.00**	2.30	**0.00**	2.63	**0.00**	3.20	**0.00**	3.90	**0.00**	3.12	**0.00**	9.67
CON3-5	50	563.70	**0.00**	3.70	**0.00**	2.69	**0.00**	0.40	**0.00**	6.86	**0.00**	6.17	**0.00**	8.95
CON3-6	50	499.05	**0.00**	3.70	**0.00**	2.75	**0.00**	2.30	**0.00**	8.54	**0.00**	9.39	**0.00**	10.77
CON3-7	50	576.48	**0.00**	1.90	**0.00**	2.75	**0.00**	2.60	**0.00**	4.26	**0.00**	4.69	**0.00**	11.05
CON3-8	50	523.05	**0.00**	3.80	**0.00**	2.46	**0.00**	1.00	**0.00**	3.89	**0.00**	3.89	**0.00**	8.03
CON3-9	50	578.25	**0.00**	2.20	**0.00**	3.37	**0.00**	2.90	**0.00**	6.33	**0.00**	5.70	**0.00**	8.02
CON8-0	50	857.17	**0.00**	4.40	**0.00**	3.65	**0.00**	5.20	**0.00**	5.40	**0.00**	4.86	**0.00**	5.21
CON8-1	50	740.85	**0.00**	3.30	**0.00**	3.02	**0.00**	2.90	**0.00**	8.46	**0.00**	6.77	**0.00**	5.69
CON8-2	50	712.89	**0.00**	2.70	**0.00**	3.08	**0.00**	2.10	**0.00**	4.79	**0.00**	3.83	**0.00**	9.50
CON8-3	50	811.07	**0.00**	2.80	**0.00**	3.99	**0.00**	2.80	**0.00**	7.21	**0.00**	6.49	**0.00**	4.69
CON8-4	50	772.25	**0.00**	2.80	**0.00**	3.69	**0.00**	3.60	**0.00**	6.70	**0.00**	6.70	**0.00**	7.40
CON8-5	50	754.88	**0.00**	5.70	**0.00**	4.18	**0.00**	3.40	**0.00**	5.74	**0.00**	6.31	**0.00**	5.84
CON8-6	50	678.92	**0.00**	3.40	**0.00**	4.09	**0.00**	7.90	**0.00**	4.36	**0.00**	4.80	**0.00**	5.33
CON8-7	50	811.96	**0.00**	2.50	**0.00**	4.03	**0.00**	3.00	**0.00**	8.38	**0.00**	9.22	**0.00**	4.12
CON8-8	50	767.53	**0.00**	3.20	**0.00**	3.42	**0.00**	3.20	**0.00**	6.16	**0.00**	4.93	**0.00**	4.98
CON8-9	50	809.00	**0.00**	3.80	**0.00**	3.48	**0.00**	2.40	**0.00**	7.19	**0.00**	6.47	**0.00**	4.81
Avg.			**0.00**	**3.27**	**0.00**	**2.92**	**0.00**	**3.06**	**0.00**	**6.13**	**0.00**	**6.16**	**0.00**	**10.56**
Processor type			T5500		Xeon		Xeon		Xeon E5-2650		Xeon E5420		i5-6600	
Processor speed			1.66 GHz		2.66 GHz		3.16 GHz		2.00 GHz		2.50 GHz		3.30 GHz	
Passmark score			584		1015		1345		1302		1073		2098	
t^c(s)			**0.91** × **?**		**1.41** × **256** × **50**		**1.96** × **10**		**3.82** × **10**		**3.15** × **6** × **10**		**10.56** × **10**	

Table B.2.: Detailed results of ALNS-PR and the state-of-the-art heuristics for the VRPSPD on the Dethloff benchmark.

			ZTK		SDBOF			SUO			VCGP			P		**ALNS-PR**		
Inst.	n	BKS	Δ^b(%)	t^b(s)	Δ^b(%)	Δ^a(%)	t^a(s)	Δ^b(%)	Δ^a(%)	t^a(s)	Δ^b(%)	Δ^a(%)	t^a(s)	Δ^b(%)	t^{ab}(s)	Δ^b(%)	Δ^a(%)	t^a(s)
r101	100	1009.95	**0.00**	5.80	**0.00**	0.06	15.81	**0.00**	0.01	65.42	**0.00**	0.16	73.80	**0.00**	33.86	**0.00**	0.49	46.61
r201	100	666.20	**0.00**	7.90	**0.00**	**0.00**	15.95	**0.00**	**0.00**	15.71	**0.00**	**0.00**	143.40	**0.00**	30.56	**0.00**	0.33	134.67
c101	100	1220.18	0.07	8.60	**0.00**	0.04	10.39	**0.00**	0.02	12.93	0.07	0.07	66.00	**0.00**	15.63	0.07	0.25	52.62
c201	100	662.07	**0.00**	4.30	**0.00**	**0.00**	8.83	**0.00**	**0.00**	9.77	**0.00**	**0.00**	97.80	**0.00**	15.55	**0.00**	**0.00**	58.92
rc101	100	1059.32	**0.00**	14.10	**0.00**	**0.00**	11.07	**0.00**	**0.00**	16.89	**0.00**	**0.00**	72.60	**0.00**	18.25	**0.00**	**0.00**	40.09
rc201	100	672.92	**0.00**	10.60	**0.00**	**0.00**	7.28	**0.00**	**0.00**	11.42	**0.00**	**0.00**	119.40	**0.00**	18.86	**0.00**	0.03	85.07
R1_2_1	200	3353.80	0.67	45.10	0.19	0.48	66.21	**0.00**	0.04	1142.05	0.05	0.32	406.20	0.19	370.83	0.17	0.59	243.15
R2_2_1	200	1665.58	**0.00**	49.40	**0.00**	**0.00**	45.30	**0.00**	**0.00**	1425.88	**0.00**	**0.00**	537.00	**0.00**	473.59	**0.00**	0.29	533.61
C1_2_1	200	3628.51	0.42	41.00	0.04	0.20	87.38	**0.00**	0.22	2874.50	0.25	0.29	435.00	0.24	917.43	0.10	0.37	263.00
C2_2_1	200	1726.59	**0.00**	56.40	**0.00**	**0.00**	65.01	**0.00**	**0.00**	1365.93	**0.00**	**0.00**	331.20	**0.00**	398.38	**0.00**	0.05	437.95
RC1_2_1	200	3303.70	0.60	51.30	0.07	0.42	71.71	**0.00**	0.09	1293.53	0.02	0.35	486.00	0.07	485.05	**0.00**	0.19	267.61
RC2_2_1	200	1560.00	**0.00**	28.10	**0.00**	**0.00**	44.71	**0.00**	**0.00**	1361.87	**0.00**	**0.00**	441.00	**0.00**	393.51	**0.00**	0.13	532.29
Avg.			**0.15**	**26.88**	**0.02**	**0.10**	**37.47**	**0.00**	**0.03**	**799.66**	**0.03**	**0.10**	**267.45**	**0.04**	**264.29**	**0.03**	**0.23**	**224.63**
Processor type			T5500		Xeon			i7			Opteron 250			Xeon E5420		i5-6600		
Processor speed			1.66 GHz		2.66 GHz			2.93 GHz			2.40 GHz			2.50 GHz		3.30 GHz		
Passmark score			584		1015			1297			649			1073		2098		
t^c(s)			**7.48 × ?**		**18.13 × 256 × 50**			**494.36 × 10**			**82.73 × 10**			**135.17 × 6 × 10**		**224.63 × 10**		
R1_4_1	400	9519.45	1.81	345.30	1.05	1.34	481.61	**0.00**	0.21	9177.90	0.30	0.78	1471.80			0.38	1.09	2241.57
R2_4_1	400	3546.49	0.73	125.00	0.14	0.31	459.15	**0.00**	0.08	9086.79	**0.00**	0.24	1474.20			0.01	0.48	3066.73
C1_4_1	400	11047.19	1.20	224.80	0.47	0.65	546.22	**0.00**	0.26	8016.83	0.27	0.60	1780.20			0.33	0.64	2456.82
C2_4_1	400	3539.50	0.28	238.20	0.19	0.55	488.56	**0.00**	0.12	10691.30	**0.00**	0.05	1785.60			0.13	0.42	2600.01
RC1_4_1	400	9447.53	2.09	160.70	0.94	1.24	513.38	**0.00**	0.32	10867.10	0.23	0.65	1790.40			0.50	1.19	2542.94
RC2_4_1	400	3403.70	0.59	315.70	**0.00**	0.03	422.61	**0.00**	**0.00**	8326.18	**0.00**	0.01	1446.00			**0.00**	0.19	2166.65
Avg. tot.			**0.47**	**96.24**	**0.17**	**0.30**	**186.73**	**0.00**	**0.08**	**3653.44**	**0.07**	**0.20**	**719.87**			**0.09**	**0.37**	**987.24**
t^c(s)			**26.79 × ?**		**90.34 × 256 × 50**			**2258.59 × 10**			**222.69 × 10**					**987.24 × 10**		

Table B.3.: Detailed results of ALNS-PR and the state-of-the-art heuristics for the VRPSPD on the Montané-All (including Montané-Medium) benchmark.

			SUO			PKKG			KK			**ALNS-PR**			**ALNS-PR**	
Inst.	n	BKS	Δ^b(%)	Δ^a(%)	t^a(s)	Δ^b(%)	Δ^a(%)	t(s)	Δ^b(%)	Δ^a(%)	t^a(s)	Δ^b(%)	Δ^a(%)	t^a(s)	f	Δ(%)
CMT6X	50	555.43	**0.00**	0.35	1.04	**0.00**	**0.00**	47.00	**0.00**	**0.00**	32.50	**0.00**	**0.00**	7.70	**555.43**	0.00
CMT6Y	50	555.43	**0.00**	0.30	1.08	**0.00**	**0.00**	47.30	**0.00**	**0.00**	32.30	**0.00**	**0.00**	8.76	**555.43**	0.00
CMT7X	75	900.12	**0.00**	0.10	4.55	0.12	0.12	70.30	**0.00**	**0.00**	52.55	0.05	0.19	24.78	**900.12**	0.00
CMT7Y	75	900.12	**0.00**	0.11	4.87	0.12	0.12	69.80	**0.00**	**0.00**	56.30	**0.00**	0.08	21.86	**900.12**	0.00
CMT8X	100	865.50	**0.00**	**0.00**	7.36	**0.00**	**0.00**	224.60	**0.00**	**0.00**	120.25	**0.00**	**0.00**	49.24	**865.50**	0.00
CMT8Y	100	865.50	**0.00**	**0.00**	7.74	**0.00**	**0.00**	162.70	**0.00**	**0.00**	127.50	**0.00**	0.08	49.24	**865.50**	0.00
CMT14X	100	821.75	**0.00**	**0.00**	5.42	**0.00**	**0.00**	228.50	**0.00**	**0.00**	85.50	**0.00**	**0.00**	22.29	**821.75**	0.00
CMT14Y	100	821.75	**0.00**	**0.00**	5.48	**0.00**	**0.00**	204.60	**0.00**	**0.00**	83.70	**0.00**	**0.00**	25.64	**821.75**	0.00
CMT13X	120	1542.86	0.00	0.04	68.72	0.00	0.02	332.70	0.00	0.00	260.25	0.01	0.11	78.76	**1541.14**	-0.11
CMT13Y	120	1542.86	0.00	0.10	73.49	0.00	0.00	375.30	0.00	0.00	250.30	0.00	0.07	89.00	**1541.14**	-0.11
CMT9X	150	1160.68	**0.00**	0.09	64.43	0.05	0.09	483.40	**0.00**	0.02	360.20	**0.00**	0.23	116.35	**1160.68**	0.00
CMT9Y	150	1160.68	**0.00**	0.16	80.86	0.05	0.08	477.10	**0.00**	0.02	350.80	**0.00**	0.05	186.37	**1160.68**	0.00
CMT10X	199	1373.40	**0.00**	0.42	552.81	1.08	1.12	1168.80	**0.00**	0.17	880.50	**0.00**	1.04	324.94	**1373.40**	0.00
CMT10Y	199	1373.40	**0.00**	0.26	547.39	1.08	1.12	1112.10	**0.00**	0.17	860.25	**0.00**	1.11	243.13	**1373.40**	0.00
Avg.			**0.00**	**0.14**	**101.80**	**0.18**	**0.19**	**357.44**	**0.00**	**0.03**	**253.78**	**0.00**	**0.21**	**89.15**		**-0.02**
Processor type			i7			Core 2 Duo T5750			Xeon E5-2650			i5-6600				
Processor speed			2.93 GHz			2.00 GHz			2.00 GHz			3.30 GHz				
Passmark score			1297			678			1302			2098				
t^c(s)			**62.93 × 10**			**115.51 × 10**			**157.49 × 10**			**89.15 × 10**				

Table B.4.: Detailed results of ALNS-PR and the state-of-the-art heuristics for the VRPSPDTL on the Salhi-VRPSPDTL benchmark.

			PKKG			**ALNS-PR**			
Inst.	n	BKS	Δ^b(%)	Δ^a(%)	t^a(s)	f^b	Δ^b(%)	Δ^a(%)	t^a(s)
CE51-5	50	**570**	**0.00**	**0.00**	98.39	**570**	0.00	**0.00**	1.69
CE76-7	75	735	0.00	0.00	250.75	**728**	-0.95	0.11	6.17
CE76-8	75	778	0.00	0.35	507.78	**777**	-0.13	-0.10	4.81
CE76-10	75	**878**	**0.00**	0.17	344.06	**878**	0.00	0.18	4.10
CE76-14	75	**1091**	**0.00**	0.11	321.56	**1091**	0.00	0.10	4.46
CE101-8	100	956	0.00	0.05	416.56	**955**	-0.10	0.25	10.03
CE101-14	100	**1175**	**0.00**	0.30	10264	1177	0.17	0.56	13.71
Avg.			**0.00**	**0.14**	**323.18**		**-0.15**	**0.16**	**5.21**
Processor type			Core 2 Duo T5750			i5-6600			
Processor speed			2.00 GHz			3.30 GHz			
Passmark score			678			2098			
t^c(s)			**104.44 × 10**			**5.21 × 10**			

Table B.5.: Detailed results of ALNS-PR in comparison to PKKG on the VRPSPDTL benchmark Polat-VRP-SPDTL.

			SUO			VCGP			**ALNS-PR**			
Inst.	n	BKS	Δ^b(%)	Δ^a(%)	t^a(s)	Δ^b(%)	Δ^a(%)	t^a(s)	f^b	Δ^b(%)	Δ^a(%)	t^a(s)
CMT1H	50	**465.02**	**0.00**	**0.00**	2.07	**0.00**	**0.00**	41.40	465.02	**0.00**	0.01	16.89
CMT1Q	50	**489.74**	**0.00**	**0.00**	1.52	**0.00**	**0.00**	33.00	489.74	**0.00**	0.28	16.99
CMT1T	50	**520.06**	**0.00**	**0.00**	1.60	**0.00**	**0.00**	31.20	520.06	**0.00**	**0.00**	18.06
CMT2H	75	**662.63**	**0.00**	**0.00**	5.16	**0.00**	**0.00**	61.20	662.63	**0.00**	**0.00**	32.78
CMT2Q	75	**731.26**	**0.00**	0.02	8.03	**0.00**	0.03	87.60	732.67	0.19	0.26	19.12
CMT2T	75	**782.77**	**0.00**	**0.00**	7.56	**0.00**	**0.00**	49.80	782.77	**0.00**	0.11	27.84
CMT3H	100	**700.94**	**0.00**	**0.00**	17.65	**0.00**	**0.00**	116.40	700.94	**0.00**	0.16	103.94
CMT3Q	100	**747.15**	**0.00**	**0.00**	9.70	**0.00**	**0.00**	79.80	747.15	**0.00**	**0.00**	77.42
CMT3T	100	**798.07**	**0.00**	**0.00**	28.76	**0.00**	**0.00**	97.20	798.07	**0.00**	0.33	83.16
CMT12H	100	**629.37**	**0.00**	**0.00**	13.93	**0.00**	**0.00**	89.40	629.37	**0.00**	0.11	66.25
CMT12Q	100	**729.25**	**0.00**	**0.00**	17.37	**0.00**	**0.00**	91.80	729.25	**0.00**	0.17	45.76
CMT12T	100	**787.52**	**0.00**	**0.00**	6.79	**0.00**	**0.00**	57.00	787.52	**0.00**	**0.00**	31.84
CMT11H	120	**818.05**	**0.00**	**0.00**	63.18	**0.00**	**0.00**	271.80	818.05	**0.00**	0.01	195.73
CMT11Q	120	**939.36**	**0.00**	**0.00**	20.35	**0.00**	**0.00**	179.40	939.36	**0.00**	**0.00**	114.33
CMT11T	120	**998.80**	**0.00**	**0.00**	19.91	**0.00**	**0.00**	140.40	998.80	**0.00**	**0.00**	69.97
CMT4H	150	**828.12**	**0.00**	0.42	80.24	**0.00**	0.18	340.20	828.12	**0.00**	0.50	185.30
CMT4Q	150	**915.27**	**0.00**	**0.00**	58.92	**0.00**	0.02	250.80	915.27	**0.00**	0.23	177.80
CMT4T	150	**990.39**	**0.00**	**0.00**	50.42	**0.00**	**0.00**	150.00	990.39	**0.00**	0.25	205.52
CMT5H	199	**978.74**	**0.00**	**0.00**	1531.73	**0.00**	0.19	297.00	978.74	**0.00**	0.41	577.38
CMT5Q	199	**1104.87**	**0.00**	0.08	1627.78	**0.00**	0.19	347.40	1104.87	**0.00**	0.42	493.59
CMT5T	199	**1218.77**	**0.00**	0.12	1802.81	**0.00**	0.14	357.60	1218.77	**0.00**	0.18	331.88
CMT6H	50	**555.43**	**0.00**	0.35	1.08	**0.00**	**0.00**	27.00	555.43	**0.00**	0.06	9.35
CMT6Q	50	**555.43**	**0.00**	0.31	1.08	**0.00**	**0.00**	27.00	555.43	**0.00**	**0.00**	9.37
CMT6T	50	**555.43**	**0.00**	0.22	1.15	**0.00**	**0.00**	27.60	555.43	**0.00**	**0.00**	9.69
CMT7H	75	**900.12**	0.05	0.08	4.47	**0.00**	0.05	61.80	900.12	**0.00**	0.14	33.31
CMT7Q	75	**900.69**	**0.00**	0.21	4.90	**0.00**	0.04	71.40	900.69	**0.00**	0.19	31.05
CMT7T	75	**903.05**	**0.00**	**0.00**	4.77	**0.00**	**0.00**	43.80	903.05	**0.00**	**0.00**	28.97
CMT8H	100	**865.50**	**0.00**	**0.00**	7.78	**0.00**	**0.00**	78.60	865.50	**0.00**	**0.00**	70.57
CMT8Q	100	**865.50**	**0.00**	**0.00**	7.50	**0.00**	**0.00**	79.20	865.50	**0.00**	0.08	69.52
CMT8T	100	**865.54**	**0.00**	**0.00**	7.18	**0.00**	**0.00**	78.60	865.54	**0.00**	0.09	57.46
CMT14H	100	**821.75**	**0.00**	**0.00**	5.37	**0.00**	**0.00**	57.60	821.75	**0.00**	**0.00**	24.33
CMT14Q	100	**821.75**	**0.00**	**0.00**	5.47	**0.00**	**0.00**	58.20	821.75	**0.00**	**0.00**	20.91
CMT14T	100	**826.77**	**0.00**	**0.00**	6.30	**0.00**	**0.00**	69.00	826.77	**0.00**	**0.00**	26.31
CMT13H	120	**1542.86**	**0.00**	0.11	73.82	**0.00**	**0.00**	168.60	1542.86	**0.00**	0.06	107.14
CMT13Q	120	**1542.86**	**0.00**	0.08	69.87	**0.00**	**0.00**	171.00	1542.86	**0.00**	0.10	74.40
CMT13T	120	1542.86	0.00	0.08	73.59	0.00	0.01	171.60	1541.14	**-0.11**	0.11	87.14
CMT9H	150	**1160.68**	**0.00**	0.13	77.95	**0.00**	**0.00**	219.60	1160.68	**0.00**	0.10	186.17
CMT9Q	150	**1161.24**	**0.00**	0.04	80.64	**0.00**	0.01	207.60	1161.24	**0.00**	0.27	182.04
CMT9T	150	**1162.55**	**0.00**	0.16	83.29	**0.00**	**0.00**	220.80	1162.55	**0.00**	0.40	166.36
CMT10H	199	1372.47	0.00	0.35	550.45	0.00	0.66	393.60	1372.20	**-0.02**	0.79	265.66
CMT10Q	199	**1374.18**	**0.00**	0.38	537.66	**0.00**	0.31	474.60	1374.18	**0.00**	0.81	315.90
CMT10T	199	**1381.04**	**0.00**	0.52	501.65	**0.00**	0.71	329.40	1381.04	**0.00**	0.90	258.24
Avg.			**0.00**	**0.09**	**178.13**	**0.00**	**0.06**	**147.79**		**0.00**	**0.18**	**117.27**
Processor type			i7			Opteron 250			i5-6600			
Processor speed			2.93 GHz			2.40 GHz			3.30 GHz			
Passmark score			1297			649			2098			
t^c(s)			**110.12 × 10**			**45.72 × 10**			**117.27 × 10**			

Table B.6.: Detailed results of ALNS-PR in comparison to SUO and VCGP on the VRPMDP benchmark Salhi-VRPMDP.

		BKS		WMZS				**ALNS-PR**			
Inst.	n	m	f	Δ^m	f^b	Δ^b(%)	t(s)	Δ^m	f^b	Δ^b(%)	t^a(s)
cdp101	100	11	992.88	0	992.88	0.00	36	0	**976.04**	-1.70	19.30
cdp102	100	10	955.31	0	955.31	0.00	38	0	**941.49**	-1.45	28.11
cdp103	100	10	897.65	0	958.66	6.80	34	0	**892.98**	-0.52	48.03
cdp104	100	10	878.93	0	944.73	7.49	35	0	**871.40**	-0.86	46.51
cdp105	100	11	983.10	0	989.86	0.69	37	**-1**	1053.12	7.12	15.97
cdp106	100	11	878.29	0	878.29	0.00	37	**-1**	967.71	10.18	17.36
cdp107	100	11	911.90	0	911.90	0.00	41	**-1**	987.64	8.31	18.06
cdp108	100	10	951.24	0	1063.73	11.83	39	0	**932.88**	-1.93	18.22
cdp109	100	10	940.49	0	947.90	0.79	21	0	**910.95**	-3.14	38.45
cdp201	100	3	591.56	0	591.56	0.00	86	0	**591.56**	0.00	24.37
cdp202	100	3	591.56	0	591.56	0.00	91	0	**591.56**	0.00	46.09
cdp203	100	3	591.17	0	591.17	0.00	88	0	**591.17**	0.00	44.19
cdp204	100	3	590.60	0	594.07	0.59	90	0	**590.60**	0.00	51.76
cdp205	100	3	588.88	0	588.88	0.00	90	0	**588.88**	0.00	36.35
cdp206	100	3	588.49	0	588.49	0.00	88	0	**588.49**	0.00	36.39
cdp207	100	3	588.29	0	588.29	0.00	85	0	**588.29**	0.00	39.83
cdp208	100	3	588.32	0	599.32	1.87	83	0	**588.32**	0.00	34.47
rdp101	100	19	1653.53	0	1660.98	0.45	43	0	**1650.80**	-0.17	22.86
rdp102	100	17	1488.04	0	1491.75	0.25	29	0	**1486.12**	-0.13	20.89
rdp103	100	14	1216.16	0	1226.77	0.87	41	**-1**	1297.01	6.65	17.83
rdp104	100	10	1000.65	0	1000.65	0.00	45	0	**984.81**	-1.58	28.40
rdp105	100	14	1399.81	0	1399.81	0.00	45	0	**1377.11**	-1.62	17.58
rdp106	100	12	1275.69	0	1275.69	0.00	37	0	**1252.03**	-1.85	27.54
rdp107	100	11	1082.92	0	1082.92	0.00	35	**-1**	1121.86	3.60	18.79
rdp108	100	10	962.48	0	962.48	0.00	41	**-1**	965.54	0.32	20.60
rdp109	100	12	1160.00	0	1181.92	1.89	46	**-1**	1194.73	2.99	16.08
rdp110	100	11	1106.52	0	1106.52	0.00	45	**-1**	1148.20	3.77	19.13
rdp111	100	11	1065.27	0	1073.62	0.78	41	**-1**	1098.84	3.15	22.15
rdp112	100	10	966.06	0	966.06	0.00	51	**-1**	1010.42	4.59	28.63
rdp201	100	4	1280.44	0	1286.55	0.48	84	0	**1253.23**	-2.12	33.44
rdp202	100	4	1100.92	0	1150.31	4.49	123	**-1**	1191.70	8.25	46.71
rdp203	100	3	950.79	0	997.84	4.95	102	0	**946.28**	-0.47	84.91
rdp204	100	2	848.01	0	848.01	0.00	120	0	**833.09**	-1.76	111.52
rdp205	100	3	1046.06	0	1046.06	0.00	116	0	**994.43**	-4.94	80.35
rdp206	100	3	959.94	0	959.94	0.00	134	0	**913.68**	-4.82	89.88
rdp207	100	2	899.82	0	899.82	0.00	85	0	**890.61**	-1.02	82.51
rdp208	100	2	739.06	0	739.06	0.00	127	0	**726.82**	-1.66	100.25
rdp209	100	3	930.26	0	947.80	1.89	111	0	**909.16**	-2.27	86.59
rdp210	100	3	983.75	0	1005.11	2.17	164	0	**939.37**	-4.51	82.84
rdp211	100	3	812.44	0	812.44	0.00	98	**-1**	904.44	11.32	85.87
rcdp101	100	15	1652.90	0	1659.59	0.40	47	**-1**	1776.58	7.48	10.59
rcdp102	100	13	1522.76	0	1522.76	0.00	41	**-1**	1583.62	4.00	19.19
rcdp103	100	11	1344.62	0	1344.62	0.00	45	0	**1283.52**	-4.54	29.08
rcdp104	100	10	1268.43	0	1268.43	0.00	47	0	**1171.65**	-7.63	22.57
rcdp105	100	14	1581.26	0	1581.54	0.02	46	0	**1548.96**	-2.04	16.95
rcdp106	100	13	1418.16	0	1418.16	0.00	41	**-1**	1392.47	-1.81	20.47
rcdp107	100	11	1360.17	0	1360.17	0.00	35	0	**1255.06**	-7.73	21.22
rcdp108	100	11	1169.57	0	1169.57	0.00	38	**-1**	1198.36	2.46	19.52
rcdp201	100	4	1513.72	0	1513.72	0.00	64	0	**1406.94**	-7.05	26.68
rcdp202	100	4	1211.12	0	1273.26	5.13	72	**-1**	1414.55	16.80	39.94
rcdp203	100	3	1123.58	0	1123.58	0.00	78	0	**1050.64**	-6.49	83.95
rcdp204	100	3	822.02	0	897.14	9.14	80	0	**798.46**	-2.87	94.85
rcdp205	100	4	1371.08	0	1371.08	0.00	62	0	**1297.65**	-5.36	33.58
rcdp206	100	3	1166.88	0	1166.88	0.00	66	0	**1146.32**	-1.76	59.36
rcdp207	100	3	1089.85	0	1089.85	0.00	75	0	**1061.84**	-2.57	79.06
rcdp208	100	3	862.89	0	862.89	0.00	73	0	**828.14**	-4.03	72.78
Avg.						**1.12**	**65.93**			**-2.32**	**42.12**
$\sum$				**0**				**-17**			
Processor type				2 x Xeon E5-2650				i5-6600			
Processor speed				2.00 GHz				3.30 GHz			
Passmark score				1910 (15283/8)				2098			
t^c(s)				**60.02 × 66 × ?**				**42.12 × 10**			

Table B.7.: Detailed results of ALNS-PR in comparison to WMZS and the previous BKS on the VRPSPDTW benchmark Wang-Medium.

		WMZS			**ALNS-PR**				
Inst.	n	m	f^b	t(s)	m	Δ^m	f^b	Δ^b(%)	t^a(s)
C1_2_1	200	21	3169.52	62	**20**	-1	2846.20	-10.20	55.54
C1_4_1	400	42	8135.35	147	**40**	-2	7533.03	-7.40	220.13
C1_6_1	600	69	19720.65	257	**63**	-6	15594.21	-20.92	765.79
C1_8_1	800	88	32801.92	1054	**82**	-6	27035.71	-17.58	1495.17
C1_10_1	1000	110	52328.78	2418	**102**	-8	44764.64	-14.46	2253.40
C2_2_1	200	6	1972.97	112	6	0	**1931.44**	-2.10	194.73
C2_4_1	400	14	5085.08	279	**12**	-2	4144.84	-18.49	776.51
C2_6_1	600	20	9509.15	926	**18**	-2	7830.16	-17.66	1652.55
C2_8_1	800	27	14573.93	3636	**24**	-3	11759.05	-19.31	3022.47
C2_10_1	1000	33	23981.11	6529	**30**	-3	17088.50	-28.74	5009.45
R1_2_1	200	22	5083.39	89	**20**	-2	4849.80	-4.60	113.46
R1_4_1	400	42	12202.62	237	**40**	-2	10671.70	-12.55	537.22
R1_6_1	600	62	25729.28	581	**59**	-3	22306.17	-13.30	1211.12
R1_8_1	800	93	51949.49	1869	**80**	-13	39348.17	-24.26	3082.84
R1_10_1	1000	115	77993.35	4539	**100**	-15	58912.62	-24.46	4651.74
R2_2_1	200	5	4372.17	295	5	0	**4042.67**	-7.54	186.02
R2_4_1	400	9	14119.64	735	9	0	**8952.24**	-36.60	506.62
R2_6_1	600	13	27294.11	2439	13	0	**17459.41**	-36.03	1558.88
R2_8_1	800	19	48611.60	7663	**18**	-1	27270.04	-43.90	3429.95
R2_10_1	1000	22	67441.51	21379	22	0	**42117.48**	-37.55	5604.70
RC1_2_1	200	20	3865.18	81	**19**	-1	3652.18	-5.51	77.05
RC1_4_1	400	40	10036.82	193	**38**	-2	9772.56	-2.63	401.10
RC1_6_1	600	60	20535.26	733	**57**	-3	19679.75	-4.17	946.06
RC1_8_1	800	88	32801.92	1620	**75**	-13	38431.09	17.16	2234.79
RC1_10_1	1000	102	66883.49	3483	**93**	-9	63953.66	-4.38	4325.54
RC2_2_1	200	4	2662.75	216	4	0	**2021.49**	-24.08	338.41
RC2_4_1	400	13	7229.22	791	**12**	-1	6621.94	-8.40	681.44
RC2_6_1	600	20	22837.36	2860	**16**	-4	12693.19	-44.42	2607.91
RC2_8_1	800	22*	39375.78*	6026	60	38	26652.10	-32.31	2827.44
RC2_10_1	1000	29*	61473.68*	13793	75	46	40643.56	-33.88	6460.91
Avg.			**2834.73**					**-16.93**	**1907.63**
$\sum$						**-102**			
Processor type		2 × Xeon E5-2650			i5-6600				
Processor speed		2.00 GHz			3.30 GHz				
Passmark score		1910 (15283/8)			2098				
t^c(s)		**2580.71 × 66 × ?**			**1907.63 × 10**				

Table B.8.: Detailed results of ALNS-PR in comparison to WMZS on the VRPSPDTW benchmark Wang-Large.

		Nagy-VRPDDP1						Nagy-VRPDDP2						Nagy-VRPDDP3					
			ALNS-PR						ALNS-PR						ALNS-PR				
Inst.	n	BKS	m	n^d	f^b	Δ^b(%)	t^a(s)	BKS	m	n^d	f^b	Δ^b(%)	t^a(s)	BKS	m	n^d	f^b	Δ^b(%)	t^a(s)
CMT1X	50	470	3	0	**468**	-0.43	9.16	1138	18	10	**1132**	-0.53	6.26	**1827**	32	25	**1827**	0.00	3.24
CMT1Y	50	**459**	3	0	**459**	0.00	8.35	1058	17	11	**1054**	-0.38	4.80	1841	32	31	**1838**	-0.16	3.85
CMT2X	75	684	6	1	**674**	-1.46	22.01	**1952**	34	17	1968	0.82	12.08	2742	48	38	**2738**	-0.15	8.48
CMT2Y	75	**650**	6	2	651	0.15	19.75	**1820**	30	22	1822	0.11	12.68	2809	47	53	**2799**	-0.36	10.11
CMT3X	100	713	5	1	**707**	-0.84	48.79	1946	28	26	**1938**	-0.41	31.78	3412	57	41	**3399**	-0.38	18.97
CMT3Y	100	705	4	0	**691**	-1.99	51.50	**1805**	27	25	1818	0.72	27.82	3425	58	49	**3415**	-0.29	18.33
CMT4X	150	862	7	1	**844**	-2.09	119.85	2742	43	55	**2714**	-1.02	102.58	4933	87	70	**4905**	-0.57	59.50
CMT4Y	150	831	7	0	**820**	-1.32	104.73	2548	40	36	**2525**	-0.90	76.57	4947	85	87	**4898**	-0.99	51.15
CMT5X	199	1062	10	16	**1015**	-4.43	203.04	3672	61	67	**3624**	-1.31	146.36	6449	119	109	**6388**	-0.95	100.30
CMT5Y	199	982	9	0	**966**	-1.63	182.71	3297	56	47	**3277**	-0.61	136.28	6568	116	129	**6430**	-2.10	115.73
CMT6X	50	**548**	6	0	**548**	0.00	5.01	1138	18	12	**1132**	-0.53	6.13	**1827**	32	25	**1827**	0.00	3.27
CMT6Y	50	**548**	6	0	**548**	0.00	5.78	1058	17	6	**1055**	-0.28	5.60	1841	31	32	**1832**	-0.49	4.15
CMT7X	75	897	11	0	**895**	-0.22	9.21	**1952**	34	17	1972	1.02	13.16	2742	48	40	**2738**	-0.15	8.41
CMT7Y	75	897	11	0	**894**	-0.33	9.66	**1820**	30	18	1836	0.88	14.73	2809	48	50	**2801**	-0.28	10.07
CMT8X	100	856	9	0	**848**	-0.93	26.91	1946	29	28	**1913**	-1.70	35.44	3412	58	43	**3400**	-0.35	22.35
CMT8Y	100	856	9	0	**844**	-1.40	25.92	**1805**	27	23	1815	0.55	27.70	3425	58	47	**3416**	-0.26	20.69
CMT9X	150	**1142**	14	0	1143	0.09	61.57	2742	44	53	**2713**	-1.06	78.91	4933	87	68	**4908**	-0.51	65.62
CMT9Y	150	**1143**	14	0	**1143**	0.00	57.22	2548	41	43	**2524**	-0.94	68.59	4947	86	86	**4907**	-0.81	53.92
CMT10X	199	1372	17	1	**1355**	-1.24	115.76	3672	61	82	**3609**	-1.72	162.93	6449	118	109	**6398**	-0.79	103.24
CMT10Y	199	1366	17	0	**1353**	-0.95	134.77	3297	56	49	**3288**	-0.27	131.46	6568	115	128	**6419**	-2.27	121.95
CMT11X	120	873	4	4	**819**	-6.19	118.57	3633	29	38	**3572**	-1.68	64.09	7427	68	62	**7321**	-1.43	33.43
CMT11Y	120	826	4	6	**764**	-7.51	108.97	**3178**	28	8	3196	0.57	22.28	7499	68	67	**7469**	-0.40	25.22
CMT12X	100	672	5	4	**646**	-3.87	54.44	2426	32	33	**2408**	-0.74	35.46	3950	60	52	**3925**	-0.63	16.59
CMT12Y	100	632	5	0	**627**	-0.79	41.12	2167	31	24	**2161**	-0.28	19.99	3897	59	63	**3896**	-0.03	16.76
CMT13X	120	1546	11	0	**1524**	-1.42	35.51	3633	29	48	**3569**	-1.76	58.70	7427	68	58	**7325**	-1.37	29.74
CMT13Y	120	1542	11	0	**1524**	-1.17	28.69	**3178**	28	10	3198	0.63	22.42	7499	69	66	**7456**	-0.57	25.40
CMT14X	100	**821**	10	0	**821**	0.00	17.13	2426	32	34	**2420**	-0.25	29.16	3950	60	54	**3925**	-0.63	16.34
CMT14Y	100	**819**	10	0	821	0.24	16.20	2167	31	23	**2161**	-0.28	19.06	3897	59	61	**3895**	-0.05	16.83
Avg.						**-1.42**	**58.65**					**-0.40**	**49.04**					**-0.61**	**35.13**
$\sum$			**234**	**36**					**951**	**865**					**1873**	**1743**			

Table B.9.: Detailed results of ALNS-PR on the VRPDDP benchmarks Nagy-VRPDDP1, Nagy-VRPDDP2, and Nagy-VRPDDP3.

			P		**ALNS-PR**			
Inst.	n	BKS	Δ^b(%)	t^{ab}(s)	n^d	f^b	Δ^b(%)	t^a(s)
SCA3-0	50	635.62	0.00	8.59	2	**629.84**	-0.91	27.31
SCA3-1	50	**697.84**	**0.00**	8.07	0	**697.84**	0.00	56.29
SCA3-2	50	**659.34**	**0.00**	12.33	0	**659.34**	0.00	40.64
SCA3-3	50	**680.04**	**0.00**	9.36	0	**680.04**	0.00	38.88
SCA3-4	50	**690.50**	**0.00**	7.14	0	**690.50**	0.00	27.37
SCA3-5	50	**659.90**	**0.00**	9.69	0	**659.90**	0.00	31.04
SCA3-6	50	**651.09**	**0.00**	7.21	0	**651.09**	0.00	32.71
SCA3-7	50	**659.17**	**0.00**	9.33	0	**659.17**	0.00	32.12
SCA3-8	50	**719.47**	**0.00**	8.93	0	**719.47**	0.00	26.19
SCA3-9	50	**681.00**	**0.00**	9.63	0	**681.00**	0.00	28.49
SCA8-0	50	**961.50**	**0.00**	9.38	0	**961.50**	0.00	18.18
SCA8-1	50	**1049.65**	**0.00**	9.44	0	**1049.65**	0.00	17.66
SCA8-2	50	**1039.64**	**0.00**	11.62	0	**1039.64**	0.00	19.82
SCA8-3	50	**979.13**	**0.00**	52.08	1	**979.13**	0.00	16.92
SCA8-4	50	**1065.49**	**0.00**	10.20	0	**1065.49**	0.00	18.67
SCA8-5	50	1027.08	0.00	13.02	1	**1022.02**	-0.49	15.31
SCA8-6	50	**969.50**	**0.00**	69.31	1	**969.50**	0.00	17.59
SCA8-7	50	1051.28	0.00	17.58	3	**1047.78**	-0.33	17.85
SCA8-8	50	**1071.18**	**0.00**	12.71	0	**1071.18**	0.00	17.31
SCA8-9	50	**1057.26**	**0.00**	100.79	1	**1057.26**	0.00	13.91
CON3-0	50	**616.52**	**0.00**	9.18	0	**616.52**	0.00	48.67
CON3-1	50	**554.47**	**0.00**	6.01	0	**554.47**	0.00	40.44
CON3-2	50	**518.00**	**0.00**	14.50	0	**518.00**	0.00	64.45
CON3-3	50	**591.19**	**0.00**	11.04	0	**591.19**	0.00	57.85
CON3-4	50	**588.79**	**0.00**	4.37	0	**588.79**	0.00	53.80
CON3-5	50	**563.70**	**0.00**	9.26	0	**563.70**	0.00	60.18
CON3-6	50	499.05	0.00	13.15	1	**498.55**	-0.10	62.10
CON3-7	50	**576.48**	**0.00**	8.43	0	**576.48**	0.00	47.41
CON3-8	50	523.05	0.00	6.61	1	**521.71**	-0.26	59.11
CON3-9	50	**578.25**	**0.00**	9.12	0	**578.25**	0.00	57.71
CON8-0	50	**857.12**	**0.00**	96.91	1	**857.12**	0.00	31.63
CON8-1	50	**739.44**	**0.00**	110.83	1	**739.44**	0.00	24.77
CON8-2	50	**706.51**	**0.00**	52.36	1	**706.51**	0.00	28.29
CON8-3	50	811.07	0.00	10.82	1	**808.66**	-0.30	39.34
CON8-4	50	**771.30**	**0.00**	40.25	1	**771.30**	0.00	26.86
CON8-5	50	**754.88**	**0.00**	7.81	0	**754.88**	0.00	31.99
CON8-6	50	**678.92**	**0.00**	6.10	0	**678.92**	0.00	29.12
CON8-7	50	**811.96**	**0.00**	11.40	0	**811.96**	0.00	31.33
CON8-8	50	**766.99**	**0.00**	31.09	1	**766.99**	0.00	29.68
CON8-9	50	797.69	0.00	102.94	6	**796.40**	-0.16	53.76
Avg.			**0.00**	**23.96**			**-0.06**	**34.82**
$\sum$					**23**			
Processor type			Xeon E5420			i5-6600		
Processor speed			2.50 GHz			3.30 GHz		
Passmark score			1073			2098		
t^c(s)			**12.25 × 6 × 10**			**34.82 × 10**		

Table B.10.: Detailed results of ALNS-PR in comparsion to P on the VRPDDP benchmark Polat-VRPDDP1.

			P		**ALNS-PR**			
Inst.	n	BKS	Δ^b(%)	t^{ab}(s)	n^d	f^b	Δ^b(%)	t^a(s)
r101	100	**1009.95**	**0.00**	67.04	0	**1009.95**	0.00	122.14
r201	100	**666.20**	**0.00**	59.58	0	**666.20**	0.00	321.07
c101	100	**1220.18**	**0.00**	26.42	0	1220.99	0.07	132.86
c201	100	**662.07**	**0.00**	30.78	0	**662.07**	0.00	148.76
rc101	100	1058.94	0.00	33.39	2	**1057.40**	-0.15	126.81
rc201	100	**672.92**	**0.00**	32.62	0	**672.92**	0.00	211.76
R1_2_1	200	3353.80	0.04	1780.00	5	**3340.19**	-0.41	523.52
R2_2_1	200	**1665.58**	**0.00**	1401.84	0	**1665.58**	0.00	1248.30
C1_2_1	200	**3628.51**	0.04	4082.55	0	3632.31	0.10	410.28
C2_2_1	200	**1726.59**	**0.00**	1123.42	0	**1726.59**	0.00	1114.79
RC1_2_1	200	**3303.70**	0.04	2168.18	5	3304.81	0.03	457.95
RC2_2_1	200	**1560.00**	**0.00**	1074.29	0	**1560.00**	0.00	1635.74
Avg.			**0.01**	**990.01**			**-0.03**	**537.83**
$\sum$					**12**			
Processor type			Xeon E5420			i5-6600		
Processor speed			2.50 GHz			3.30 GHz		
Passmark score			1073			2098		
t^c(s)			**506.33 × 6 × 10**			**537.83 × 10**		

Table B.11.: Detailed results of ALNS-PR in comparison to P on the VRPDDP benchmark Polat-VRPDDP2.

		100%				75%				50%				25%				10%			
			ALNS-PR				ALNS-PR				ALNS-PR				ALNS-PR				ALNS-PR		
Inst.	n	BKS	f^b	$\Delta^b(\%)$	t^a(s)	BKS	f^b	$\Delta^b(\%)$	t^a(s)	BKS	f^b	$\Delta^b(\%)$	t^a(s)	BKS	f^b	$\Delta^b(\%)$	t^a(s)	BKS	f^b	$\Delta^b(\%)$	t^a(s)
A1	25	**229884**	**229884**	0.00	0.62	**229884**	**229884**	0.00	0.61	**228023**	**228020**	0.00	0.70	**225341**	**225339**	0.00	0.63	**224307**	**224304**	0.00	0.55
A2	25	180439	**180117**	-0.18	0.53	180450	**180117**	-0.18	0.61	**172511**	**172509**	0.00	0.48	**169970**	**169968**	0.00	0.67	**169500**	**169497**	0.00	0.45
A3/A4	25	158433	**155795**	-1.67	0.84	**154523**	**154522**	0.00	0.57	**149516**	**149514**	0.00	0.69	143283	**142872**	-0.29	0.93	**142034**	**142032**	0.00	0.90
B1	30	242495	**239077**	-1.41	1.05	241197	**238984**	-0.92	0.87	**238988**	**238984**	0.00	0.87	**234027**	**234024**	0.00	0.75	**232436**	**232430**	0.00	1.00
B2	30	198149	**198045**	-0.05	1.14	202715	**197952**	-2.35	1.04	193196	**191606**	-0.82	1.10	182231	**179225**	-1.65	1.22	182231	**179196**	-1.67	1.26
B3	30	173835	**169368**	-2.57	0.86	169190	**167419**	-1.05	0.99	153774	**152200**	-1.02	1.14	**146709**	**146706**	0.00	1.99	**145702**	**145699**	0.00	1.38
C1	40	260140	**250557**	-3.68	2.33	**250557**	**250557**	0.00	2.09	**246189**	**246189**	0.00	2.04	244184	**243845**	-0.14	2.57	240246	**238004**	-0.93	2.66
C2	40	224787	**215019**	-4.35	1.65	224794	**215019**	-4.35	1.39	211991	**205126**	-3.24	1.81	209353	**202653**	-3.20	1.83	202183	**198167**	-1.99	1.98
C3/C4	40	200195	**195365**	-2.41	2.20	207024	**195054**	-5.78	1.97	190696	**181119**	-5.02	2.18	174480	**174198**	-0.16	2.80	176263	**169202**	-4.01	3.26
D1/D2	38	318300	**316711**	-0.50	1.25	**314618**	**314621**	0.00	0.79	**314618**	**314621**	0.00	0.83	311266	**309591**	-0.54	1.06	**308696**	**308696**	0.00	1.21
D3	38	241452	**239482**	-0.82	0.97	236664	**234358**	-0.97	0.93	233652	**232597**	-0.45	1.07	226232	**220851**	-2.38	1.13	225336	**220851**	-1.99	1.58
D4	38	211288	**205834**	-2.58	1.40	210480	**198856**	-5.52	1.72	195368	**192769**	-1.33	1.88	188411	**184166**	-2.25	1.60	182928	**182499**	-0.23	1.84
E1	45	253558	**238880**	-5.79	1.73	**238747**	**238747**	0.00	1.54	238747	**237190**	-0.65	2.30	231073	**227500**	-1.55	2.32	224650	**220740**	-1.74	2.29
E2	45	223650	**212262**	-5.09	3.46	223974	**212129**	-5.29	2.49	207579	**198119**	-4.56	3.11	197050	**190048**	-3.55	4.46	198476	**190048**	-4.25	3.68
E3	45	217763	**206658**	-5.10	3.67	229597	**206525**	-10.05	3.13	192305	**187675**	-2.41	3.09	188968	**183791**	-2.74	4.99	186525	**182549**	-2.13	4.70
F1/F2	60	276713	**263175**	-4.89	4.78	275712	**261774**	-5.06	4.95	269338	**253737**	-5.79	6.32	261795	**246657**	-5.78	6.92	250039	**244818**	-2.09	5.08
F3	60	262287	**241970**	-7.75	4.17	260831	**235669**	-9.65	5.53	240760	**226612**	-5.88	5.34	223918	**217967**	-2.66	6.93	219273	**213490**	-2.64	8.09
F4	60	248238	**233861**	-5.79	4.95	244507	**223213**	-8.71	5.80	229230	**217089**	-5.30	6.97	212470	**204313**	-3.84	9.97	211083	**198711**	-5.86	6.13
G1	57	316209	**305001**	-3.54	3.46	323054	**305001**	-5.59	3.13	316997	**300964**	-5.06	5.17	301871	**296045**	-1.93	5.47	298309	**295951**	-0.79	4.95
G2	57	262137	**245441**	-6.37	3.04	259071	**240950**	-6.99	5.23	234653	**234302**	-0.15	4.78	256104	**234268**	-8.53	5.33	237195	**234062**	-1.32	8.04
G3/G4	57	243394	**229506**	-5.71	5.18	237680	**223006**	-6.17	5.34	217963	**216805**	-0.53	6.13	222770	**214933**	-3.52	6.71	215418	**215322**	-0.04	5.77
G5	57	235707	**218484**	-7.31	6.79	227340	**209413**	-7.89	4.78	223223	**204589**	-8.35	6.62	211184	**202050**	-4.33	6.43	203854	**200520**	-1.64	7.77
G6	57	247591	**213457**	-13.79	4.44	225017	**201770**	-10.33	5.80	210451	**191082**	-9.20	7.38	203668	**189927**	-6.75	8.99	192196	**188696**	-1.82	8.45
H1	68	284488	**268933**	-5.47	4.91	279261	**266888**	-4.43	7.64	261276	**249953**	-4.33	9.83	239087	**237142**	-0.81	9.48	236540	**234555**	-0.84	10.42
H2	68	296774	**253366**	-14.63	5.99	280952	**249803**	-11.09	10.12	251865	**231550**	-8.07	11.80	226552	**218642**	-3.49	10.55	217359	**212840**	-2.08	11.58
H3/H4	68	302895	**247449**	-18.31	8.02	269975	**241149**	-10.68	11.16	240853	**221265**	-8.13	14.12	214351	**205778**	-4.00	10.85	208908	**202030**	-3.29	9.47
H5/H6	68	275934	**246121**	-10.80	10.37	268266	**236155**	-11.97	8.11	228830	**214760**	-6.15	10.12	213306	**201421**	-5.57	11.86	208955	**196715**	-5.86	11.45
I1	90	365077	**347837**	-4.72	12.78	360486	**347837**	-3.51	12.29	358323	**335837**	-6.28	16.02	353935	**324505**	-8.32	16.78	330750	**319899**	-3.28	14.73
I2	90	354304	**309946**	-12.52	11.51	332209	**309946**	-6.70	12.64	317656	**293522**	-7.60	19.15	289686	**278309**	-3.93	19.77	275940	**271639**	-1.56	21.83
I3/I4/I5	90	368042	**294509**	-19.98	17.15	358742	**286148**	-20.24	17.28	287803	**264449**	-8.11	23.47	266319	**247580**	-7.04	19.72	250220	**240680**	-3.81	18.89
J1	94	380016	**335004**	-11.84	14.03	352363	**334063**	-5.19	16.81	340645	**329442**	-3.29	21.35	339141	**324759**	-4.24	19.92	330235	**323819**	-1.94	24.78
J2	94	354542	**310417**	-12.45	13.68	323085	**307280**	-4.89	16.06	316751	**294712**	-6.96	20.30	296512	**292732**	-1.27	21.85	300584	**290764**	-3.27	20.30
J3	94	351824	**279220**	-20.64	15.27	303957	**271822**	-10.57	28.18	278173	**252473**	-9.24	21.73	266619	**249469**	-6.43	28.85	265430	**249688**	-5.93	26.35
J4	94	318167	**296533**	-6.80	22.75	314083	**287902**	-8.34	18.26	294595	**276208**	-6.24	25.07	282575	**271812**	-3.81	25.30	282725	**273643**	-3.21	22.05
K1	113	445021	**394068**	-11.45	22.74	406113	**384685**	-5.28	33.39	389440	**368372**	-5.41	32.28	371304	**355427**	-4.28	33.43	356970	**350422**	-1.83	33.90
K2/K3	113	468828	**362128**	-22.76	34.79	418710	**348576**	-16.75	29.52	403276	**330431**	-18.06	34.69	329701	**315875**	-4.19	30.06	319350	**311759**	-2.38	29.46
K4	113	435073	**348947**	-19.80	27.08	401243	**333219**	-16.95	41.07	343598	**312541**	-9.04	44.21	308501	**295929**	-4.08	39.06	298461	**291793**	-2.23	42.35
L1	150	508933	**425748**	-16.34	40.52	522364	**423730**	-18.88	43.39	489902	**407219**	-16.88	57.71	444667	**392125**	-11.82	74.45	415670	**387766**	-6.71	85.51
L2/L3	150	470331	**401231**	-14.69	51.92	473495	**393916**	-16.81	59.79	450087	**375280**	-16.62	64.27	407072	**358242**	-12.00	80.14	377344	**351938**	-6.73	66.44
L4/L5	150	455947	**384635**	-15.64	57.19	437409	**373552**	-14.60	76.63	431652	**354556**	-17.86	78.43	366808	**341024**	-7.03	69.81	354543	**330359**	-6.82	83.05
M1/M2	125	443331	**397255**	-10.39	35.64	431664	**386885**	-10.37	39.29	401431	**369624**	-7.92	50.00	384954	**358206**	-6.95	66.86	369165	**355275**	-3.76	39.48
M3	125	448935	**376158**	-16.21	36.17	395342	**361141**	-8.65	30.68	363880	**344206**	-5.41	45.24	347320	**335260**	-3.47	49.40	340765	**333532**	-2.12	43.77
M4	125	391298	**348436**	-10.95	39.57	387711	**332690**	-14.19	45.84	332140	**312291**	-5.98	58.16	322823	**303565**	-5.97	44.56	310964	**302075**	-2.86	58.74
N1/N2	150	505039	**408062**	-19.20	53.17	506900	**400947**	-20.90	64.42	449625	**383470**	-14.71	71.47	400453	**372057**	-7.09	67.64	396404	**366832**	-7.46	75.17
N3/N4	150	501193	**394334**	-21.32	50.46	458584	**386237**	-15.78	84.82	413482	**365325**	-11.65	83.27	394661	**350307**	-11.24	75.72	367762	**344849**	-6.23	68.52
N5/N6	150	483845	**374864**	-22.52	72.67	441319	**359308**	-18.58	76.25	404513	**335740**	-17.00	76.02	352094	**320147**	-9.07	82.22	328354	**311463**	-5.14	87.31
Avg.				**-9.36**	**15.63**			**-8.09**	**18.37**			**-6.10**	**20.88**			**-4.17**	**21.61**			**-2.71**	**21.49**

Table B.12.: Detailed results of ALNS-PR on the VRPRMDP benchmark Nagy-VRPRMDP1 for different percentages of unavailable vehicle capacity in case of mixed loads.

		100%				75%				50%				25%				10%			
			ALNS-PR				ALNS-PR				ALNS-PR				ALNS-PR				ALNS-PR		
Inst.	n	BKS	f^b	Δ^b(%)	t^a(s)	BKS	f^b	Δ^b(%)	t^a(s)	BKS	f^b	Δ^b(%)	t^a(s)	BKS	f^b	Δ^b(%)	t^a(s)	BKS	f^b	Δ^b(%)	t^a(s)
EIL22_2	21	372	**371**	-0.27	0.42	367	**358**	-2.45	0.48	353	**345**	-2.27	0.48	327	**326**	-0.31	0.44	326	**324**	-0.61	0.41
EIL22_3	21	368	**366**	-0.54	0.41	367	**366**	-0.27	0.42	349	**348**	-0.29	0.54	349	**348**	-0.29	0.48	342	**341**	-0.29	0.48
EIL22_5	21	382	**375**	-1.83	0.38	377	**375**	-0.53	0.37	355	**352**	-0.85	0.36	360	**341**	-5.28	0.45	342	**341**	-0.29	0.43
EIL23_2	22	**677**	**677**	0.00	0.60	621	**598**	-3.70	0.57	555	**531**	-4.32	0.87	532	**526**	-1.13	0.70	527	**526**	-0.19	0.74
EIL23_3	22	645	**640**	-0.78	0.60	**577**	**577**	0.00	0.56	**553**	**553**	0.00	0.78	**533**	**533**	0.00	0.87	527	**526**	-0.19	0.69
EIL23_5	22	648	**623**	-3.86	0.59	**571**	**571**	0.00	0.60	**514**	**514**	0.00	0.56	**514**	**514**	0.00	0.55	527	**514**	-2.47	0.54
EIL30_2	29	574	**501**	-12.72	1.16	566	**480**	-15.19	1.49	**456**	**456**	0.00	1.42	448	**432**	-3.57	1.26	437	**429**	-1.83	1.65
EIL30_3	29	538	**537**	-0.19	1.12	508	**506**	-0.39	1.04	494	**484**	-2.02	1.49	478	**475**	-0.63	1.15	481	**475**	-1.25	1.03
EIL30_5	29	530	**514**	-3.02	0.95	500	**491**	-1.80	1.15	478	**475**	-0.63	0.95	478	**475**	-0.63	0.99	478	**475**	-0.63	1.06
EIL33_2	32	774	**738**	-4.65	1.13	745	**724**	-2.82	1.19	715	**712**	-0.42	2.48	697	**694**	-0.43	1.31	692	**691**	-0.14	1.75
EIL33_3	32	777	**750**	-3.47	1.41	736	**721**	-2.04	1.56	699	**694**	-0.72	1.62	684	**681**	-0.44	1.91	684	**680**	-0.58	1.43
EIL33_5	32	779	**736**	-5.52	1.33	732	**711**	-2.87	1.48	716	**699**	-2.37	1.52	694	**686**	-1.15	1.67	687	**686**	-0.15	1.14
EIL51_2	50	607	**559**	-7.91	4.69	617	**538**	-12.80	5.19	532	**507**	-4.70	4.83	505	**477**	-5.54	8.69	475	**470**	-1.05	6.01
EIL51_3	50	630	**548**	-13.02	6.80	564	**530**	-6.03	5.16	525	**492**	-6.29	5.80	504	**488**	-3.17	5.52	505	**488**	-3.37	4.54
EIL51_5	50	651	**565**	-13.21	5.08	564	**549**	-2.66	5.17	524	**505**	-3.63	5.43	519	**492**	-5.20	5.92	501	**493**	-1.60	5.17
EIL76a_2	75	831	**690**	-16.97	10.64	838	**669**	-20.17	14.52	759	**625**	-17.65	16.41	709	**602**	-15.09	19.56	694	**595**	-14.27	17.20
EIL76a_3	75	875	**715**	-18.29	12.97	797	**678**	-14.93	12.64	761	**639**	-16.03	16.94	736	**622**	-15.49	17.95	732	**622**	-15.03	16.96
EIL76a_5	75	831	**694**	-16.49	10.46	819	**678**	-17.22	12.98	814	**650**	-20.15	14.95	783	**643**	-17.88	14.88	775	**642**	-17.16	14.40
EIL76b_2	75	856	**713**	-16.71	12.19	824	**690**	-16.26	13.22	812	**652**	-19.70	13.43	792	**630**	-20.45	17.64	789	**623**	-21.04	15.42
EIL76b_3	75	897	**734**	-18.17	10.60	907	**709**	-21.83	10.93	871	**667**	-23.42	10.80	840	**657**	-21.79	12.36	869	**652**	-24.97	12.66
EIL76b_5	75	955	**736**	-22.93	10.10	949	**718**	-24.34	11.77	932	**691**	-25.86	14.52	914	**689**	-24.62	11.90	908	**689**	-24.12	8.18
EIL76c_2	75	830	**739**	-10.96	7.48	789	**729**	-7.60	9.76	710	**691**	-2.68	13.65	**660**	661	0.15	14.07	**641**	658	2.65	15.94
EIL76c_3	75	837	**768**	-8.24	9.34	800	**747**	-6.63	12.84	726	**720**	-0.83	9.57	**682**	708	3.81	11.04	**667**	706	5.85	11.68
EIL76c_5	75	826	**781**	-5.45	9.60	**746**	781	4.69	10.86	**724**	763	5.39	8.38	**701**	752	7.28	10.83	**699**	745	6.58	11.23
EIL76d_2	75	829	**801**	-3.38	11.75	**775**	794	2.45	9.53	**694**	786	13.26	9.75	**645**	758	17.52	8.39	**618**	748	21.04	10.75
EIL76d_3	75	**832**	873	4.93	8.85	**736**	864	17.39	7.81	**674**	838	24.33	8.31	**656**	820	25.00	10.78	**642**	816	27.10	9.77
EIL76d_5	75	**820**	919	12.07	10.61	**725**	918	26.62	8.40	**684**	907	32.60	11.11	**674**	895	32.79	11.67	**663**	894	34.84	9.39
EIL101a_2	100	1087	**827**	-23.92	34.59	935	**779**	-16.68	30.77	831	**754**	-9.27	38.67	771	**726**	-5.84	47.07	761	**700**	-8.02	35.71
EIL101a_3	100	1028	**846**	-17.70	35.02	939	**808**	-13.95	37.27	844	**760**	-9.95	25.84	793	**746**	-5.93	24.44	777	**737**	-5.15	25.92
EIL101a_5	100	968	**859**	-11.26	21.68	892	**810**	-9.19	32.49	882	**784**	-11.11	28.38	811	**766**	-5.55	29.84	788	**758**	-3.81	30.20
EIL101b_2	100	1046	**927**	-11.38	19.39	958	**889**	-7.20	24.67	949	**863**	-9.06	23.50	895	**849**	-5.14	25.39	887	**833**	-6.09	27.16
EIL101b_3	100	1040	**971**	-6.63	19.44	1008	**957**	-5.06	17.59	997	**919**	-7.82	21.04	940	**898**	-4.47	20.64	941	**898**	-4.57	23.92
EIL101b_5	100	1040	**1008**	-3.08	21.70	1039	**982**	-5.49	21.46	1024	**957**	-6.54	24.70	997	**954**	-4.31	24.65	987	**952**	-3.55	22.84
Avg.				**-8.05**	**9.18**			**-5.73**	**9.88**			**-4.03**	**10.27**			**-2.66**	**11.06**			**-1.95**	**10.50**

Table B.13.: Detailed results of ALNS-PR on the VRPRMDP benchmark Nagy-VRPRMDP2 for different percentages of unavailable vehicle capacity in case of mixed loads.

		100%				75%				50%				25%				10%			
			ALNS-PR				ALNS-PR				ALNS-PR				ALNS-PR				ALNS-PR		
Inst.	n	BKS	f^b	Δ^b(%)	t^a(s)	BKS	f^b	Δ^b(%)	t^a(s)	BKS	f^b	Δ^b(%)	t^a(s)	BKS	f^b	Δ^b(%)	t^a(s)	BKS	f^b	Δ^b(%)	t^a(s)
CMT1T	50	587	**570.16**	-2.87	1.96	547	**544.30**	-0.49	2.80	536	**520.26**	-2.94	3.18	522	**520.06**	-0.37	2.19	528	**520.06**	-1.50	2.75
CMT1Q	50	600	**572.78**	-4.54	2.16	573	**536.94**	-6.29	2.88	529	**502.74**	-4.96	3.42	503	**495.90**	-1.41	2.95	501	**490.50**	-2.10	3.04
CMT1H	50	657	**562.83**	-14.33	5.00	613	**541.62**	-11.64	3.53	532	**509.46**	-4.24	4.98	505	**486.34**	-3.70	5.55	475	**472.45**	-0.54	5.97
CMT2T	50	835	**806.31**	-3.44	5.33	832	**806.49**	-3.07	5.91	819	**797.10**	-2.67	6.09	811	**786.29**	-3.05	7.21	799	**788.00**	-1.38	6.64
CMT2Q	50	827	**806.74**	-2.45	7.17	810	**794.87**	-1.87	6.76	774	**751.44**	-2.91	5.90	749	**738.88**	-1.35	7.31	761	**734.70**	-3.46	11.37
CMT2H	50	849	**743.59**	-12.42	4.19	817	**733.72**	-10.19	4.11	759	**696.51**	-8.23	5.78	709	**667.23**	-5.89	8.54	694	**663.92**	-4.33	8.65
CMT3T	75	873	**830.78**	-4.84	13.22	849	**813.84**	-4.14	16.70	835	**801.19**	-4.05	25.40	850	**800.83**	-5.78	20.55	822	**798.14**	-2.90	18.97
CMT3Q	75	968	**847.04**	-12.50	26.27	915	**820.85**	-10.29	15.11	838	**773.72**	-7.67	22.56	813	**758.46**	-6.71	21.16	792	**747.87**	-5.57	13.72
CMT3H	75	1053	**840.48**	-20.18	22.26	935	**790.42**	-15.46	14.98	831	**763.69**	-8.10	27.36	771	**735.85**	-4.56	23.38	761	**713.26**	-6.27	35.57
CMT4T	75	1169	**1044.65**	-10.64	48.58	1059	**1019.24**	-3.75	42.29	1047	**998.85**	-4.60	40.11	1049	**997.30**	-4.93	54.83	1038	**995.09**	-4.13	53.46
CMT4Q	75	1261	**1018.53**	-19.23	46.63	1137	**982.61**	-13.58	46.65	1072	**950.11**	-11.37	46.12	954	**928.36**	-2.69	58.36	961	**918.37**	-4.44	61.95
CMT4H	75	1152	**1007.42**	-12.55	49.81	1213	**956.78**	-21.12	51.94	1057	**907.39**	-14.15	82.63	935	**864.25**	-7.57	64.68	915	**846.46**	-7.49	85.99
CMT5T	100	1421	**1286.40**	-9.47	76.85	1323	**1255.80**	-5.08	61.46	1290	**1229.89**	-4.66	78.85	1304	**1226.36**	-5.95	102.90	1294	**1223.13**	-5.48	87.67
CMT5Q	100	1463	**1248.80**	-14.64	96.21	1357	**1192.12**	-12.15	79.09	1251	**1158.37**	-7.40	101.19	1278	**1125.69**	-11.92	87.10	1178	**1117.39**	-5.15	137.84
CMT5H	100	1484	**1155.86**	-22.11	80.16	1486	**1124.77**	-24.31	74.30	1299	**1076.95**	-17.09	106.92	1134	**1022.67**	-9.82	126.06	1082	**995.03**	-8.04	135.64
CMT6T	100	619	**602.29**	-2.70	2.52	595	**577.43**	-2.95	1.99	561	**555.43**	-0.99	3.01	556	**555.43**	-0.10	3.12	562	**555.43**	-1.17	3.43
CMT6Q	100	629	**606.49**	-3.58	2.54	591	**577.24**	-2.33	2.23	561	**560.24**	-0.13	2.48	559	**555.43**	-0.64	2.95	559	**555.43**	-0.64	2.15
CMT6H	100	630	**612.49**	-2.78	2.09	591	**588.97**	-0.34	1.66	568	**563.39**	-0.81	3.33	557	**555.43**	-0.28	2.39	559	**555.43**	-0.64	2.54
CMT7T	100	922	**915.16**	-0.74	6.25	916	**914.21**	-0.20	6.03	918	**903.05**	-1.63	6.51	907	**903.05**	-0.44	6.93	921	**903.05**	-1.95	6.23
CMT7Q	100	999	**945.62**	-5.34	4.98	960	**929.89**	-3.14	4.03	938	**906.94**	-3.31	8.60	912	**904.44**	-0.83	8.58	910	**902.95**	-0.77	7.55
CMT7H	100	997	**944.49**	-5.27	5.08	948	**931.54**	-1.74	3.88	908	**906.07**	-0.21	7.51	901	**901.37**	0.04	8.88	903	**900.84**	-0.24	5.72
CMT8T	100	903	**885.84**	-1.90	15.21	945	**873.19**	-7.60	13.15	908	**868.28**	-4.37	19.08	877	**865.54**	-1.31	15.34	883	**865.54**	-1.98	14.89
CMT8Q	100	1015	**949.45**	-6.46	15.64	992	**894.02**	-9.88	10.90	895	**868.17**	-3.00	13.83	866	**865.50**	-0.06	13.45	874	**865.50**	-0.97	14.10
CMT8H	100	1069	**956.99**	-10.48	11.41	961	**904.17**	-5.91	11.84	887	**876.23**	-1.21	16.21	890	**865.54**	-2.75	20.16	878	**865.50**	-1.42	15.61
CMT9T	120	1285	**1195.04**	-7.00	47.12	1279	**1182.35**	-7.56	42.90	1215	**1162.99**	-4.28	60.07	1232	**1162.99**	-5.60	60.77	1222	**1162.68**	-4.85	49.01
CMT9Q	120	1452	**1221.24**	-15.89	49.02	1255	**1188.75**	-5.28	38.07	1229	**1163.65**	-5.32	51.40	1224	**1161.51**	-5.11	47.01	1220	**1161.24**	-4.82	57.63
CMT9H	120	1418	**1265.25**	-10.77	26.72	1284	**1203.85**	-6.24	33.38	1221	**1171.58**	-4.05	53.58	1222	**1162.71**	-4.85	66.35	1214	**1162.39**	-4.25	39.66
CMT10T	120	1546	**1454.31**	-5.93	128.36	1515	**1421.09**	-6.20	140.17	1464	**1404.73**	-4.05	95.72	1466	**1391.36**	-5.09	124.56	1462	**1398.30**	-4.36	129.28
CMT10Q	120	1727	**1506.73**	-12.75	87.31	1539	**1431.27**	-7.00	90.16	1441	**1406.65**	-2.38	126.90	1476	**1387.70**	-5.98	115.91	1419	**1386.17**	-2.31	136.48
CMT10H	120	1688	**1512.01**	-10.43	80.25	1551	**1458.42**	-5.97	77.33	1462	**1410.66**	-3.51	91.55	1448	**1392.48**	-3.83	108.39	1447	**1377.45**	-4.81	118.61
CMT11T	150	1358	**1052.93**	-22.46	30.28	1143	**1019.18**	-10.83	29.36	1174	**1002.75**	-14.59	30.88	1192	**999.04**	-16.19	25.87	1210	**1000.51**	-17.31	28.50
CMT11Q	150	1367	**1014.01**	-25.82	35.21	1162	**982.70**	-15.43	38.21	1144	**953.04**	-16.69	52.03	1053	**947.27**	-10.04	42.42	1066	**944.38**	-11.41	36.66
CMT11H	150	1488	**961.21**	-35.40	44.50	1439	**934.33**	-35.07	50.86	1124	**892.46**	-20.60	70.57	946	**835.50**	-11.68	72.40	948	**820.26**	-13.47	143.25
CMT12T	150	869	**815.70**	-6.13	14.29	837	**792.63**	-5.30	11.05	816	**787.52**	-3.49	12.47	822	**787.52**	-4.19	11.07	816	**787.52**	-3.49	9.80
CMT12Q	150	917	**814.91**	-11.13	11.73	819	**788.89**	-3.68	13.36	790	**754.27**	-4.52	15.49	765	**739.48**	-3.34	17.95	761	**735.48**	-3.35	19.99
CMT12H	150	968	**749.95**	-22.53	10.51	782	**736.04**	-5.88	9.90	725	**700.87**	-3.33	13.05	716	**657.07**	-8.23	20.28	672	**646.02**	-3.87	28.65
CMT13T	199	1722	**1586.00**	-7.90	19.00	1692	**1548.96**	-8.45	20.01	1704	**1541.70**	-9.52	29.35	1647	**1541.14**	-6.43	25.20	1646	**1542.97**	-6.26	27.82
CMT13Q	199	1800	**1590.89**	-11.62	27.33	1727	**1561.97**	-9.56	22.03	1626	**1544.28**	-5.03	33.30	1648	**1541.61**	-6.46	28.90	1648	**1545.72**	-6.21	29.26
CMT13H	199	1835	**1629.01**	-11.23	18.36	1714	**1574.32**	-8.15	24.25	1643	**1545.72**	-5.92	29.42	1647	**1544.93**	-6.20	29.81	1647	**1545.67**	-6.15	36.02
CMT14T	199	863	**852.81**	-1.18	8.04	850	**832.99**	-2.00	9.84	872	**828.09**	-5.04	9.37	849	**826.77**	-2.62	9.24	853	**826.77**	-3.07	9.42
CMT14Q	199	1016	**898.55**	-11.56	7.20	868	**863.42**	-0.53	11.36	861	**831.58**	-3.42	7.64	872	**823.34**	-5.58	6.99	826	**821.75**	-0.52	7.05
CMT14H	199	1077	**938.57**	-12.85	6.59	958	**878.00**	-8.35	6.68	898	**834.46**	-7.08	10.13	826	**821.75**	-0.52	6.49	851	**821.75**	-3.44	8.56
Avg.				**-10.67**	**28.65**			**-7.83**	**27.46**			**-5.80**	**34.14**			**-4.62**	**35.58**			**-4.20**	**39.46**

Table B.14.: Detailed results of ALNS-PR on the VRPRMDP benchmark Nagy-VRPRMDP3 for different percentages of unavailable vehicle capacity in case of mixed loads.

Julian Hof

Curriculum Vitae

Wissenschaftlicher Werdegang

01/2013–09/2018 **Wissenschaftlicher Mitarbeiter**, *Lehrstuhl für Wirtschaftsinformatik und Operations Research, Technische Universität Kaiserslautern*.

12/2012 **Diplom**, *Wirtschaftsingenieurwesen Fachrichtung Maschinenbau*, Technische Universität Kaiserslautern.

06/2006 **Allgemeine Hochschulreife**, *Gymnasium Ottweiler*.