universität wien

MASTERARBEIT | MASTER'S THESIS

Titel | Title

A solution method for the deposit refund waste collection problem

verfasst von | submitted by

Flora Luise Obermüller BA

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

Master of Science (MSc)

Wien | Vienna, 2025

Studienkennzahl lt. Studienblatt | Degree programme code as it appears on the student record sheet:	UA 066 915
Studienrichtung lt. Studienblatt | Degree programme as it appears on the student record sheet:	Masterstudium Betriebswirtschaft
Betreut von | Supervisor:	Univ.-Prof. Mag. Dr. Karl Franz Dörner Privatdoz.

Abstract

In this master's thesis a solution method for the deposit refund waste collection problem is presented. We define this problem as a waste collection problem for bottles and cans made of recyclable plastic and metal with periodic characteristics where collection demand data is based on estimated demand data as well as collection data gathered from the collection machine which is connected to the internet. Additionally minimum- and maximum thresholds of customers, referring to the minimum amount required for pickup and maximum amount allowed for storage are incorporated, alongside vehicle capacity constraints. The proposed solution method consists of two phases, where the first stage aims at solving the PVRP with an ALNS metaheuristic with estimated demand data, and the second phase adapts the resulting routes each day before route execution based on the collection machine data using heuristic methods to make sure minimum and maximum thresholds as well as vehicle capacity are not violated. The effectiveness of the first phase is tested by first comparing the total distance resulting from periodic vehicle routing with benchmark instances, for which the solution-values achieved in this thesis are overall 2.34% worse. Then, example data is used to illustrate tours and test the second phase of the solution method, by comparing how much this method changes the initial routes resulting from the first stage, where the total-distance is on average 0.62% longer in the second stage.

Abstract (deutsch)

In dieser Master-Arbeit wird eine Lösungsmethode für die Sammlung von Pfandabfall präsentiert. Wir definieren das Problem als ein Abfallsammelproblem für Flaschen und Dosen aus wiederverwertbarem Kunststoff und Metall mit periodischen Merkmalen, bei dem die Sammelnachfragedaten auf geschätzten Nachfragedaten sowie auf Daten basieren, die von dem mit dem Internet verbundenen Sammelautomat gesammelt werden. Zusätzlich werden minimale und maximale Schwellenwerte der Kunden, die sich auf die Mindestmenge für die Abholung und die maximal zulässige Menge für die Lagerung beziehen, sowie Beschränkungen der Fahrzeugkapazität berücksichtigt. Die vorgeschlagene Lösungsmethode besteht aus zwei Phasen, wobei die erste Phase darauf abzielt, das PVRP mit einer ALNS-Metaheuristik mit geschätzten Nachfragedaten zu lösen, und die zweite Phase, die resultierenden Routen jeden Tag vor der Routenausführung auf der Grundlage der Daten des Sammelautomaten mit heuristischen Methoden anpasst, um sicherzustellen, dass die Mindest- und Höchstschwellenwerte sowie die Fahrzeugkapazität nicht unter- bzw. überschritten werden. Die Effektivität der ersten Phase wird getestet, indem zunächst die Gesamtdistanz, die sich aus der periodischen Tourenplanung ergibt, mit Benchmark-Instanzen verglichen wird, wo die Werte dieser Arbeit durchschnittlich 2.34 % schlechter sind. Anschließend werden die mit der Methode erzielten Touren anhand von Beispieldaten veranschaulicht und die zweite Phase der Lösungsmethode getestet, indem verglichen wird, inwieweit diese Methode, die in der ersten Phase erzielten Touren verändert, wobei die Gesamtdistanz in der zweiten Phase im Durchschnitt 0.62 % länger ist.

Table of contents

List of Abbreviations

ACO ant colony optimization

ALNS Adaptive large neighborhood search

ARP arc routing problem

BSA backtracking search algorithm

CVRP capacitated vehicle routing problem

CW Clarke and Wright

CV Coefficient of variation

DCM discrete choice model

DVRP dynamic vehicle routing problem

GA genetic algorithm

GRASP greedy randomized search procedure

LNS large neighborhood search

MDVRP multi depot vehicle routing problem

NS neighborhood search

PVRP periodic vehicle routing problem

PSO particle swarm optimization

SA simulated annealing

SEO social engineering optimizer

TS tabu search

TSP travelling salesman problem

VNS variable neighborhood search

VRP vehicle routing problem

List of figures

List of tables

1 Introduction

In 2025, a deposit refund system for plastic bottles and beverage cans was introduced in Austria. Under this system, vendors who sell beverages are required to take back empty containers, for which consumers in return receive a 25 cent refund per item (EWP Recycling Pfand Österreich gGmbh, 2024a). The system was implemented in response to relatively low collection rates in the past, where only around 70 percent of plastic bottles were collected through regular recycling infrastructure. With the new deposit refund system, Austria aims to achieve a collection rate of 80 percent, in line with the EU Single-Use Plastics Directive, which sets a target of 77 percent (European Commission, n.d.). To little surprise, reaching this target rate is influenced by whether a country in the EU had implemented a similar system (EWP Recycling Pfand Österreich gGmbh, 2024a).

While this system clearly supports Austria's efforts towards a more circular economy, its implementation comes with several logistical challenges. One of which is the collection of waste from the vendors in terms of routing the collection vehicles, hence in which order the individual collection sites are to be visited with which vehicle. For this problem, the publicly available information on how the Austrian deposit refund waste collection is structured serves as an inspiration for the problem setting described in this thesis. Especially regarding the waste collection machines, which are connected to the internet and their ability to communicate the amount of waste collected with logistics providers (EWP Recycling Pfand Österreich gGmbh, 2024b).

Hence, using the information provided, this thesis aims to explore the arising problem by presenting one possible method to solve it, which can also be seen as a more generalized approach for solving the routing of deposit refund systems with smart collection machines. Here the focus lies on developing an approach that is effective at minimizing the distance travelled by the collection vehicles but also simple to implement and operationally use in practice. To do so, our method, which was implemented using C#, follows a two-step structure. Here an initial periodic plan is created first, where each vendor can be visited up to five times per week, which then gets adjusted each day based on the data from the collection machines in the second step. For the initial planning, an ALNS algorithm and for the adaptation of tours, a heuristic solution method will be presented in this thesis. For testing the effectiveness of the first part of the solution method is tested using benchmark instances by Cordeau et al. (1997). To illustrate the full solution method and testing how the tour adaptation changes the distance travelled and its effect on tours, example data with real coordinates of supermarkets and estimated demands using data of sales volumes of beverages in Austria, alongside population data, is created.

The remainder of this thesis is structured as follows: Chapter 2 presents theoretical background relevant to the topic. Chapter 3 describes the problem setting. Chapter 4 reviews the existing literature on vehicle routing and smart waste collection. Chapter 5 introduces the solution method. Chapter 6 describes the data used for testing. Chapter 7 presents and discusses the results of the experiments. Finally, Chapter 8 concludes the thesis and suggests directions for future research.

2 Theoretical background

This chapter aims to give an overview of the essentials of the VRP, CVRP as well as the PVRP. Then the fundamentals of the solution methods used in this thesis will be elaborated on to gain a better understanding of the method proposed in this thesis.

2.1 Fundamentals of (periodic) vehicle routing

First proposed by Dantzig and Ramser (1959), which aimed to optimize the routing of oil tankers as a generalization of the TSP[1] with returns to depot and first introduced with multiple vehicles by Clarke and Wright (1964), the VRP has become a widely studied problem within the field of logistics where the goal is to optimize the routing from one or multiple starting points to cities, customers or other locations while considering problem-specific constraints (Zhang et al., 2022).

The standard variant of the VRP, namely the CVRP, consists of a graph $G = (V, A)$, with $V = \{v_0, v_1, \dots, v_k\}$ being the vertex set (set of nodes) and $A = \{(v_i, v_j): v_i, v_j \in V, i \neq j\}$ being the arc set (set of direct connections between nodes), where Vertex v_0 corresponds to the depot and the rest of vertices i.e., $V \setminus \{v_0\}$ to the locations of customers or cities, each having a non-negative demand q_i and non-negative service duration d_i. Each arc in the arc set is associated with a certain travel time, distance or cost c_{ij}. The CVRP then aims to construct routes for a set of m vehicles while not exceeding their corresponding capacity $Q_1, \dots, Qm$. and the total tour duration D_k and making sure that each route begins and ends at the depot and assigning each customer to exactly one route (Cordeau et al., 1997). An example of the resulting graph can be seen in figure 1. It should be noted that some common objectives when it comes to VRP include minimizing the total cost (which usually depends on things such as fixed and variable costs for the vehicles), minimizing the total distance travelled or minimizing the number of vehicles used (Zhang et al., 2022).

The PVRP extends this problem by requiring customers to be visited within a multi-period planning horizon. Meaning, instead of planning routes only for single periods, e.g., a day, a customer should now be visited one or several times within multiple periods, e.g., five days. Generally, the dates when customers should be visited are not set, but a list with possible options can be provided (Ahmadi Basir et al., 2024). The idea was first introduced by Beltrami and Bodin (1974) in a study about waste collection, where some locations should be visited

[1] The TSP describes a problem where a salesperson (or any other person/vehicle specific to the problem) must visit a set of locations exactly once such that total travel costs are minimized and then return to the starting point (Dantzig et al., 1954).

Monday, Wednesday, and Friday or Tuesday, Thursday, and Saturday, hence three times a week and others six times. The name PVRP was first mentioned by Gaudioso and Paletta (1992), who required customers to have a minimum and maximum number of days between visits.

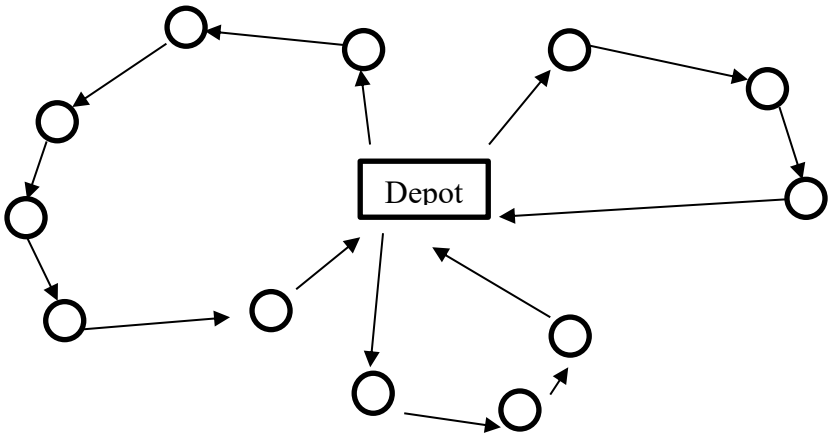

Figure 1 VRP diagram

In Foster and Ryan (1976) a first attempt to formulate the PVRP as a vehicle scheduling problem with periodic assignments was made. Another paper that should be mentioned when it comes to the earliest efforts to formulate the PVRP is Christofides and Beasley (1984), who use two sets of decision variables, one for assigning customers to schedules according to their frequency and one for routing the individual days in their integer programming formulation of the problem.

In the following, the problem definition and formulation used by Cordeau et al. (1997) will be further elaborated since the benchmark instances used later in this thesis stem from the same paper. They define the PVRP on a multigraph $G = (V, A)$, consisting of the vertex set $V = \{v_0, v_1, \dots, v_n\}$ and the arc set $A = \left\{(v_i, v_j)^{k,l}\right\}$. For each arc the non- negative costs c_{ijkl} proportional to the travel time of vehicle k from customer i to customer j on day l is known, which reduces to c_{ij} for $k = 1, \dots, m$ and $l = 1, \dots, t$, if not dependent on the vehicle and day. Then the binary constant a_{rl} which is one if day l belongs to a combination of visits r and the binary decision variables x_{ijkl} which turn one if a vehicle k visits customer j after i on day l, where i does not equal j, are defined. Since each customer i has a specified frequency of service e_i for example $e_i = 3$ within a planning horizon of t days and a resulting set of possible visit combinations C_i for example $C_i = \{\{1,3,5\}, \{2,4,6\}\}$, meaning that the customer must be visited twice, either on day one, three and five or day two, four and six, the third set of decision variables y_{ir} turn one if the visit combination $r \in C_i$ is assigned to customer i.

The problem can then be formulated as follows (taken out of Cordeau et al. (1997)):

Minimize $\sum_{i=0}^{n}\sum_{j=0}^{n}\sum_{k=l}^{m}\sum_{l=1}^{t} c_{ijkl}\, x_{ijkl}$, (1)

subject to:

$\sum_{r\in C_i} y_{ir} = 1 \qquad (i = 1, \dots, n);$ (2)

$\sum_{j=0}^{n}\sum_{k=1}^{m} x_{ijkl} - \sum_{r\in C_i} a_{rl} y_{ir} = 0 \qquad (i = 1, \dots, n; l = 1, \dots, t);$ (3)

$\sum_{i=0}^{n} x_{ihkl} - \sum_{j=0}^{n} x_{hjkl} = 0 \qquad (h = 0, \dots, n; k = 1, \dots, m; l = 1, \dots, t);$ (4)

$\sum_{j=1}^{n} x_{0jkl} \leq 1 \qquad (k = 1, .. m; l = 1, \dots, t);$ (5)

$\sum_{i=0}^{n}\sum_{j=0}^{n} q_i x_{ijkl} \leq Q_k \qquad (k = 1, \dots, m; l = 1, \dots, t)$ (6)

$\sum_{i=0}^{n}\sum_{j=0}^{n} (c_{ijkl} + d_i) x_{ijkl} \leq D_k \qquad (k = 1, \dots, m; l = 1, \dots., t);$ (7)

$\sum_{v_i\in S}\sum_{v_j\in S}(x_{ijkl} \leq |S| - 1 \quad (k = 1, \dots, m; l = 1, \dots, t; S \subseteq V \setminus \{0\};\ |S| \geq 2);$ (8)

$x_{ijkl} \in \{0,1\} \qquad (i = 0, \dots, n; j = 0, \dots, n; k = 1, \dots, m; l = 1, \dots, t);$ (9)

$y_{ir} \in \{0,1\} \qquad (i = 1, \dots, n; r \in C_i)$ (10)

Where the objective is to minimize total costs (1) subject to the constraints that each customer must be allocated to one possible visit combination (2) while also making sure that each customer is only visited on the days within the visit combination (3). Also, each vehicle which is servicing a customer on a given day should leave the customer on the same day (4), while only using each vehicle once per day (5). Route duration and capacity for each vehicle constraints are formulated in constraints (6) and (7), with q_i and d_i referring to the demand and duration of i and it should be noted that if the fleet is homogenous $D_k = D$ and $Q_k = Q$ for $k = 1, \dots m$. Constraint (8) eliminates subtours and constraints (9) and (10) ensure that the decision variables are binary.

2.2 Solution methods for the (P)VRP

As the VRP, even in its standard CVRP form without a requirement of assigning customers to days, can be classified as a NP-hard problem (Toth & Vigo, 2002), many different solution methods for solving the (P)VRP, have been introduced throughout the years. VRP solution

algorithms can be classified into two main subcategories, the first one being exact algorithms and the second one being approximation algorithms. Exact algorithms try to find the optimal solution by exploring all possible solutions; one commonly used example of this is the column generation algorithm. Even though exact algorithms have the advantage of being able to find the optimal solution, the complexity of the VRP results in high computational time for this class of algorithms. Hence, approximation algorithms that try to find solutions - in the best case - close to the optimum in shorter time than exact methods can be more efficient, especially for larger instances. These approximate methods can be further split up into two main classes- namely heuristics and metaheuristic approaches, where heuristics refer to problem-specific algorithms and metaheuristics to more general algorithms, which can be applied to different types of problems. Heuristics can then be further split up into construction heuristics, meant to construct a solution "from scratch", like the savings algorithm, which will be further explained in section 2.2.1, improvement heuristics, aimed at improving a given solution, like inter-route improvement and two-phased algorithms, which split the problem into two phases to reduce complexity, like cluster-first route second. Metaheuristics, on the other hand, can be divided into local searches, like TS or ALNS, where the latter will be further explained in section 2.2.2, that explore the solution space by moving the current solution to another in the solution neighborhood. The other class of metaheuristics is made up of population-based methods, like GAs, which, often inspired by natural processes, create new solutions from a set of solutions, by merging them or by learning based processes (Konstantakopoulos et al., 2022).

For the special case of the PVRP, some of the earliest attempts to solve the problem include Beltrami and Bodin (1974), which present the approach of creating routes using the CW savings algorithm and then allocating them to days, and another approach where they assign customers to days randomly and then individually route each day. Christofides and Beasley (1984), use a heuristic where they assign customers to days by prioritizing customers with limited delivery options and larger quantities and aim at minimizing costs and solving the resulting VRPs. To refine the solution, they reassign customers between delivery days and simplify the resulting VRPs by solving them as TSPs and median problems.

When it comes to metaheuristic approaches to finding a solution for the PVRP, TS, which was first presented for the PVRP by Cordeau et al. (1997) appears to be among the most influential. Cordeau et al. (1997), who use the same algorithm to solve the periodic TSP and the MDVRP, start by assigning customers randomly to delivery days according to their possible visit combinations and then use insertion heuristics to construct routes. The solution space is then explored with different neighborhood moves, for example, by moving customers to other

routes, which, in standard TS fashion, gets then, for a couple of iterations, prevented from being reinserted back into the route it was removed from. Also, infeasible solutions are permitted using a penalty function in the objective function.

After Cordeau et al. (1997), Hemmelmayr et al. (2009) present the most noteworthy improvement to the best-known solutions for the benchmark instances used and created in the former. They apply a VNS to the initial solution of the problem found with a CW savings algorithm, which consists of exploring neighborhoods and switching to a new random solution in a new neighborhood, when a local optimum is reached. Some other metaheuristic approaches to solve the problem include hybrid evolutionary algorithms (Vidal et al., 2012), who achieved to improve the solution quality compared to Hemmelmayr et al. (2009), ACO (Matos& Oliviera, 2004) and GRASP (Goncalves et al., 2005).

In the following, the basics of the heuristic and metaheuristic algorithms used in this thesis will be described. Note that the description refers to the general working mechanisms of the algorithms in relation to VRP and its variants, and not the adaptations which have to be made when handling a PVRP, as the specific adaptations and selected operators will be discussed in section 5.1.

2.2.1 Clarke and Wright savings algorithm

The savings algorithm presented by Clarke and Wright (1964) refers to, as already mentioned, a construction heuristic aimed at solving the CVRP and was used to construct the initial solution for the ALNS in this thesis. It constructs routes aimed at minimizing the total distance travelled by vehicles with a given (varying) capacity.

The algorithm begins with initial routes, where for each customer the route depot-customer-depot is created. These routes are then merged in such a way that maximum savings in cost are achieved. Hence savings refer to the reduction in travel distance if both customers are served by the same vehicle instead of separate routes. This can be formulated as $\lambda_{ij} = c_{i0} + c_{0j} - c_{ij}$, where λ_{ij} refers to the savings value of customer i and j, c_{i0} and c_{0j} to the cost of travelling to the depot from customer i and from the depot to customer j, whereas c_{ij} refers to the distance of travelling from i to j. The savings value for each pair of customers is calculated and documented in a matrix with the dimension of $(n+1) \times (n+1)$. This can technically also be done before initializing the routes, since the savings remain the same over the course of the algorithm. Then the highest element in this matrix gets chosen to decide which of the routes will be merged, while respecting vehicle capacity constraints and not merging with routes that

have already been merged in a conflicting way. Also note that routes can only be joined with nodes at the beginning of the route (after the depot) or at the end (before the depot). This is repeated until no more feasible merges can be achieved (Fikejz et al., 2024).

2.2.2 Adaptive large neighborhood search

The ALNS refers to a local search metaheuristic that starts with an initial solution s, which can be constructed via a construction heuristic or by using repair operators (will be explained in the following). This initial solution can be required to be feasible but can also be permitted to be infeasible and then calculated with a generalized cost function, which adds penalty costs based on the level of violations of constraints. The ALNS then explores the neighborhoods of the solution by partially destroying and repairing the initial solution (Voigt, 2025).

It should be noted that this framework was first introduced by Ropke and Pisinger (2006) and is based on LNS proposed by Shaw (1998). The key difference here is that LNS explores the solution space by using one destroy and one repair operator. In contrast to that ALNS chooses between multiple operators at each iteration via a roulette wheel selection, which means that each operator gets assigned a probability to be selected based on its past performance when it comes to improving the solution. Hence, if we work with k destroy or repair operators with weights $w_i \in \{1,2, \dots, k\}$, the probability of selecting an operator is:

$$\frac{w_j}{\sum_{i=1}^{k} w_i}$$

At each iteration a random number gets drawn to be compared with the operator probabilities, which are normalized and summed cumulatively respective to their scores. Please note that destroy and repair operators get selected independently of each other. After applying the selected operators, their success parameters are being updated (Ropke& Pisinger, 2006).

When it comes to destroy and repair operators, they are also referred to as removal and insertion parameters in the context of the VRP. As suggested by the name, removal operators aim at removing part of the solution, hence removing a set of customers or routes from the original solution (Voigt, 2025). Over the years a large variety of removal and insertion operators have been introduced to literature (see Voigt (2025) for an overview and classification of operators). Some examples of removal operators include randomly removing customers (random removal) (Ropke&Pisinger, 2006), and removing customers that are related, for example, by their distance to a seed customer (related removal) (Hemmelmayr et al. 2012).

Removal can also be done based on information obtained in the current solution, for example, by removing the worst-cost customer by calculating for which customer their removal results in the highest reduction of costs (Ropke& Pisinger, 2006). Insertion operators, on the other hand, aim at reinserting the customers obtained by the removal step back into the current routes. Here we must differentiate between what position a customer gets reinserted in and in what order the removed nodes are handled (Voigt, 2025). Some examples include inserting customers within their best position in random order (greedy insertion) (Lei et al., 2011), in lowest cost order (best insertion heuristic) or with highest route regret (Ropke& Pisinger, 2006). Finally, an optional local search method can be added to enhance the algorithm (Voigt, 2025).

After this process the newly derived solution can be accepted simply if it is better than the initial criterion or according to another criterion (acceptance criterion), for example, by assigning some probability of accepting worse solutions (Santini et al., 2018).

The last essential component of the ALNS needed as input is a stopping condition or criterion after which the algorithm terminates, for example, after a certain number of iterations (without improvement) or runtime (Voigt, 2025). Figure 2 summarizes the ALNS framework; note that q refers to a subset of customers N.

Input: Starting solution s
Output: Best solution s^*

$s^* \leftarrow s$
while stopping condition is no met **do**
  Remove(), Insert()← ChooseOperators()
  $(s^{new}, N) \leftarrow$ Remove (s, q)
  $s^{new} \leftarrow$ Insert (s^{new}, C')
  $s^{new} \leftarrow$ LocalSearch (s^{new})
  if $f(s^{new}) < f(s^*)$ **then**
    $s^* \leftarrow s^{new}, s \leftarrow s^{new}$
  else if accept $(f(s^{new}))$ **then**
    $s \leftarrow s^{new}$
  end
  UpdateParameters()
end

Figure 2 ALNS pseudocode (adapted from (Voigt, 2025))

3 Problem Description

In the following section, the problem of routing vehicles for collecting deposit refund waste from collection machines - at least for the Austrian case- will be explained in a bit more detail for better understanding. Note that the following information was found on the website of EWP Recycling Pfand Österreich gGmbH (central department for handling deposit waste in Austria) and could change as time evolves.

As already mentioned, each vendor selling plastic bottles and beverage cans must also employ a method that consumers are able to return the resulting waste. This can be done manually or by using a deposit collection machine, where the former is recommended for smaller vendors, specifically vendors not selling that many drinks, like drugstores, and the latter for bigger vendors, like supermarkets. For this thesis, we will concentrate exclusively on waste collection performed by the collection machines for the following reasons: First, the bags collected manually can be retrieved by third parties (like wholesalers, using their own collection systems). Secondly, because the waste collected manually needs to be counted first, it can be taken to regional collection points designated solely for counting (not sorting) or to those that serve both purposes. Consequently, it would be wise to pick them up separately from bags collected by machines. Lastly, to reduce the size of the problem while focusing on its online characteristics. Where, for the case of manual collection, the amount of waste that vendors must take back must correspond to the usual sales quantity per purchase, this is not the case for machine collection (EWP Recycling Pfand Österreich gGmbH, 2024c).

The machine collection will now be explained further. As mentioned, machines are connected to the internet and can count the number of bottles/cans, collected in waste bags provided by EWP and thrown into the machine by consumers, as well as communicate the number of collected bags to the logistics service provider. There is a minimum amount of waste needed to be picked up by the service as well as a maximum amount a store can handle. The collected bags are then brought to the depot of the logistics provider and stored there until they are picked up by EWP, to be brought to the sorting plant, where the waste is sorted according to its type (metal or plastic). Each vendor can be visited between one and five times per week. How often they get visited depends on the expected number of bags per year and the registered amount of waste counted by the collection machines and will be further elaborated in section 6.2.2 (EWP Recycling Pfand Österreich gGmbH, 2024c). Summing up all this information, we

can extend the standard PVRP model described in section 2.1 with the following information that needs to be incorporated[2]:

- If the machine demand data q'_i of customer i is below the minimum pickup threshold t_{min}, hence $q'_i < t_{min}$ he should not be picked up, which we can linearly express as, where M refers to a large constant:

$$q'_i - t_{min} < -M * (1 - x_{ijkl}) \quad (i = 1, \dots, n; j = 0, \dots, n; k = 1, \dots, m; l = 1, \dots, t)$$

- If the machine demand data q'_i of customer i has reached the maximum pickup threshold t_{max} , hence $q'_i \geq t_{max}$ he should be picked up, which we can linearly express as, where z_{jl} is a binary constant that turns one if $q'_i \geq t_{max}$:

$$z_{il} = \begin{cases} 1, \text{if } q'_i \geq t_{max} \\ 0, \quad \text{otherwise} \end{cases} \qquad (i = 1, \dots, n; \mathrm{l} = 1, \dots, t)$$

$$q'_i - t_{max} \leq M * z_{il} \quad (i = 1, \dots, n; l = 1, \dots, t)$$

$$q'_i - t_{max} \geq -M * (1 - z_{il}) \quad (i = 1, \dots, n; l = 1, \dots, t)$$

$$\sum_{j=0}^{n} \sum_{k=1}^{m} x_{ijkl} \geq z_{il} \quad (i = 1, \dots, n; l = 1, \dots t)$$

- Also the sum of the updated demands should not exceed the vehicle capacity Q_k: $\sum_{i=0}^{n} \sum_{j=0}^{n} q'_i x_{ijkl} \leq Q_k \qquad (k = 1, \dots, m; l = 1, \dots, t)$

.

[2] Please note that this notation and incorporation refer to the interpretation of the model as a PVRP, with the additional constraints mentioned above. It is also possible to apply alternative formulations for the description of the problem, for instance, in case a dynamic approach is chosen for planning the routes, hence creating them based on the current demand values without the initial PVRP routes.

4 Literature Review

In this section the application of the PVRP in the context of waste collection in literature will be discussed. Then an overview of the literature concerning the use of smart waste bins in routing problems will be given.

4.1 Periodic waste collection

Even though many articles on the routing of waste collection do not consider periodicity (see, e.g., Benjamin& Beasley, 2010; Hannan et al., 2018; Yuliza et al., 2023) or are modeled as an ARP[3] (see, e.g., Liu et al., 2014, Tirkolaee et al., 2023; Tirkolaee et al., 2018) where in this case all customers have the same frequency and customers living on the same street are generally serviced together, the literature on the PVRP and its variants also have a rich history in the context of (recycling) waste collection, for various sub-types of waste (for literature reviews on routing problems in the context of waste collection see, e.g., Hess et al., 2024; Li et al.; 2025, Liang et al., 2022). As already mentioned, even the first known publication considering a VRP with periodic requirements by Beltrami and Bodin (1974) deals with solving a waste collection problem, for the specific case of industrial garbage in New York City. Also, the paper by Matos and Oliviera (2004), which has already been mentioned in the context of solution methods in section 2.2 uses data for collecting domestic waste at 8087 sites in Viseu, Portugal, which they split into 202 geographical areas. They propose a two-step procedure by first constructing an initial infeasible solution with ACO, without considering scheduling constraints. This solution then gets transformed into a feasible one in a second step. Bommisetty et al. (1998) aim at optimizing the collection of recyclable waste at Northern Illinois University for the one truck that is available. They address the problem in terms of which buildings to collect recycling waste each week, by using a nearest insertion heuristic. Baptista et al. (2002) in their case study, solve a PVRP for the collection of recycling paper containers in Almada, Portugal. To solve the problem, they adapted the approach of Christofides and Beasley (1984), discussed earlier, such that aspects specific to their problem description, like the asymmetry of the distance matrix and a profit maximization objective are being covered.

In Angelelli and Speranza (2002), the effect on the operational costs of using three different types of waste collection systems in terms of the vehicle used is discussed. One example here is a side loader type vehicle, which only requires one operator since it

[3] The ARP refers to a problem, where with a set of predefined edges, with a certain positive (in the case of capacitated ARP) or zero demand the objective is to find best tours that cover all edges (Tirkolaee et al., 2018).

automatically empties containers. They model the problem as PVRP and solve it with a TS for the case of Val Trompia, Italy, and Antwerp, Belgium.

Teixeira et al. (2004) solve the PVRP in a case study for a region in Portugal over a planning horizon of one month and for the case separately collecting three different types of waste (glass, paper, plastic/metal) by first clustering the 1642 collection sites, then deciding the type of waste to collect for each work shift and finally selecting the sites to collect and forming routes. Each of these steps is done via problem-specific heuristics.

Another example of waste collection as a PVRP is a study by Nuortio et al. (2006), who aim to optimize municipal solid waste collection with different types of waste, required to be collected separately and varying needs for collection in two regions of Eastern Finland. They consider this problem as a PVRP with time windows since, in their case, collection and disposal must be done in a certain time frame and also note how waste collection generally is a stochastic problem since the amount of collected waste depends on various factors and address this problem by working with an average accumulation rate of waste for each separate container type using historical values. For solving the resulting problem, they use a guided variable neighborhood thresholding metaheuristic and a heuristic scheduling algorithm. Bianchi-Aguiar et al. (2012) aim at optimizing the municipal waste collection plan of Ponte de Lima, Portugal, with varying container visits, hence a PVRP. They solve the model with a decomposition approach, where they first assign customers to days, then assign customers to vehicles for each day and finally solve the resulting TSPs.

Other papers deal with specific types of waste collection problems, such as Hemmelmayr et al. (2013), who present a model and solution method based on VNS and dynamic programming for a periodic waste collection model with intermediary facilities. Also, more specific types of waste, with certain requirements, can be considered in PVRP problems. Examples of this include Coene et al. (2010), who consider the case of animal waste collected from supermarkets, slaughterhouses and butchers by a Belgium-based company, where a differentiation between low- and high-risk waste must be made since both types cannot be transported in the same vehicle. Another example is the case of infectious medical waste collected from hospitals and brought to an incinerator; here Shih and Chang (2001) solve this problem for 384 hospitals in Tainan, Taiwan, via a two- step approach where they first determine a set of individual routes for each vehicle via dynamic programming and then assign the routes to days of the week with mixed integer programming.

4.2 Waste collection routing problems with smart bins

In recent years, the use of smart waste bins, hence, in most cases, bins that are equipped with sensors to communicate fill levels to decision-makers, such that only full bins are collected and in order to deal with the stochastic nature of waste collection problems, that has already been mentioned. Many papers consider this problem as a DVRP, hence updating or constructing routes dynamically during execution. One early example of this is Johansson (2006), who aims at evaluating operational costs of different dynamic and static routing and scheduling policies using real-time data from bins equipped with smart sensors in a municipality in Sweden. Faccio et al. (2011), who suggest a vehicle routing model that uses real-time input for the replenishment level of bins, for the vehicles, for the bins visited and for the location of the vehicle, to minimize the number of vehicles, travel time and travelled distance, while respecting the capacity of bins, among other constraints such as time windows. They introduce two parameters, one communicating the maximum percentage of bins allowed to exceed capacity and another for the optimal replenishment level of a bin. This aids in the decision of which bins to service, after which routes are optimized, using the real-time traceability data. This novel framework gets solved via a heuristic method and tested using data from a city in Italy.

Anagnostopoulos et al. (2015) use real-time information in a dynamic waste collection problem to aid with immediately collecting waste at sites with a high priority, which are equipped with sensors, such as hospitals. They suggest and test four different types of models and algorithms to deal with high-priority bins. Ramos et al. (2018) evaluate three different managerial approaches when it comes to dynamically defining routes for waste collection. Whereas the first approach is based on the heuristic presented by Faccio et al. (2011). They propose enhancing this method, by also considering service levels and a method which adds a heuristic to the enhanced method that decides on which day the model should be run for profit maximization. The methods are tested via a case study for recyclable waste collection in 14 municipalities in Portugal.

Jorge et al. (2022) propose a hybrid-metaheuristic method to solve a smart waste collection problem with workload concerns, which in their case refers to maximum shift duration of workers and route workload balance. The suggested metaheuristic approach consists of a look-ahead heuristic deciding which days collection is necessary and the bins that need to be emptied, as well as a SA/NS based algorithm to decide which bins are profitable and for route planning. They report a profit increase case study for a waste collection company based in Portugal if the method were to be implemented.

Other case studies of smart waste collection problems include the case of Casablanca (Idrissi et al., 2024) and South Korea (Roy et al., 2022), where for the latter a time dependent penalty concept is introduced, making sure full waste bins are emptied in a timely manner. For solving this problem, they also use a hybrid-metaheuristic approach based on VNS and ACO.

Akhtar et al. (2017) suggest an adapted BSA to be used when combining CVRP with smart bins. They put forth the idea of a threshold waste level for waste bins, aiming to decrease the number of bins that need emptying by identifying an optimal range and thereby minimizing distance alongside a model for scheduling decisions. Hannan et al. (2018) also make use of the suggested threshold waste level and scheduling to optimize route efficiency in a smart waste collection CVRP model. They propose a modified version of a particle swarm algorithm to solve the problem. Both Akhtar et al. (2017) and Hannan et al. (2018) found that a threshold fill level of 70% to 75% performed the best in their research.

Even though considered implemented in other work (see e.g., Faccio et al., Akhtar et al., 2017; Hannan et al., 2018), some recent work places an even greater emphasis on including environmental objectives into waste collection with smart bins. One example is Lu et al. (2020) who propose a sensor-based system to classify and collect waste to enhance sustainability and optimize the workload of vehicles. They introduce a new hybrid algorithm based on GA and whale optimization to solve the model and test it with a real-world case of an area in Shanghai. Similar to Anagnostopoulos et al. (2015), Wu et al. (2020) also consider a waste collection problem with priority considerations for places such as hospitals while including environmental concerns, for which a hybrid algorithm based on PSO and SA is implemented.

Salehi-Amiri et al. (2022), aim to minimize pollution and recovery value of waste in their smart waste collection problem, where they consider the special case of a two-echelon problem, in which waste is first transported in a low-capacity vehicle, brought it to a separation center and afterwards in a higher capacity vehicle to a recycling center. They approach this as a two-stage problem and consider uncertainties to handle the varying conditions and qualities of waste. After modelling, they test different meta-heuristic algorithms, such as SA and SEO. Rahmanifar et al. (2023) also discuss a similar two-echelon model considering CO2 emissions and costs for which problem they present several new heuristics to solve it, alongside metaheuristic methods (SA, SEO).

Another specific case discussed in recent literature on waste collection with smart bins is the use of multi-compartment vehicles, such that different types of waste can be transported in the same vehicle. Bouleft and Elhilali (2023) discuss the case on DVRP with multi-department vehicles, for which they use a hybridized GA to solve the resulting problem.

Mohammadi et al. (2023), also suggest using a DCM in addition, to evaluate at each point of decision making the likelihood of selecting a geographical area to be visited based on fill levels and cost of travel in their DVRP waste collection problem with multi-compartment vehicles. They propose a hybrid algorithm based on GA and PSO to solve it.

This thesis adds to the existing body of literature by first determining the PVRP plan with ALNS for the case of deposit-refund waste collection and then adapting it to daily changes in customer demand. Which means that instead of using a fully dynamic or static approach, a "middle-ground" between those two approaches is presented.

5 Solution method

As already mentioned, to solve the deposit-refund waste collection problem, a two-step procedure was implemented. The first step being a routing plan optimized with the expected frequencies and demand values of customers, done with an ALNS metaheuristic approach. This is done to achieve a good initial plan that serves as the basis for serving the customers. The second step then uses a simple heuristic at the beginning of each day to adapt the routes based on the data received from the collection machines, since at the time of planning the initial routes the exact values to be collected are not known, due to the stochastic nature of waste collection. For the initial plan, our objective was to minimize the total distance of routes. The second step aims to incorporate new circumstances, such as bags exceeding vehicles or customers reaching the maximum amount of waste they can store. In this chapter the proposed solution method will be discussed in detail. Note that for all solution methods the capacity Q_k of vehicle k was assumed to be the same for all vehicles, hence $Q_k = Q$.

5.1 ALNS Algorithm

As discussed in section 2.2.2, the ALNS works with an initial solution which gets iteratively destroyed and repaired using different problem-specific operators that get selected with a probability, as well as an acceptance and stopping criterion and an optional local search. All the chosen components and their implementation logic will be described in the following.

5.1.1 Initial Solution construction

For constructing the initial solution to be then optimized with the ALNS, an adapted CW savings algorithm was implemented. Note that a description of the general mechanism of the savings algorithm can be found in section 2.2.1**.** To determine which customer is routed on which day(s), similar to other authors (see, e.g., Beltrami& Bodin, 1974; Hemmelmayr et al., 2009), customers are assigned to days randomly, and then each day is routed like a standard CVRP. This is based on possible visit combinations C_i of each customer $i \in N$ which are stated in the problem data; hence for all randomly chosen i one combination $r \in C_i$ gets chosen randomly. The customer then gets added to each of the days l included in r. Then the savings algorithm is solved every day in the planning horizon t. While our savings are computed as in Clarke and Wright (1964), our tour-constructing mechanism differs in some aspects. Instead of initializing each customer in a tour containing the depot, we construct routes dynamically by iterating over the sorted savings list. Hence, we can differentiate between three cases for merging tours: First, both customers i and j in a savings pair λ_{ij} have not been assigned to a

tour, second, one of them has already been assigned to a tour, third, both have been assigned, but to different tours. Note that, as mentioned, tours can only be merged directly after or directly before the depot; the same logic applies to unassigned customers. Also note that customers can only be assigned to new tours once. Since some customers might not be able to be routed using the savings algorithm procedure without violating capacity constraints, especially since the random allocation to routes does not consider total capacity each day, a backup strategy needed to be implemented; hence it was decided to run the saving algorithm until there are no unassigned customers, which gets determined with a helper method that checks how many customers are not assigned according to their frequency. The summarized pseudocode for the savings algorithm used in this thesis can be found in figure 3.

Input: set of customers N, number of vehicles m, vehicle capacity Q, number of days t
Output: Initial solution s

```
1   s ← ∅
2   while |unassigned Customers| > 0 do
3   s^new ← ∅
4       foreach customer i ∈ I do
5         AssignCustomersToDays()    //randomly based on possible combinations
6       foreach day l ← 1 to t do
7         s^new ←SolveSavings (N, m, l)
8           foreach customer pair i, j
9           Calculate savings λ_ij = c_i0 + c_0j − c_ij
10          Sort  λ_ij in descending order
11          foreach pair i, j in sorted savings do
12            if both i, j are unassigned && q_i + q_j ≤ Q   //demand
13              merge to tour
14            else if i is unassigned&& j is assigned||i is assigned && j is unassigned
15              if tour of assigned customer ends or starts with the customer
16                merge to tour
17            else if i is assigned&& j is assigned to different tours
18              if tour of i start with i&& tour of j ends with j|| ends with i && starts with j
19                merge to tour
20        until no more feasible merges found
21  s ← s^new
22  return s
```

Figure 3 Savings Algorithm Pseudocode

5.1.2 ALNS general information

The number of times the ALNS algorithm proposed in this thesis is performed depends on a value defined by the decision-maker; in other words, the stopping criterion is the number of iterations K. It was chosen, since this criterion allows for intuitive adaptation based on desired runtime, quality of solution and given instance size.

For the acceptance of a new solution, a mechanism that allows us to also accept a solution s^{new} that is slightly worse than the so far best solution s^*, this helps to diversify the solution space and escape local optima. To achieve this, an SA-based acceptance criterium was implemented, which will be described in the following. Note that the SA acceptance criterium is very commonly used for the ALNS and was also implemented in the first introduction of the ALNS by Ropke and Pisinger (2006). The formulas used were adapted from Santini et al. (2018), they describe that when working with an SA algorithm a new solution is selected based on the probability:

$$e^{\frac{f(s^{new})-f(s^*)}{T}}$$

Using a reference value of the solution z, which in our case was defined as the total distance of the best initial solution found with the savings algorithm, the temperature T can be set to:

$$p = e^{\frac{z-hz}{T}} \Rightarrow \ln p = \frac{z(1-h)}{T} \Rightarrow T = \frac{z(1-h)}{\ln p}$$

Where $p \in [1,0]$ is the probability with which new solutions of cost $f(s^{new}) = h \cdot z$ are accepted. In our case a probability of 0.5 was chosen, hence:

$$T = \frac{z(1-h)}{\ln 0.5}$$

The value h defines how much worse a solution can be to be eligible to be selected with probability p, can be defined by the user for two cases: at the start of the ALNS h^{start}, hence in the first iteration and at the end h^{end}. These values can be defined by the user and result in a starting temperature T^{start} and a final temperature T^{end}. To determine the cooling rate, which influences how the temperature is updated, i.e., decreased after each iteration, the following formula was implemented, where K refers to the number of iterations (Santini et al., 2018):

$$T \leftarrow T \cdot (T^{end}/T^{start})^{1/K}$$

It should be mentioned that for our algorithm a new solution s^{new} gets accepted if and only if there are no unassigned customers in s^{new}, which gets validated by checking if each customer i is assigned respective to his frequency e_i. The SA-based acceptance criterium is summarized in figure 4.

Input: Current solution s, best solution s^*, number of iterations K, initial temperature T^{start}
Output: s^*

For $k \leftarrow 1$ *to* K **do**
  Generate neighborhood solution s^{new}
  if no unassigned customer **then**
    if $f(s^{new}) < f(s) \parallel \text{rand}(0,1) \leq e^{\frac{f(s)-f(s^{new})}{T}}$ **then**
      $s \leftarrow s^{new}$
      if $f(s^{new}) < f(s^*)$ **then**
        $s^* \leftarrow s^{new}$
    else $s \leftarrow s^*$
  else $s \leftarrow s^*$
  update(T)
return s^*

Figure 4 Simulated Annealing acceptance

Another component concerning the general working mechanisms of our ALNS implementation that should be mentioned refers to the way in which the success scores of removal and insertion operators are updated. In our case, the success rates are only increased if the newly found solution s^{new} of an iteration is strictly better than the overall best solution s^*, hence if a worse solution is accepted due to simulated annealing or if the new solution s^{new} is better than the current solution s, but not better than s^* the success counter of parameters do not get updated. Note that this is contrary to Ropke and Pisinger (2006), who include these cases when updating success scores. If the new solution was not eligible for selection we use the best solution for further processing.

Last, it should be noted that in our implementation the number of customers that get removed from the solution is determined dynamically instead of being set to the same static value throughout the ALNS procedure. To do so the removal percentage $q \in [0,1]$ decreases dynamically based on the current iteration, which results again in more diversification in the beginning and intensification in the end. With maximum removal percentage q_{max} and the minimum q_{min} -both defined by the user-, number of iterations K and current iteration k , the formula implemented to do so is:

$$q = q_{max} - \frac{k}{K} \cdot (q_{max} - q_{min})$$

5.1.3 Removal operators

A total of three removal operators were used in the final ALNS algorithm, namely two different random removal operators and a worst removal operator. The selected operators are inspired by (some of) the (A)LNS operators mentioned in Ropke and Pisinger (2006) but have been updated to better suit the periodic requirements in this thesis. The reasons for selecting the heuristics presented in the following were having a good balance between diversification and intensification, which is achieved by using random removal operators, adding a stochastic element alongside worst removal. Also, their ease of implementation, as well as performance in experiments, were considered as criteria for inclusion in the algorithm. It was decided against using more than three operators to allow for a focused selection of them based on their success scores and their respective weights in the roulette wheel selection process.

The first random removal operator is referred to as partial random removal, since for this operator we randomly select customers to be removed from a random day they are planned for, hence not removing the customer from all days and all tours. This is implemented by creating a new set of customers N' based on the days and tours the customers are currently assigned to, meaning that each customer i is as many times in N' as they are in the solution (since we only accept solutions without unassigned customers this is the same number as their frequency e_i), and then randomly selects a subset of removed customers R of size $q \cdot |N'|$, where q is the removal percentage. Pseudocode for this heuristic can be seen in figure 5.

Input: current working solution s, removal percentage q
Output: removed customers R

```
1  N' ← ∅
2  foreach day l in s do
3    foreach tour t in l do
4      foreach customer i in t with i ≠ depot do
5        N' ← N'{(l, t, i)}
6        R ← randomly select q * |N'| elements from N'
7  foreach (l, t, i) ∈ R do
8    remove i from tour t on day l
9  return R
```

Figure 5 Partial random removal heuristic

The second removal operator, namely random removal, is a generalization of the partial random removal operator, with the key difference being that instead of removing customers from only one randomly selected occurrence, here customers are removed from all tours and days they are

assigned to. To implement this, the set of all customers N' (all customers excluding the depot) is considered, and a random subset $q \cdot |N'|$, is selected from it, where q refers to the removal percentage. Each customer is then removed from every tour and day they occur. To avoid issues when processing the removed customers with an insertion heuristic, each removed customer is added to the removal set according to their frequency e_i. Pseudocode for the random removal heuristic can be seen in figure 6.

Input: current working solution s, removal percentage q, set of customers N
Output: removed customers R

```
1  N' ← N \ {depot}
2  R ← ∅
3  N'' ← randomly select q · |N'| from N'
4  foreach customer i in N'' do
5    foreach day l in s do
6      foreach tour t in l do
7        if i ∈ t then
8          remove i from tour t on day l
9    for i ← 1 to e_i do
10     add i to R
11 return R
```

Figure 6 Random removal heuristic

Lastly, an adapted worst removal heuristic was implemented. Due to the periodic structure of the problem, each customer is associated with different costs across all days and tours they are assigned to. Hence, we can calculate the marginal cost Δ_i for each customer i, which represents the additional distance which is caused by including i in a tour, compared to the distance of the same tour without i. This value is calculated for every occurrence of i and then summed up and averaged to obtain an average for each customer in the subset N' that refers to $N' = N \setminus \{depot\}$. Then a smaller subset of $q \cdot |N'|$ customers with the highest average Δ_i is removed from all tours and days, where q refers to the removal percentage. Note that as in the random removal heuristic, customers are then added to the list of removed customers R according to their frequency for consistency. Pseudo-code for the worst removal operator can be seen in figure 7.

Input: current working solution s, removal percentage q
Output: removed customers R

```
1  R ← ∅
2  foreach day l in s do
3    foreach tour t in l do
4      foreach customer i ≠ depot in t do
5        Δ_i ← dist(i − 1, i) + dist(i, i + 1) − dist(i − 1, i + 1)
6        avg_Δi ← compute average Δ_i for each customer i
7        N'' ← select top q · |N'| customers in N' with highest avg_Δi
8  foreach i in N'' do
9    foreach day l in s do
10     foreach tour t in l do
11       if i ∈ t then
12         remove i from tour t on day l
13   for i ← 1 to e_i do
14     add i to R
15   return R
```

Figure 7 Worst removal heuristic

5.1.4 Insertion operators

Since ALNS requires methods to reinsert customers removed via a removal heuristic, two insertion heuristics were implemented in the final version of the algorithm presented in this thesis, namely a cheapest insertion in random order and a regret-based order insertion heuristic. The two different methods are again inspired by Ropke and Pisinger (2006) and are described in more detail in the following. Note that for the implementation of both heuristics, a Boolean helping method which checks whether or not a customer can be inserted into a tour based on all his visit combinations C_i was used. The method first determines the set of days on which customer i is already in the current working solution s. If the customer is assigned to no days in the current solution, an insertion is considered valid as long as a given day l is included in any of the visit combinations $r \in C_i$. In all other cases hence, if the customer is already scheduled for any of the given days and tour in s an insertion is accepted if and only if the resulting set of assigned days is part of a visit combination. Note that for runtime optimization this method might be skipped, hence always return true for customers with $e_i = 1$ if they can be assigned to any day and for customers where frequency e_i=t, where t refers to the total number of days. In the latter case a check if a customer has already been assigned on a given

day should be performed before the method to validate the insertion. The pseudocode for this method is provided in figure 8.

Input: customer i, day l, current working solution s
Output: true if insertion is valid, false otherwise

```
1  assignedDays ← days where customer i is assigned
2  if assignedDays = ∅ then
3    if there is a visit combination r in possible visit combinations C_i where l ∈ r then
4      return true
5  else
6  simulateInsertion ← assignedDays ∪ {l}
7    if in there is a r ∈ C_i such that simulateInsertion ⊆ r then
8      return true
9  return false
```

Figure 8 Verify insertion method

The cheapest insertion in random order heuristic used in this thesis aims to reinsert customers in the current working solution s into their respective minimum cost position. In other words, for every customer $i \in R$, where R is the set of removed customers, all possible positions in between two consecutive nodes are evaluated for each day $l = 1, \dots, t$ in the planning horizon, given a day is eligible for insertion. To diversify the order in which nodes are inserted throughout the ALNS iterations, R is always evaluated in random order. After that, similar to the worst removal method, the marginal cost of insertion, hence the change of the objective function when inserting c is $\triangle f_{c,l,t=}dist(i,c) + dist(c,j) - dist(i,j)$, where i and j refer to two consecutive customers within a tour t on a day l. We can than define the minimum cost position for each customer c as $m_c = min_{l=1,\dots,t;t\in T}\{\Delta f_{c,l,t}\}$ (Ropke& Pisinger, 2006). Contrary to Ropke and Pisinger (2006), who compare all min_c and choose the customer with the lowest m_c to be inserted in the respective position, we simply add every customer to their position with m_c in random order, which could help with diversification. Pseudocode for the implementation of the cheapest insertion heuristic can be seen in figure 9.Note that we refer to costs as distance since this is our objective.

Input: set of removed customers R, current solution s (without R)
Output: updated solution s with reinserted customers

```
1  Shuffle R randomly
2    foreach customer c ∈ R do
3      cheapestDay ← −1
4      cheapestTour ← null
5      cheapestPosition ← −1
6      minDistance ← ∞
7    foreach day l = 1 ← t in s do
8      if c is already inserted on day l then
9        continue
10     if IsValidInsertion(c, l, s) = false then
11       continue
12     foreach tour t in day l do
13       if c ∪ t ≤ Q then   //vehicle capacity
14         for each pair of consecutive customers (i, j) in t do
15           △ f_{c,l,t=} dist(i, c) + dist(c, j) − dist(i, j)
16           if △ f_{c,l,t=} Δ < bestCost then
17             update minDistance, cheapestDay, cheapestTour, cheapestPosition
18     If cheapestTour ≠ null then
19       insert c at cheapestPosition in cheapestTour on cheapestDay in s
20   return s
```

Figure 9 Cheapest insertion random order heuristic

Since cheapest insertion leads to some customers c with high m_c being postponed to the last iterations, which then leads to a lack of opportunities for inserting these expensive requests, since many tours might be full at the end of the insertion process. One method to combat this issue is using a regret-based order heuristic, where the regret value is defined as the difference in costs resulting from inserting a customer in his best position versus his second-best position, in other words, for the PVRP the regret value $m_c{}^*$ can be denoted as $m_c{}^* = \triangle f_{c,x_{c2}} - \triangle f_{c,x_{c1}}$ where x_{ck} for $k \in \{1, \ldots, m\}$ refers to the day and route for which the insertion of customer c has the k-lowest insertion cost, where in this case k is 2. Out of these $m_c{}^*$ the maximum value is chosen, and the respective customer is inserted in their best regret position. Note that any number of regret positions might be considered by simply summing up the difference between the best insertion and the k-lowest insertion costs; in our case only two positions where considered (Ropke& Pisinger, 2006). Pseudocode for the regret-based order insertion heuristic can be seen in figure 10.

Input: set of removed customers R, current working solution s (without R)
Output: updated solution s with reinserted customers

```
1  while |R| < 0 do
2      highestRegretCustomer ← null
3      highestRegretPosition ← −1
4      highestRegretDay ← −1
5      highestRegretTour ← null
6      highestRegret ← −∞
7     foreach customer c ∈ R do
8       L ← ∅      //insertion cost List
9        foreach day l = 1 ← t do
10         if already inserted on l then
11          continue
12         if IsValidInsertion(c, l, s) = false then
13          continue
14          foreach tour t in day l do
15            if c ∪ t ≤ Q then
16              foreach pair of consecutive nodes (i, j) in t do
17                  △ f_{c,l,t=} dist(i, c) + dist(c, j) − dist(i, j)
18                    add (△ f_{c,l,t=} c, l, t, position i + 1) to L
19          if |L| < 2 then
20            continue
21         sort L by △ f_{c,l,t=} ascending
22         regret ←sum of the cost difference between L(0) and L(1)
23         if regret > highestRegret then
24           save c, l, t, and position
25       if no feasible insertion found then
26            break
27    insert highestRegretCustomer at highestRegretPosition in highestRegretTour on
        highestRegretDay in s
28    remove highestRegretCustomer from R
29   return s
```

Figure 10 Regret based order insertion heuristic

5.1.5 Local search

To further optimize the solutions obtained with the removal and insertion heuristics after the customer insertion in each iteration of the ALNS, a 2-opt heuristic is performed on each of the tours. 2-opt refers to a local search heuristic which, given a tour or set of tours, tries to iteratively replace two edges of a tour with two other edges in the same tour, as long as it results in a shorter total distance, hence comparing the current edge costs with the costs obtained when reconnecting the nodes. Hence, for each tour, for all pairs of customers i and j where $(i < j)$, excluding the depot, we calculate $\triangle_{i,j} = [dist(i-1,j) + dist(i,j+1)] - [dist(i-1,i) + dist(j,j+1)]$. If $\triangle_{i,j}$ is < 0, we reverse the segment $[i,j]$ (Englert et al., 2014). Note that since we are dealing with the PVRP, the 2-opt is applied to every day and all tours of the respective day. Pseudocode for the 2-opt can be seen in figure 11.

Input: tour t
Output: 2-opt optimized tour t
improvement ← true
1 **while** improvement true **do**
2 improvement ← false
3 **if** number of nodes n in $t < 4$ **then** //only for tours with more than 3 customers
4 **return**
5 **for** $i \leftarrow 1$ to $n - 3$ **do**
6 **for** $j \leftarrow i + 1$ *to* $n - 2$ **do**
7 currentDistance ← $dist(i-1,i) + dist(j,j+1)$
8 newDistance ← $dist(i-1,j) + dist(i,j+1)$
9 **if** newDistance < currentDistance **then**
10 reverse segment $[i,j]$ in tour t
11 improvement ← true
12 **return** t

Figure 11 2-opt heuristic

5.2 Adaptation heuristic

As mentioned for the second part of the solution method, an adaptation heuristic that adapts the initial routes found by the ALNS each day before tour execution based on the data from the collection machines. The general framework here aims to first exclude customers from tours which are under the minimum amount of waste required for pickup, then we incorporate customers who have reached their maximum number of stored bags earlier than planned, followed by adapting vehicles exceeding their capacity with the current number of bags in a tour. Lastly, a 2-opt local search is applied for the potential improvement of adapted routes.

The individual steps will be discussed in the following. This process is done for all days and weeks in the planning horizon before route execution. Please note that for executing tests on the adaptation heuristic, a waste-data simulation is needed, which is described in section 6.2.5. In the following, we will be referring to waste levels consisting of bags and the amount of waste in the currently used bag stemming from this simulation method.

5.2.1 Removing customers under minimum pickup threshold

The reason for first removing customers who have not reached their minimum pickup threshold t_{min} is that it makes sense to first remove customers to gain new capacity before inserting customers or adapting vehicle capacity. Note that the minimum pickup threshold is set by the user or, in consequence, the decision-maker, which affects how many customers can be removed from the final tours. The method in general works very straightforwardly by first copying the initial tours from the ALNS for a given day l since it is the first method. Then the customers i with current demand in bags q'_i under t_{min} are collected and removed from the tours. Pseudo-code for this method can be seen in figure 12.

Input: initialTours, adaptedTours, customers with updated demands N', currentDay l, minPickupThreshold t_{min}
Output: adaptedTours

```
1  copy tours from initialTours[l] into adaptedTours[l]
2  customersToRemove R ← ∅
3  foreach customer i in adaptedTours[l] do
4    if i ≠ depot and updated demand q'_i < t_min then
5      Add i to R
6  foreach tour t in adaptedTours[l] do
7    remove all i ∈ R
8  return adaptedTours
```

Figure 12 Removing customers under minimum threshold

5.2.2 Inserting customers over maximum threshold

As noted, this method aims to insert customers who have reached the maximum threshold t_{max} for waste collection earlier than planned. Therefore, this method applies only to customers who are not included in any route for each given current day l that reach or exceed this threshold. Like the first adaptation method, we first collect customers for which this is the case and then insert them using either the cheapest or regret insertion, where cheapest insertion is only used if there are fewer customers to be inserted than regret positions (in our case this is two9

considered. Since we only adapt one day at once, the insertion methods are only performed for one day, instead of how it is done in the ALNS, where we consider all days. Another change is that instead of restricting possible positions for insertions only to tours with free capacity, we allow slight capacity violations. Customers can be inserted into a tour t of vehicle k if it has some free capacity, therefore $q'_t < Q$, with q'_t being the total demand of a tour of vehicle k, even if adding the customer would result in a larger total number of bags than Q. The reason for allowing this violation is that we adapt vehicles with capacity violations afterwards. By only allowing for insertions with free capacity, some good insertions could be overlooked, or if all vehicles are almost full, inserting a customer could be infeasible. Therefore, it could be a good choice to allow violations and adapt capacity afterwards. Note that we also don't allow customers to be inserted into empty tours. Apart from the small changes just mentioned, the cheapest insertion and regret insertion apply the same logic as discussed in section 5.1.4, therefore we refrain from explaining it in more detail here and showing pseudo-code for the insertion methods. Pseudocode for the general method for adapting customers over the maximum threshold can be seen in figure 13.

Input: adaptedTours, customers with updated demands N', day l, vehicleCapacity Q, maxPickupThreshold t_{max}
Output: updated adaptedTours

```
1  customersToInsert R ← ∅
2  foreach customer i in N' do
3    if i not assigned in any tour on l and q'_t ≥ t_max then
4      R ← R ∪ {i}
5  if |R| < 2 then        // number of regret positions considered
6  CheapestInsertion(adaptedTours, R, l, Q)
7  else
8  RegretInsertion(adaptedTours, R, l, Q)
9  return adaptedTours
```

Figure 13 Insert customers over maximum threshold

5.2.3 Adaptation of vehicle capacity

After inserting customers who have reached their maximum threshold earlier, the capacity Q of some vehicles might be exceeded with the updated demand data of customers initially planned for or customers added during adaptation. Therefore, for each vehicle k with tour t where $q'_t > Q$, we first try to relocate or remove customers until $q'_t \leq Q$, where q'_t refers to the total updated demand of a tour. This is done iteratively by first trying to relocate a customer $i \in t$ in

another tour $t' \neq t$, without increasing the total distance or exceeding Q. The position for this insertion gets calculated via cheapest insertion, similar to the methods discussed before, and as mentioned, no increase is allowed, hence distance of tour t and u after inserting i in u should be smaller or equal than before inserting i therefore $D_{t'}+D_{u'} \leq D_t+D_u$. If this is not possible for any customer in t, a i with q'_i under target threshold t_{tar} which we refer to as the minimum of a range for the ideal number of bags collected per pickup. Not to be confused with t_{min} mentioned earlier. If this is also not possible, customers, which have not reached their maximum capacity, therefore under t_{max} are removed, these values are all set by the decision maker. Note that for both removal steps, the customer i with the lowest demand q'_i gets removed first. Pseudocode for the general vehicle capacity adaptation and the relocation heuristic can be seen in figure 14 and 15.

Input: adaptedTours, updatedCustomers N', day l, vehicle capacity Q, target pickup threshold t_{tar}, maximum pickup threshold t_{max}
Output: adaptedTours

```
1  foreach tour t ∈ adaptedTours[l] do
2    while q'_t > Q do
3      success = false
4      success ← relocateCustomer(adaptedTours, N', t, l, Q)  //true if relocation
5      if ! success then
6        removed = false
7        R ← {i ∈ N' < t_tar}
8        if |R| > 0
9          order R by increasing q'_i
10         Remove first i in R
11         removed← true
12           if ! removed then
13             R ← {i ∈ N' < t_max}
14             if |R| > 0
15               order R by increasing q'_i
16               Remove first i in R
17               removed← true
18       success = removed
19     if ! success then     //no adaptation possible
20       break
21   return adaptedTours
```

Figure 14 Adaptation of tours over vehicle capacity

Input: adaptedTours, updatedCustomers N', current tour over capacity t, vehicle capacity Q, day l
Output: true if a relocation successful,
false otherwise

```
1  foreach customer i / {depot} ∈ t do
2   D_t ← current distance t
3   D_t' ← distance without i
4   q'_t' ← updated demand without i
5    foreach other tour u ∈ adaptedTours[l] where u ≠ t do
6     if q'_u + q'_t > Q then
7       Continue
8     cheapestDistance ← ∞
9     cheapestPosition ← 0
10    foreach valid insertion position i in u do
11     Δ = dist(i − 1, c) + dist(c, i) − dist(i − 1, i)
12      if Δ < cheapestDistance then
13       cheapestDistance ← Δ
14       cheapestPosition ← i
15       D_u ← original distance u
16       D_u' ← distance with c inserted at bestPosition
17    if D_u' + D_t' ≤ D_u + D_t then
18      update demands of t and u
19    return true
20   else
21   return false
```

Figure 15 Relocation of customers heuristic

5.2.4 Local search

The last step after adapting each tour is to apply a 2-opt local search to the adapted tours. Since 2-opt has also been applied during the ALNS process, improvements should mainly happen in tours where insertions have taken place during the adaptation process. Hence, this method aims to improve such tours. Because 2-opt has already been discussed, we refrain from showing pseudocode and refer to section 5.2.4.

6 Data

In this chapter, the benchmark instances for the PVRP used to test the effectiveness of the ALNS algorithm will be described further, as well as the example data used to illustrate the ALNS solution for the periodic plan and the adaptation heuristic making up the second step of the solution method. Note that the example data does not aim to be a one hundred percent accurate depiction of reality, since some general as well as customer information could not be obtained. Hence, it is hypothetical data that also comes with underlying assumptions to reduce complexity, which will be elaborated further on in this chapter.

6.1 Benchmark instances

The benchmark instances used to test the first part of the solution methods are a selection from the list of benchmark instances used in Cordeau et al. (1997). Hence, we refer to them as Cordeau_al_number. It should be mentioned that instances originally stem from other authors and in some cases, are adapted VRP sets. Cordeau et al. (1997) have made small adaptations to two of the original sets, only one of them, Cordeau_al_11 being used in this thesis. For this data, set the original data by Russell and Igo (1979), some of the customers have varying demands for each day. To combat this, Cordeau et al. (1997) introduce customers with the same location for each of the possible demand options that each require a visit on a different day. Also, one customer was to be visited on day six in the original data, for which Cordeau et al. (1997) account by introducing an additional customer. For this thesis a total of nine benchmark instances out of the 32 used in Cordeau et al. (1997) were selected to be tested. It was decided against testing all of them since running each of the sets multiple times with a high number of iterations would result in a total computation time exceeding the scope of this thesis. Also, for some instances, due to tight capacity constraints, finding a feasible initial solution with the savings algorithm proposed in this thesis was not possible. Other criteria, apart from feasibility for the selection of benchmark instances set for further testing, were the number of days; we preferred instances with five collection days since this is also the number of days for our case of the deposit refund waste collection (EWP Recycling Pfand Österreich gGmbH, 2024b). Also, varying sizes of sets ranging from 50 to 417 customers were considered to see how the ALNS algorithm performs depending on the number of customers.

As for general information provided by the datasets, the number of vehicles, customers and days is given, as well as the capacity of vehicles and maximum tour length. Apart from the customer ID and their x and y coordinates, for each customer their service duration, demand, frequency of visits, number of possible visit combinations, and a list of possible visit

combinations are listed as customer information. It should be noted that each of the instances used does not include a maximum tour length. Hence, service time is zero and assumes that each vehicle has the same capacity. Cordeau et al. (1997) encode the various visit combinations with decimal equivalents to their related binary bit string. For instance, for a period of five days, code 10 equals the visit combination 01010, meaning the customer can be visited on days 2 and 4 (Dorronsoro, n.d.). The sets have been downloaded from http://www.vrp-rep.org/ (VRP-REP, 2017) and are summarized in table 1.

Table 1 Benchmark instances

Name	**#Customers**	**#Vehicles**	**Capacity**	**Days**	**Origin VRP instance**	**Adapted to/ origin PVRP instance**
Cordeau_al_02	50	3	160	5	Eilon et al. (1971)	Christofides and Beasley (1984)
Cordeau_al_03	50	1	160	5	Eilon et al. (1971)	Christofides and Beasley (1984)
Cordeau_al_05	75	6	140	5	Eilon et al. (1971)	Christofides and Beasley (1984)
Cordeau_al_08	100	5	200	5	Eilon et al. (1971)	Christofides and Beasley (1984)
Cordeau_al_10	100	4	200	5	Eilon et al. (1971)	Christofides and Beasley (1984)
Cordeau_al_11	139	4	235	5	-	Russell and Igo (1979)
Cordeau_al_12	163	3	140	5	Cook and Russell (1978)	Russell and Gribbin (1991)
Cordeau_al_13	417	9	2000	7	-	Russell and Gribbin (1991)
Cordeau_al_20	184	4	60	4	-	Chao et al. (1995)

6.2 Illustrative example data

For the illustrative example data information obtained by the EWP, statistical data and coordinates of supermarkets in Lower Austria were used to create six data sets of varying sizes, which are summarized in table 6. They include the same structure of information as the data in

Cordeau et al. (1997) with an additional column for the weekly amount of waste collected since this is needed for simulating daily demand. The details of the data collection process will be elaborated in the following. Note that the demand data and frequency determined for customers is the one being used to determine the routes in the first stage of the solution process and might change for the second stage of the solution when using the (simulated) machine data.

6.2.1 General information

The EWP states that collection of waste is done from Monday till Friday. Hence, the planning horizon t is five days (EWP Recycling Pfand Österreich gGmbH, 2024b). The capacity of the vehicles used results from the approximate number of bags fitting into a truck with 13,6m length, 2,45m width and 2,5m height of the loading compartment. Note that in reality other trucks or varying size trucks might be used. With the measurements given, the resulting volume is 83300 litres (LKW Walter, n.d.). EWP states three possible types of bag sizes available for different types of collection machines being used, namely half pallet size, EURO pallet size and rolling containers (EWP Recycling Pfand Österreich gGmbH, 2024b). To reduce complexity, it was assumed that each customer has a machine with medium-sized bags, hence EURO pallet size. The associated bag size is 85x65x170 (in centimetres) resulting in a volume of 1700 liters (EWP Recycling Pfand Österreich gGmbH, 2024c). Therefore, each truck can transport roughly 50 bags, assuming they can be stacked, which is the case for waste. The number of trucks used was determined dynamically during the solution process, therefore set to the minimum number of trucks needed to achieve a feasible initial solution with the savings algorithm. Note that like the benchmark instances used in Cordeau et al. (1997), we did not include a tour length or service time constraint in our data sets.

6.2.2 Customer information for Lower Austria

For obtaining customer information - in our case supermarkets - it was assumed that each supermarket has a collection machine for collecting deposit refund waste, hence is not collecting the waste manually. Therefore, customer coordinates can be obtained by searching for all possible supermarkets in a selected area. As already mentioned for this thesis the Austrian federal state, Lower Austria (excluding Vienna), was selected to collect customer data and out of this data different data sets were extracted for testing, in consequence all supermarkets needed to be collected in this area. To do so, Overpass Turbo (*Overpass Turbo*, n.d.), a web-based data filtering tool for OpenStreetMap, was used to extract coordinates of supermarkets in Lower Austria. The search query, which was created with the Overpass query assistant "Wizard" and the resulting nodes on the map can be seen in figure 16 and 17.

```
1  [out:json][timeout:25];
2  // fetch area “Niederösterreich” to search in
3  {{geocodeArea:Niederösterreich}}->.searchArea;
4  // gather results
5  nwr["shop"="supermarket"](area.searchArea);
6  // print results
7  out geom;
```

Figure 16 Overpass query for supermarkets in Lower Austria

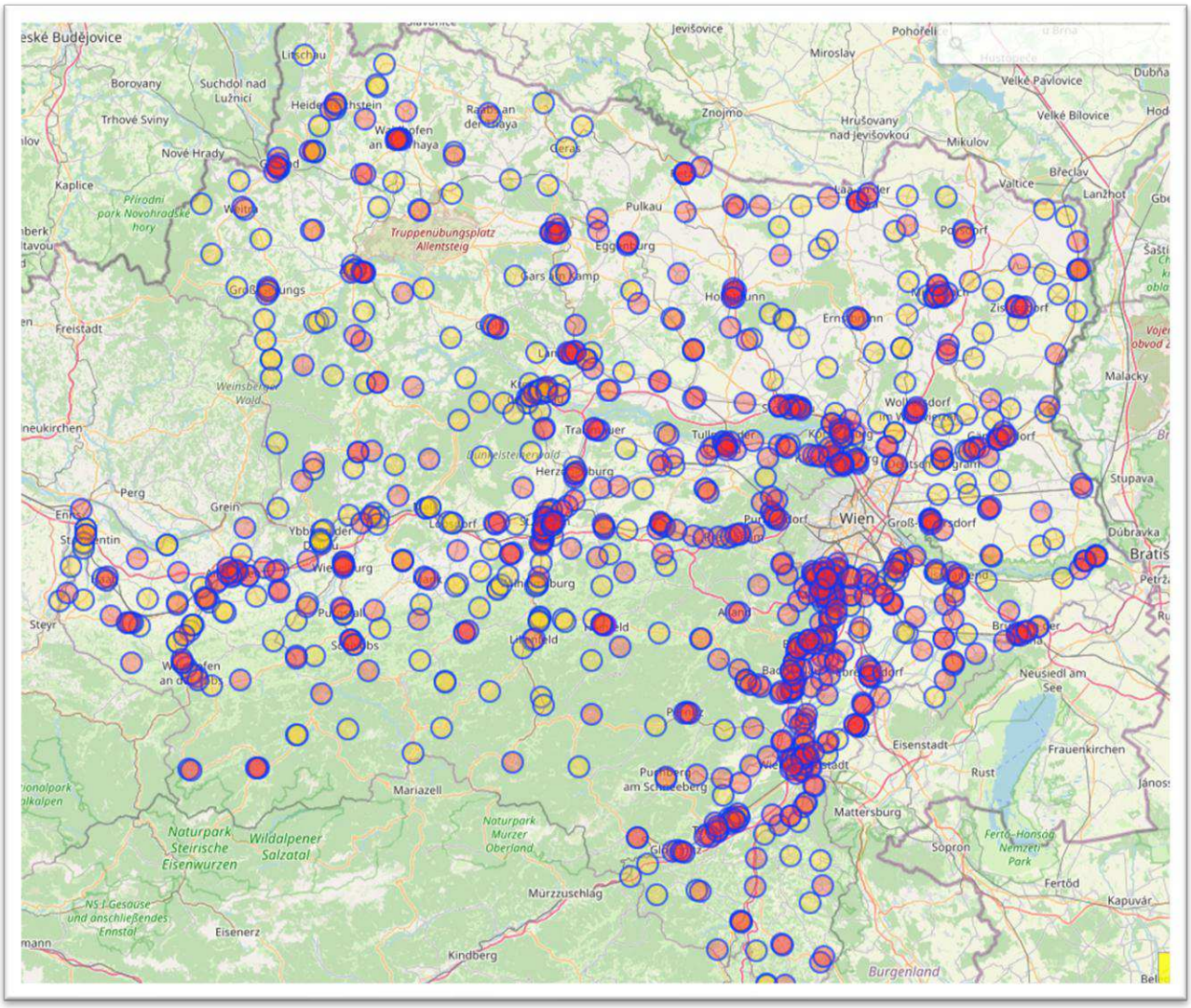

Figure 17 Supermarkets in Lower Austria on map

The resulting data file was then transformed with a C# method into a sorted file with supermarkets, their coordinates and the city/municipality they are in, if retrievable. If not, they were added manually, using the coordinates in Google maps (*Google maps*, n.d.). There were

a total of 1051 nodes found which, after filtering out shops that aren't supermarkets in the strong sense (e.g., smaller shops like butchers), a total of 929 were used for further data collection.

The weekly expected demand of supermarkets, which was also used to determine the number of pickups and bags per pickup, was estimated by the following procedure: First, the approximate amount of waste in litres per person per year was estimated using statistical data from the Austrian Economic Chamber on the sales market of drinks before the deposit refund system was implemented. The report differentiates between single-use and multi-use drink packages since, for the new deposit refund system, packages that are meant for recycling - hence technically single-use in their original form- are collected, we only look at the sales data for single-use packages, excluding single-use glass, cartons and cups. From the total number of drink sales, the number of sales for single-use plastic bottles using their sales percentage was calculated (all in litres) (WKO, 2024). Using a goal collection rate of 80%, which was the stated amount by EWP (EWP Recycling Pfand Österreich gGmbH, 2024a), this should result in three litres of deposit refund waste per person per week in Austria (population data for Austria was found in Statistik Austria, 2025). The details on the calculation can be found in table 2.

Table 2 Waste per person in Austria calculation

	Percentage	Number
Total sales 2023 in liters		3.426.000.000
Single-use plastic bottles in liters	42.7%	1.462.902.000+
Metal cans in liters	14.6%	500.196.000
Total sales plastic/ metal in liters		1.963.098.000
Collection rate	80%	1.570.478.400/
Population Austria 2023		9.104.770
Waste/person/year in liters (rounded)		172
Waste/person/week in liters (rounded)		3

Then, using population data for each municipality in Austria - now for the year 2024 - from Statistik Austria (2025), we can calculate an approximate amount of waste in litres per municipality by simply multiplying the population times three. Finally, to determine the waste per supermarket, for each supermarket the total amount of waste per week was divided by the number of supermarkets with the municipality or city of the supermarket for all supermarkets in Lower Austria. The waste of municipalities without a supermarket was split up among the supermarkets in the closest neighbouring municipality with one or multiple supermarkets. To

approximate this, the coordinates of one point in the municipality were used to calculate the closest point in a municipality with supermarket, according to Euclidian distance (this was done in Excel). The point coordinates were again obtained using Overpass Turbo (*Overpass Turbo*, n.d.), and a C# method for data extraction. The search query used can be seen in figure 18. If it wasn't found for a municipality or a village within the municipality, it was added manually using a random point in the desired area in Google maps (*Google maps*, n.d.). After the demand calculation, the values for expected weekly waste for the whole Lower Austria example data set ranged from 1557 to 21456 litres of waste, with the average being ~5550 litres. Again, this serves as an approximation since the demand of supermarkets depends on various factors, such as supermarket size and more detailed distributions of the population.

```
1  [out:csv(name, ::lat, ::lon)][timeout:600];
2  {{geocodeArea:Niederösterreich}}->.searchArea;
3  (node["place"~"city|town|village|municipality"](area.searchArea);
4  );
5  out center;
```

Figure 18 Overpass query municipality and city coordinates

What was then left to calculate were the frequencies and estimated number of bags per pickup for each supermarket, which corresponds to the demand value used for optimizing. As already mentioned, the same machine- and in consequence, bag-size was assumed for all customers. The EWP states that for the selected bag size, approximately 550 to 700 compacted pieces can fit into one bag. Since we calculated the amount of waste in litres, we set the amount fitting in each bag to the lower end, hence 550, since drinks come in various bottle and can sizes, sometimes larger but often also smaller than 1 litre. Using this information, we can then calculate the approximate amount of waste in bags per week (rounded to integers). The frequencies corresponding to the expected amount of bags per week were also stated by the EWP and can be seen in table 3 (EWP Recycling Pfand Österreich gGmbH, 2024b).

Table 3 Pickups per week according to the number of waste bags

Bags per week	**Planned pickups per week**
2-7	1
8-14	2
15-21	3
22-28	4
More than 28	5

A comparison of the prevalence of different frequencies and the number of bags after calculation for the total of supermarkets in the Lower Austria data set can be seen in table 4, where it is apparent that two was the most common frequency and five the most common number of bags per pickup. It should be noted that a bag number of eight is only possible for customers with a frequency of five pick-ups, since for every other frequency, the maximum number of bags per pick-up is seven.

Table 4 Frequency and bag number prevalence

Frequency	**Prevalence**
1	228
2	604
3	67
4	23
5	7
Bag number	**Prevalence**
1	0
2	0
3	6
4	212
5	328
6	225
7	156
8	2

The possible visit combinations were determined such that for customers being visited 2 to 4 times a week, the risk of them accumulating too much waste due to an uneven distribution of pick-up days was reduced. Meaning, while customers with a frequency of one can be picked up any day of the week and customers with a frequency of five each day of the week, the rest should be evenly distributed throughout the week. It should be noted that on Saturday, waste cannot be picked up, but it still increases because stores are open that day. Hence, for customers with a frequency of two, pickup on Monday or Friday with two days in between pickups would be advisable. For customers with a frequency of three or four, a pickup on Monday and Friday

should be planned, with the case of frequency three having one day in between pickups. The resulting visit combinations can be seen in table 5 and were encoded to decimal equivalents to their binary bit string in the data file.

Table 5 Possible visit combinations

Frequency	Visit Combinations
1	{Monday}; {Tuesday}; {Wednesday}; {Thursday}; {Friday}
2	{Monday, Thursday}; {Tuesday, Friday}
3	{Monday, Wednesday, Friday}
4	{Monday, Tuesday, Wednesday, Friday}; {Monday, Tuesday, Thursday, Friday}; {Monday, Wednesday, Thursday, Friday}
5	{Monday, Tuesday, Wednesday, Thursday, Friday}

6.2.3 Resulting example instance sets

Since we now have determined all parameters needed, hence the coordinates, expected bags per pickup, frequency, visit combinations and expected weekly demands for the whole Lower Austria data set, we can now extract smaller datasets from this set. The areas were selected based on the districts and cities the supermarkets are in, hence using one district or several neighbouring districts. The criteria for these clusters were obtaining relatively square areas since this aids with efficient routing as well as data sets of different sizes. This process resulted in six data sets, ranging from 44 to 220 customers. The selected areas can be seen in figure 19 and the summary in table 6. Please note that some districts were not included in the final data sets. For depots, one of the (recycling-)waste collection centres located in the selected area was chosen.

Figure 19 Example data map (created with Datawrapper)

Table 6 Example data sets summary

Name	Districts/ cities	Supermarkets	Depot	Vehicles	Frequencies
LowerAustria_1	Melk	44	WSZ Pöchlarn	2	1-5
LowerAustria_2	Mödling	68	Altstoffsammelzentrum Maria Enzersdorf	4	1-4
LowerAustria_3	Amstetten Scheibbs Waidhofen an der Ybbs	99	Wertstoffsammelzentrum Waidhofen Ybbs	5	1,2,3,5
LowerAustria_4	Gmünd Waidhofen an der Thaya Horn Zwettl Krems (district) Krems (city)	129	Altstoffsammelzentrum Rapottenstein	6	1-4
LowerAustria_5	Lilienfeld Baden Neunkirchen Wr. Neustadt (city) Wr. Neustadt (district)	206	Wertstoffsammelzentrum der Abfallbehandlungsanlage Wr. Neustadt	12	1-4

LowerAustria_6	Hollabrunn Mistelbach Korneuburg Tulln Gänserndorf	220	Altstoffsammelzentrum Korneuburg	14	1-5

6.2.4 Other information

For the second part of the solution method, minimum and maximum waste thresholds of customers t_{min} and t_{max} must be incorporated, as well as the target collection value t_{tar}, we also must define these values before testing. Note that we did not include these values in the data file but added them directly into the program. The minimum and maximum threshold levels were set according to the already mentioned bags per pickup stated by the EWP, where pickup starts with 2 bags per week, hence we set $t_{min} = 2$. Since for customers with eight bags per week an additional pickup needs to be planned, we chose $t_{max} = 8$. Lastly, they refer to the target number of collected bags per week as 5 to 7 bags, resulting in $t_{tar} = 5$ (EWP Recycling Pfand Österreich gGmbH, 2024b).

6.2.5 Simulating machine data

Since the second part of the solution method requires actual demand levels at the start of each planning day of customers in terms of collected bags, we need a way to simulate such data by using the data sets, which are used for planning the initial tours. We do this by assuming that in week 0 the tours were executed exactly as stated in the plan created with the ALNS. Then, for the first day, which is Monday in week 1, we look for the last day on which a customer has been picked up, for example for a customer with a pickup on Monday and Thursday, the last pick-up day would be Thursday. We then calculate the waste that has accumulated since that day. Please note that we assume that the collection takes place in the morning and only the number of bags that have been registered, i.e., simulated in the morning are being picked up, therefore if the last pickup day was Thursday the waste collected after Thursday morning is part of the simulation. Also note that on Saturday waste is not picked up but can still accumulate, which must be considered when simulating the bag number and waste values. For instance, if the last pick-up day was Thursday (week 0), waste has accumulated for Thursday, Friday, and Saturday. By using the expected weekly waste in the original instance data, we can calculate the expected amount of waste for the number of days since the last pickup. We then add noise by multiplying this with a random number centered around 1 based on the standard deviation Z of waste values for customers (the desired Z value is set by the user). This is done by using a

Box muller transformation[4]. The resulting value in the expected amount of waste then gets transformed into full bags and the amount of waste in the currently used bag. For all other days after the first day of week one, we simply add the amount of waste generated on one day to the previous value, if the waste has not been picked up yet, expect on Mondays (expect Monday week 1), where due to Saturday the amount for two days is added to the previous levels. If the waste has been picked up, we start from zero, hence on the next day after the pickup only waste from one day is calculated (also respecting if Saturday in-between). Note that noise gets added to all the cases just described like on Monday, week 1. Also note that we set the standard deviation value to 0.25 for our waste simulation. Example output for the waste simulation can be seen in figure 20

```
--- Waste Simulation Week 1, Day 1 ---
Customer 1: 6 full bags, 134 waste level
Customer 2: 8 full bags, 114 waste level
Customer 3: 1 full bags, 520 waste level
Customer 4: 7 full bags, 415 waste level
Customer 5: 4 full bags, 124 waste level
Customer 6: 2 full bags, 350 waste level
Customer 7: 9 full bags, 528 waste level
Customer 8: 3 full bags, 12 waste level
Customer 9: 4 full bags, 507 waste level
Customer 10: 3 full bags, 280 waste level
Customer 11: 4 full bags, 233 waste level
Customer 12: 4 full bags, 239 waste level
Customer 13: 5 full bags, 241 waste level
Customer 14: 3 full bags, 371 waste level
Customer 15: 3 full bags, 27 waste level
Customer 16: 5 full bags, 165 waste level
Customer 17: 5 full bags, 450 waste level
Customer 18: 4 full bags, 547 waste level
Customer 19: 5 full bags, 417 waste level
Customer 20: 7 full bags, 296 waste level
Customer 21: 3 full bags, 500 waste level
Customer 22: 2 full bags, 417 waste level
Customer 23: 3 full bags, 348 waste level
Customer 24: 3 full bags, 157 waste level
Customer 25: 2 full bags, 523 waste level
Customer 26: 3 full bags, 452 waste level
Customer 27: 2 full bags, 531 waste level
Customer 28: 4 full bags, 297 waste level
Customer 29: 4 full bags, 194 waste level
Customer 30: 4 full bags, 36 waste level
Customer 31: 6 full bags, 201 waste level
Customer 32: 2 full bags, 365 waste level
Customer 33: 2 full bags, 260 waste level
Customer 34: 5 full bags, 121 waste level
Customer 35: 3 full bags, 449 waste level
Customer 36: 2 full bags, 330 waste level
Customer 37: 2 full bags, 93 waste level
Customer 38: 3 full bags, 162 waste level
Customer 39: 1 full bags, 99 waste level
Customer 40: 7 full bags, 61 waste level
Customer 41: 6 full bags, 518 waste level
Customer 42: 5 full bags, 364 waste level
Customer 43: 5 full bags, 208 waste level
Customer 44: 1 full bags, 400 waste level
Customer 45: 2 full bags, 38 waste level
```

Figure 20 Example waste simulation output for Lower_Austria_1

[4] The Box muller transformation takes two independent uniformly distributed numbers $u_1$and u_2 within the interval [0,1], which are then transformed into a random value z with a mean of zero and a standard deviation of 1, hence can be then multiplied by the desired standard deviation. The formula used is $z = \sqrt{-2\log(u_2)} * \sin(2\pi u_1)$ Note that, since the transformation works in polar coordinates a pair of z can be created, by exchanging sin with cos, hence there are technically two formulas but only one used in our implementation (Scott, 2011).

7 Results

In this chapter the results for the ALNS algorithm presented in this thesis will first be compared to the selected set of benchmark instances presented in Cordeau et al. (1997). Then for the illustrative example data, results for the ALNS and the adaptation heuristic, alongside a tour visualization will be shown and discussed. All experiments were performed with a Lenovo i7 laptop using C# in Visual Studio. Note that the distance between two nodes in the following results were calculated using Euclidean distances for the benchmark instances, but for the illustrative example data, Haversine[5] distances were used for a more realistic depiction of distances between customers.

7.1 Benchmark results

In the following results for the effectiveness of the ALNS for the PVRP will be described, alongside a discussion of parameter settings and the effectiveness of different aspects of the algorithm.

7.1.1 Parameter settings

Since multiple parameters in the ALNS must be set by the user, our justification for certain settings of these values is now provided. For all tests, 6000 iterations were chosen as the stopping criterion, as a tradeoff of solution quality vs. algorithm runtime. This setting allows for better comparison of results, even though for smaller instances fewer iterations might have been sufficient, and larger instances might have profited from more iterations. But as can be seen in the results, the largest benchmark instance runtime was already around an hour with the number of iterations chosen, in consequence it was decided against increasing it any further.

As for the settings of values needed for determining the temperature in the SA acceptance criterion the p value, hence the probability to select a solution fit for selection according to the other values, was set to $p = 0.5$ in all experiments, since this value was also used as a reference value by Santini et al. (2018), and we neither wanted to favor selecting nor not selecting a fitting solution -in this context- which is achieved by going 50/50. The lower value for determining the SA temperature h_{end} was set to $h_{end} = 0.01$ in all experiments to only allow for small deviations from the best solution at the end of the algorithm. For the dynamic removal percentage, the minimum removal percentage q_{min} was set to $q_{min} = 0.05$ in all experiments to achieve some exploration of the solution while not increasing runtime too

[5] Used to calculate the distance between two points on the earth's surface (assuming the earth is a sphere) usings its radius. The formula is: $2r\ arc\sin\sqrt{sin^2(\frac{\emptyset_2-\emptyset_1}{2}) + \cos(\emptyset_1) * \cos(\emptyset_1) * sin^2(\frac{\lambda_2-\lambda_1}{2})}$ with $\emptyset$ being the latitude and λ being the longitude (Maria et al., 2020).

much, since it is heavily impacted by this parameter. For the upper values of both q_{max} and h_{start} an analysis of their impact of the solution quality and runtime for five runs of one instance- namely Cordeau_al_08- has been done and the average value of solution and runtime can be seen in table 7 and table 8. The other values were set as just described in this analysis. It was done in 0.1 steps for h_{start} and 0.2 steps for q_{max}. As can be seen in these experiments, the average solution quality generally increases with q_{max}. To little surprise, the increase of this value also drastically impacts the runtime of the algorithm. The case of h_{start} is not as apparent, but for a q_{max} of 0.1 and 0.5 the best solution values were achieved with a $h_{start} = 0.3$. According to these results, solution quality seems to increase and then decrease again, with its size. Note that this was not the case for $q_{max} = 0.3$ where its increase also led to an increase in solution quality. Regarding these results, the choice of h_{start} should not really impact the runtime of the algorithm. Since $h_{start} = 0.3$ and $q_{max} = 0.5$, achieved the best average solution, these values were chosen as settings for our algorithm. But it should be mentioned again that this analysis is limited to Cordeau_al_08 and was only done for five runs, in consequence might not reflect ideal settings for all instances but seem like an overall reasonable choice. Also note that we refrained from choosing a higher q_{max} due to runtime concerns.

Table 7 Impact of hstart and qmax on solution quality

h_{start}/q_{max}	0.1	0.3	0.5
0.1	2151.936	2131.258	2105.708
0.2	2158.306	2133.23	2100.22
0.3	2155.67	2143.934	**2093.27**
0.4	2172.804	2134.712	2102.484
0.5	2156.064	2122.304	2110.414

Table 8 Impact of hstart and qmax on runtime in seconds

h_{start}/q_{max}	0.1	0.3	0.5
0.1	15.36	87.37	286.9
0.2	15.45	112.96	261.77
0.3	16.49	107.06	292.82
0.4	10.41	112.37	212.76
0.5	16.39	112.25	293.97

7.1.2 Comparison to benchmark instances

In the following the solution results of the selected benchmark instances will be compared to the results of Cordeau et al. (1997), who, as mentioned, implemented a TS algorithm in their solution framework for the PVRP. A total of 10 runs were performed for all selected instances. Note that for more runs or if run another time for ten runs, the results will most likely be slightly better or worse for most instances.

In terms of average solution quality, the average objective value was 0 to 2.75% higher than the values obtained by Cordeau et al. (1997) for most instances. For the largest instance, Cordeau_al_13, with 417 customers and seven days it is 9.42% higher, which might be due to an insufficiently large number of iterations. The results for the average distance of Cordeau_al_11 were 0.41% better than those of Cordeau et al. (1997), showing a slight improvement for this instance set. The mean for the average difference in distance between all chosen sets was 2.34%.

The best solution obtained during the ten runs was 0 to 3.24% higher for most instances, except for Cordeau_al_13 where it was 9.58% higher. The average here was 2.67%. For 6000 iterations the average runtime T was below 12 minutes for all instances except Cordeau_al_13 which, due to the large number of customers and seven days, took on average 57.74 minutes. In general, the runtime was around 17.06% lower than the time stated in Cordeau et al. (1997), but it should be mentioned that for runtime comparisons don't make too much sense since this value heavily depends on the hardware and software used and can only be seen as an approximate reference. For detailed results we refer to table 9

Table 9 Comparision to benchmark instances

Instance Cordeau_al_	**ALNS**	**TS**	**%**	**ALNS best**	**TS best**	**%**	**T ALNS**	**T TS**	**%**
02	1366.64	1330.09	2.75	1335.47	1322.87	0.95	0.77	0.6	27.78
03	527.728	524.61	0.59	524.61	524.61	0.00	0.44	3.18	-86.08
05	2108.255	2061.36	2.27	2080.02	202.7.99	2.57	1.52	7.46	-79.63
08	2102.911	2054.9	2.34	2077.22	2034.15	2.12	4.37	8.97	-51.24
10	1674.739	1629.96	2.75	1634.14	1595.84	2.40	3.58	5.57	-35.64
11	814.217	817.56	-0.41	803.84	779.29	3.15	3.92	9.72	-59.66
12	1256.873	1239.58	1.40	1234.61	1195.88	3.24	4.81	11.51	-58.22

13	3941.969	3602.76	9.42	3848.02	3511.62	9.58	48.09	57.74	-16.71
20	8367.4	8367.4	0.00	8367.4	8367.4	0.00	19.39	6.34	205.87
ALL	-	-	2.34	-	-	2.67	-	-	-17.06

7.1.3 Additional measures

In this section an analysis of additional measures of results of the ALNS for the PVRP instances by Cordeau et al. (1997) will be provided. First, we want to show the average percentage difference between the initial solution result, obtained with the adapted CW savings algorithm, and the result after the ALNS process. This value ranged from 13.70 to 57.23% with all except Cordeau_al_03 under 33% and an average of 28.31% for all instances, which shows that the ALNS helps to significantly improve the solution compared to just using a savings algorithm. Detailed results can be seen in table 10, here the percentage refers to how much shorter the ALNS solution is compared to the savings solution.

Table 10 Improvement of initial savings distance with ALNS

Cordeau_al_	02	03	05	08	10	11	12	13	20	ALL
%	22.00	57.23	13.70	16.68	28.45	29.05	32.38	25.95	29.32	28.31

To test how consistent the results are, we calculated the CV for each instance, which is the ratio of the standard deviation to the mean and is used to measure the relative dispersion of data (Arachchige et al., 2022). This value ranged from 0 to 1.65% with a mean of 1.07%, indicating that, at least for the runs in these experiments, good solutions can be achieved relatively consistently with the presented ALNS algorithm. These results can be seen in table 11.

Table 11 CV results

Cordeau_al_	02	03	05	08	10	11	12	13	20	ALL
CV in %	1.28	1.23	0.78	0.94	1.65	0.8	1.36	1.53	0.00	1.07

Since it might be interesting for further studies to see how the individual removal and insertion operators performed in the test, we want to provide information on their success rates. As noted, we refer to success if the overall best solution has been improved with an operator. In general, the average total successful application of operators was relatively low compared to the total number of iterations, ranging from 22.2 to 92.7 and increasing with problem size. As for individual operators, for all instances, partial removal was the most successful removal

operator, making up on average 68.97% of the total average successful applications, followed by random removal with 25.94%. The least successful operator was worst removal, only making up around 5.09%. This might be due to the fact that partial random removal does not remove customers from all their days and tours, consequently, reinserting them could result in smaller, more efficient changes of tours. As for insertion operator success rates, regret insertion performed better on all instances, with an overall average of 89.21% of successful applications. Respectively, cheapest insertion in random order was only successful 10.79%. Detailed results can be found in table 12. As for the 2-opt local search, a better solution in terms of improving the current working solution was found in an average of 40.96% of iterations for all instance sets, ranging from 11.86 to 93.20%, with all sets except Cordeau_al_13 below 50%, generally indicating that 2-opt was a good addition to the ALNS. Detailed results can be seen in table 13.

Table 12 Operator success

Operators /Cordeau_al_	**02**	**03**	**05**	**08**	**10**	**11**	**12**	**13**	**20**	**ALL**
Total avg. success	25.6	22.2	42.2	49.2	47.7	57	69.2	92.7	28.6	48.27
Success rates removal										
Partial rand. removal %	70.7	43.69	68.72	89.02	83.86	67.37	74.86	60.63	61.89	68.97
Rand. removal %	24.61	39.64	30.57	8.33	12.37	31.23	24.71	38.94	23.08	25.94
Worst removal %	4.69	16.67	0.71	2.64	3.77	1.40	0.43	0.43	15.03	5.09
Success rates insertion										
Cheapest insertion %	26.95	17.57	8.77	9.35	5.24	4.26	1.01	1.19	22.73	10.79
Regret insertion %	73.05	82.43	91.23	90.65	94.76	95.74	98.99	98.81	77.27	89.21

Table 13 2-opt success

Cordeau_al_	02	03	05	08	10	11	12	13	20	ALL
% of iterations	32.26	11.86	49.86	46.01	42.15	42.18	23.51	93.20	25.19	40.96

7.2 Illustrative example data results

In this section the results for the illustrative example data from the six datasets located in Lower Austria, Austria, will be discussed for the ALNS and the adaptation heuristic. As no comparison values are available for these instances, we will look at results in terms of improvement compared to the savings algorithm, runtime, CV and visualized tours for analysis of the ALNS results. For the adaptation heuristic, the average distance change after adaptation and the impact of the different steps will be provided, alongside a tour visualization. Note that all visualizations were done with the Python extension Folium.

7.2.1 ALNS results

Please note that the example data sets were also run for 6000 iterations for ten runs with the same parameter settings used for benchmark instances. As mentioned, Haversine distances were used for the calculation of distance, so the distance values mentioned can be seen as a rough correspondence with kilometres travelled (straight line distance). The average distance for the ALNS ranged from 278,20 in LowerAustria_2 (Mödling), which is the second smallest data set with 68 customers, to 3785.20 for the largest dataset with 221 customers (Hollabrunn, Mistelbach, Korneuburg, Tulln, Gänserdorf).

The average improvement of distance achieved with ALNS compared to the initial savings distance ranged from 11.17 to 27.29% with an average of 18.35% for all instances, indicating a lower improvement compared to the average achieved for the benchmark instances, which was 28.31%. The average CV ranged from 1.07 to 1.86%, with an average of 1.63% for all sets, hinting at less consistency than the benchmark instances, for which this value was this average was 1.07% across ten runs. The runtime increased roughly exponentially with instance size, ranging from only 0.26 minutes for the set with 44 customers to 47.74 minutes for the set with 220 customers. The reason for all these values being slightly “worse” than the values achieved for the benchmark instances could be the result of more restrictive visit combinations for customers with two and three visits per week, where the former was the most prevalent, leading to fewer opportunities for exploration of the solution space. Detailed results can be found on table 14.

Table 14 Illustrative example data ALNS results

Data set	ALNS	ALNS best	Improvement savings %	CV %	T in min
LowerAustria_1	486.48	480.16	27.29	1.07	0.26
LowerAustria_2	278.20	272.67	18.37	1.44	1.07
LowerAustria_3	1342.81	1284.56	21.39	1.77	4.02
LowerAustria_4	2481.19	2404.25	15.47	1.78	14.07
LowerAustria_5	2497.98	2440.89	16.39	1.84	39.89
LowerAustria_6	3785.2	3731.18	11.17	1.86	47.74
ALL	-	-	18.35	1.63	-

In the following figures 21 to 25, visualized tours for the best ALNS result of the initial tours will be shown for day one for five of the data sets to provide a representative overview of the ALNS solutions. In table 15 the total distance for day one, the number of vehicles, total demand, demand range and mean of demand values for that day are listed. These values indicate sub-ideal allocation of demand in some cases, which could be due to the objective of minimizing the total distance instead of the number of vehicles used.

Table 15 Visualized tours data day 1

Dataset	Distance	Vehicles	Total demand	Demand range	Mean demand
LowerAustria_1	97.82	2/2	90	41-49	45.5
LowerAustria_2	78.18	4/4	182	34-50	43.33
LowerAustria_4	530.12	6/6	260	18-50	45
LowerAustria_5	702.19	12/12	590	10-50	49.17
LowerAustria_6	1018.87	12/14	590	47-50	49.17

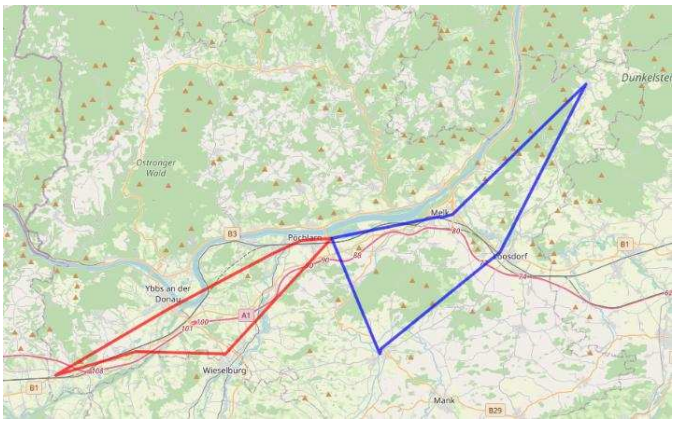

Figure 21 LowerAustria_1 Day 1

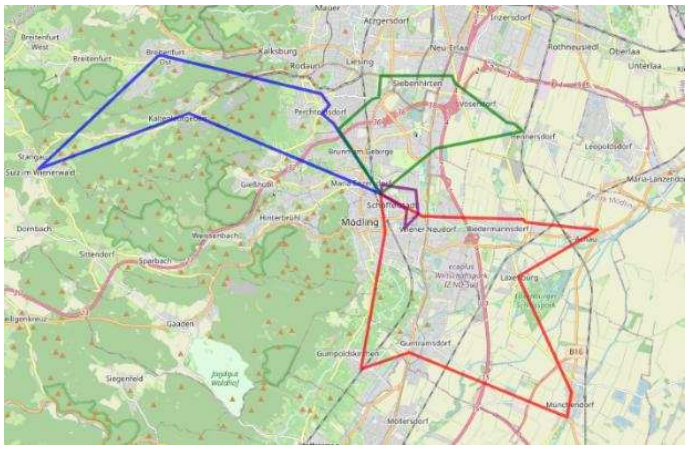

Figure 22 LowerAustria_2 day 1

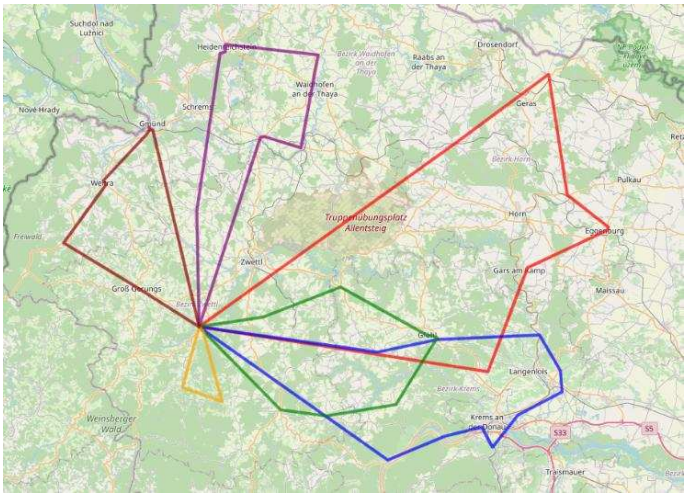

Figure 23 LowerAustria_4 day 1

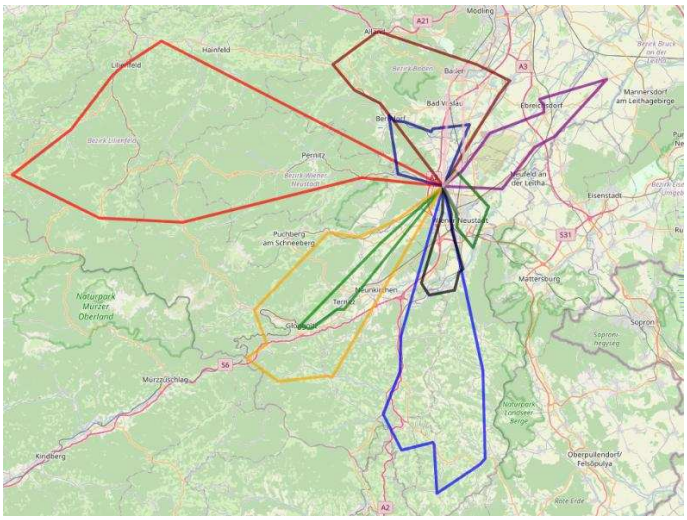

Figure 24 LowerAustria_5 day 1

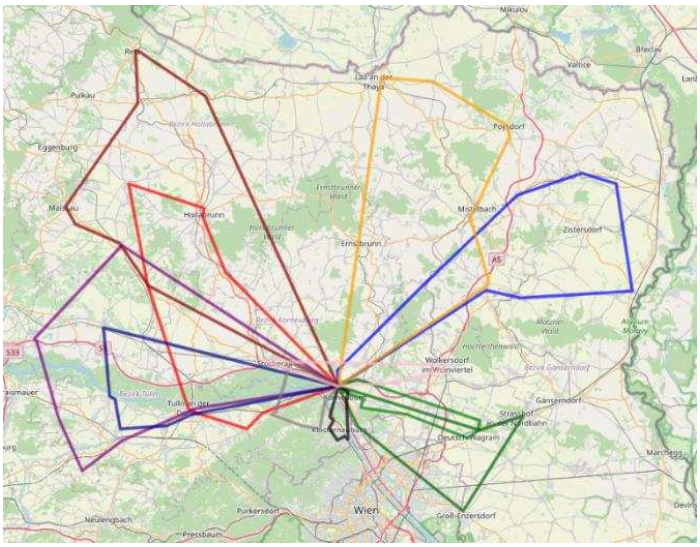

Figure 25 LowerAustria_6 day 1

In figures 27 to 31 the best ALNS result for all five days will be presented for the dataset LowerAustria_3 (Amstetten, Scheibbs, Waidhofen an der Ybbs) with 99 customers to gain a better understanding of the effect of different frequencies and corresponding visit combinations of customers. Table 16 provides the corresponding values for distances and demand. As stated in table 14 the total distance for this week was 1284.56. Apart from day two, all vehicles were efficiently used in this ALNS run. For day three only two out of five vehicles were used, and the distance travelled was the lowest out of all distances for this week. This is since the most prevalent customer visiting frequency for this set was two, with 66 customers, for which day three is not part of any visit combination.

Table 16 LowerAustria_3 best ALNS results

Day	Distance	Vehicles	Total Demand	Demand range	Mean demand
1	260.37	4/5	195	48-50	48.75
2	291.91	5/5	208	12-50	41.60
3	150.67	2/5	90	48-49	48.5
4	233.08	4/5	199	49-50	49.75
5	328.53	5/5	241	42-50	48.20

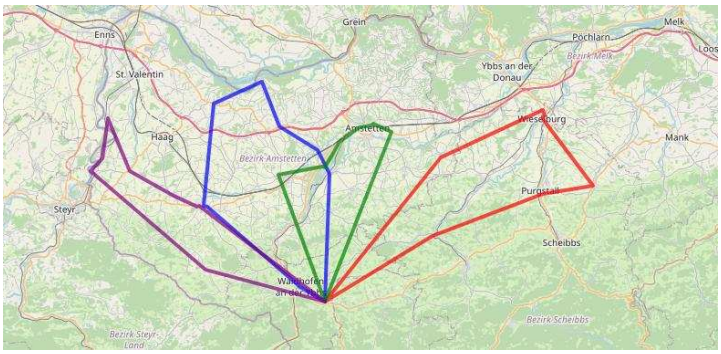

Figure 26 LowerAustria_3 day 1

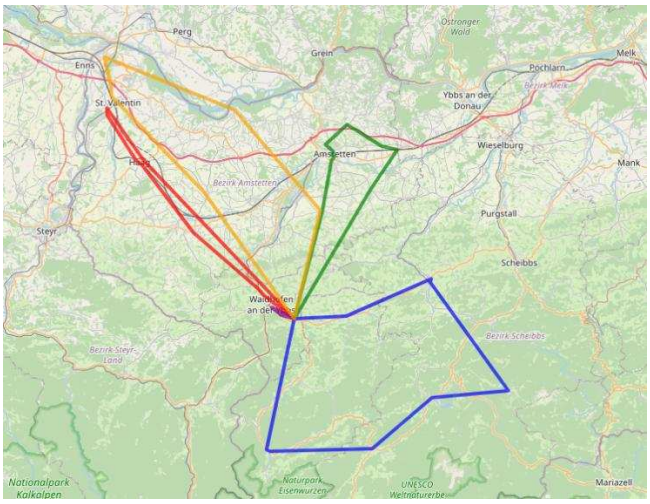

Figure 27 LowerAustria_3 day 2

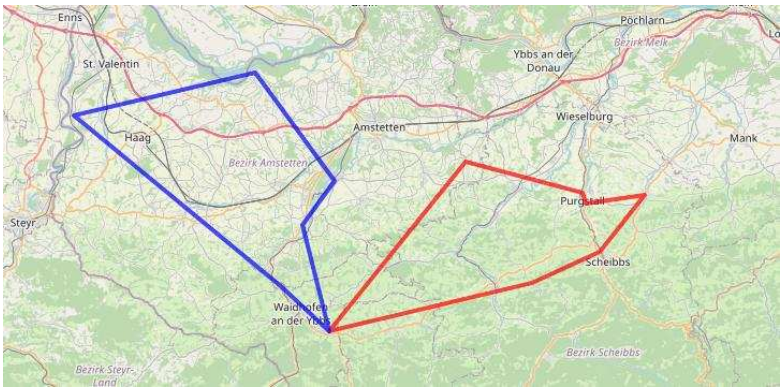

Figure 28 LowerAustria_3 day 3

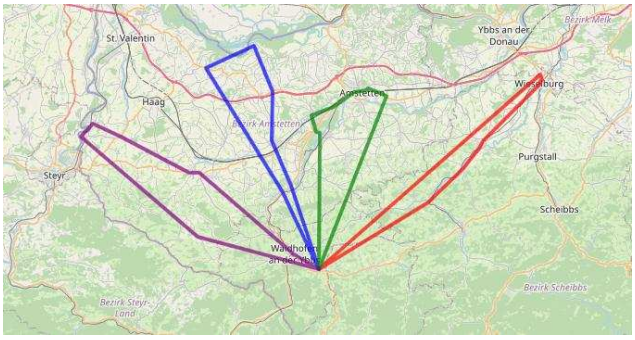

Figure 29 LowerAustria_3 day 4

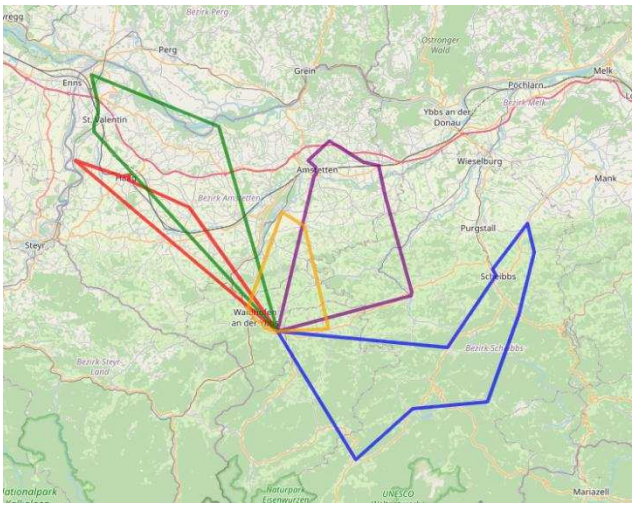

Figure 30 LowerAustria_3 day 5

7.2.2 Adaptation heuristic results

Please note that we tested the adaptation heuristic for four weeks after each of the ten ALNS runs, using the initial ALNS solution for adaptation. As mentioned, a standard deviation value of 0.25 was used for simulating updated demand values. For the adaptation heuristic, hence customers being added or removed from tours due to updated demand values, the average distance across four adapted weeks was 0.01 to 0.82% shorter than the initial routes for three datasets and 1.04 to 2.37% longer for three datasets, with an average of 0.62% increased distance after adaptation. Detailed results can be found in table 17.

Table 17 Adaptation heuristic general results

Dataset	**ALNS avg.**	**Adapt. avg.**	**Avg. % gap ALNS → Adapt**
LowerAustria_1	486.48	486.41	-0.01
LowerAustria_2	278.2	275.95	-0.82
LowerAustria_3	1324.81	1340.24	-0.09
LowerAustria_4	2481.19	2540.25	2.37
LowerAustria_5	2497.98	2528.98	1.23
LowerAustria_6	3785.20	3824.11	1.04
ALL	-	-	0.62

When looking at the individual adaptation steps, removing customers under the minimum pickup threshold t_{min} from the tour each day led to a distance reduction between 0.03 and

0.29% with an average of 0.10% compared to the initial ALNS distance. Note that this value depends on the chosen t_{min}, which was set to two in this, which is quite low compared to the expected demand values of customers and corresponding simulated demand values for the datasets. Consequently, only a few customers could be removed during the adaptation process.

The inclusion of customers that reached their maximum pickup threshold t_{max} earlier than expected led to an increase of distance between 1.7 and 3.52%, with an average of 2.27% for all datasets, when compared to the distances after the first adaptation steps. The adaptation of vehicles over the capacity Q resulted in a decrease between 0.69 and 2.56% and an average of 1.32%, compared to the distances of the preceding step, mainly due to the removal of customers under t_{tar} and t_{max}. The 2-opt local search managed to improve this distance between 0.05 and 0.19% with an average of 0.11%, which does not indicate a big improvement with this step. Detailed results can be found in table 18. Note that, as mentioned, the percentage gaps refer to the previous step, except for removing customers under t_{min}, where it refers to the difference to the ALNS results.

Table 18 Impact of Adaptation steps (in percent)

Dataset	**Remove under t_{min}**	**Insert over t_{max}**	**Adapt due to Q**	**2-opt**
LowerAustria_1	-0.13	2.56	-2.56	-0.16
LowerAustria_2	-0.29	1.7	-2.19	-0.19
LowerAustria_3	-0.07	1.9	-0.69	-0.05
LowerAustria_4	-0.04	3.52	-0.98	-0.13
LowerAustria_5	-0.05	2.19	-0.72	-0.05
LowerAustria_6	-0.03	1.74	-0.78	-0.08
ALL	-0.1	2.27	-1.32	-0.11

In the following 32 to 41 a tour visualization for the adapted tours of week two for the dataset LowerAustria_2 (Mödling) with 68 customers compared to the initial ALNS tours for the best run will be provided. The corresponding numbers of removed customers under t_{min}, inserted customers and removed and moved customers due to capacity constraints can be found in table 19. Here, adapting a tour due to capacity restrictions was the most apparent adaptation process,

which took place eight times, followed by inserting customers, which happened three times. For this week a 2-opt local search found an improvement on none of the days. Please note that in the tour visualization for vehicles 1 to 4 the respective colors were red, blue, green, and purple.

Table 19 Visualization data Lower_Austria_2

Day	**Distance original**	**Distance adapted**	**Demand original**	**Demand adapted**	**Remove under**	**Insert over**	**Adapt due to Q**	**2-opt**
1	78.18	71.74	182	168	-	-	Vehicle 2 2 cust. removed	-
2	41.5	42.86	101	97	-	Vehicle 3 1 cust inserted	-	-
3	40.32	42.29	49	46	Vehicle 1 cust. removed	Vehicle 1 1 cust. inserted	Vehicle 1 1 cust. removed	-
4	58.44	66.92	196	190	Vehicle 1 1 cust. removed	Vehicle 2 1 cust. inserted	1 cust. moved from vehicle 1 to 2; Vehicle 3 2 cust. removed; Vehicle 4 1 cust. removed	-
5	54.24	54.15	130	122	-	-	Vehicle 2 1 cust. removed	-
ALL	272.68	277.96	658	623	2	3	8	-

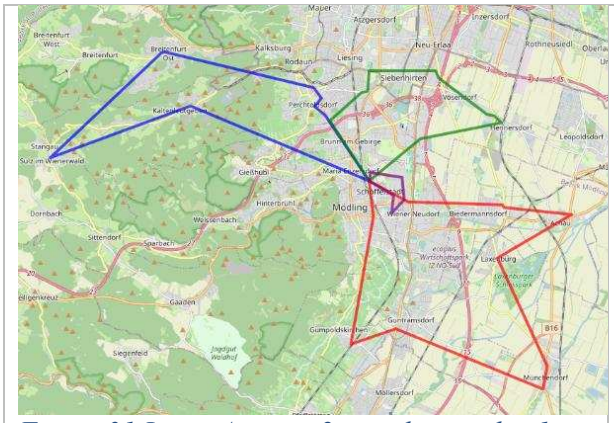

Figure 31 LowerAustria_2 initial tours day 1

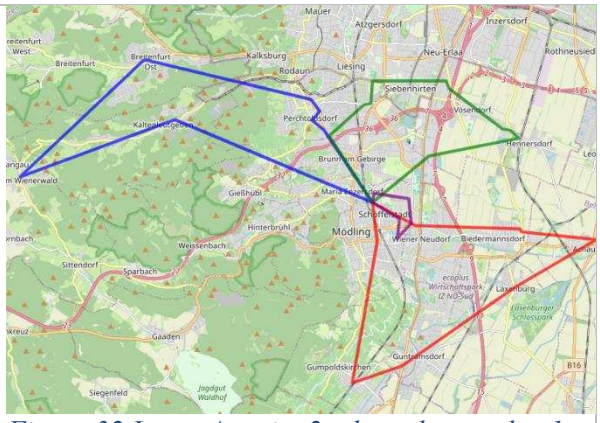

Figure 32 LowerAustria_2 adapted tours day 1

Figure 33 LowerAustria_2 initial tours day 2

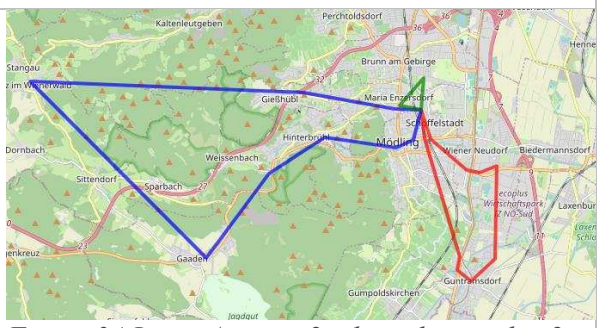

Figure 34 LowerAustria_2 adapted tours day 2

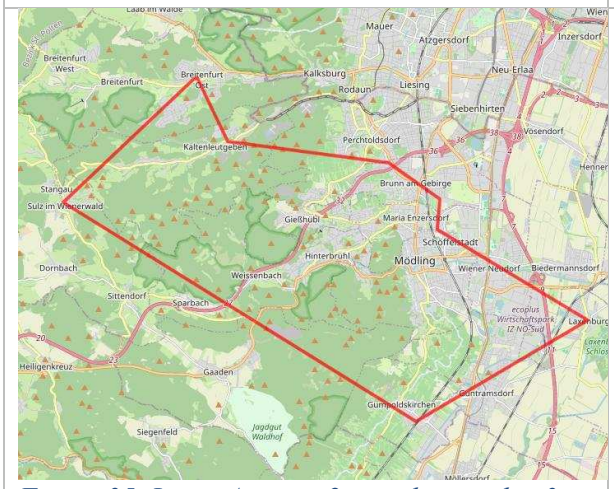

Figure 35 LowerAustria_2 initial tours day 3

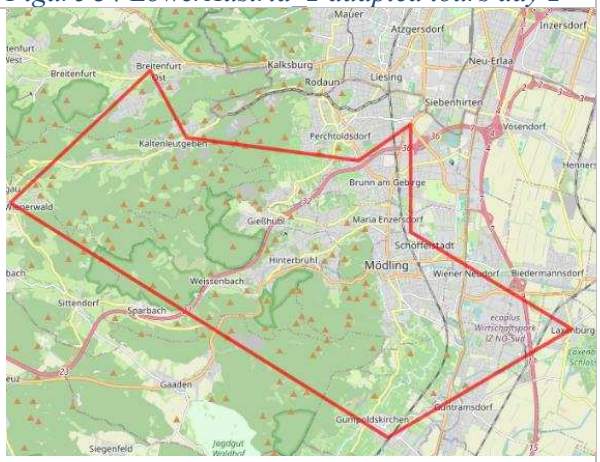

Figure 36 LowerAustria_2 adapted tours day 3

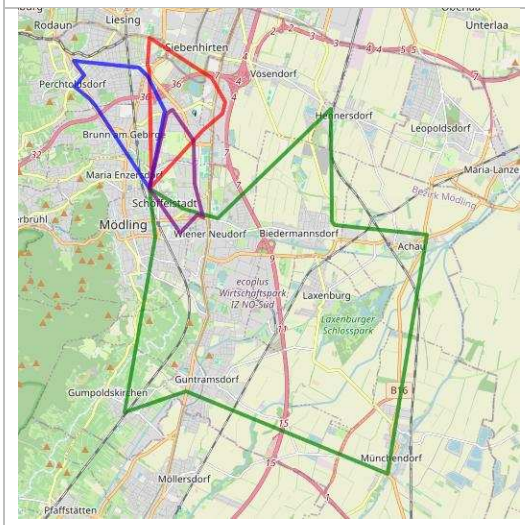

Figure 37 LowerAustria_2 initial tours day 4

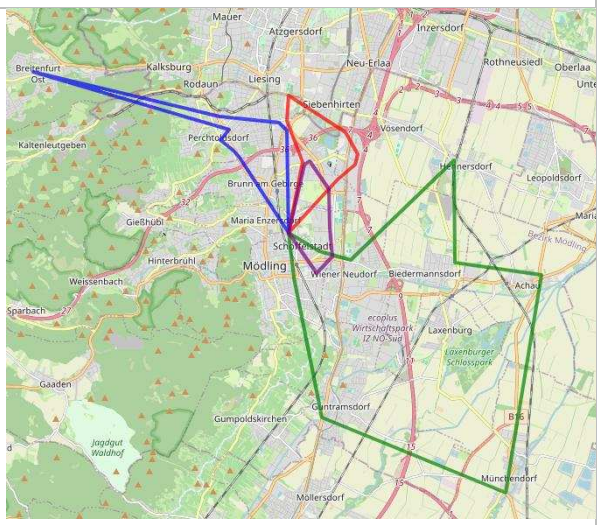

Figure 38 LowerAustria_2 adapted tours day 4

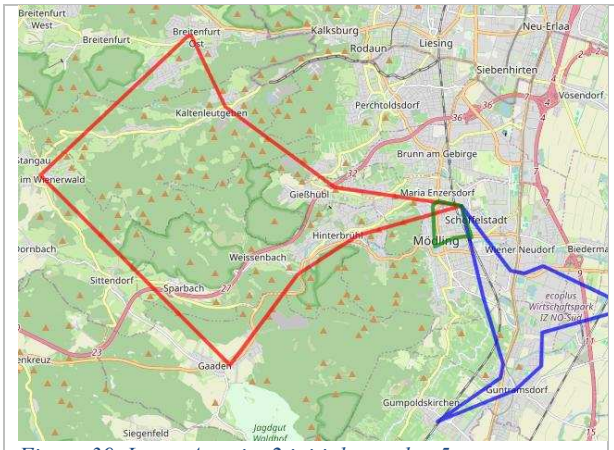

Figure 39 LowerAustria_2 initial tour day 5

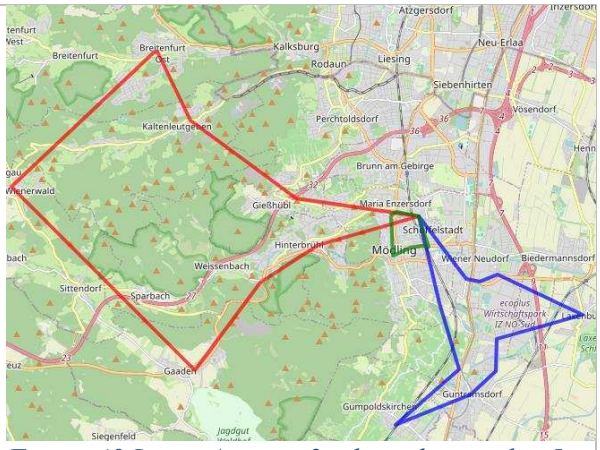

Figure 40 LowerAustria_2 adapted tours day 5

8 Conclusion

This thesis addressed vehicle routing in the context of deposit refund waste collection for the Austrian case, aiming to find a solution that incorporates periodic customer visits as well as customer demand data in terms of waste collected obtained from the internet-connected collection machines. To solve this problem, a two-step solution approach was introduced and implemented, in which the first step routes are created with an ALNS metaheuristic algorithm. This algorithm used an adapted CW savings algorithm to create initial routes which were then improved by using three destroy and two repair operators and a simulated annealing-based acceptance criterion followed by a 2-opt local search to improve solution quality. For adapting the resulting routes at the beginning of each day based on the demand data of collection machines, a route adaptation heuristic was introduced to exclude customers below a certain customer pick-up threshold or include the ones exceeding their threshold before their intended collection date, as well as adaptations for not exceeding the vehicle capacity.

To test the presented solution method, the first part was tested using a selected set of benchmark instances by Cordeau et al. (1997), where the ALNS solution was on average 0.41% better than the average results by Cordeau for one instance and less than 2.76% worse for all other instances except one. The tests suggest that the proposed ALNS algorithm is effective for producing good quality periodic vehicle routing solutions, with relatively low variation between runs and significant improvement compared to a solution resulting from only using a savings algorithm.

Furthermore, several real-world instances were created with coordinates of supermarkets and waste depots in Lower Austria, Austria. For these instances, initial routes achieved with ALNS also significantly improved the savings solution with an average of 18.35% but showed slightly more variation between runs. The adaptation heuristic, which was tested using simulated machine data for the different collection sites, led to overall shorter routes in three cases and longer distances in three others, with a low overall change in distance with an average of 0.62%. Suggesting effective adaptation of routes, at least for the data sets and waste simulation method used for testing.

In general, the results indicate that the presented method works effectively and could serve as a solid basis for creating and adapting routes for the collection of deposit refund waste. Even though the creation of the initial routes comes with a high runtime for larger instances, which might often be the case for real-world settings of these types of problems using the simple heuristic method suggested for daily adaptation does not require significant computational

effort and can therefore be easily operated, even with limited resources, while also not significantly worsening the overall distance travelled by vehicles.

However, despite the promising results, some limitations and further considerations should be acknowledged. First, the data for the illustrative example relies on statistics for beverage sales and was done in litres instead of different sized bottles/cans. Second, the overall structure as well as the mechanisms of the simulation of waste levels might not reflect actual waste generation patterns, which are also influenced by various factors such as holidays. Third, only minimization of distance was considered as an objective for the solution, but other factors like travel time, number of vehicles used, time windows or traffic might just be as important from a decision-maker's perspective. Additionally, testing the ALNS algorithm with a higher number of iterations on larger instances would allow for a better assessment of its scalability and robustness.

Apart from the just-mentioned considerations, further research could assess how the two-step approach presented in this thesis performs against a full-on dynamic route creation approach in terms of obtained results but also managerial considerations such as the plannability of vehicle fleets and ease of implementation. Moreover, incorporating predictive analysis for demands for the adaptation heuristic could enhance its applicability in real-world settings as well as routing to the collection sites where the bottles and cans are counted, instead of only considering intermediate depots.

Bibliography

Ahmadi Basir, S., Şahin, G., & Özbaygın, G. (2024). A comparative study of alternative formulations for the periodic vehicle routing problem. *Computers & Operations Research*, *165*, 106583. https://doi.org/10.1016/j.cor.2024.106583

Akhtar, M., Hannan, M. A., Begum, R. A., Basri, H., & Scavino, E. (2017). Backtracking search algorithm in CVRP models for efficient solid waste collection and route optimization. *Waste Management (Elmsford)*, *61*, 117–128. https://doi.org/10.1016/j.wasman.2017.01.022

Anagnostopoulos, T., Kolomvatsos, K., Anagnostopoulos, C., Zaslavsky, A., & Hadjiefthymiades, S. (2015). Assessing dynamic models for high priority waste collection in smart cities. *The Journal of Systems and Software*, *110*, 178–192. https://doi.org/10.1016/j.jss.2015.08.049

Angelelli, E., & Speranza, M. G. (2002). The application of a vehicle routing model to a waste-collection problem: two case studies. *The Journal of the Operational Research Society*, *53*(9), 944–952. https://doi.org/10.1057/palgrave.jors.2601402

Arachchige, C. N. P. G., Prendergast, L. A., & Staudte, R. G. (2022). Robust analogs to the coefficient of variation. *Journal of Applied Statistics*, *49*(2), 268–290. https://doi.org/10.1080/02664763.2020.1808599

Baptista, S., Oliveira, R. C., & Zúquete, E. (2002). A period vehicle routing case study. *European Journal of Operational Research*, *139*(2), 220–229. https://doi.org/10.1016/S0377-2217(01)00363-0

Beltrami, E. J. & Bodin, L. D. (1974). Networks and vehicle routing for municipal waste collection. *Networks*, *4*(1), 65–94. https://doi.org/10.1002/net.3230040106

Benjamin, A. M., & Beasley, J. E. (2010). Metaheuristics for the waste collection vehicle routing problem with time windows, driver rest period and multiple disposal facilities. *Computers & Operations Research*, *37*(12), 2270–2280. https://doi.org/10.1016/j.cor.2010.03.019

Bianchi-Aguiar, T., Carravilla, M. A., & Oliveira, J. F. (2012). Municipal waste collection in Ponte de Lima, Portugal - A vehicle routing application. *OR Insight*, *25*(4), 185–198. https://doi.org/10.1057/ori.2011.23

Bommisetty, D., Dessouky, M., & Jacobs, L. (1998). Scheduling collection of recyclable material at Northern Illinois University campus using a two-phase algorithm. *Computers & Industrial Engineering*, *35*(3), 435–438. https://doi.org/10.1016/S0360-8352(98)00127-2

Bouleft, Y., & Elhilali Alaoui, A. (2023). Dynamic multi-compartment vehicle routing Problem for Smart Waste Collection. *Applied System Innovation*, *6*(1), 30. https://doi.org/10.3390/asi6010030

Chao, I., Golden, B. L. & Wasil, E. (1995). An improved heuristic for the period vehicle routing problem. *Networks*, *26*(1), 25–44. https://doi.org/10.1002/net.3230260104

Christofides, N. & Beasley, J. E. (1984). The period routing problem. *Networks*, *14*(2), 237–256. https://doi.org/10.1002/net.3230140205

Clarke, G., & Wright, J. W. (1964). Scheduling of Vehicles from a central depot to a number of delivery points. *Operations Research*, *12*(4), 568–581. https://doi.org/10.1287/opre.12.4.568

Coene, S., Arnout, A., & Spieksma, F. C. R. (2010). On a periodic vehicle routing problem. *The Journal of the Operational Research Society*, *61*(12), 1719–1728. https://doi.org/10.1057/jors.2009.154

Cook, T. M., & Russell, R. A. (1978). A simulation and statistical analysis of stochastic vehicle routing with timing constraints. *Decision Sciences*, *9*(4), 673–687. https://doi.org/10.1111/j.1540-5915.1978.tb00753.x

Cordeau, J.-F., Gendreau, M., & Laporte, G. (1997). A tabu search heuristic for periodic and multi-depot vehicle routing problems. *Networks*, *30*(2), 105–119. https://doi.org/10.1002/(SICI)1097-0037(199709)30:2<105::AID-NET5>3.0.CO;2-G

Dantzig, G. B., & Ramser, J. H. (1959). The truck dispatching problem. *Management Science*, *6*(1), 80–91. https://doi.org/10.1287/mnsc.6.1.80

Dantzig, G., Fulkerson, R., & Johnson, S. (1954). Solution of a large-scale traveling-Salesman problem. *Journal of the Operations Research Society of America*, *2*(4), 393–410. https://doi.org/10.1287/opre.2.4.393

Datawrapper (2024, 8. Februar). Datawrapper. Retrieved March 28, 2025 from https://www.datawrapper.de/

Dorronsoro, B. (n.d.). *Description for files of Cordeau's instances. The VRP Web*. Retrieved March 28, 2025 from https://www.bernabe.dorronsoro.es/vrp/index.html?/Problem_Instances/VRPLIBDesc.html

Eilon, S., Watson-Gandy, C. D. T., & Christofides, N. (1971). *Distribution management : mathematical modelling and practical analysis* (1. publ.). Griffin.

Englert, M., Röglin, H., & Vöcking, B. (2014). Worst case and probabilistic analysis of the 2-opt algorithm for the TSP. *Algorithmica*, *68*(1), 190–264. https://doi.org/10.1007/s00453-013-9801-4

EWP Recycling Pfand Österreich gGmbH (2024a, Oktober). Das Pfandsystem für Einweggetränkeverpackungen.https://www.recycling-pfand.at/downloads/ewp-infoblatt-das-pfandsystem-okt.2024.pdf?1729176808

EWP Recycling Pfand Österreich gGmbH (2024b, März). Der Einwegpfand kommt 2025: Was Produzenten und Rücknehmer jetzt wissen müssen! https://www.recycling-pfand.at/downloads/praesentation-webinar-ruecknahme-mit-automaten-handout.pdf?1711004723

EWP Recycling Pfand Österreich gGmbH (2024c, November). Rücknehmer- Handbuch. Retrieved March 28, 2025 from https://www.recycling-pfand.at/downloads/ruecknehmer-handbuch-v2-november-2024.pdf?1733228495

European Commission (n.d.) *Single-use plastics.* https://environment.ec.europa.eu/topics/plastics/single-use-plastics_en

Faccio, M., Persona, A., & Zanin, G. (2011). Waste collection multi objective model with real time traceability data. *Waste Management (Elmsford)*, *31*(12), 2391–2405. https://doi.org/10.1016/j.wasman.2011.07.005

Fikejz, J., Brázdová, M., Jánošíková, Ľ., & Miwa, T. (2024). Modification of the Clarke and Wright algorithm with a dynamic savings matrix. *Journal of Advanced Transportation*, *2024*(1). https://doi.org/10.1155/2024/8753106

Foster, B. A., & Ryan, D. M. (1976). An integer programming approach to the vehicle Scheduling Problem. *Operational Research Quarterly (1950)*, *27*(2), 367–384. https://doi.org/10.1057/jors.1976.63

Gaudioso, M. & Paletta, G. (1992). A heuristic for the periodic vehicle routing problem. *Transportation Science*, *26*(2), 86–92. https://doi.org/10.1287/trsc.26.2.86

Google Maps. (n.d.). Google Maps. Retrieved March 28, 2025 from https://www.google.com/maps/?entry=wc.

Goncalves, L. B., Ochi, L. S., & Martins, S. L. (2005). A GRASP with adaptive memory for a period vehicle routing problem. In *International Conference on Computational Intelligence for Modelling, Control and Automation and International Conference on Intelligent Agents, Web Technologies and Internet Commerce (CIMCA-IAWTIC'06)*, *1*, 721–727. https://doi.org/10.1109/CIMCA.2005.1631349

Hannan, M. A., Akhtar, M., Begum, R. A., Basri, H., Hussain, A., & Scavino, E. (2018). Capacitated vehicle-routing problem model for scheduled solid waste collection and route optimization using PSO algorithm. *Waste Management (Elmsford)*, *71*, 31–41. https://doi.org/10.1016/j.wasman.2017.10.019

Hemmelmayr, V., Doerner, K. F., Hartl, R. F., & Rath, S. (2013). A heuristic solution method for node routing based solid waste collection problems. *Journal of Heuristics*, *19*(2), 129–156. https://doi.org/10.1007/s10732-011-9188-9

Hemmelmayr, V. C., Cordeau, J.-F., & Crainic, T. G. (2012). An adaptive large neighborhood search heuristic for two-echelon vehicle routing problems arising in city logistics. *Computers & Operations Research*, *39*(12), 3215–3228. https://doi.org/10.1016/j.cor.2012.04.007

Hemmelmayr, V. C., Doerner, K. F., & Hartl, R. F. (2009). A variable neighborhood search heuristic for periodic routing problems. *European Journal of Operational Research*, 195(3), 791–802. https://doi.org/10.1016/j.ejor.2007.08.048

Hess, C., Dragomir, A. G., Doerner, K. F., & Vigo, D. (2024). Waste collection routing: a survey on problems and methods. *Central European Journal of Operations Research*, 32(2), 399–434. https://doi.org/10.1007/s10100-023-00892-y

Idrissi, A., Benabbou, R., Benhra, J., & Haji, M. E. (2024). Smart waste collection based on vehicle routing optimization: Case of Casablanca city. *Procedia Computer Science*, *236*, 194–201. https://doi.org/10.1016/j.procs.2024.05.021

Johansson, O. M. (2006). The effect of dynamic scheduling and routing in a solid waste management system. *Acta Materialia*, *26*(8), 875–885. https://doi.org/10.1016/j.wasman.2005.09.004

Jorge, D., Pais Antunes, A., Rodrigues Pereira Ramos, T., & Barbosa-Póvoa, A. P. (2022). A hybrid metaheuristic for smart waste collection problems with workload concerns. *Computers & Operations Research*, *137*, 105518. https://doi.org/10.1016/j.cor.2021.105518

Konstantakopoulos, G. D., Gayialis, S. P., & Kechagias, E. P. (2022). Vehicle routing problem and related algorithms for logistics distribution: a literature review and classification. *Operational Research*, *22*(3), 2033–2062. https://doi.org/10.1007/s12351-020-00600-7

Lei, H., Laporte, G., & Guo, B. (2011). The capacitated vehicle routing problem with stochastic demands and time windows. *Computers & Operations Research*, *38*(12), 1775–1783. https://doi.org/10.1016/j.cor.2011.02.007

Li, W., Ng, T. F., Ibrahim, H., & Wang, S. L. (2025). A literature review of the state of the art of sustainable waste collection and vehicle routing problem. *Journal of the Air & Waste Management Association (1995)*, *75*(1), 3–26. https://doi.org/10.1080/10962247.2024.2415298

Liang, Y.-C., Minanda, V., & Gunawan, A. (2022). Waste collection routing problem: A mini-review of recent heuristic approaches and applications. *Waste Management & Research*, *40*(5), 519–537. https://doi.org/10.1177/0734242X211003975

Liu, J., He, Y. F., & Zhang, A. P. (2014). The waste collection vehicle arc routing problem with turn constraints. *Advanced Materials Research*, *1010–1012*, 1000–1005. https://doi.org/10.4028/www.scientific.net/AMR.1010-1012.1000

LKW Walter. (n.d.). Equipment im LKW-Ladungstransport: Abmessungen/ Verladekapazitäten eines Planen-LKWs. Retrieved March 28, 2025, from https://www.lkw-walter.com/at/de/produkte-und-services/strassentransporte

Lu, X., Pu, X., & Han, X. (2020). Sustainable smart waste classification and collection system: A bi-objective modeling and optimization approach. *Journal of Cleaner Production*, *276*, 124183. https://doi.org/10.1016/j.jclepro.2020.124183

Maria, E., Budiman, E., Haviluddin, H., & Taruk, M. (2020). Measure distance locating nearest public facilities using Haversine and Euclidean methods. *Journal of Physics: Conference Series*, 1450. https://doi.org/10.1088/1742-6596/1450/1/012080.

Matos, A. C., Oliveira, R. C., Gambardella, L. M., Blum, C., Dorigo, M., Birattari, M., Stützle, T., & Mondada, F. (2004). An experimental study of the ant colony system for the period vehicle routing problem. In *ANT COLONY OPTIMIZATION AND SWARM INTELLIGENCE, PROCEEDINGS* (Vol. 3172, pp. 286–293). Springer Berlin Heidelberg. https://doi.org/10.1007/978-3-540-28646-2_26

Mohammadi, M., Rahmanifar, G., Hajiaghaei-Keshteli, M., Fusco, G., Colombaroni, C., & Sherafat, A. (2023). A dynamic approach for the multi-compartment vehicle routing problem in waste management. *Renewable & Sustainable Energy Reviews*, *184*, 113526. https://doi.org/10.1016/j.rser.2023.113526

Nuortio, T., Kytöjoki, J., Niska, H., & Bräysy, O. (2006). Improved route planning and scheduling of waste collection and transport. *Expert Systems with Applications*, *30*(2), 223–232. https://doi.org/10.1016/j.eswa.2005.07.009

Overpass Turbo. (n.d.). Overpass Turbo. Retrieved March 28, 2025 from https://overpass-turbo.eu/

Rahmanifar, G., Mohammadi, M., Sherafat, A., Hajiaghaei-Keshteli, M., Fusco, G., & Colombaroni, C. (2023). Heuristic approaches to address vehicle routing problem in the Iot-based waste management system. *Expert Systems with Applications*, *220*, 119708. https://doi.org/10.1016/j.eswa.2023.119708

Ramos, T. R. P., de Morais, C. S., & Barbosa-Póvoa, A. P. (2018). The smart waste collection routing problem: Alternative operational management approaches. *Expert Systems with Applications*, *103*, 146–158. https://doi.org/10.1016/j.eswa.2018.03.001

Ropke, S., & Pisinger, D. (2006). An adaptive large neighborhood search heuristic for the pickup and delivery problem with time windows. *Transportation Science*, *40*(4), 455–472. https://doi.org/10.1287/trsc.1050.0135

Roy, A., Manna, A., Kim, J., & Moon, I. (2022). IoT-based smart bin allocation and vehicle routing in solid waste management: A case study in South Korea. *Computers & Industrial Engineering*, *171*, 108457. https://doi.org/10.1016/j.cie.2022.108457

Russell, R. A. & Gribbin, D. (1991). A multiphase approach to the period routing problem. *Networks*, *21*(7), 747–765. https://doi.org/10.1002/net.3230210704

Russell, R. & Igo, W. (1979). An assignment routing problem. *Networks*, *9*(1), 1–17. https://doi.org/10.1002/net.3230090102

Salehi-Amiri, A., Akbapour, N., Hajiaghaei-Keshteli, M., Gajpal, Y., & Jabbarzadeh, A. (2022). Designing an effective two-stage, sustainable, and IoT based waste management system. *Renewable & Sustainable Energy Reviews*, *157*, 112031. https://doi.org/10.1016/j.rser.2021.112031

Santini, A., Ropke, S., & Hvattum, L. M. (2018). A comparison of acceptance criteria for the adaptive large neighbourhood search metaheuristic. *Journal of Heuristics*, *24*(5), 783–815. https://doi.org/10.1007/s10732-018-9377-x

Scott, D. W. (2011). Box-Muller transformation. *Wiley Interdisciplinary Reviews. Computational Statistics*, *3*(2), 177–179. https://doi.org/10.1002/wics.148

Shaw, P., Puget, J.-F., & Maher, M. (1998). Using constraint programming and local search methods to solve vehicle routing problems. In *Lecture notes in computer science* (Vol. 1520, pp. 417–431). Springer Berlin Heidelberg. https://doi.org/10.1007/3-540-49481-2_30

Shih, L. H., & Chang, H. C. (2001). A routing and scheduling system for infectious waste collection. *Environmental Modeling & Assessment*, *6*(4), 261–269. https://doi.org/10.1023/A:1013342102025

Statistik Austria. (2025). *Bevölkerung zu Jahres-/Quartalsanfang*. Retrieved March, 28, 2025 from https://www.statistik.at/statistiken/bevoelkerung-und-soziales/bevoelkerung/bevoelkerungsstand/bevoelkerung-zu-jahres-/-quartalsanfang

Teixeira, J., Antunes, A. P., & de Sousa, J. P. (2004). Recyclable waste collection planning—a case study. *European Journal of Operational Research*, *158*(3), 543–554. https://doi.org/10.1016/S0377-2217(03)00379-5

Tirkolaee, E. B., Goli, A., Gütmen, S., Weber, G.-W., & Szwedzka, K. (2023). A novel model for sustainable waste collection arc routing problem: Pareto-based algorithms. *Annals of Operations Research*, *324*(1–2), 189–214. https://doi.org/10.1007/s10479-021-04486-2

Tirkolaee, E. B., Mahdavi, I., & Mehdi Seyyed Esfahani, M. (2018). A robust periodic capacitated arc routing problem for urban waste collection considering drivers and crew's working time. *Waste Management (Elmsford)*, *76*, 138–146. https://doi.org/10.1016/j.wasman.2018.03.015

Toth, P., & Vigo, D. (2002). Models, relaxations and exact approaches for the capacitated vehicle routing problem. *DISCRETE APPLIED MATHEMATICS*, *123*(1), 487–512. https://doi.org/10.1016/S0166-218X(01)00351-1

Vidal, T., Crainic, T. G., Gendreau, M., Lahrichi, N., & Rei, W. (2012). A Hybrid Genetic Algorithm for Multidepot and Periodic Vehicle Routing Problems. *Operations Research*, *60*(3), 611–624. https://doi.org/10.1287/opre.1120.1048

Voigt, S. (2025). A review and ranking of operators in adaptive large neighborhood search for vehicle routing problems. *European Journal of Operational Research*, *322*(2), 357–375. https://doi.org/10.1016/j.ejor.2024.05.033

VRP-REP. (2017). Dataset: Cordeau_al_1997_PVRP. http://www.vrp-rep.org/datasets/item/2017-0009.html.

Wirtschaftskammer Österreich. (2024). Mehrweganteil beim Getränkeabsatz in Österreich nach Art der Getränke im Jahr 2023. https://www-wko-at.uaccess.univie.ac.at/oe/netzwerke/umsetzungsbericht-nachhaltigkeitsagenda-2023.pdf

Wu, H., Yang, B., & Tao, F. (2020). Optimization of vehicle routing for waste collection and transportation. *International Journal of Environmental Research and Public Health*, *17*(14), 1–26. https://doi.org/10.3390/ijerph17144963

Yuliza, E., Suprihatin, B., Bangun, P. B. J., Puspita, F. M., Indrawati, & Octarina, S. (2023). Waste collection vehicle routing problem with time windows for route optimization of garbage transport vehicles. *Science & Technology Indonesia*, *8*(1), 66–70. https://doi.org/10.26554/sti.2023.8.1.66-70

Zhang, H., Ge, H., Yang, J., & Tong, Y. (2022). Review of vehicle routing problems: Models, classification and solving algorithms. *Archives of Computational Methods in Engineering*, *29*(1), 195–221. https://doi.org/10.1007/s11831-021-09574-x

Appendix

Appendix A Detailed results savings algorithm benchmark instances

	Cordeau_al_02	Cordeau_al_03	Cordeau_al_05	Cordeau_al_08	Cordeau_al_10	Cordeau_al_11	Cordeau_al_12	Cordeau_al_13	Cordeau_al_20
1	1725.89	1219.23	2365.02	2553.63	2399.36	1124.66	1761.22	5197.12	11835.82
2	1739.73	1300.76	2457.31	2554.2	2384.11	1197.53	1875.78	5375.66	12324.77
3	1875.65	1233.26	2489.66	2539.67	2301.68	1139.39	1719.72	5420.99	12577.85
4	1730.16	1203.44	2443.21	2466.17	2260.93	1189.02	1755.81	5242.65	10918.76
5	1790.92	1070.25	2402.7	2483.29	2338.54	1155.2	1999.14	5326.34	12392.01
6	1706.85	1271.08	2427.97	2477.75	2294.76	1175.65	1932.1	5225.97	10984.52
7	1709.58	1299.93	2417.58	2585.44	2443.41	1129.7	1839.77	5456.21	13033.74
8	1685.36	1266.74	2493.2	2549.9	2236.62	1143.39	1811.15	5130.14	12210.93
9	1838.64	1148.25	2466.5	2491.55	2391.32	1098.35	1993.94	5532.51	11607.76
10	1740.07	1386.23	2471.75	2546.44	2369.82	1130.99	1943.76	5354.68	10932.1

Appendix B Detailed results ALNS benchmark instances

	Cordeau_al_02	Cordeau_al_03	Cordeau_al_05	Cordeau_al_08	Cordeau_al_10	Cordeau_al_11	Cordeau_al_12	Cordeau_al_13	Cordeau_al_20
1	1385.02	524.61	2097.54	2077.22	1653.71	820.89	1240.86	3848.85	8367.4
2	1335.47	524.61	2080.02	2087.23	1634.14	806.47	1250.78	3917.95	8367.4
3	1364.82	527.67	2097.85	2101.12	1697.59	808.44	1254.66	3933.27	8367.4
4	1378.58	524.61	2120.98	2150.32	1673.92	810.21	1257.97	4018.1	8367.4
5	1367.44	524.61	2133.92	2094.57	1714.56	803.84	1294.35	4006.46	8367.4
6	1362.48	524.61	2118.86	2090.51	1634.77	821.8	1239.83	3954.81	8367.4
7	1368.58	524.61	2109	2100.91	1712.55	815.54	1234.61	3979.31	8367.4
8	1400.51	531.02	2096.62	2094.04	1659.27	824.3	1265.76	3899.82	8367.4
9	1349.97	546.32	2097.06	2111.39	1678.62	814.3	1275.59	3848.02	8367.4
10	1353.57	524.61	2130.7	2121.8	1688.26	816.38	1254.32	4013.1	8367.4

Appendix C Detailed results savings example data

	NOE_1	NOE_2	NOE_3	NOE_4	NOE_5	NOE_6
1	668.69	370.76	1688.84	2927.32	2880.95	4385.79
2	672.32	363.54	1748.28	2894.82	2966.11	4232.34
3	642.88	340.29	1635.41	3045.55	3021.43	4261.59
4	616.04	353.25	1594.04	2997.64	2967.81	4249.06
5	755.97	250.08	1674.75	2965.32	3019.81	4157.76
6	654.31	374.6	1599.82	2904.27	3047.49	4199.42
7	646.41	338.73	1728.48	2862.28	2907.81	4267.38
8	698.61	344.21	1725.65	2925.44	3036.03	4343.58
9	659.92	355.72	1710.17	2875.56	3043.12	4255.48
10	694.39	356.48	1763.8	2961.1	2993.33	4274.27

Appendix D Detailed results ALNS example data

	NOE_1	NOE_2	NOE_3	NOE_4	NOE_5	NOE_6
1	498.08	277.58	1297.97	2518.92	2444.41	3731.18
2	485.4	277.09	1341.14	2422.46	2549.48	3782.47
3	484.88	281.23	1316.83	2477.61	2500.45	3801.43
4	487.88	278.88	1343.32	2530.71	2507.56	3827.76
5	484.91	273.16	1322.53	2531.43	2440.89	3976.06
6	484.91	272.67	1284.56	2435.14	2597.47	3731.98
7	481.4	276.06	1372.77	2404.25	2485.25	3735.41
8	493.57	279.78	1314.35	2516.34	2472.67	3743.7
9	480.16	278.06	1333.38	2498.36	2518.19	3757.52
10	483.6	287.5	1321.29	2476.71	2463.4	3764.48

Appendix E Detailed results example data after adaptation heuristic

	NOE_1	NOE_2	NOE_3	NOE_4	NOE_5	NOE_6
1	489.54	276.21	1318.21	2559.94	2439.41	3829.72
2	491.8	273.25	1350.67	2501.14	2609.5	3840.55
3	488.77	277.33	1339.73	2507.25	2512.58	3839.21
4	473.09	282.09	1342.25	2702.48	2601.25	3806.18
5	481.11	264.81	1323.11	2572.63	2446.84	3988.43
6	488.94	267.59	1289.67	2487.37	2642.74	3732.37
7	485.31	277.77	1393.59	2427.72	2509.51	3785.69
8	501.91	275.5	1336.72	2613.86	2513.34	3787.04

9	471.71	279.59	1371.25	2526.85	2516.62	3809.38
10	491.93	285.32	1337.17	2503.21	2498.03	3822.49

Appendix F Detailed results change in distance after removing customers under minimum threshold

	NOE_1	NOE_2	NOE_3	NOE_4	NOE_5	NOE_6
1	0.21	0.04	0.08	0.8	0	0.79
2	0	4.72	1.17	2.98	1.58	0.68
3	0.01	0.13	0.01	1.72	1.44	1.92
4	0	0.21	0.01	1.59	0.46	0.52
5	5.84	2.49	0.01	2.31	0.22	2.58
6	0.02	0.08	0.19	0	1.51	2.63
7	0	0.01	0.01	0.01	6.74	0.26
8	0.45	0.12	0.87	0.15	0.23	2.16
9	0	0.13	4.67	0.37	0.05	0.34
10	0.01	0.15	2.4	0.03	0.88	0.11

Appendix G Detailed results change in distance after inserting customers over maximum threshold

	NOE_1	NOE_2	NOE_3	NOE_4	NOE_5	NOE_6
1	27.71	9.81	29.35	62.55	8.1	136.68
2	17.84	10.96	14.56	106.88	79.79	82.23
3	16.59	4.87	37.18	63.32	26.26	72.67
4	2.52	6.73	1.32	209.84	120.62	21.45
5	14.68	0.12	10.94	85.55	28.91	54.73
6	11.13	7.69	17.95	90.92	71.67	45.86
7	7.86	5.11	27.84	41.59	59.18	92.07
8	19.48	2.64	30.94	135.73	59.44	67.76
9	4.92	3.39	67.62	54.77	11.75	83.43
10	19.2	1.33	27.72	36.43	52.16	76.98

Appendix H Detailed results change in distance after adapting tours over capacity

	NOE_1	NOE_2	NOE_3	NOE_4	NOE_5	NOE_6
1	35.19	10.98	9.01	17.35	11.89	29.06
2	11.21	8.29	2.76	21.87	16.41	20.43
3	10.06	7.85	14.04	24.81	11.11	31.57
4	17.31	2.08	2.22	28.8	25.4	42.17
5	11.34	5.95	9.55	39	21.71	34.68
6	6.4	12.35	10.77	36.46	23.16	39.92
7	3.94	3.18	6.32	17.68	29.97	39.36
8	10.53	6.81	7.03	34.68	16.83	19.13
9	12.68	1.38	24.64	23.84	12.51	27.06
10	9.86	3.01	8.66	8.46	15.6	18.33

Appendix I Detailed results change in distance after 2-opt

	NOE_1	NOE_2	NOE_3	NOE_4	NOE_5	NOE_6
1	0.85	0.15	0.03	3.39	1.21	8.28
2	0.23	1.78	1.1	3.34	1.77	3.04
3	2.64	0.79	0.23	7.15	1.58	1.39
4	0	1.22	0.15	7.67	1.07	0.34
5	1.3	0.03	0.8	3.04	1.04	5.09
6	0.68	0.34	1.88	2.23	1.74	2.92
7	0	0.2	0.69	0.43	1.22	2.17
8	0.16	0	0.67	3.37	1.72	3.14
9	0.69	0.35	0.44	2.07	0.75	4.16
10	1	0.35	0.78	1.44	1.04	0.53