Erasmus University Rotterdam
Erasmus School of Economics
Bachelor Thesis BSc^2 in Econometrics and Economics

An Adaptive Large Neighborhood Search algorithm for the Green Mixed Fleet Vehicle Routing Problem with Partial Recharging and Time Windows

Tijn Hoogeveen (499802)

Supervisor:	Ymro Hoogendoorn
Second assessor:	dr. Riley Badenbroek
Date:	29th June 2023

Abstract

In this thesis, we address the challenge of solving the Green Mixed Fleet Vehicle Routing Problem with Partial Recharging and Time Windows. In this version of the vehicle routing problem, a fleet consisting of both electric vehicles and internal combustion commercial vehicles is dispatched from a central depot to serve a set of customers, with the goal of minimizing the costs while restricting CO_2 emissions. We compare the performance of two heuristics in solving this problem: iterated local search (ILS) and adaptive large neighborhood search (ALNS). Our results demonstrate that the ALNS consistently outperforms the ILS, producing better solutions in 91% of the instances and improving the initial solution by an additional 10 percentage points more than the ILS on average. The ALNS also performs better in handling infeasible initial solutions, achieving feasibility for all such instances, while the ILS fails in three instances. Although the ALNS has longer computational times, its average running time remains under a minute even for the largest instances (with 100 customers).

1 Introduction

In the past decade, policymakers have become increasingly concerned about greenhouse gas emission reduction. As the transportation industry is one of the biggest contributors to these emissions, policies aiming to decrease emissions in this sector are crucial (Zelasky & Buonocore, 2021). There is thus a need to develop transportation and distribution systems that are both ecologically friendly and effective. Therefore, several researchers have started to view these transportation problems from an ecologically conscious perspective by integrating sustainability objectives into classic optimization problems. One of the most common of these transportation problems is the Vehicle Routing Problem (VRP). This problem optimizes a set of routes for a fleet of vehicles to traverse in order to deliver goods to a given set of customers. VRPs with ecologically conscious objectives included are known as Green-VRPs (G-VRPs).

Increasingly more research is done on these G-VRPs in the last years (Fernández Gil, Lalla-Ruiz, Gómez Sánchez, Castro et al., 2022). These G-VRPs take into account the CO_2 emissions from internal combustion commercial vehicles (ICCVs) and aim to find methods to decrease these emissions. In many cases, this is done by replacing ICCVs with electric vehicles (EVs). The VRP using electric vehicles is known as the Electric Vehicle Routing Problem (EVRP). The main advantages of using EVs compared to ICCVs is that they do not produce CO_2 emissions and that they are more silent. The major disadvantages of EVs are their limited battery capacities, long charging times, and the limited amount of public charging stations available. The EVRP, therefore, includes battery constraints of the vehicles and a set of recharging stations into the classical VRP.

This thesis will examine a mixed fleet model, in which both EVs and ICCVs are dispatched. A limit is set on the CO_2 emissions of the fleet. Furthermore, all customers will have time windows in which they need to be visited, and the model will allow for partial recharging at a charging station for the EVs. This model is called the Green Mixed Fleet Vehicle Routing Problem with Partial Recharging and Time Windows, or GMFVRP-PRTW, and has been introduced by Macrina, Di Puglia Pugliese, Guerriero and Laporte (2019).

The GMFVRP-PRTW, and G-VRPs in general, are NP-hard combinatorial optimization

problems. Solving them to optimality becomes computationally expensive for large instances. Therefore, various heuristics have been introduced to efficiently generate a solution to approach the optimal solution. In this thesis, two heuristical approaches will be implemented to solve the GMFVRP-PRTW, namely iterated local search (ILS) and adaptive large neighborhood search (ALNS).

The ILS metaheuristic first creates an initial solution and then uses a local search algorithm to iteratively improve this solution. In every iteration, this local search heuristic will move one customer to another route at a position such that the total cost decrease is maximized. The implementation of this heuristic will be based on Macrina et al. (2019).

The ALNS also starts with the creation of an initial solution. The search process of the ALNS is then characterized by a selection of destroy and repair heuristics, which denote the "neighborhoods" which will be searched. At each iteration, both a destroy and a repair heuristic will be applied to the current solution, after which is determined whether the new solution is accepted. This selection process allows slightly worse solutions to still be selected. The implementation will be based on Pisinger and Røpke (2007), modified for the GMFVRP-PRTW.

The advantages of the ALNS compared to the ILS are that due to the broader search, stemming from the larger neighborhoods, the randomization in neighborhood selection, and the chance that worse solutions are accepted, better minima can be found, as the ILS is likely to get stuck in a local minimum earlier. Therefore, the ALNS could lead to better solutions than the ILS. The main disadvantage of the ALNS is that it is computationally slower than the ILS.

In this thesis, the performance of both the ILS and the ALNS on the GMFVRP-PRTW will be compared, with a focus on the quality of the algorithms' solutions and computation speeds. Furthermore, we will examine the solutions derived from these algorithms.

In Section 2, we will cover the literature review. In Section 3, the problem statement will be defined. In Section 4, we will describe the iterated local search heuristic. In Section 5 the adaptive local search neighborhood heuristic will be covered. In Section 6, we will describe the numerical experiment and show the results. We will provide the conclusion and discussion in Section 7.

2 Literature review

Dantzig and Ramser (1959) were the first to introduce the concept that is now known as the Vehicle Routing Problem (VRP), which they named the "Truck Dispatching Problem." Their work focused on creating a model that addressed the efficient transportation of oil from a central depot to multiple gas stations using a fleet of identical trucks, with the aim of minimizing the total distance traveled. Building upon their foundation, Clarke and Wright (1964) expanded the problem into a broader context, formalizing the VRP framework. The VRP involves optimizing the distribution of goods to geographically dispersed customers from a central depot, considering trucks with varying capacities. Over the years, the VRP has become a highly researched area within Operations Research.

The introduction of green objectives in VRPs is a relatively new phenomenon. The first G-VRPs calculated CO_2 emissions and included them in the objective function. Figliozzi (2011) modeled the fuel spent as a proxy for the emissions. Bektaş and Laporte (2011) were the first to

explicitly model the emissions. They called this problem, with the emissions incorporated into the objective function, the Pollution Routing Problem (PRP).

Other versions of G-VRPs included the introduction of alternative fuel vehicles. Erdoğan and Miller-Hooks (2012) created such a G=VRP with alternative fuel vehicles. In this problem, a vehicle's fuel tank can be charged at designated charging stations, and its fuel consumption is proportional to the distance traveled by the vehicle.

This problem is extended by Schneider, Stenger and Goeke (2014), who introduced the Electric Vehicle Routing Problem with Time Windows (EVRP-TW). In the case of an EVRP, implementing time windows is more complex than for regular VRPs, as a charging strategy needs to be constructed which considers the charging times and their effect on the time windows. Schneider et al. (2014) computed these charging times by using their battery state of charge on arrival at a charging station, modeling a full recharge at each charging station visit.

Goeke and Schneider (2015) considered a green mixed fleet VRP including time window (GMFVRP-TW), in which both EVs and ICCVs are dispatched. They introduced a realistic energy consumption model considering factors like speed, vehicle mass, and gradient and proposed three objective functions: minimizing distance, minimizing energy and labor costs, and considering battery replacement costs after depreciation. Like Schneider et al. (2014), they modeled a full recharge at each charging station.

Macrina et al. (2019) built on this GMFVRP-TW, and extended it by allowing a partial recharging strategy (GMFVRP-PRTW), as this can lead to better solutions. Furthermore, they took into account the CO_2 emissions produced by the ICCVs and explicitly restricted these.

As mentioned in the introduction, these complex vehicle routing problems can only be solved to optimality using exact methods for small instances. Therefore, heuristics are used to obtain high-quality solutions for large instances of these problems.

The earlier mentioned paper by Macrina et al. (2019) uses an Iterated Local Search (ILS) for this purpose. The ILS starts from an initial solution and iteratively searches for an improvement in the solution. The ILS implemented by Macrina et al. (2019) specifically searches for improvement following from moving one customer to another route at each iteration.

Another popular heuristic used in VRPs is the Adaptive Large Neighborhood Search (ALNS). This method has been introduced by Røpke and Pisinger (2006a), for the pickup and delivery problem with time windows. The ALNS searches for a much broader neighborhood, by making multiple changes at each iteration, and by allowing to accept solutions with a worse objective value. Pisinger and Røpke (2007) build a framework to implement the ALNS for a variety of VRPs. This framework will be used to apply the ALNS to the GMFVRP-PRTW,

The ALNS has been implemented on several G-VRPs. For example, by Bektaş and Laporte (2011) for the earlier introduced PRP and by Goeke and Schneider (2015) for the GMFVRP-TW. The main difference between the implementation of the latter and this thesis is that we will allow for partial recharging, and will restrict CO_2 emissions, in a similar fashion as Macrina et al. (2019).

3 Problem definition

The problem definition is based on that of Macrina et al. (2019), with some minor alterations. We define N as the set of customers and R the set of recharging stations. Each recharging station is copied σ times, resulting in set R', with $|R'| = (\sigma + 1)R$. This ensures each station can be visited at most $\sigma + 1$ times. A special node is the depot, from which each vehicle departs and where each vehicle returns. This depot is duplicated to indicate a starting depot s and an ending depot t. We define $V = R \cup N \cup \{s, t\}$ and $V' = R' \cup N \cup \{s, t\}$. The graph on which the problem is defined is then $G = (V', A)$, where A is the set of all possible arcs between two nodes in V'.

The fleet consists of two types of vehicles: electric vehicles (EVs) and conventional, i.e., internal combustion commercial vehicles (ICCVs). In notation, the superscripts E and C will be used to refer to electric and conventional vehicles respectively. The fleet sizes for both vehicle types are unrestricted. All routes begin at s and end at t. All customers must be visited by exactly one vehicle, and all customers $i \in N$ have a demand of q_i in kg and a service time of s_i in hours. Furthermore, each customer has a time window $[e_i, l_i]$, denoted in hours since the start time of the problem, in which the customer needs to be visited, i.e., the arrival time needs to be in this window. For each arc $(i, j) \in A$, the distance is denoted by d_{ij} in km, and the travel time by t_{ij} in hours. All nodes $i \in V'$ have coordinates (x_i, y_i), and the distance between two nodes is given by the Euclidian distances. The travel time is a linear function of the distance traveled, as it is assumed that the vehicle speed is constant. The costs in euros per km are given by c_{ij}^E and c_{ij}^C for EVs and ICCVs respectively. The time limit of a single route is T, this is established by setting $l_t = T$. Furthermore, the vehicles have load capacities Q^E and Q^C, and electric vehicles have battery capacity B^E. All recharging stations $i \in R'$ have charging speed ρ, in kW. The parameter π denotes the coefficient of energy consumption, in kWh/km. Partial recharging is allowed at each recharging station. The decision variables in the problem are

- x_{ij}^E, a binary variable denoting whether an electric vehicle traverses from node i to j, with $i, j \in V'$,
- x_{ij}^C, a binary variable denoting whether a conventional vehicle traverses from node i to j, with $i, j \in V'$,
- u_i^E, the load left (in kg) in the electric vehicle after visiting node $i \in V'$,
- u_i^C, the load left (in kg) in a conventional vehicle after visiting node $i \in V'$,
- τ_i, the arrival time (in seconds) of the vehicle at node $i \in V'$,
- z_{ij}, the battery charge (in kWh) available at arrival at node j after directly before having visited node i, defined for $i, j \in V'$, and
- g_{ij}, the energy recharged (in kWh) at recharging station $i \in R'$, before travelling to node $j \in V'$ immediately afterwards.

Table 1: Notation in the MIP formulation

Symbol	Type	Description	Domain	Unit
N	Set	Set of customers		
R	Set	Set of recharging stations		
R'	Set	Set of recharging stations, including σ copies of each station		
V	Set	Set of both customers and recharging stations ($V = N \cup R$)		
V'	Set	Set of both customers and recharging stations, including copies ($V' = N \cup R'$)		
A	Set	Set of all possible arcs between two nodes in V' ($A = \{(i,j) : i,j \in V', i \neq j\}$)		
s,t	Element	Depot, denoting starting point (s) and ending point (t) of all routes	$s,t \in R'$	
Q^E, Q^C	Parameter	Loading capacities of electric (E) and conventional (C) vehicles		kg
π	Parameter	Constant coefficient of energy consumption		kWh/km
ρ	Parameter	Constant recharging speed		kW (=kWh/hour)
B^E	Parameter	Maximum battery capacity of an electric (E) vehicle		kWh
q_i	Parameter	Demand of customer $i \in N$ ($q_i = 0, \forall i \in R'$)	$i \in V'$	kg
s_i	Parameter	Service time for customer i	$i \in N$	hours
e_i, l_i	Parameter	Time window for node i, from e_i to l_i	$i \in V'$	hours
(x_i, y_j)	Parameter	Coordinates of node i	$i \in V'$	km
c_{ij}^E, c_{ij}^C	Parameter	Travel costs for electric (E) and conventional (C) vehicles from i to j	$i,j \in V'$	€/km
d_{ij}	Parameter	Travel distance from i to j	$i,j \in V'$	km
t_{ij}	Parameter	Travel time from i to j	$i,j \in V'$	hours
x_{ij}^E	Decision variable	1 if an electric (E) vehicle travels from i to j (0 otherwise)	$i,j \in V'$	
x_{ij}^C	Decision variable	1 if a conventional (C) vehicle travels from i to j (0 otherwise)	$i,j \in V'$	
z_{ij}	Decision variable	Energy available when arriving at node j from the node i	$i,j \in V'$	kWh
g_{ij}	Decision variable	Energy recharged at node i for travelling to node j	$i \in R', j \in V'$	kWh
τ_i	Decision variable	Arrival time of the vehicle to the node i	$i \in V'$	hours
u_i^E	Decision variable	Amount of load left in electric vehicle after visiting node i	$i \in V'$	kg
u_i^C	Decision variable	Amount of load left in conventional vehicle after visiting node i	$i \in V'$	kg
$\varepsilon(u_i^C)$	Function	Models the emission factor of a conventional (C) vehicle using its load at i	$i \in V'$	kg CO_2 / km

For the ICCVs, an overall CO_2 emission limit is in place. The emissions are computed using function $\varepsilon(u_i^C)$, which models the emission factor of an ICCV using its load at a given node i. The function is a piecewise constant function, which is defined as

$$\varepsilon(u_i^C) = \begin{cases} 0.77 & \text{if} \quad 0 \quad \leq u_i^C \leq \quad 0.25Q^C \\ 0.83 & \text{if} \quad 0.25Q^C \quad < u_i^C \leq \quad 0.50Q^C \\ 0.90 & \text{if} \quad 0.50Q^C \quad < u_i^C \leq \quad 0.75Q^C \\ 0.95 & \text{if} \quad 0.75Q^C \quad < u_i^C \leq \quad Q^C \end{cases} \qquad \forall i \in V', \tag{1}$$

and denotes the CO_2 emitted in kg/km. These values are based on Macrina et al. (2019), computed using a vehicle with capacity $Q^C = 10000$ kg.

All notations used and their explanations can be found in Table 1. The mixed integer problem at hand is:

$$\text{minimize} \quad \sum_{(i,j)\in A} c_{ij}^E d_{ij} x_{ij}^E + \sum_{(i,j)\in A} c_{ij}^C d_{ij} x_{ij}^C, \tag{2a}$$

$$\text{subject to} \quad \sum_{j\in V'} (x_{ij}^E + x_{ij}^C) = 1 \qquad i \in N, \tag{2b}$$

$$\sum_{j\in V'} x_{ij}^E \leq 1 \qquad i \in R', \tag{2c}$$

$$\sum_{j\in V'\setminus\{s\}} x_{ij}^E - \sum_{j\in V'\setminus\{t\}} x_{ji}^E = 0 \qquad i \in V', \tag{2d}$$

$$\sum_{j\in V'\setminus\{s\}} x_{ij}^C - \sum_{j\in V'\setminus\{t\}} x_{ji}^C = 0 \qquad i \in V, \tag{2e}$$

$$\sum_{i\in V'\setminus\{s\}} x_{si}^E - \sum_{j\in V'\setminus\{t\}} x_{jt}^E = 0, \tag{2f}$$

$$\sum_{i\in V'\setminus\{s\}} x_{si}^C - \sum_{j\in V'\setminus\{t\}} x_{jt}^C = 0, \tag{2g}$$

$$u_j^E \leq u_i^E - q_i x_{ij}^E + Q^E(1 - x_{ij}^E) \qquad i \in V'\setminus\{t\}, j \in V'\setminus\{s\}, \tag{2h}$$

$$u_j^C \leq u_i^C - q_j x_{ij}^C + Q^C(1 - x_{ij}^C) \qquad i \in V\setminus\{t\}, j \in V\{s\}, \tag{2i}$$

$$u_t^E = 0, \tag{2j}$$

$$u_t^C = 0, \tag{2k}$$

$$u_j^E \leq Q^E \qquad j \in V', \tag{2l}$$

$$u_j^C \leq Q^C \qquad j \in V, \tag{2m}$$

$$\tau_j \geq \tau_i + (t_{ij} + s_i)x_{ij}^E - M(1 - x_{ij}^E) \qquad i \in N, j \in V', \tag{2n}$$

$$\tau_j \geq \tau_i + (t_{ij} + s_i)x_{ij}^C - M(1 - x_{ij}^C) \qquad i \in N, j \in V, \tag{2o}$$

$$\tau_j \geq \tau_i + t_{ij}x_{ij}^E + \frac{1}{\rho}g_{ij} - M(1 - x_{ij}^E) \qquad i \in R', j \in V', \tag{2p}$$

$$e_j \leq \tau_j \leq l_j \qquad j \in V', \tag{2q}$$

$$z_{ij} \leq (z_{hi} + g_{ij}) - \pi d_{ij}x_{ij}^E + M(1 - x_{ij}^E) + M(1 - x_{hi}^E) \qquad h, i, j \in V', i \neq s, i \neq j, i \neq h, j \neq h, \tag{2r}$$

$$z_{sj} \leq B^E - \pi d_{sj}x_{sj}^E + M(1 - x_{sj}^E) \qquad j \in V'\setminus\{s\}, \tag{2s}$$

$$g_{ij} \leq B^E - z_{hi} + M(1 - x_{ij}^E) + M(1 - x_{hi}^E) \qquad h, j \in V', i \in R', \tag{2t}$$

$$\sum_{(i,j)\in A} \varepsilon(u_i^C) d_{ij} x_{ij}^C \leq UB, \tag{2u}$$

$$x_{ij}^E, x_{ij}^C \in \{0,1\} \quad i,j \in V', \qquad u_i^E, u_i^C, z_{ij}, \tau_i \geq 0 \quad i \in V', \qquad g_{ij} \geq 0 \quad i \in R', j \in V'. \tag{2v}$$

The objective function (2a) minimizes the total costs. The total costs are dependent on the costs of the distance traveled by the EVs and the ICCVs. Constraints (2b) ensure each customer is visited exactly once. Constraints (2c) ensure that each copy of a recharging station is visited at most once. Constraints (2d)-(2e) are the flow conservation constraints. Constraints (2f)-(2g) ensure that all vehicle routes start and end at the depot. Constraints (2h)-(2i) model the load change after visiting a customer, using this customer's demand. Together with Constraints (2j)-(2k), which ensure the vehicles are empty when they return at the depot, they model the course of the vehicle load during the routes. Constraints (2l)-(2m) ensure that the maximum capacity of a vehicle is not exceeded. Constraints (2n)-(2o) model the arrival time at a node after serving a customer at the previous node, while Constraints (2p) model the arrival time at a node after using a recharging station at the previous node. Constraints (2q) ensure that each customer is visited within its time window. Constraints (2r) model the course of the battery charge of an EV, where Constraints (2s) initialize this charge. Constraints (2t) model the partial recharging. They ensure that at a recharging station, there can be recharged at most until the battery is completely recharged. Constraint (2u) models that the upper bound on the emissions, UB, cannot be exceeded. Finally, Constraints (2v) define the ranges for the decision variables, which ensure that the load, and charge at any moment cannot be negative. Furthermore, they define that the time at any node, and the amount recharged at any recharging station are nonnegative, and ensure the binary nature of the route selection variables.

4 Iterated local search (ILS) heuristic

In this section, the ILS heuristic, based on Macrina et al. (2019), will be explained. The general structure of the heuristic is described in Algorithm 1.

Algorithm 1 Iterated local search (ILS)

```
Generate the initial solution η^init (see Section 4.1)
Set current solution η' = η^init
while Stop condition is not verified do
    if current solution η' is feasible then
        Apply the local search improvement heuristic (see Section 4.2.1)
    else
        Apply the local search improvement heuristic using the penalty function (see Section 4.2.2)
    end if
end while
return: best solution η* = η'
```

First, an initial solution η^{init} is generated using the process described in Section 4.1. Then, until the stop criterion is satisfied, the local search is applied. This process will be explained in Section 4.2. In case the initial solution is feasible, the improvement heuristics described in Section 4.2.1 are used to improve the solution. If the initial solution is infeasible, a penalty function is introduced to seek a feasible solution. This function is explained in 4.2.2. In case a feasible solution is found by this process, afterwards the regular improvement heuristic is used.

4.1 Initial solution generation

An initial solution is generated by the sequential insertion heuristic, as provided by Macrina et al. (2019), based on Solomon (1987). Algorithm 2 provides the structure of this algorithm.

Algorithm 2 Sequential insertion heuristic (SIH)

```
Clustering: divide N into C', and E' (see Section 4.1.1)
Insertion heuristic: obtain η^C from C' (see Section 4.1.2)
if customers in C' not served by η^C then
    Update E' by including unserved customers
end if
Insertion heuristic: obtain η^E (see Section 4.1.3)
return: solution η' = η^C ∪ η^E
if customers in E' not served by η^E then
    Repeat conventional insertion heuristic (see Section 4.1.2) on unserved customers, while relaxing emission constraint
end if
```

First, the customers (set N) are divided into two clusters to indicate whether they will be served by EVs (set E') or ICCVs (set C'). This clustering process is described in Section 4.1.1. After the clustering process, the insertion heuristic for conventional routes, described in Section 4.1.2, is used to construct routes that serve the customers in C'. If it is infeasible to serve some of these customers due to the emission constraint, these unserved customers will be moved to set C'. Next, the insertion heuristic for electrical routes, described in Section 4.1.3, will be executed to construct electrical routes to serve the remaining customers. In case there exist some customers that cannot be served by electrical vehicles due to battery capacity constraints, despite recharging attempts, these customers will be moved to set C'. Finally, these last unserved customers will be served by routes constructed by the insertion heuristic for conventional routes, but this time while relaxing the emission constraint. It is thus possible that the initial solution is infeasible. In those cases, the initial solution will only violate the emission constraint.

4.1.1 Clustering algorithm

The clustering algorithm will divide the customers, N, into two subsets, E' and C', which will be served by EVs and ICCVs respectively. The clustering process is described in Algorithm 3.

In order to find these two clusters, we initialize sets E and C both with a copy of s, the starting depot. To select customers into a cluster, scores are defined for each customer $i \in N$ for each cluster: p_i^E and p_i^C, both with range [1,10]. The computation of the scores is explained in the following paragraphs. In each iteration, the customer with the highest value for p_i^E and the customer with the highest value for p_i^C are selected into clusters E and C respectively. In case one customer has the highest value for both scores, this customer is placed into the cluster for which it has the highest score. Afterwards the scores are recalculated for the remaining customers and the process is repeated until each customer is selected into one of the two clusters. Finally, the depot is removed from sets E and C in order to obtain the final clusters E' and C'.

Algorithm 3 Clustering Algorithm

initialize: sets E and C both only containing start depot s
while $N \backslash (E \cup C) \neq \emptyset$ **do**
    (Re)calculate the barycenters B^E and B^C
    Compute p_i^E and p_i^C for all $i \in N \backslash (E \cup C)$
    Define $i_E^* = \text{argmax}_{i \in N \backslash (E \cup C)} \{p_i^E\}$
    Define $i_C^* = \text{argmax}_{i \in N \backslash (E \cup C)} \{p_i^C\}$
    if $i_E^* \neq i_C^*$ **then**
        Assign i_E^* to E and i_C^* to C
    else if $p_{i_E^*}^E > p_{i_C^*}^C$ **then**
        Assign customer i_E^* to E
    else
        Assign customer i_C^* to C
    end if
end while
Remove s from E and C to obtain E' and C'
return: E' and C'

EV case For electric vehicles, the score of customer i is determined by the difference in distance between i and the customer already selected in E closest to the barycenter of E, scaled by the maximum such difference among customers already selected in E. Mathematically, this gives

$$p_i^E = 10 - \frac{d_{iB^E} - d_{\min}^E}{d_{\max}^E - d_{\min}^E} \cdot 9, \tag{3}$$

where d_{iB^E} is the Euclidian distance between customer i and B^E, the barycenter of set E with coordinates given by

$$x_{B^E} = \frac{1}{|E|} \sum_{j \in E} x_j, \tag{4}$$

$$y_{B^E} = \frac{1}{|E|} \sum_{j \in E} y_j, \tag{5}$$

and with $d_{\min}^E$ and $d_{\max}^E$ defined as

$$d_{\min}^E = \min_{j \in N} d_{jB^E}, \tag{6}$$

and

$$d_{\max}^E = \max_{j \in N} d_{jB^E}. \tag{7}$$

ICCV case For conventional vehicles, the score of customer i is determined by two factors: the difference in distance between i and the customer in C closest to the barycenter of C, scaled by the maximum such difference among customers already selected in C, and the demand of i, scaled by the minimum and maximum demands in the total set N, denoted by $q_{\min}$ and $q_{\max}$ respectively. Mathematically

$$p_i^C = \lambda \; pDist_i^C + (1 - \lambda) pQ_i, \tag{8}$$

with λ a hyperparameter in the range $[0, 1]$, controlling the relative weight of the two scores, and where

$$pQ_i = 10 - \frac{q_i - q_{\min}}{q_{\max} - q_{\min}} \cdot 9, \tag{9}$$

$$pDist_i^C = 10 - \frac{d_{iB^C} - d_{\min}^C}{d_{\max}^C - d_{\min}^C} \cdot 9, \tag{10}$$

with B^C, $d_{\min}^C$ and $d_{\max}^C$ defined in similar fashion as for E.

4.1.2 Insertion algorithm conventional routes

A conventional route with vehicle number k, denoted as Z_k^C, is initialized by serving the customer $i' \in C' \backslash C^*$ (where C^* denotes the customers already visited by a conventional route) with the lowest value for l_i, the closure of its time window. Now $Z_k^C = (s, i', t)$. Afterward, the algorithm iteratively computes for all unvisited customers $u \in C' \backslash (C^* \cup Z_k^C)$, what the most efficient position is to visit customer u on current route Z_k^C. This is done by computing the added cost for a given customer u for all possible positions on the route. Denote the current route as $(s, i_1, i_2, ..., i_m, t)$. Then for a given customer u, and for each possible position $n \in \{1, ..., m, m+1\}$, the algorithm computes

$$f_1(u, n) = c_{i_{n-1},u}^C d_{i_{n-1},u} + c_{u,i_n}^C d_{u,i_n} - c_{i_{n-1},i_n}^C d_{i_{n-1},i_n}. \tag{11}$$

This is in line with the idea used in Macrina et al. (2019). Inserting a customer at position n denotes adding it between customer i_{n-1} and customer i_n, where i_0 denotes s, and i_{m+1} denotes t. After (11) is computed for all possible positions n, the minimum is determined: $n_u^* = \text{argmin}_{n \in \{1,...,m,m+1\}}\{f_1(u, n)\}$, where n_u^* is the optimal position to place customer u.

After this is done for each customer, (12) is used to compute the cost saving of adding a customer u to its optimal position n_u^* on the current route, compared to serving it directly from the depot.

$$f_2(u, n_u^*) = c_{s,u}^C d_{s,u} + c_{u,t}^C d_{u,t} - f_1(u, n_u^*) \tag{12}$$

Now u^* denotes the customer with the maximum value of (12). This is the customer that we preliminarily add to route Z_k^C. We check whether the load and time constraints are verified for the route with the new customer added, and whether the total emission constraint is verified. If all constraints hold, the customer is definitively added to the route and the algorithm will search for a potential new customer to add to the route. If the emission constraint does hold, but the time window and/or load constraints cannot hold with the new customer, customer u^* will not be added, this route is terminated, and a new route is initialized. If the total emission constraint does not hold, this customer will not be added to the route, no new route will be initialized, and all remaining customers in C' that have not been served will be moved to E'.

A formal description of the conventional route insertion algorithm can be found in Appendix A.1.

4.1.3 Insertion algorithm electrical routes

The electrical route insertion algorithm is similar to the previously described insertion algorithm for conventional routes. The main difference is the creation of routes visiting recharging stations. A route is initially created in a similar fashion as the conventional route, however evidently without checking for emission constraints. After a route creation process is terminated because the time window or load conditions do not allow for an extra customer, the route is checked for battery charge feasibility. In case the route is infeasible because of an empty battery along the route, a recharging station is added to the route. Specifically, at the position before the node where the charge runs out, the nearest recharging station is inserted, if it is feasible to reach this recharging station. If this is not feasible, the position is iteratively put one node before, until a charging station can feasibly be reached. The recharge is taken such that the end depot can be reached if the battery capacity allows it. Otherwise, the battery will be recharged until it is full, and new recharging stations are added in a similar fashion until the battery constraints are satisfied. Afterwards, the time constraints will be rechecked. If they are not satisfied, a feasible solution is constructed by iteratively removing customers. After each customer removal, the energy charged is optimized for each recharging station, and superfluous recharging stations are removed. If it is not possible to serve all customers in E', the remaining customers are transferred back into C' and the emission constraint is relaxed. Again, the formal pseudo-algorithms for the EV insertion algorithm and the route repairing algorithm can be found in Appendix A.1.

4.2 Local search

After the initial solution generation, the local search algorithm searches iteratively for a better solution. For a fixed number of iterations, the improvement heuristic searches for an improvement to the solution by moving customers to a new route (see Section 4.2.1). In case the initial solution generated by the SIH is infeasible, a penalty function is introduced to find a feasible solution. Section 4.2.2 describes this process.

4.2.1 Improvement heuristic

The local search improvement heuristic consists of three strategies: the optimal customer change within the conventional routes, the optimal customer change within the electrical route, and the optimal customer change between all routes. The following paragraphs will specify the exact algorithms used. At each iteration, randomly one of the three strategies is chosen. Formal descriptions of these algorithms can be found in Appendix A.2.

Strategy 1: best customer change in conventional routes This strategy searches for the maximal cost saving by moving a customer from a certain conventional route to another conventional route. It does so by iteratively searching for each customer, for each other route, the cheapest position on that route to move the customer to. It then computes the maximal cost saving between all options. Before computing these costs, for each node insertion at each position, first, the constraints are checked. In case this yields an infeasible solution, this move will not be considered a possible option.

Strategy 2: best customer change in electrical routes This strategy searches for the maximal cost saving by moving a customer from a certain electrical route to another electrical route. It does so in a similar fashion as strategy 1, but it includes recharging stations where necessary and possible, in a similar manner as in the electrical route insertion. The key difference is that here no customers will be removed in case the time constraints are violated due to the addition of a recharging station. For such cases the insertion at this position is seen as infeasible.

Strategy 3: best customer change in all routes The final strategy combines strategies 1 and 2 by searching for the maximal cost saving by moving a customer from any route to any other route. In case the customer is moving to an electrical route and the battery constraints are violated, it uses the previously mentioned route repairing algorithm to obtain a feasible route, where possible.

4.2.2 Penalty function

In case an infeasible solution is generated by the SIH, the improvement heuristic is modified to ensure a feasible solution is found. In particular, as the emission constraint has been relaxed in this case, it is necessary to obtain a solution with feasible emissions. The objective function is modified to

$$z'(\eta) = z(\eta) + \theta e(\eta), \tag{13}$$

where θ is the penalty factor and $e(\eta)$ denotes the penalty function, defined as

$$e(\eta) = \max \left\{ 0, \sum_{(i,j)\in A} \varepsilon(u_i^C) d_{ij} x_{ij}^C - UB \right\}. \tag{14}$$

Penalty factor θ is initialized at 1, but increases with 10% at each iteration until the emission constraint is satisfied. Starting from the infeasible solution, the approach described in 4.2.1 is applied with the alternative objective function. This efficiently searches for a high-quality solution that is feasible.

5 Adaptive large neighborhood search (ALNS) heuristic

In this section, the ALNS will be described. The general structure of the ALNS is described in Algorithm 4. The ALNS begins with the same initial solution as the ILS, described in Section 4.1. From this initial solution, the search procedure starts. Within each iteration, it selects destroy and repair neighborhoods based on a roulette wheel selection. This selection process is described in Section 5.2. A new solution is generated by applying the selected neighborhood heuristics to the current solution. The probability of acceptance of the new solution is described in Section 5.1. If the new solution is accepted, it becomes the current solution and if its objective function value is better than the best solution, the current global minimum is updated. Based on the new solution's results, the scores of the neighborhoods are updated. The temperature, which regulates the acceptance probability is also adjusted after each iteration. In case the initial solution is violating the emission constraint, perturbation is applied first. This process is

described in Section 5.3. The neighborhoods are described in Section 5.4. The loop continues until the stop criteria are satisfied, which are described in Section 6.2.3.

Algorithm 4 Adaptive large neighborhood search (ALNS)

```
Generate the initial solution η^init (same as in the ILS)
set η, the current solution, and η*, the current best solution, to η^init
while Stop criteria not met do
    select destroy neighborhood N^- and repair neighborhood N^+ using roulette wheel selection, based on {π_j}
    generate a new solution η' from current solution η using the heuristics corresponding to chosen destroy and repair neighborhoods
    determine whether new solution η' is accepted using current temperature T
    if new solution η' is accepted then
        set current solution to new solution, η = η'
        if z(η) < z(η*) then
            set current best to current solution, η* = η
        end if
    end if
    update scores π_j of N^- and N^+
    update temperature T
end while
return: the best solution η*
```

5.1 Solution acceptance

Simulated annealing is used as the master search framework. After each iteration, first we check whether the new solution η' satisfies all constraints. If this is not the case, η' is automatically rejected. If all constraints are satisfied, the probability to accept η', and replace current solution η is computed with the following formula

$$P(\text{accept } x' \text{ at iteration } i) = \min\left\{ \mathrm{e}^{\frac{z(\eta)-z(\eta')}{T_i}}, 1 \right\}. \tag{15}$$

From this formula, it can be seen that when $z(\eta) > z(\eta')$, i.e., new solution η' has lower costs than current solution η, η' is always accepted. The temperature mechanism plays a crucial role in guiding the search process. The temperature, denoted by T_i, with i the iteration number, is updated after each iteration. Higher temperatures increase the probability of accepting solutions that are worse than the current solution. To simulate an annealing process, the temperature gradually decreases over time. This decrement is controlled by the cooling rate hyperparameter c, which takes a value in the range $(0, 1)$. The temperature at iteration i is determined using the recursive formula

$$T_i = T_{i-1} \cdot c. \tag{16}$$

The starting temperature T_0 is selected, such that in the first iteration, a solution with increase in objective value of w times the initial solutions objective value $z(\eta^{\text{init}})$ has a 50% chance to get accepted. Here, w represents a hyperparameter that influences the acceptance

probability threshold, in the range $(0, 1)$. The starting temperature is obtained by rewriting this condition of the starting temperature

$$\mathrm{e}^{\frac{z(\eta^{\text{init}})-(1+w)z(\eta^{\text{init}})}{T_0}} = 0.5, \tag{17}$$

to

$$T_0 = -\frac{w \cdot z(\eta^{\text{init}})}{\ln(0.5)}. \tag{18}$$

5.2 Neighborhood selection

Roulette wheel selection is used at each iteration to select the destroy and repair mechanisms. After each iteration, the probabilities to select these neighborhoods are updated.

For each destroy and repair neighborhood the probability to be selected is initially equal. Afterwards, the probabilities to select a neighborhood are updated based on their successes in creating good solutions. Each neighborhood has a score, which is updated after it has been selected in an iteration. Specifically, the score is incremented by the following values after the corresponding results:

1. The score of neighborhood j is incremented by σ_1 if solution η', generated using neighborhood j, is accepted.
2. The score of neighborhood j is incremented by σ_2 if the objective value of solution η', generated using neighborhood j, is better than the current solution, i.e., $z(\eta') < z(\eta)$.
3. The score of neighborhood j is incremented by σ_3 if the objective value of solution η', generated using neighborhood j, is better than the all-time best solution, i.e., $z(\eta') < z(\eta^*)$.

Note that event 3 implies events 2 and 1, and that event 2 implies event 1. These events are similar to the events used in Pisinger and Røpke (2007), with the exception that they include the condition that a solution needs to be original in criteria 1 and 2. Furthermore, in Pisinger and Røpke (2007) the criteria are formulated such that they are mutually exclusive.

The probability to select destroy neighborhood j, denoted by π_j^-, is recomputed after each iteration by dividing the score of neighborhood j by the sum of all destroy neighborhood scores. This is done similarly for the probabilities of the repair neighborhoods, denoted with π_j^+. The scores of all neighborhoods are initialized at $10 \cdot (\sigma_1 + \sigma_2 + \sigma_3)$ at the start. This ensures that successes in the first iterations do not skew the probability distribution too much. This is done as an alternative to the smoothened score calculations used in Pisinger and Røpke (2007).

5.3 Dealing with infeasible initial solutions

In the case that initial solution η^{init} is infeasible, i.e., does not satisfy the emission constraint, the acceptance criteria are loosened in order to obtain a feasible solution as soon as possible. Firstly, in this case a new solution can be accepted while violating the emission constraints, as long as the emissions of η' are lower than those of the current solution η. Furthermore, to maximize the perturbation speed, $z(\eta)$ in (15) is set to infinity as long as the current solution does not satisfy the emission constraints. In practice, this means that any solution that produces

lower emissions than the current solution will be accepted in the perturbation phase.
Additionally, in order to increase the neighborhood space, the maximum number of customers that will be removed by the destroy neighborhoods, $qMax$, which will be described in Section 5.4.1, is enlarged to the total number of customers.
After any feasible solution is found, the original acceptance criteria is used from that point onward. This means that from that point on only feasible solutions will be accepted and with the probability of the original formula in 15. $qMax$ will also return to its original value.

5.4 Destroy and repair neighborhoods

In this subsection, the search neighborhoods are described. First, in Section 5.4.1 the destroy neighborhoods are explained. Then, in Section 5.4.2 the repair neighborhoods are described. Finally, in Section 5.4.3 is explained how noise is introduced to certain neighborhoods.

5.4.1 Destroy neighborhoods

For all destroy heuristics except the route removal, first, the number of customers to be removed, q, needs to be selected. q is a randomly generated integer on the interval $qMin$ and $qMax$, all with equal probability.

1: Route removal Randomly select a route and delete this route. All routes have equal probability to be selected.

2: Random customer removal Randomly select q customers and remove them from their respective routes.

3: Removal of q worst customers Compute the removal score for each customer. This means the difference in the costs of the route with and without this customer. For the computation of the score without this customer, in case the route is served by an electric vehicle, all recharging stations will be reevaluated and removed if redundant. The worst customer is removed. Repeat q times.

4: Removal of q worst customers with noise The same algorithm as the previous one is used with an adjustment. After computing the removal score for each customer, noise is added to the removal score function, after which the maximum is chosen. The noise generation process is further described in Section 5.4.3.

5.4.2 Repair neighborhoods

The following four heuristics are used to obtain full solutions from the partial solutions obtained by executing one of the destroy neighborhoods. For all repair neighborhoods it is possible that from a given partial solution, no feasible solution can be obtained. In that case, the partial solution will be discarded. The current solution will not be updated and will again be used in the next iteration.

1: Electric routes first, conventional second Insert one or multiple electric routes in the same way as in the construction heuristic, until none of the unserved customers can be feasibly added to these new routes. If unserved customers remain, serve them by initializing new conventional vehicles, until no customers can be added without violating the emission constraint. In case there are remaining customers unserved, insert them into current routes using the greedy algorithm (see repair neighborhood 3).

2: Conventional routes first, electrical second Insert one or multiple conventional routes in the same way as in the construction heuristic, until none of the unserved customers can be feasibly added to these new routes. If there remain customers that are unserved, serve them by initializing new electrical vehicles, until no customers can be feasibly added. In case there are remaining customers unserved, insert them into current routes using the greedy heuristic.

3: Greedy algorithm Insert customers using the greedy algorithm as described. For all customers that have been removed in the destroy algorithm, compute the added costs to add this customer for all routes and all positions. Select the cheapest customer and insert it at its best possible place. Repeat this process until all customers are served. The pseudocode is given in Appendix A.3.

4: Regret-2 heuristic The regret-2 heuristic is similar to the greedy algorithm, but instead of selecting the cheapest customer, the customer with the highest difference in added costs between its cheapest and second-cheapest position is selected and inserted at its cheapest place. The pseudocode is again given in Appendix A.3.

5.4.3 Noise introduction

In order to make the heuristics less deterministic, noise is added in the following cases. For destroy neighborhood 4, noise is always introduced. Furthermore, all repair neighborhoods can be executed with or without noise. The selection whether repair neighborhoods will be executed with or without noise is done using similar roulette wheel selection as previously described in Section 5.2. Separate scores are kept track of for the case with and the case without noise.

The noise is generated by drawing a number from a uniform distribution between 0 and the distance between the depot and the customer furthest from the depot. The noise is generated for and added to each comparison score. This means that it will be added to the costs of each customer for each position of each route, after which the comparison as mentioned in the heuristics will take place.

6 Numerical experiments

In this section, the numerical experiments will be described and their results will be given. First, in Section 6.1 the problem instances are introduced and explained. Subsequently, in Section 6.2 the parameter selection is described. Finally, Section 6.3 shows the results of the experiments.

6.1 Problem instances

For the numerical experiments, the instances created by Schneider et al. (2014) will be used as in Macrina et al. (2019). These are derived from the well-known instances of Solomon (1987), with the addition of recharging stations. These instances provide the sets of customers (N), recharging stations (R), the depot (s, t), demands ($q_i, \forall i \in N$), service times ($s_i, \forall i \in N$) time windows ($e_i, l_i, \forall i \in V$), and coordinates of all nodes, from which the distances ($d_{ij}, \forall (i,j) \in A$) will be computed using Euclidian distance. The costs are assumed to be unitary (1 €/ km), for all vehicles and all routes. The travel times are established by computing $t_{ij} = \frac{d_{ij}}{v}$, with v the constant velocity, specified in each instance. The velocity is assumed to be the same for ICCVs and EVs. Furthermore, all problem instances specify the battery capacity for the EVs, B^C, the vehicle loading capacities, which are assumed to be equal for Q^E and Q^C. The parameters defining the coefficient of energy consumption, π, and the recharging speed, ρ, are also defined for each problem instance. Because of the artificial nature of the instances, the emission coefficient function $\varepsilon(u_i^C)$ is used as introduced in Section 3, regardless of the value of Q^C, as provided in the instance. For example, if the vehicle capacity is specified at 200 kg for a specific instance, and the load at a given node is 160 kg, or 0.8 Q^C, $\varepsilon(u_i^C)$ will take a value of 0.95 kg CO_2 / km.

Upper bound emissions The upper bound will be computed by taking $\alpha \cdot UB_{\max}$, for $\alpha \in \{0.25, 0.5, 0.75\}$. Here, $UB_{\max}$ is computed by constructing an initial solution with only ICCVs, without maximum emissions, using the ICCV insertion heuristic described in Section 4.1.2.

Instances In total, there are 81 instances used in this thesis. The instances are subdivided in clustered (denoted with C), random (R) and partially random and partially clustered (RC) customer sets. Furthermore, the instances are divided in short (denoted by a 1) and long time windows (2). The instances are of small sizes (with $n = 5, 10, 15$), medium sizes ($n = 25, 30$) and large sizes ($n = 100$). The small and large sizes are obtained from Goeke (2019). The medium size instances are constructed by taking the first 25 and 30 customers from the large instances while keeping all recharging stations. For all instances, there are three possible emission upper bounds, bringing the total instances to 243. The instances are identified by codes that provide this information about their customer spread, scheduling horizon, customer size, and emission bounds. Let's consider the code C102C15_0.25. Here, "C" indicates that it is a clustered instance. The "1" represents short scheduling, Then, "02" is the instance number. Subsequently "C15" denotes 15 customers, and the emission upper bound is calculated using $\alpha = 0.25$.

6.2 Model parameters

This subsection will explain the model parameters chosen. In Section 6.2.1 we explain the parameter tuning done to obtain the optimal λ used in the initial solution. Next, in Section 6.2.2 we provide the parameters used in the ALNS. Finally, in Section 6.2.3 the termination criteria are given.

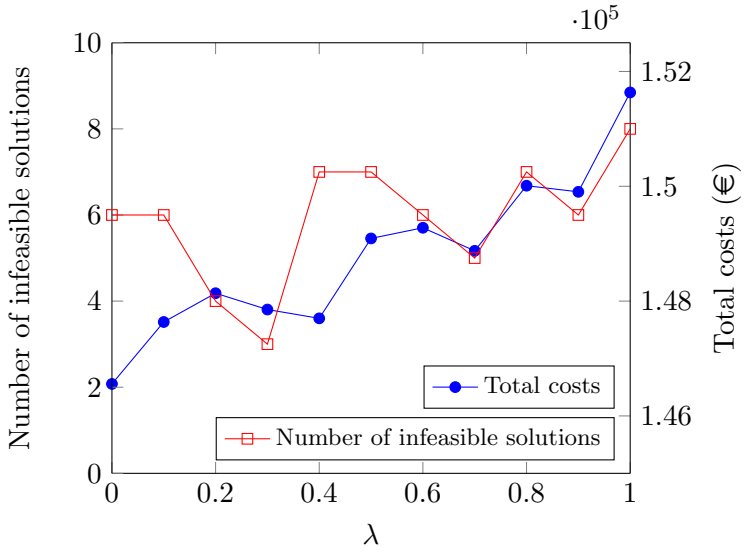

Figure 1: Graph plotting the total costs for all instances after performing the ILS, and the number of ILS solutions violating the emission constraint, for a range of values of λ

6.2.1 Lambda

To optimize the λ hyperparameter, the iterated local search (ILS) algorithm is executed on the full set of 243 problems (comprising 81 instances for each of the 3 specified α values). Parameter λ, defined on the interval [0, 1], is incremented in steps of 0.1. Figure 1 illustrates the results of the ILS, specifically the number of infeasible solutions violating the emission constraint, as well as the sum of the objective values of these solutions, for different lambda values. During lambda selection, the primary criterion is the number of feasible solutions, while total costs serve as a secondary criterion. However, total costs do not account for penalties incurred by solutions exceeding emission limits and are significantly influenced by the largest instances.

Based on the findings, a λ value of 0.3 is chosen since it yields the fewest infeasible solutions (3). Furthermore, in terms of objective value, only three other λ values perform better than 0.3. It is worth noting that relying on the initial solutions provided by the Sequential Iterated Heuristic (SIH), before the ILS application, is inadequate for lambda selection. These initial solutions exhibit a higher number of infeasible solutions, ranging from 79 (for $\lambda = 1.0$) to 51 (for $\lambda = 0.1$). Additionally, the λ value that performs best before perturbation does not necessarily perform best after perturbation, indicating that these values are unreliable predictors of the number of feasible solutions after perturbation.

In the case of the Adaptive Large Neighborhood Search (ALNS), the initial solution's importance diminishes due to its extensive neighborhood search. To ensure a fair comparison, the same λ value as the one selected for the ILS ($\lambda = 0.3$) is used

6.2.2 ALNS parameters

Sigmas To stimulate neighborhoods that obtain a new best solution, σ_3 is given double the value of σ_1 and σ_2, which are given the same value. In practice $\sigma_1 = \sigma_2 = 5$, and $\sigma_3 = 10$ are used, but these exact values are arbitrary, only their relative size is important. These scores differ from Røpke and Pisinger (2006b), as the computations of the scores also differ.

Temperature parameters The starting temperature is initialized using $w = 0.05$. The cooling rate is set at $c = 0.998$. Both of these values are taken from Røpke and Pisinger (2006b).

Settings of qMin and qMax The minimum number of customers to be removed in an iteration, $qMin$, is set to 1, to ensure the algorithm can find small changes in the routes to optimize the solution, the maximum number of customers to be removed, $qMax$, is set at 40% of the customers in the instance. In case of a solution violating the emission constraint, $qMax$ is set to 100% of the customers, to increase the change per iteration in the phase seeking a feasible solution. The general maximum of 40% stems from Røpke and Pisinger (2006a), whereas the maximum in case of an infeasible solution, and the minimum are adjusted to this specific problem.

6.2.3 Termination criteria

ILS The ILS is executed for at most $\Delta = 1000$ iterations, or if no improvement in the solution has been found for 100 iterations. These two parameters are chosen as arbitrarily large numbers, as in practice the algorithm will find a local minimum faster. As the algorithm still operates in a very small computation time, we do not decrease these values. The number 100 is a safe choice for the improvement condition, as the only randomness in the model is the strategy selection, and all three strategies are deterministic, as seen in Section 4.2.

We want the model to terminate if all three strategies do not lead to an improvement, as that means the ILS is not able to provide any better solution. As the probability that any strategy is not chosen in subsequent 100 iterations is of the order 10^{-18}, we assume that this is sufficiently small and that in practice the ILS will only terminate under these conditions if the best solution possible by the ILS is found.

ALNS The ALNS will be executed for at most $\Delta = 25000$ iterations and will terminate earlier if one of the following two termination conditions is satisfied:

1. for 250 subsequent iterations, no new solution has been accepted
2. for 500 subsequent iterations, no new global minimum has been found

These criteria will ensure that the algorithm will automatically ensure a trade-off between computation time and results.

6.3 Results

In this subsection, we will provide the results of the numerical experiments. First, in Section 6.3.1 we compare the performance of the ILS and the ALNS, based on their solution quality and computational times. Next, in Section 6.3.2, the convergence to the optimal solutions is analysed for both heuristics. Then, in Section 6.3.3, we analyze the performance of the destroy and repair neighborhoods used in the ALNS. Finally, in Section 6.3.4 we will analyze visualizations of the solutions for a specific instance.

Table 2: Comparison performance ALNS vs ILS

Metric	**ILS**	**ALNS**
Instances with the best solution	8 / 243	222 / 243
Average percentage improvement from initial solution	23.84%	34.03%
Successful feasibility reparation attempts	55 / 58	58 / 58
Average running time small instances ($n = 5, 10, 15$)	0.01 s	0.08 s
Average running time medium instances ($n = 25, 30$)	0.06 s	0.57 s
Average running time large instances ($n = 100$)	1.40 s	38.75 s

6.3.1 Comparison performance ILS and ALNS

The key comparison metrics are presented in Table 2. Among the total of 243 instances, the Adaptive Large Neighborhood Search (ALNS) yields a superior solution in 222 cases (91%), while the Iterated Local Search (ILS) outperforms in 8 cases (3%), and both heuristics achieve the same objective value in 13 cases (5%).

On average, the ILS demonstrates a 23.84% improvement over the initial solution, whereas the ALNS achieves an average improvement of 34.03%.

Regarding the emission constraint, all 58 initial solutions that violated it are successfully perturbed to feasible solutions by the ALNS. However, the ILS fails to produce a feasible solution in three of these cases.

In terms of efficiency, the ILS proves to be a faster algorithm, particularly for large instances ($n = 100$). For these instances, the ALNS has an average running time of 38.75 seconds. The ILS terminates, on average, after 1.40 seconds.

Similarly, for small ($n = 5, 10, 15$) and medium ($n = 25, 30$) instances, the ILS exhibits faster performance. However, for these instances, both algorithms provide solutions almost instantaneously, as the average running time of both the ILS and the ALNS is below 1 second.

Table 3 shows the percentage improvements of the ILS and ALNS compared to the initial solutions for small instances, grouped by the emission limit and the instance customer spread type. Table 4 shows the same information for medium and large instances.

From these tables, it can be concluded that the ILS is able to find larger improvements for instances with clustered customers than for instances with random customers. For the ALNS the same is the case for instances larger than 10 customers. Both algorithms have varying performances on the mixed instances.

This does not necessarily mean that the algorithms lead to better performance for these instances, as it could also be that the initial solution is of poorer quality for the instances with random customers. To properly assess this, it is necessary to compare the results to those of exact solutions.

Another trend is that the improvement for both heuristics highly increases with an increasing customer size and looser emission limits. These findings are highlighted by Table 5.

This table also shows that the difference in performance between the ILS and the ALNS is the lowest for instances with 10 customers and loosely restricted emissions ($\alpha = 0.75$), with a percentage point difference of 5.67%. This difference is the highest for instances with 15 customers and medium emission restrictions ($\alpha = 0.5$), where the percentage point difference is

Table 3: Average improvement % compared to the initial solution, for small instances

	$n = 5$		$n = 10$		$n = 15$	
Instance type	**ILS**	**ALNS**	**ILS**	**ALNS**	**ILS**	**ALNS**
Clustered						
$\alpha = 0.25$	10.80%	13.64%	16.27%	23.54%	19.65%	31.37%
$\alpha = 0.50$	9.28%	16.68%	11.05%	16.20%	15.08%	35.98%
$\alpha = 0.75$	13.02%	18.51%	9.03%	13.18%	27.76%	37.28%
Average	11.03%	16.28%	12.12%	17.64%	20.83%	34.88%
Random						
$\alpha = 0.25$	4.22%	15.83%	9.99%	20.84%	8.37%	18.88%
$\alpha = 0.50$	12.25%	16.01%	8.74%	18.22%	20.70%	28.01%
$\alpha = 0.75$	9.93%	21.96%	9.74%	15.60%	25.15%	33.98%
Average	8.80%	17.93%	9.49%	18.22%	18.07%	26.96%
Random/Clustered						
$\alpha = 0.25$	7.24%	17.59%	9.70%	17.62%	5.51%	21.92%
$\alpha = 0.50$	3.54%	14.10%	11.17%	21.22%	15.76%	26.70%
$\alpha = 0.75$	12.92%	16.99%	18.16%	25.16%	24.98%	30.09%
Average	7.90%	16.23%	13.01%	21.33%	15.41%	26.24%
Overall Average	9.24%	16.81%	11.54%	19.07%	18.11%	29.36%

Table 4: Average improvement % compared to the initial solution, for medium and large instances

	$n = 25$		$n = 30$		$n = 100$	
Instance type	**ILS**	**ALNS**	**ILS**	**ALNS**	**ILS**	**ALNS**
Clustered						
$\alpha = 0.25$	32.68%	45.07%	35.32%	47.57%	42.22%	52.71%
$\alpha = 0.5$	38.41%	53.77%	31.55%	45.58%	49.28%	58.31%
$\alpha = 0.75$	43.85%	53.02%	38.82%	45.78%	43.53%	59.78%
Average	38.31%	50.62%	35.23%	46.31%	45.01%	56.94%
Random						
$\alpha = 0.25$	22.02%	30.38%	15.80%	23.51%	27.73%	34.00%
$\alpha = 0.5$	21.19%	27.04%	19.80%	24.76%	38.10%	44.31%
$\alpha = 0.75$	25.39%	33.95%	23.24%	33.58%	35.68%	42.31%
Average	22.87%	30.46%	19.61%	27.29%	33.84%	40.21%
Random/Clustered						
$\alpha = 0.25$	31.14%	44.32%	27.65%	44.43%	30.70%	38.17%
$\alpha = 0.5$	34.59%	51.35%	29.92%	48.60%	32.59%	46.39%
$\alpha = 0.75$	33.58%	53.36%	35.88%	53.92%	37.94%	48.07%
Average	33.10%	49.68%	31.15%	48.98%	33.74%	44.21%
Overall average	31.43%	43.58%	28.66%	40.86%	37.53%	47.12%

Table 5: Average improvement % compared to the initial solution, for each emission limit and customer size

	$\alpha = 0.25$		$\alpha = 0.50$		$\alpha = 0.75$	
Instance size	**ILS**	**ALNS**	**ILS**	**ALNS**	**ILS**	**ALNS**
$n = 5$	7.42%	15.69%	8.36%	15.60%	11.96%	19.16%
$n = 10$	11.99%	20.67%	10.32%	18.55%	12.31%	17.98%
$n = 15$	11.18%	24.06%	17.18%	30.23%	25.96%	33.78%
$n = 25$	28.61%	39.92%	31.40%	44.05%	34.27%	46.78%
$n = 30$	26.25%	38.50%	27.09%	39.65%	32.64%	44.43%
$n = 100$	33.55%	41.63%	39.99%	49.67%	39.05%	50.06%

13.05%. In general, for larger instances, the gap tends to increase, with an interesting exemption at $n = 100$. Furthermore, for smaller instances, the gap tends to be higher in the case of tighter emission restrictions, while for the largest instances, a reverse effect can be seen.

Detailed results for each individual instance can be found in Appendix B.

6.3.2 Convergence to optimal solution in ILS and ALNS

Figure 2 illustrates a plot of the objective value for solutions obtained at each iteration of the Adaptive Large Neighborhood Search (ALNS) and Iterated Local Search (ILS) algorithms.

The graph highlights the substantial disparity in convergence speed between the two algorithms. The ILS achieves termination in under 200 iterations, while the ALNS requires nearly 1300 iterations to reach termination. During the initial iterations, the ILS consistently attains superior solutions compared to the slower ALNS. It is only around the 600th iteration that the ALNS approaches the ILS optimum and finally surpasses the ILS solution around the 800th iteration.

This plot provides insight into why the ILS is faster but less effective in discovering optimal solutions. The ILS rapidly converges to a local minimum, while the ALNS, due to its exploration of a larger neighborhood, proceeds at a slower pace. Although the ALNS process is slower, it ultimately leads to superior solutions.

The scatter plot of the solutions reveals additional interesting observations. Only feasible solutions are depicted, resulting in a relatively sparse representation with solutions shown in blue for acceptance and in red for rejection. The broad range of objective values displayed reflects the extensive exploration undertaken by the ALNS across a wide neighborhood.

By examining the accepted and rejected solutions, the solution acceptance criteria become apparent. Notably, all solutions that improve upon the current solution are accepted, and slightly inferior solutions may also be accepted, with random effects involved. The influence of temperature can be observed, as earlier iterations accept significantly poorer solutions compared to the current solution, while the acceptance of such solutions diminishes in later iterations.

After an initial exploration around the minimum discovered near the 800th iteration, the current solution diverges from this minimum and fails to find solutions near the optimum thereafter. This could indicate that there is room for improvement in the final part of the process, by focusing on a local search starting from the last found optimum.

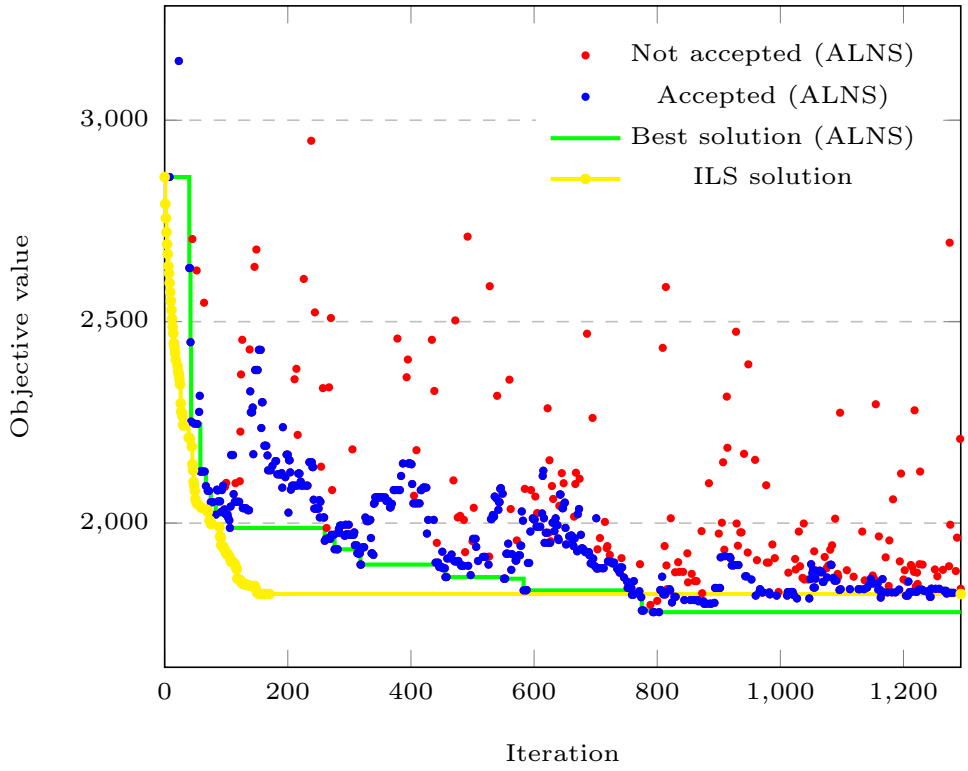

Figure 2: Plot of the objective values of the current and optimal solutions at each iteration for the ALNS and ILS for instance R101_0.25

Table 6: Overview of average values of π after termination

Neighborhood	π			
Mean value π^{noise}	0.43			
For $j =$	1	2	3	4
Mean value π_j^-	0.19	0.27	0.27	0.27
Mean value π_j^+	0.12	0.13	0.39	0.36

6.3.3 Analysis of π-values

Let us revisit the neighborhoods described in Section 5.4. An overview of the average π-values after the termination of the Adaptive Large Neighborhood Search (ALNS) is presented in Table 6.

Based on the results, it can be seen that repair neighborhoods without noise outperform those with noise, as $\pi^{\text{noise}} < 0.5$. However, the difference in performance is not large, suggesting that both the repair neighborhoods with and without noise lead to successful solutions. Furthermore, it can be concluded that destroy neighborhoods 2 (random removal of q customers), 3 (removal of worst q customers), and 4 (removal of worst q customers with noise) demonstrate comparable performance. On the other hand, the route removal neighborhood exhibits notably worse performance. Lastly, repair neighborhoods 3 (greedy insertion) and 4 (regret-2 insertion) vastly outperform repair neighborhoods 1 (electrical route insertion) and 2 (conventional route insertion).

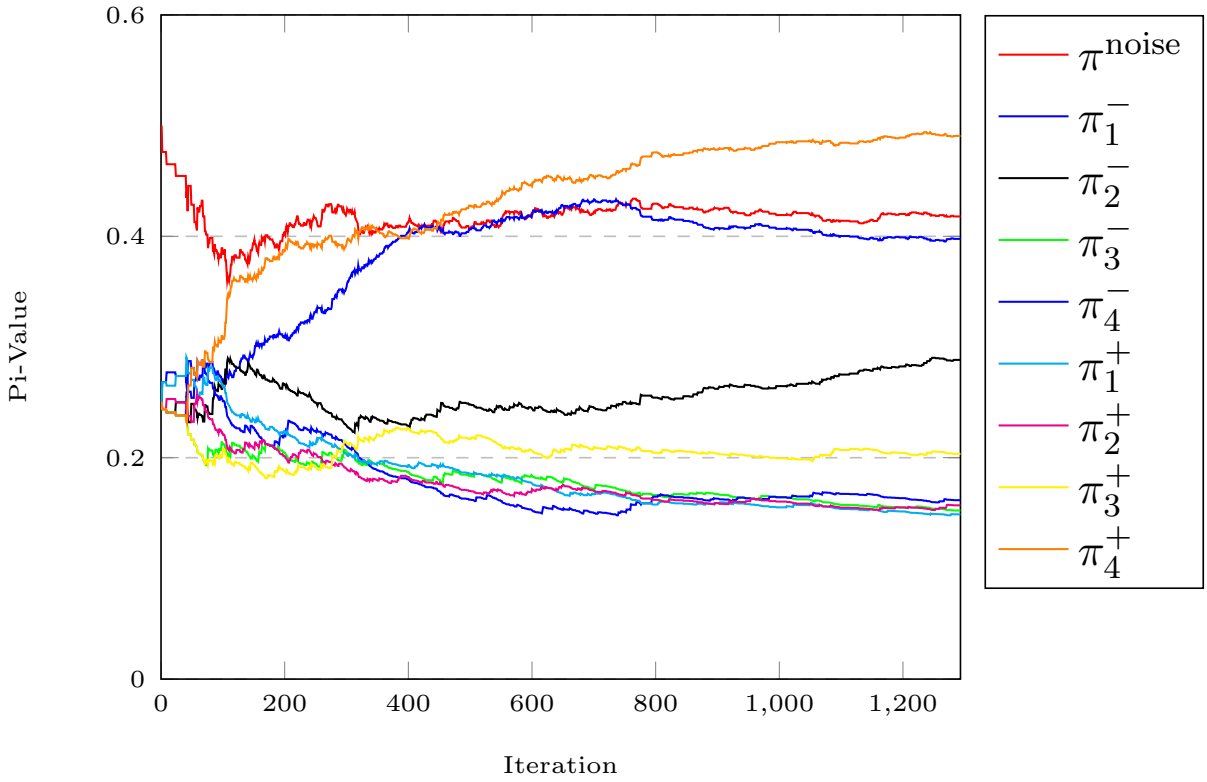

Figure 3: Plot of the pi-values for each neighborhood at each iteration for the ALNS for instance R101_0.25

Figure 3 presents the progression of the pi values for each iteration of instance R101_0.25.

Destroy neighborhood 1 (route removal) performs exceptionally well for this instance, despite its overall poorer average performance across all instances. Among the repair neighborhoods, regret-2 insertion stands out as it achieves strong performance, consistent with its average performance across all instances. The noise score demonstrates comparable performance to the average score of this π-value.

The π-values exhibit a slower initial movement due to the initialization of all scores at $10 \cdot (\sigma_1 + \sigma_2 + \sigma_3)$. However, following this initial sluggish phase, most π-values rapidly converge to their final values and experience minimal change beyond approximately the 600th iteration.

6.3.4 Solution visualisations

In this subsection, we will analyze the solutions provided by the algorithms for a specific instance (RC108C15_0.25) to give an illustration of these solutions. For this instance, the initial solution is unable to generate a feasible solution. This initial solution, which violates the emission constraint, is visualized in Figure 4. The ILS is able to construct a feasible solution from this solution, visualized in Figure 5. Due to the penalty function for the excess emissions, the ILS effectively moves customers from conventional routes to electric routes, leading to a solution with permitted emissions. This new solution does lead to higher costs than the original infeasible solution. It can be seen that the ALNS solution, in Figure 6 has changed much more compared to the initial solution. In this case, all customers served by conventional vehicles in the initial solution have been moved to electric routes. Furthermore, the large neighborhood search

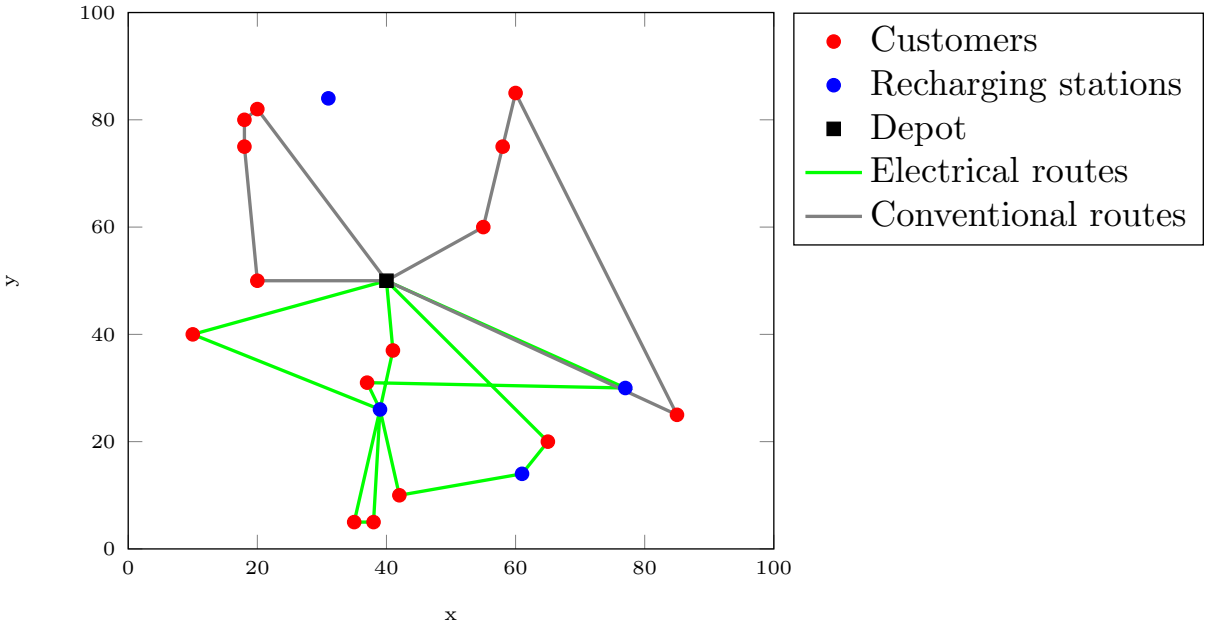

Figure 4: The initial solution for instance RC108C15_0.25

has led to a much more efficient solution, using only three vehicles. This example illustrates the capacities of both algorithms to deal with infeasible solutions and accentuates the larger neighborhood search used in the ALNS.

7 Conclusion

This thesis has covered the Green Mixed Fleet Vehicle Routing Problem with Partial Recharging and Time Windows. In this version of the vehicle routing problem, both electrical and conventional vehicles are used to serve a set of customers. The total travel costs are minimized while adhering to a limit on the CO_2 emissions and respecting the time windows of all customers. Electric vehicles have a limited battery capacity and can recharge during their routes.

Both an iterated local search heuristic, and an adaptive large neighborhood search heuristic have been implemented to obtain solutions to instances of this problem. Both heuristics use the same initial solution as a starting point of their search. Subsequently, the results of these heuristics have been compared to each other on solution quality and computational speed.

In conclusion, the ALNS outperforms the ILS algorithm with major improvements in results. The ALNS achieves better outcomes in 91% of instances and, on average, exhibits a 10 percentage point improvement over the initial solution compared to the ILS.

Moreover, the ALNS demonstrates superior capability in obtaining feasible solutions from infeasible initial starting points. It has successfully obtained feasible solutions for all instances. In contrast, the ILS fails to produce feasible solutions in three instances, and its feasibility reparation performance worsens as the quality of the initial solutions declines. This has been seen in the λ parameter tuning in Section 6.2.1.

While the ALNS does have longer computational times, particularly for instances with a size of $n = 100$, the average running time remains below one minute. Therefore, the extended

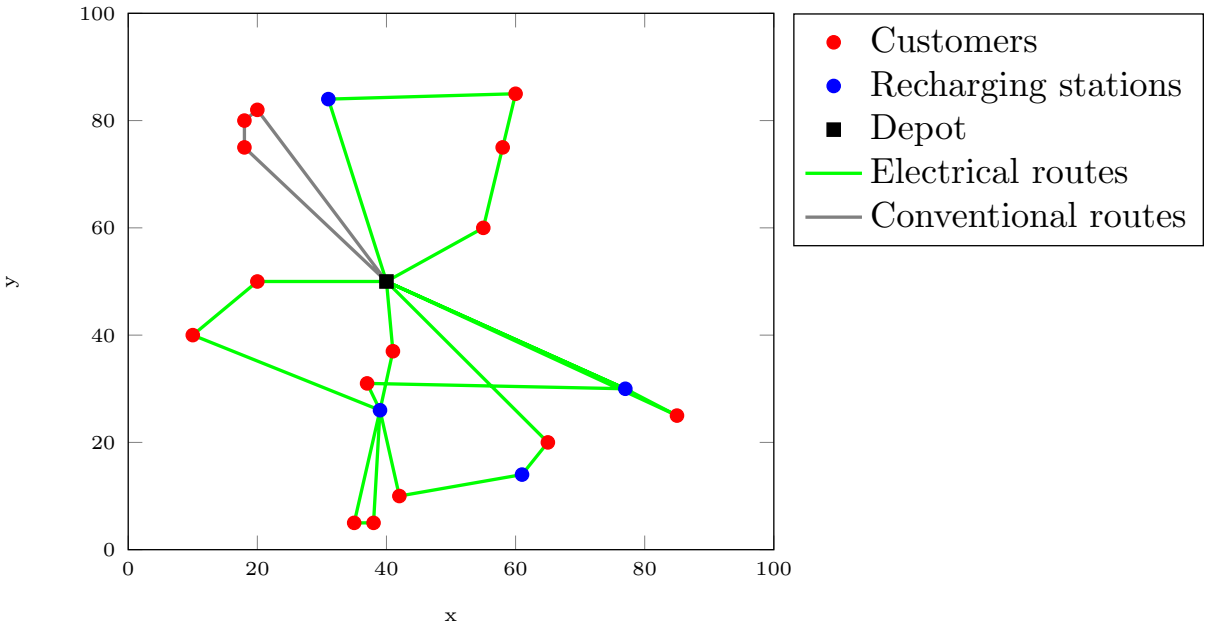

Figure 5: The ILS solution for instance RC108C15_0.25

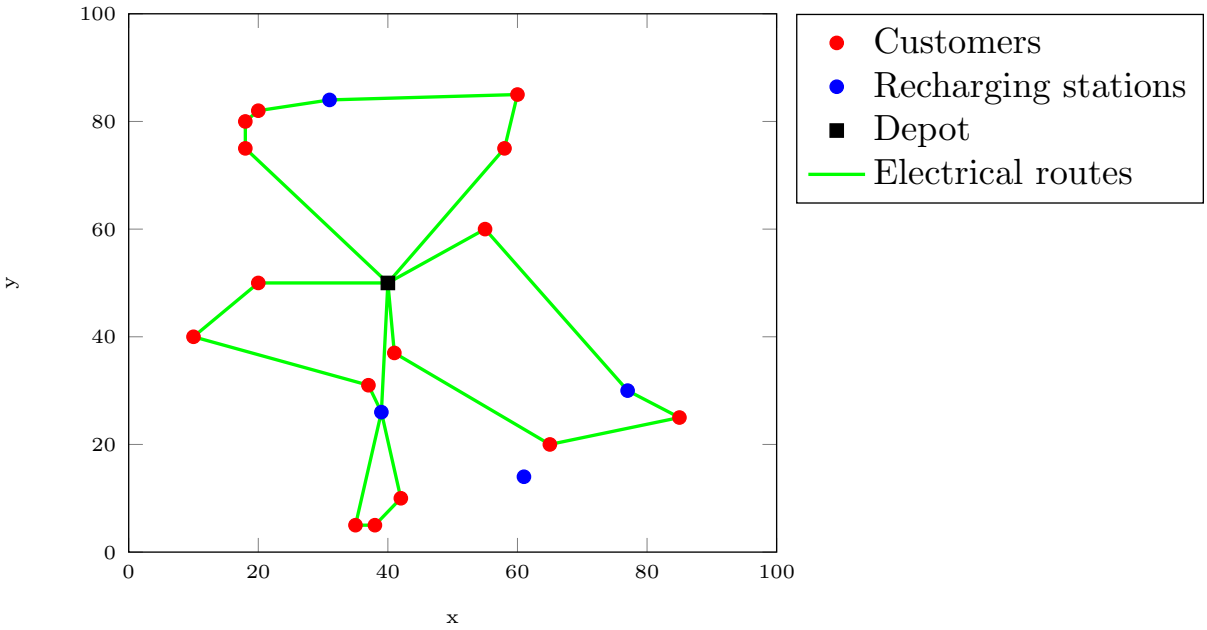

Figure 6: The ALNS solution for instance RC108C15_0.25

running time is only a minor drawback for instances up to this size. However, in practice, there are scenarios where much larger customer sets exist. Because of exponentially increasing computation times, these running times could become infeasible for practical implementation in these cases. Then, the ALNS' efficiency needs to be reevaluated.

Examining the neighborhood success scores obtained during the ALNS process, it is clear that the greedy and regret-2 insertion heuristics excel in generating high-quality feasible solutions from partial solutions. Furthermore, the relatively similar scores for the destroy neighborhoods suggest that all utilized destroy neighborhoods contribute value to the search process. Lastly, the score for noise introduction, which does not differ much from 0.5, indicates that employing repair neighborhoods both with and without noise yields favorable performance.

Future research could focus on conducting additional analysis to identify the optimal hyperparameters for the ALNS, aiming to improve its performance. Moreover, enhancing the termination conditions for the ALNS could lead to performance improvement. A potential approach could involve selecting the current global minimum as a starting point after a period of not reaching a new global minimum. Another possibility to achieve this goal is executing the ILS using the optimal solution found by the ALNS. Such a method has been implemented by Goeke and Schneider (2015). This strategy could particularly be interesting, considering that the ILS outperformed the ALNS in 8 instances. It suggests that there is potential for improvement by placing more emphasis on discovering solutions close to the current best solution, especially in the final stage of the process.

Additionally, expanding the range of destroy and repair neighborhoods is crucial to enhance the ALNS's effectiveness. The current selection of neighborhoods is relatively limited, and incorporating a broader variety would likely yield better results.

Furthermore, the ALNS could benefit from the implementation of a more advanced scoring mechanism to compute the probabilities to select neighborhoods. Currently, there is no mechanism in place to reward original solutions. If all three scoring measures include the requirement for a solution to be original, it is likely to accelerate the ALNS and improve its results since finding original solutions is crucial for its search performance.

References

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

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.

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

Erdoğan, S. & Miller-Hooks, E. (2012). A green vehicle routing problem. *Transportation research part E: logistics and transportation review*, *48*(1), 100–114.

Fernández Gil, A., Lalla-Ruiz, E., Gómez Sánchez, M., Castro, C. et al. (2022). A review of heuristics and hybrid methods for green vehicle routing problems considering emissions. *Journal of Advanced Transportation*, *2022*.

Figliozzi, M. A. (2011). The impacts of congestion on time-definitive urban freight distribution networks CO2 emission levels: Results from a case study in Portland, Oregon. *Transportation Research Part C: Emerging Technologies*, *19*(5), 766–778.

Goeke, D. (2019). E-VRPTW instances. *Mendeley Data*. doi: 10.17632/h3mrm5dhxw.1

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

Macrina, G., Di Puglia Pugliese, L., Guerriero, F. & Laporte, G. (2019). The green mixed fleet vehicle routing problem with partial battery recharging and time windows. *Computers & Operations Research*, *101*, 183-199. doi: https://doi.org/10.1016/j.cor.2018.07.012

Pisinger, D. & Røpke, S. (2007). A general heuristic for vehicle routing problems. *Computers & operations research*, *34*(8), 2403–2435.

Røpke, S. & Pisinger, D. (2006a). An adaptive large neighborhood search heuristic for the pickup and delivery problem with time windows. *Transportation science*, *40*(4), 455–472.

Røpke, S. & Pisinger, D. (2006b). A unified heuristic for a large class of vehicle routing problems with backhauls. *European Journal of Operational Research*, *171*(3), 750–775.

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

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

Zelasky, S. E. & Buonocore, J. J. (2021). The social costs of health-and climate-related on-road vehicle emissions in the continental united states from 2008 to 2017. *Environmental Research Letters*, *16*(6), 065009.

A Pseudocode algorithms

A.1 Initial solution insertion algorithms

Algorithm 5 Insertion algorithm conventional routes (IAC)

Initialize $k = 1$, the vehicle number
Initialize C^*, the set of customers already served by ICCVs, as $\emptyset$
while $C' \backslash C^* \neq \emptyset$ **and** algorithm termination condition not activated **do**
    Define $i' = \text{argmin}_{i \in C' \backslash C^*}\{l_i\}$
    Initialize new route $Z_k^C = (s, i', t)$
    while $C' \backslash (C^* \cup Z_k^C) \neq \emptyset$ **and** route termination condition not activated **do**
        Let $Z_k^C = (s, i_1, , i_2, ..., i_m, t)$ be the current route
        for each unserved customer $u \in C' \backslash (C^* \cup Z_k^C)$ **do**
            for each position $n \in \{1, ..., m, m+1\}$ in current route Z_k^C **do**
                Compute $f_1(u, n) = c^C_{i_{n-1},u} d_{i_{n-1},u} + c^C_{u,i_n} d_{u,i_n} - c^C_{i_{n-1},i_n} d_{i_{n-1},i_n}$ ▷ This computes the added cost of inserting customer u at position n. Adding u at position n means adding it between customers i_{n-1} and i_n, where $i_0 = s$ and $i_{m+1} = t$
            end for
            Compute $n_u^* = \text{argmin}_{n \in \{1,...,m,m+1\}}\{f_1(u, n)\}$, the optimal position for u.
            Compute $f_2(u, n_u^*) = c^C_{s,u} d_{s,u} + c^C_{u,t} d_{u,t} - f_1(u, n_u^*)$ ▷ This formula computes the difference in costs between serving u on this route versus serving u from the depot.
        end for
        Compute $u^* = \text{argmax}_{u \in C' \backslash (C^* \cup Z_k^C)}\{f_2(u, n_u^*)\}$
        Compute feasibility of adding u^* on route Z_k^C at $n_{u^*}^*$ by checking: load and time constraints for this route, and total emission constraints
        if all constraints verified **then**
            Add u^* to route Z_k^C at position $n_{u^*}^*$
        else
            Activate route termination condition ▷ This route will not be extended further. The last customer will not be added.
        end if
    end while
    if all constraints verified **then** ▷ Extra check needed for the case where the route only contains the initialization
        Add route Z_k^C to solution η^C
        Add customers from Z_k^C to C^*
        Update k (+1)
    else
        Activate algorithm termination condition ▷ This part will be reached when adding a route containing only the initialization is infeasible, in this case the insertion algorithm should be terminated.
    end if
end while
if there exist customers in C' unserved by η^C **then**
    Move these customers from C' to E'
end if
return: η^C and updated C' and E'

Algorithm 6 Route repairing algorithm for electrical vehicles

1: **input:** a route $Z_k^E = (s, i_1, i_2, ..., i_m, t)$ which does not satisfy the battery constraints
2: Check for all customers i_n, whether battery is empty before arrival, i.e., $z_{i_{n-1},i_n} < 0$
3: Let n' denote the lowest n for which this is the case
4: Find the closest recharging station (RS) u', by computing for which RS the added cost at position n' (between $i_{n'-1}$ and $i_{n'}$ is minimal, i.e., minimize $f_1(u, n')$ (see 11) over all $u \in R'$ holding n' constant
5: Check whether inserting u' at n' satisfies $z_{i_{n'-1},i_{u'}} \geq 0$
6: **while** feasible RS not found **do**
7: Update RS position $n' = n' - 1$
8: Find RS u' for the updated position
9: Recheck the battery constraint $z_{i_{n'-1},u'} \geq 0$ for the new RS and position
10: **end while**
11: Compute the necessary charge needed to reach t from u' using the route.
12: Compute maximum recharge available at u': $B^C - z_{i_{n'-1},u'}$
13: **if** necessary charge is higher than $B^C - z_{i_{n'-1},u'}$ **then**
14: Set $g_{u',i_{n'}} = B^C - z_{i_{n'-1},u'}$ and search for a new recharging station by going back to line 2.
15: **else**
16: Set $g_{u',i_{n'}}$ equal to the necessary charge ▷ Now all battery charge constraints hold
17: **end if**
18: Recheck all time window (TW) constraints
19: **while** TW constraints do not hold **do**
20: **if** there are unnecessary RSs on the route **then** ▷ The route with this RS deleted can satisfy battery constraints
21: Delete the first unnecessary RS on the route
22: **else**
23: Delete the first customer on the route for which TW do not hold
24: **end if**
25: Recheck all TW constraints
26: **end while**
27: **return:** the repaired route

Algorithm 7 Insertion algorithm electrical routes (IAE)

Initialize $k = 1$, the vehicle number
Initialize E^*, the set of customers already served by EVs, as $\emptyset$
while $E' \backslash E^* \neq \emptyset$ **and** algorithm termination condition not activated **do**
    Define $i' = \text{argmin}_{i \in E' \backslash E^*}\{l_i\}$
    Initialize new route $Z_k^E = \{s, i', t\}$
    while $E' \backslash (E^* \cup Z_k^E) \neq \emptyset$ **and** route termination condition not activated **do**
        Let $Z_k^E = (s, i_1, , i_2, ..., i_m, t)$ be the current route
        for each unserved customer $u \in E' \backslash (E^* \cup Z_k^E)$ **do**
            for each position $n \in \{1, ..., m, m+1\}$ in current route Z_k^E **do**
                Compute $f_1(u, n) = c^E_{i_{n-1},u} d_{i_{n-1},u} + c^E_{u,i_n} d_{u,i_n} - c^E_{i_{n-1},i_n} d_{i_{n-1},i_n}$ ▷ This computes the added cost of inserting customer u at position n. Adding u at position n means adding it between customers i_{n-1} and i_n, where $i_0 = s$ and $i_{m+1} = t$
            end for
            Compute $n_u^* = \text{argmin}_{n \in \{1,...,m,m+1\}}\{f_1(u, n)\}$, the optimal position for u.
            Compute $f_2(u, n_u^*) = c^E_{s,u} d_{s,u} + c^E_{u,t} d_{u,t} - f_1(u, n_u^*)$ ▷ This formula computes the difference in costs between serving u on this route versus serving u from the depot.
        end for
        Compute $u^* = \text{argmax}_{u \in C' \backslash (E^* \cup Z_k^E)}\{f_2(u, n_u^*)\}$
        Compute feasibility of adding u^* on route Z_k^E at $n^*_{u^*}$ by checking: load and time constraints for this route
        if both constraints verified **then**
            Add u^* to route Z_k^E at position $n^*_{u^*}$
        else
            Activate route termination condition ▷ This route will not be extended further. The last customer will not be added.
        end if
        Check battery constraints, if they are not verified, use algorithm 6 to repair the route
    end while
    if all constraints verified **then** ▷ Extra check needed for the case where the route only contains the initialization
        Add route Z_k^E to solution η^C
        Add customers from Z_k^E to E^*
        Update k (+1)
    else
        Activate algorithm termination condition ▷ This part will be reached when adding a route containing only the initialization is infeasible, in this case the insertion algorithm should be terminated.
    end if
end while
if there exist customers in E' unserved by η^E **then**
    Move these customers from E' to C'
    Create new routes for these customers by violating the emission constraint
end if
return: η^E

A.2 Local search improvement heuristics

Algorithm 8 Customer change conventional routes (CCC)

Initialize with temporary solution η^C a set of routes Z_k^C
for each route $Z_k^C \in \eta^C$ **do**
    for each customer $u \in Z_k^C$ **do**
        Compute added cost of u in its current route (vs dropping u)
        for each route $Z_l^C \in \eta^C \backslash Z_k^C$ **do**
            for each position $n \in Z_l^C$ **do**
                Compute cost of adding u at position n
                Check feasibility load and TW
            end for
            Store cheapest feasible position n to add u on route Z_l^C
        end for
        Store cheapest route to feasibly move u to
        Compare costs of the new route to the current route
    end for
    Store customer with biggest cost decrease on route Z_k^C
end for
Store customer with the biggest cost decrease among all routes
If the cost decrease is positive, move the customer to the new route

Algorithm 9 Customer change electrical routes (CCE)

Initialize with temporary solution η^C a set of routes Z_k^C
for each route $Z_k^E \in \eta^E$ **do**
    for each customer $u \in Z_k^E$ **do**
        Compute added cost of u in its current route (vs dropping u) ▷ In the case of dropping u, check if there exist unnecessary recharging stations (RSs)
        for each route $Z_l^E \in \eta^C \backslash Z_k^E$ **do**
            for each position $n \in Z_l^E$ **do**
                Compute cost of adding u at position n
                Check feasibility of load, TW and battery capacity constraints
                If battery capacity infeasible: construct a feasible route by inserting a recharging station, if possible, and compute total added costs
            end for
            Store cheapest feasible position n to add u on route Z_l^E
        end for
        Store cheapest route to feasibly move u to
        Compare costs of new route to current route
    end for
    Store customer with biggest cost decrease on route Z_k^C
end for
Store customer with biggest cost decrease among all routes
If the cost decrease is positive, move the customer to the new route

A.3 ALNS neighborhood heuristics

Algorithm 10 Greedy insertion algorithm

while There exist unserved customers **do**
    for each unserved customer **do**
        for each existing route, at each position, compute the feasibility and added costs of inserting the current customer.
        in case the current route is electric, reevaluate all recharging stations and if necessary add a new one
        keep track of the cheapest route and position to insert this customer
    end for
    insert the cheapest customer to the cheapest possible route at the cheapest position
end while

Algorithm 11 Regret-2 insertion algorithm

while There exist unserved customers **do**
    for each unserved customer **do**
        for each existing route, at each position, compute the feasibility and added costs of inserting the current customer.
        in case the current route is electric, reevaluate all recharging stations and if necessary add a new one
        keep track of the cheapest and second-cheapest route and position to insert this customer
        store the regret-2 score of this customer: the difference between the first and second-cheapest positions to insert this customer
    end for
    insert the customer with the highest regret-2 score to the cheapest possible route at the cheapest position
end while

B Results for each instance

In the following tables, the results of each individual instance are shown. For each instance, the costs of the initial solution, the costs of the ILS solution, and the costs of the ALNS are shown, as well as the running times of the ILS and the ALNS.

Table 7: Results for $\alpha = 0.25$

Instance	Costs Init. (€)	Costs ILS (€)	RT ILS (s)	Costs ALNS (€)	RT ALNS (s)
n = 5					
C101C5_0.25	250.04	250.04	0.03	250.04	0.07
C103C5_0.25	185.73	165.73	0.01	165.67	0.01
C206C5_0.25	265.97	249.83	0.00	236.58	0.03
C208C5_0.25	244.20*	179.86	0.00	164.34	0.02
R104C5_0.25	168.40	139.95	0.01	136.69	0.01
R105C5_0.25	202.98*	202.63*	0.00	156.08	0.01
R202C5_0.25	143.39	143.39	0.00	128.88	0.01
R203C5_0.25	201.77	201.77	0.00	179.06	0.05
RC105C5_0.25	256.97	239.46	0.00	238.05	0.04
RC108C5_0.25	294.52*	258.75	0.01	253.93	0.03
RC204C5_0.25	247.19	243.63	0.00	185.44	0.04
RC208C5_0.25	221.72	202.76	0.00	167.98	0.02
n = 10					
C101C10_0.25	480.18*	412.51	0.02	392.10	0.11
C104C10_0.25	343.16	322.28	0.02	273.93	0.08
C202C10_0.25	326.21	243.20	0.02	243.20	0.05
C205C10_0.25	327.06	263.42	0.03	228.28	0.04
R102C10_0.25	326.85	269.18	0.03	249.19	0.05
R103C10_0.25	225.22	217.05	0.02	203.94	0.04
R201C10_0.25	267.77	217.68	0.02	217.68	0.04
R203C10_0.25	339.46*	336.00*	0.01	232.68	0.13
RC102C10_0.25	515.73*	470.51*	0.02	456.23	0.07
RC108C10_0.25	483.01*	412.53	0.02	388.79	0.05
RC201C10_0.25	395.26	348.08	0.01	310.06	0.03
RC205C10_0.25	427.22	374.85	0.02	350.75	0.05
n = 15					
C103C15_0.25	563.37	394.73	0.03	399.82	0.16
C106C15_0.25	484.16	387.98	0.02	271.09	0.19
C202C15_0.25	461.29	440.55	0.03	369.56	0.14
C208C15_0.25	445.75	337.33	0.03	300.55	0.11
R102C15_0.25	522.13*	425.89	0.06	413.97	0.15
R105C15_0.25	405.03	362.22	0.03	336.08	0.06
R202C15_0.25	444.80*	454.55	0.03	358.00	0.24
R209C15_0.25	378.18*	352.94	0.01	309.15	0.23
RC103C15_0.25	405.41	400.18	0.03	393.88	0.05
RC108C15_0.25	553.51*	594.93	0.03	389.13	0.08
RC202C15_0.25	545.60	444.43	0.05	405.44	0.17
RC204C15_0.25	440.67*	397.95	0.01	310.81	0.21
n = 25					
C101C25_0.25	594.22	287.14	0.07	266.56	0.34
C102C25_0.25	477.05	296.52	0.05	251.17	0.23
C103C25_0.25	399.25	330.44	0.05	220.38	0.34
C104C25_0.25	250.52	220.23	0.07	200.16	0.26
C105C25_0.25	566.06	313.99	0.08	238.02	0.25
R101C25_0.25	803.69	648.21	0.09	595.60	0.20
R102C25_0.25	715.35	584.35	0.07	517.90	0.66
R103C25_0.25	741.70*	501.08	0.11	457.57	0.38
R104C25_0.25	559.40*	451.77	0.07	404.83	0.84
R105C25_0.25	774.01	613.33	0.07	522.85	0.28
RC101C25_0.25	1003.58	607.84	0.05	552.78	0.24
RC102C25_0.25	819.16	544.97	0.07	393.98	0.61
RC103C25_0.25	646.05*	407.71	0.09	326.39	0.37
RC104C25_0.25	571.37*	402.34	0.05	320.82	0.26
RC105C25_0.25	708.23	592.61	0.05	485.50	0.39
n = 30					
C101C30_0.25	636.28	341.11	0.06	259.48	0.30
C102C30_0.25	567.47	351.62	0.06	250.40	0.58
C103C30_0.25	565.10	381.60	0.06	246.04	0.44
C104C30_0.25	288.97	245.36	0.09	254.45	0.80
C105C30_0.25	577.33	319.89	0.08	263.64	0.30
R101C30_0.25	796.04	710.45	0.09	685.58	0.63
R102C30_0.25	784.04	614.71	0.09	639.62	0.24
R103C30_0.25	717.24	619.31	0.08	478.35	1.49
R104C30_0.25	640.70	497.61	0.12	429.60	1.17
R105C30_0.25	731.94	654.03	0.07	592.87	0.28
RC101C30_0.25	1306.61*	923.31	0.11	728.11	0.43
RC102C30_0.25	987.30*	725.10	0.08	538.47	0.95
RC103C30_0.25	967.38*	614.27	0.15	514.45	0.52
RC104C30_0.25	740.56*	556.69	0.08	393.03	0.77
RC105C30_0.25	981.98*	775.57	0.09	602.10	0.75
n = 100					
C101_0.25	2857.65	1527.06	2.36	1042.12	57.09
C102_0.25	2823.62	1265.52	1.91	1092.93	74.08
C103_0.25	2201.71	1454.70	1.87	1062.73	43.42
C104_0.25	1830.87*	1165.35	2.14	1282.61	14.06
C105_0.25	2535.34	1544.14	1.64	1088.57	67.77
R101_0.25	2858.86*	1823.64	1.61	1779.37	18.93
R102_0.25	2433.93*	1686.48	1.86	1508.94	75.10
R103_0.25	1969.11*	1388.95	2.26	1486.69	20.49
R104_0.25	1609.13*	1531.34	0.92	1114.45	115.46
R105_0.25	2462.45*	1540.86	2.25	1502.42	37.83
RC101_0.25	3369.67*	1958.01	1.59	2007.89	17.73
RC102_0.25	2686.25*	1864.62	1.69	1566.85	59.45
RC103_0.25	2574.74*	1656.89	2.00	1426.83	36.33
RC104_0.25	1744.40*	1529.02	1.39	1255.64	173.03
RC105_0.25	2764.39*	1851.51	2.19	1764.99	12.01

Note: * denotes that a solution violates the emission constraint

Table 8: Results for $\alpha = 0.5$

Instance	Costs Init. (€)	Costs ILS (€)	RT ILS (s)	Costs ALNS (€)	RT ALNS (s)
n = 5					
C101C5_0.5	249.93	249.93	0.02	234.82	0.07
C103C5_0.5	185.73	165.73	0.01	165.67	0.02
C206C5_0.5	267.99	267.99	0.01	221.98	0.03
C208C5_0.5	244.20*	179.86	0.01	164.34	0.03
R104C5_0.5	166.50	136.45	0.00	136.45	0.02
R105C5_0.5	197.19	156.08	0.01	156.08	0.02
R202C5_0.5	143.39	128.88	0.00	128.78	0.01
R203C5_0.5	210.58	210.58	0.00	179.06	0.02
RC105C5_0.5	255.20	244.57	0.00	235.67	0.02
RC108C5_0.5	249.87	249.87	0.00	249.87	0.01
RC204C5_0.5	247.19	243.63	0.00	176.00	0.03
RC208C5_0.5	221.72	202.76	0.01	177.47	0.17
n = 10					
C101C10_0.5	446.84*	405.44	0.01	376.90	0.07
C104C10_0.5	370.80	298.59	0.01	271.20	0.09
C202C10_0.5	312.90	264.47	0.02	243.15	0.12
C205C10_0.5	226.02	226.02	0.01	226.02	0.04
R102C10_0.5	304.56	264.74	0.01	244.75	0.05
R103C10_0.5	225.22	217.05	0.01	195.26	0.08
R201C10_0.5	244.63	213.99	0.01	208.63	0.08
R203C10_0.5	311.14	293.29	0.01	232.68	0.16
RC102C10_0.5	495.98*	430.00	0.02	406.86	0.04
RC108C10_0.5	438.43*	400.63	0.02	345.33	0.03
RC201C10_0.5	377.31	320.69	0.01	315.28	0.19
RC205C10_0.5	467.06	430.95	0.04	330.55	0.04
n = 15					
C103C15_0.5	522.41	438.07	0.05	342.00	0.14
C106C15_0.5	497.89	387.98	0.06	270.44	0.11
C202C15_0.5	494.90	415.49	0.04	367.00	0.19
C208C15_0.5	490.62	460.86	0.07	304.83	0.22
R102C15_0.5	533.49	447.04	0.02	391.90	0.15
R105C15_0.5	454.55	368.61	0.01	335.88	0.16
R202C15_0.5	469.94*	375.99	0.02	357.21	0.12
R209C15_0.5	434.93	314.46	0.03	280.93	0.19
RC103C15_0.5	454.95	400.18	0.02	388.11	0.11
RC108C15_0.5	552.49*	533.12	0.02	389.04	0.16
RC202C15_0.5	607.28	491.31	0.03	386.30	0.13
RC204C15_0.5	458.27	328.21	0.01	338.54	0.14
n = 25					
C101C25_0.5	612.69	351.55	0.04	241.42	0.37
C102C25_0.5	520.20	351.79	0.05	218.30	0.32
C103C25_0.5	456.04	257.78	0.03	218.31	0.50
C104C25_0.5	308.98	221.62	0.05	193.13	0.51
C105C25_0.5	612.69	335.15	0.07	241.42	0.21
R101C25_0.5	775.95	615.45	0.06	585.08	0.32
R102C25_0.5	661.62	547.26	0.05	501.32	0.52
R103C25_0.5	595.53*	469.50	0.06	442.20	0.30
R104C25_0.5	556.04*	434.53	0.05	404.81	0.38
R105C25_0.5	757.11	568.06	0.05	503.95	0.25
RC101C25_0.5	1054.05	684.57	0.05	489.72	0.31
RC102C25_0.5	798.79	466.29	0.07	393.58	0.28
RC103C25_0.5	901.35	379.69	0.06	330.49	0.90
RC104C25_0.5	574.91	492.68	0.04	326.96	1.47
RC105C25_0.5	770.32	584.70	0.03	415.87	0.51
n = 30					
C101C30_0.5	620.15	374.48	0.07	256.77	0.75
C102C30_0.5	410.15	305.10	0.04	244.09	0.55
C103C30_0.5	459.16	306.31	0.06	238.25	0.96
C104C30_0.5	310.95	261.30	0.06	241.83	1.35
C105C30_0.5	620.15	351.83	0.09	257.50	2.06
R101C30_0.5	797.31	680.72	0.07	678.39	0.79
R102C30_0.5	810.25	611.71	0.06	596.08	1.24
R103C30_0.5	671.98	519.00	0.05	467.46	1.79
R104C30_0.5	679.18	531.20	0.06	454.84	1.00
R105C30_0.5	699.08	591.85	0.06	566.27	1.96
RC101C30_0.5	1265.16*	863.92	0.06	703.82	0.89
RC102C30_0.5	1078.69*	720.87	0.07	528.99	0.48
RC103C30_0.5	998.82*	750.12	0.08	449.05	0.88
RC104C30_0.5	740.56*	538.88	0.05	439.64	1.53
RC105C30_0.5	1126.22*	759.27	0.05	540.92	0.53
n = 100					
C101_0.5	2697.31	1382.89	1.42	1037.91	42.01
C102_0.5	2724.44	1278.39	1.63	994.54	32.45
C103_0.5	2252.80	1165.45	1.37	1047.90	22.00
C104_0.5	2188.83	1202.73	1.44	1131.80	26.69
C105_0.5	2847.84	1387.17	1.47	1003.35	40.21
R101_0.5	2960.45*	1808.88	1.02	1632.74	46.40
R102_0.5	2566.17*	1553.35	0.96	1430.16	41.94
R103_0.5	2178.72	1345.43	1.60	1194.75	85.68
R104_0.5	1827.79	1211.18	1.24	1065.68	50.40
R105_0.5	2418.98	1447.96	1.12	1316.14	39.77
RC101_0.5	3496.65*	2894.89	0.86	1755.67	22.27
RC102_0.5	2923.32*	1833.75	1.34	1613.35	23.85
RC103_0.5	2528.95*	1660.74	1.61	1371.91	28.61
RC104_0.5	2101.77*	1422.31	1.13	1201.22	57.18
RC105_0.5	2760.72	1606.70	1.02	1414.28	54.67

Note: * denotes that a solution violates the emission constraint

Table 9: Results for $\alpha = 0.75$

Instance	Costs Init. (€)	Costs ILS (€)	RT ILS (s)	Costs ALNS (€)	RT ALNS (s)
n = 5					
C101C5_0.75	249.93	249.93	0.01	234.82	0.05
C103C5_0.75	161.26	161.26	0.00	161.26	0.02
C206C5_0.75	267.99	233.92	0.00	221.98	0.04
C208C5_0.75	334.32*	202.68	0.00	164.34	0.01
R104C5_0.75	200.28	136.45	0.00	136.45	0.02
R105C5_0.75	203.15	202.57	0.00	153.49	0.02
R202C5_0.75	154.35	154.35	0.00	128.78	0.02
R203C5_0.75	210.58	194.68	0.00	179.06	0.02
RC105C5_0.75	248.91	233.90	0.01	231.64	0.01
RC108C5_0.75	266.41	249.87	0.00	249.87	0.01
RC204C5_0.75	263.89*	184.13	0.00	176.00	0.02
RC208C5_0.75	213.99	194.27	0.00	167.98	0.03
n = 10					
C101C10_0.75	451.32	392.05	0.01	366.52	0.07
C104C10_0.75	305.82	270.16	0.01	270.16	0.05
C202C10_0.75	241.09	239.89	0.01	239.89	0.06
C205C10_0.75	288.96	257.63	0.01	226.02	0.04
R102C10_0.75	304.56	264.74	0.01	245.58	0.02
R103C10_0.75	212.27	204.10	0.01	191.33	0.05
R201C10_0.75	234.16	213.99	0.01	208.63	0.07
R203C10_0.75	301.15	260.70	0.01	234.13	0.15
RC102C10_0.75	479.14	407.55	0.01	388.01	0.02
RC108C10_0.75	473.23	400.91	0.01	345.33	0.03
RC201C10_0.75	384.46	304.33	0.01	304.33	0.10
RC205C10_0.75	485.20	380.51	0.01	321.39	0.03
n = 15					
C103C15_0.75	547.30	435.62	0.03	343.98	0.08
C106C15_0.75	458.81	333.97	0.02	266.40	0.12
C202C15_0.75	433.07	381.57	0.01	363.38	0.15
C208C15_0.75	644.49	312.30	0.02	296.85	0.15
R102C15_0.75	503.45	441.42	0.01	377.76	0.06
R105C15_0.75	505.15*	370.63	0.01	312.44	0.08
R202C15_0.75	519.77*	385.23	0.01	357.07	0.10
R209C15_0.75	461.02	296.17	0.01	269.65	0.14
RC103C15_0.75	473.50	390.76	0.01	386.94	0.06
RC108C15_0.75	507.60	368.60	0.01	366.70	0.07
RC202C15_0.75	679.10	497.98	0.01	399.42	0.24
RC204C15_0.75	458.27	328.21	0.01	306.42	0.14
n = 25					
C101C25_0.75	579.72	308.77	0.03	242.64	0.30
C102C25_0.75	462.44	256.57	0.03	213.97	0.24
C103C25_0.75	459.59	240.33	0.03	211.99	0.20
C104C25_0.75	335.40	235.25	0.05	193.13	0.35
C105C25_0.75	554.05	274.58	0.04	238.50	0.06
R101C25_0.75	841.22	627.11	0.04	582.75	0.20
R102C25_0.75	672.67	536.74	0.04	499.83	0.20
R103C25_0.75	688.50	510.78	0.04	419.05	0.35
R104C25_0.75	622.34	468.07	0.05	404.99	0.39
R105C25_0.75	825.12	572.10	0.04	500.94	0.20
RC101C25_0.75	995.33	682.74	0.06	478.75	0.11
RC102C25_0.75	747.88	506.17	0.04	384.41	0.34
RC103C25_0.75	809.03	446.40	0.09	308.53	0.22
RC104C25_0.75	584.42	409.41	0.03	305.21	0.23
RC105C25_0.75	828.24	584.70	0.03	359.06	0.13
n = 30					
C101C30_0.75	564.26	274.48	0.04	259.02	0.22
C102C30_0.75	451.30	305.10	0.03	244.09	0.49
C103C30_0.75	462.71	280.47	0.04	238.25	0.24
C104C30_0.75	377.59	272.59	0.04	250.67	0.65
C105C30_0.75	535.68	304.60	0.04	285.06	0.47
R101C30_0.75	896.68	711.70	0.05	664.40	0.96
R102C30_0.75	852.96	660.71	0.07	578.32	1.17
R103C30_0.75	694.26	568.33	0.05	477.98	0.31
R104C30_0.75	686.09	517.89	0.05	396.20	0.61
R105C30_0.75	846.22	589.37	0.05	538.12	0.91
RC101C30_0.75	1252.61	854.54	0.05	701.36	0.75
RC102C30_0.75	1066.14	695.50	0.05	517.32	0.73
RC103C30_0.75	1112.52*	655.18	0.06	441.45	0.44
RC104C30_0.75	994.67	632.70	0.06	383.06	0.38
RC105C30_0.75	1018.64	658.55	0.04	485.66	0.55
n = 100					
C101_0.75	2682.59	1403.87	1.17	1037.91	27.01
C102_0.75	2554.23	1188.10	1.11	1016.28	13.55
C103_0.75	2369.72	1874.64	1.26	993.59	7.12
C104_0.75	2070.19*	1170.76	1.46	889.48	36.20
C105_0.75	2653.50	1268.82	1.27	1000.61	27.61
R101_0.75	2946.34	1813.17	0.85	1588.58	32.05
R102_0.75	2570.54	1659.00	1.03	1467.97	8.02
R103_0.75	2140.67	1412.72	1.01	1167.32	32.95
R104_0.75	1726.62*	1226.51	1.03	1241.54	16.44
R105_0.75	2535.47	1483.08	1.00	1292.51	23.55
RC101_0.75	3630.02	2061.73	0.78	1685.57	23.21
RC102_0.75	2915.79	1892.63	1.19	1558.95	10.41
RC103_0.75	2663.98	1600.57	1.05	1305.49	24.18
RC104_0.75	2047.24	1439.98	0.87	1176.08	12.46
RC105_0.75	2760.72	1606.70	0.97	1471.06	12.06

Note: * denotes that a solution violates the emission constraint

Table 10: Overview of attached code files

File name	Description
Main.java	The main class that executes the program
Coordinate.java	The Coordinate class, providing x and y coordinates
Node.java	The Node class, extending the Coordinate class, used as the basis for all nodes
Customer.java	The Customer class, extending the Node class, used to model customers
RechargingStation.java	The RechargingStation class, extending the Node class, used to model recharging stations
Instance.java	The Instance class, contains all information of a specific instance
Route.java	The Route class, used to model finished routes and routes in progress
Clustering.java	Class that executes the clustering process
ConventionalRoutes.java	Class that executes the conventional routes insertion algorithm
ElectricRoutes.java	Class that executes the electrical routes insertion algorithm
ImprovementHeuristic.java	Class that executes the ILS improvement heuristic
AdaptiveLargeNeighbourhoodSearch.java	Class that implements the ALNS improvement heuristic
SolutionVisualisation.java	Class that can be used to visualize solutions

C Code

Table 10 describes the code files that are attached to this thesis in a zip-file.