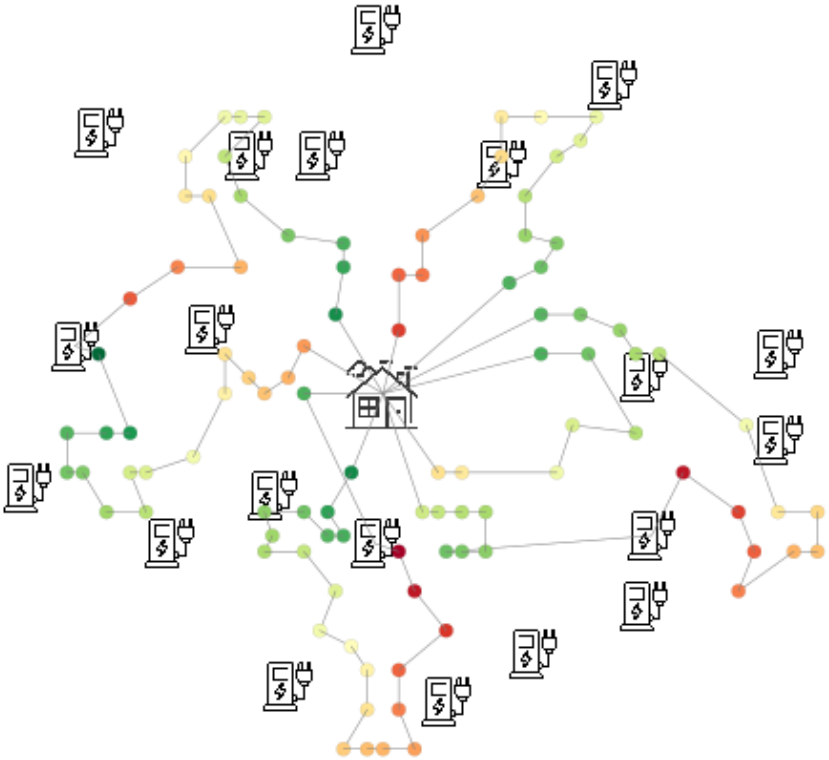

Route planning for electric vehicles using Adaptive Large Neighborhood Search

Master's Thesis in Mathematical Engineering and Computational Science

Regina Gustavsson
Ellinor Sorpola Svenningsson

DEPARTMENT OF MATHEMATICAL SCIENCES
CHALMERS UNIVERSITY OF TECHNOLOGY
Gothenburg, Sweden 2025
www.chalmers.se

Master's thesis 2025

Logistic route planning for electric vehicles using Adaptive Large Neighborhood Search

REGINA GUSTAVSSON
ELLINOR SORPOLA SVENNINGSSON

CHALMERS
UNIVERSITY OF TECHNOLOGY

Department of Mathematical Sciences
Division of Applied Mathematics and Statistics
Chalmers University of Technology
Gothenburg, Sweden 2025

Logistic route planning for electric vehicles using Adaptive Large Neighborhood Search
REGINA GUSTAVSSON
ELLINOR SORPOLA SVENNINGSSON

Supervisor: Joseph Löfving, Volvo Group Trucks Technology and the Department of Mathematical Sciences
Examiner: Ann-Brith Strömberg, the Department of Mathematical Sciences

Master's Thesis 2025
Department of Mathematical Sciences
Division of Applied Mathematics and Statistics
Chalmers University of Technology
SE-412 96 Gothenburg
Telephone +46 31 772 1000

Cover: Solution routes generated using ALNS for the data instance `c201`.

Typeset in LaTeX
Printed by Chalmers Reproservice
Gothenburg, Sweden 2025

Logistic route planning for electric vehicles using Adaptive Large Neighborhood Search

REGINA GUSTAVSSON
ELLINOR SORPOLA SVENNINGSSON
Department of Mathematical Sciences
Chalmers University of Technology

Abstract

The Electric Vehicle Routing Problem (EVRP) is an extension of the well-known Traveling Salesperson Problem (TSP) including battery limitations and multiple vehicles. Variants of this problem are formulated as mathematical optimization models consisting of objective functions and constraints, that may vary depending on the specific variant and the assumptions made.

A method that can be used to solve this problem is the Adaptive Large Neighborhood Search (ALNS) which is a so-called meta-heuristic method where the solution is destroyed and repaired iteratively to improve the solution over time. ALNS is flexible and can be designed to fit various formulations of the EVRP. In this project the aim is to investigate how our design and its implementation performs on different variants of the problem.

To evaluate the performance, ALNS was compared against a MILP solver by giving them the same initial solution and time limit. The evaluations were performed on data of different sizes as well as with different customer distributions. The best-found objective value from an article using the same data instances and a problem formulation similar to ours was used as an additional reference.

The evaluation displayed the efficiency of the ALNS, even if it was not possible to make any strong conclusions due to the quality of the reference solutions for the larger data instances. However, the algorithm and its implementation show great promise for further development, both for adapting it to other problem formulations and for improving overall computational performance.

Keywords: route planning, Electric Vehicle Routing Problem, Adaptive Large Neighborhood Search, mathematical optimization, meta-heuristic, load-dependent.

Acknowledgements

We would like to express our sincere gratitude to all those who were involved in and supported us throughout the completion of this master thesis project. Our heartfelt thanks go to Volvo Group for providing us with the opportunity to explore this subject, and to our team at Volvo for answering all of our technical questions, which greatly helped us in developing a clear understanding of the problem.

We would also like to extend a special thanks to our supervisor, Joseph Löfving, for his support during the whole project, and for the valuable insights and ideas that helped us move forward and develop our work.

Regina Gustavsson and Ellinor Sorpola Svenningson, Gothenburg, June 2025

List of Acronyms

Below is the list of acronyms that have been used throughout this thesis listed in alphabetical order:

ALNS	Adaptive Large Neighborhood Search
C-EVRPTW	Capacitated Electric Vehicle Routing Problem with Time Windows
EVRP	Electric Vehicle Routing Problem
EVRPTW	Electric Vehicle Routing Problem with Time Windows
LD-EVRPTW	Load-Dependent Electric Vehicle Routing Problem with Time Windows
MILP	Mixed-Integer Linear Programming
SA	Simulated Annealing
VRP	Vehicle Routing Problem

Contents

1
Introduction

One of the biggest challenges in today's society is climate change, and it is a well-known fact that reducing the emissions generated by fossil fuels is necessary. In 2019, about 23% of the total global emissions was from the transportation sector, with road transportation as the biggest contributor [1]. Due to increased awareness concerning the environment and emissions, there have been changes in customer interest and regulations, leading to fossil fuels being phased out and replaced by greener options as energy sources.

This project is a collaboration with Volvo Group, which, by its own account, is the largest manufacturer of electric trucks in Europe and North America [2]. Currently, this new technology is not as profitable as fossil fuel-driven trucks. This is due to factors such as long charging times, limited access to charging stations, and expensive batteries. One way to mitigate these issues would be to offer a service that uses optimization techniques to suggest efficient routes for deliveries for a fleet of electric trucks. This problem is known as the Electric Vehicle Routing Problem (EVRP) and in this work, the efficacy of the Adaptive Large Neighborhood Search algorithm (ALNS) for solving the EVRP will be investigated.

1.1 The Electric Vehicle Routing Problem

The Vehicle Routing Problem (VRP) is a generalization of the Traveling Salesperson Problem (TSP). TSP is a well-known NP-hard optimization problem where the goal is to find an optimal route for a single vehicle or salesperson to visit a number of locations exactly once, before returning to the starting position [3]. In VRP, we can instead have multiple vehicles that take different routes, where each location is visited by exactly one vehicle. In EVRP, a variant of the VRP, the goal is to find the optimal routes for electric vehicles, taking into account factors such as the need for charging stops and the resulting downtime. When solving the EVRP, the use of ALNS has shown promising results [4]. However, using electric trucks for deliveries is still relatively new and research about the EVRP is limited. Meanwhile, the VRP is a more covered topic, also in regard to using the ALNS algorithm as a solution method. Therefore, there is potential in utilizing the existing research on the VRP when applying ALNS to the EVRP.

1.2 An overview of the ALNS algorithm

ALNS was first introduced in 2006 [5], and is an algorithm where so called remove and insert operators are used to systematically improve the solution, by repeatedly destroying the current solution and then repairing it [4]. In the case of the EVRP, this typically corresponds to removing and adding requests or charging stops from or to the routes in the solution. It is also possible to remove or add entire routes. The remove and insert operators that perform the best are more likely to be used in later iterations, which is what makes the algorithm adaptive.

1.3 Previous work

The two components of this project, the ALNS and the EVRP, has over the years been well researched considering different aspects. In this section, the aim is to give a brief overview of previous research on this subject.

As a method for heuristically solving vehicle routing problems and other complex optimization problems, ALNS has shown great promise since it is a flexible algorithm that can be designed and adjusted according to the problem requirements [4]. Some aspects that can be varied are the remove and insert operators as well as the criteria for accepting a new found solution. Examples to indicate the applicability of ALNS include VRPs with pickups and deliveries [5] and VRPs that allow multiple visits to the depot to pick up more goods to be delivered [6]. Even if ALNS as a solution algorithm for different routing problems has become increasingly popular [4], there are other heuristics that also show promising results. Examples of such methods are Variable Neighborhood Search (VNS) [7], Ant Colony Optimization (ACO) [8] and Genetic Algorithms (GA) [9].

ALNS is an extension of the Large Neighborhood Search (LNS), which was introduced in 1998 to solve a version of the VRP. The difference between the methods is that while ALNS uses multiple operators with adaptive likelihoods, LNS uses the same remove and insert operator throughout all iterations [10].

The discharging of the battery can be approximated in different ways, where the simplest approach is assuming linear discharging depending on the distance or the travel time. Load-dependent discharging for the EVRP was introduced by Lin et al. in 2016 [11]. They approximated the energy consumption as a linear function depending on both the travel distances and the weight of the vehicle upon arrival. Their formula is an adaptation of the energy cost formulae used for modeling energy consumption in heavy duty vehicles. The load-dependent discharging can be combined with more complex nonlinear recharging functions [12].

1.4 Aim

The aim of this project is to evaluate the performance of ALNS for three different variants of the EVRP. The intention is to investigate if it is possible to implement an ALNS that works well for our different problem formulations without too many modifications. The intended outcomes are a collection of similar models describing

the selected versions of the EVRP, a working implementation of ALNS for these variants, and a review of how well the implementation works.

The main research questions of this project are:

- How should we design the ALNS algorithm so that it can be used on the different variants of the EVRP chosen for this project?
- How does ALNS perform on the chosen variants of the EVRP with different aspects such as time windows, partial charging or load-dependent discharging?
- How should we construct remove and insert operators that improve our results?

1.5 Limitations

To limit the scope of the project only three variants of the EVRP will be considered. Since the focus of the project has been the method, rather than the modeling, more care has been taken when constructing the algorithm. The operators, updating schemes, and other variational components in the ALNS are inspired or adapted from previous research and are based on what works well for the VRP. Since the focus is on a more general implementation of ALNS for the EVRP, neither the algorithm nor the code is optimized for one specific problem variant.

The performance of ALNS has been evaluated using one dataset, originally generated for VRPs with time windows but later modified to fit the EVRP [13]. For a more robust analysis it is preferred to also use other datasets and empirical data, to investigate its performance on real-world problems. To measure the performance of the algorithm, long evaluation times are to be desired. However, due to time restrictions, the results are limited to shorter runs of up to 10 minutes for the largest data instances.

1.6 Outline

The report begins with Chapter 2 which provides a theoretical background for the EVRP, including the three variants considered in this project, as well as an overview of ALNS. A more detailed description of ALNS and its implementation is presented in Chapter 3 and evaluated in Chapter 4 using various numerical examples. The results are further discussed in Chapter 5, together with conclusions and suggestions for future work.

2
Theory

This chapter includes some theoretical background to the problem and the different variants included in the project, together with some information regarding ALNS.

2.1 The Electric Vehicle Routing Problem

The goal of the EVRP is to plan routes for multiple electric vehicles such that they visit a set of locations exactly once before returning back to the starting position. The start position is called the depot and locations that need to be visited are called customers. For the EVRP in general, each customer has a request for a delivery of goods and may only be visited by one vehicle. During the trip, the electric vehicle requires charging, which can be done at charging stations. There is no restriction on how many times a vehicle is allowed to visit a charging station, and the same charging station may be visited by multiple vehicles.

In this project, three distinct formulations of the EVRP are considered. The first problem variant is a Capacitated EVRP with Time Windows (C-EVRPTW). In this context, capacitated means that all electric vehicles have a maximum load they can carry. A time window consist of a start and end time within which the visit to the customer needs to be scheduled. The other two problems are variants of the C-EVRPTW including partial charging and load-dependent discharging respectively. These are explained in Sections 2.1.2 and 2.1.3.

In this project, the goal is to minimize the total distance traveled by the electric vehicles, but this can be exchanged for other objectives such as minimizing the energy consumption or the total time spent on travel and service (the time it takes to unload the vehicle at the customers).

2.1.1 The Capacitated EVRP with Time Windows

To model the C-EVRPTW, additional assumptions are needed:

- all vehicles need to charge fully before leaving the depot and the charging stations,
- all vehicles travel at constant speeds and road elevations are not considered,
- the vehicles only perform deliveries,
- there is a limited number of vehicles that may be used,
- the fleet is homogeneous, that is, all vehicles are identical,

- the electric vehicles charge at a constant speed, regardless of the current state of charge, and
- the energy consumption is given by a constant factor times the distance traveled.

The problem can be formulated as a graph problem with nodes and arcs. The nodes are the customers, charging stations and the depot, and the arcs represent the shortest paths between them. The C-EVRPTW can be formulated as (2.1) which is an adaptation of the model introduced by Kucukoglu et al. [14]. Definitions of parameters and variables can be found in Table 2.1. A discrepancy between the problem formulation and the model is that the latter only allows each node to be visited at most once by any vehicle. This causes problems since each vehicle needs to start and end their route in the depot, and multiple vehicles are allowed to visit the charging stations more than once. To solve this, the depot is represented as two nodes, with indices 0 and $N+1$, and each charging station is represented by a predefined number of copies. Each charging station copy may then only be visited once.

$$\min \quad \sum_{i \in V_0'} \sum_{j \in V_{N+1}'} \sum_{k \in K} d_{ij} x_{ij}^k \tag{2.1a}$$

s.t.

$$\sum_{j \in V_{N+1}' \setminus \{i\}} \sum_{k \in K} x_{ij}^k = 1 \qquad \forall i \in V \tag{2.1b}$$

$$\sum_{j \in V_{N+1}'} \sum_{k \in K} x_{ij}^k \leq 1 \qquad \forall i \in F' \tag{2.1c}$$

$$\sum_{j \in V'} x_{0j}^k \leq 1 \qquad \forall k \in K \tag{2.1d}$$

$$\sum_{i \in V_0'} x_{ij}^k = \sum_{i \in V_{N+1}'} x_{ji}^k \qquad \begin{matrix} \forall k \in K \\ \forall j \in V' \end{matrix} \tag{2.1e}$$

$$y_j^k - y_i^k \leq Q(1 - x_{ij}^k) - h \cdot d_{ij} \cdot x_{ij}^k \qquad \begin{matrix} \forall i \in V \\ \forall j \in V_{N+1}' \\ \forall k \in K \end{matrix} \tag{2.1f}$$

$$y_j^k \leq Q - h \cdot d_{ij} \cdot x_{ij}^k \qquad \begin{matrix} \forall i \in F' \cup \{0\} \\ \forall j \in V_{N+1}' \\ \forall k \in K \end{matrix} \tag{2.1g}$$

$$\sum_{i \in V} \sum_{j \in V_{N+1}'} q_i \cdot x_{ij}^k \leq C, \qquad \forall k \in K \tag{2.1h}$$

$$p_i + (t_{ij} + s_i) \sum_{k \in K} x_{ij}^k \leq p_j + \ell_0 (1 - \sum_{k \in K} x_{ij}^k) \qquad \begin{matrix} \forall i \in V_0 \\ \forall j \in V_{N+1}' \end{matrix} \tag{2.1i}$$

$$p_i + t_{ij} \cdot x_{ij}^k + g(Q - y_i^k) \leq p_j + (\ell_0 + g \cdot Q)(1 - x_{ij}^k) \quad \forall i \in F' \quad \forall j \in V'_{N+1} \quad \forall k \in K \tag{2.1j}$$

$$x_{ij}^k \in \{0, 1\} \quad \forall i \in V'_0 \quad \forall j \in V'_{N+1} \quad \forall k \in K \tag{2.1k}$$

$$e_i \leq p_i \leq \ell_i \quad \forall i \in V'_{0,N+1} \tag{2.1l}$$

$$0 \leq y_j^k \leq Q \quad \forall j \in V'_0 \quad \forall k \in K \tag{2.1m}$$

$$y_0^k = Q \quad \forall k \in K \tag{2.1n}$$

Table 2.1: Definitions of sets, parameters and variables used in (2.1).

	Sets
V	The set of customers represented by indices, $V = \{1, 2, \ldots, N\}$.
V_0	The set of customers including the start depot node 0, $V_0 = V \cup \{0\}$.
F	The set of charging stations.
F'	The set of charging station copies used for multiple visits to charging station.
V'	The set of customers and charging station copies, $V' = V \cup F'$.
V'_0	The set of customers, charging station copies, and the start depot node 0, $V'_0 = V' \cup \{0\}$.
V'_{N+1}	The set of customers, charging station copies, and the end depot node $N+1$, $V'_{N+1} = V' \cup \{N+1\}$.
$V'_{0,N+1}$	The set of customers, charging station copies, and the start and end depot nodes 0 and $N+1$, $V'_{0,N+1} = V' \cup \{0, N+1\}$.
K	The set of available electric vehicles.
	Parameters
d_{ij}	The distance between node i and j, $i, j \in V'_{0,N+1}$.
t_{ij}	The time it takes to travel between node i and j, $i, j \in V'_{0,N+1}$.
Q	The battery capacity, must be positive.
h	The energy consumption rate per unit distance traveled, must be non-negative.
g	The recharging rate at charging stations, must be positive.
C	The weight capacity of the vehicles in weight units, must be positive.
q_i	The demand of node $i \in V'_{N+1}$ given in weight units. For a customer it must be in the interval $[0, C]$ and for charging stations and the end depot it needs to be 0.
s_i	The service time at node $i \in V_0$ given in time units, must be positive.
e_i	The earliest time to start service at node $i \in V'_{0,N+1}$ given in time units, must be in the interval $[0, \ell_i]$.
ℓ_i	The latest time to start service at node $i \in V'_{0,N+1}$ given in time units, must be $\geq e_i$.

Variables	
x_{ij}^k	Decision variable that is equal to 1 if vehicle k travels from node i to node j, and 0 otherwise, $i, j \in V'_{0,N+1}, i \neq j, k \in K$.
y_i^k	Decision variable that tracks the battery level of vehicle k upon arriving at node i, $i \in V'_0, k \in K$.
p_i	Decision variable that tracks the arrival time of a vehicle at node i, $i \in V'_{0,N+1}$.

It is important to mention that there are many limitations in the model such as the charging and discharging of the battery being estimated by linear functions, that all vehicles are identical, and that worker conditions, like breaks, are not taken into consideration. Another limitation, more specific to the way of modeling, is the use of copies to handle multiple visits to the charging stations. The number of copies needed is problem specific and could affect the solution if too few are used since it limits the number of visits to each charging station.

2.1.2 The C-EVRPTW with the partial charging policy

In the previous model, all electric vehicles need to depart from the charging stations with a full battery. Using the partial charging policy, vehicles are instead allowed to leave without charging fully. However, all vehicles still start their routes with full batteries. To model this change, one additional variable, $Y_i \in [0, Q]$, is required. It tracks the battery level when departing from node $i \in F' \cup \{0\}$. Modifying (2.1) to follow the partial charging policy results in the model (2.2) [14].

$$\min \quad \sum_{i \in V'_0} \sum_{j \in V'_{N+1}} \sum_{k \in K} d_{ij} x_{ij}^k \tag{2.2a}$$

s.t.

$$(2.1b) - (2.1f)$$

$$y_j^k - Y_i \leq Q(1 - x_{ij}^k) - h \cdot d_{ij} \cdot x_{ij}^k \quad \forall i \in F' \cup \{0\}, \ \forall j \in V'_{N+1}, \ \forall k \in K \tag{2.2b}$$

$$(2.1h) - (2.1i)$$

$$p_i + t_{ij} \cdot x_{ij}^k + g(Y_i - y_i^k) \leq p_j + (\ell_0 + g \cdot Q)(1 - x_{ij}^k) \quad \forall i \in F', \ \forall j \in V'_{N+1}, \ \forall k \in K \tag{2.2c}$$

$$(2.1k) - (2.1m)$$

$$y_i^k \leq Y_i \leq Q \quad \forall i \in F' \cup \{0\}, \ \forall k \in K \tag{2.2d}$$

$$y_0^k = Y_0 = Q \quad \forall k \in K \tag{2.2e}$$

2.1.3 The C-EVRPTW with load-dependent discharging

In the previous two models, the energy consumption rate has been assumed to be constant. If also considering the load of the vehicle it may instead be estimated by a linear function $\Phi_1 + \Phi_2 M$ where M is the total mass of the vehicle including the load. The energy consumption parameters Φ_1 and Φ_2 are calculated as

$$\Phi_1 = \frac{0.5 C_\mathrm{d} \rho A v^3}{1000\epsilon} \tag{2.3}$$

and

$$\Phi_2 = \frac{g C_\mathrm{r} v}{1000\epsilon} \tag{2.4}$$

where C_d is the coefficient for aerodynamic drag, ρ is the air density, A is the frontal surface area of the vehicle, v is the speed, ϵ is the vehicle drive train efficiency, g is the gravity force constant and C_r is the coefficient of rolling resistance [15]. Using this function when modeling the C-EVRPTW, we obtain the load-dependent C-EVRPTW, denoted LD-EVRPTW.

To model the LD-EVRPTW, the additional variable u_j^k—tracking the load of vehicle k upon departure from node j—is introduced together with the constant m, that represents the mass of an empty vehicle. The resulting model is (2.5), which builds on (2.1). The additional constraints needed for this model are adapted from [15].

$$\min \quad \sum_{i \in V_0'} \sum_{j \in V_{N+1}'} \sum_{k \in K} d_{ij} x_{ij}^k \tag{2.5a}$$

s.t.

$$(2.1\mathrm{b}) - (2.1\mathrm{e})$$

$$\begin{aligned} y_i^k - y_j^k - [\Phi_1 + \Phi_2(u_i^k + m)]t_{ij} \geq \\ (-Q - [\Phi_1 + (C+m)\Phi_2]t_{ij})(1 - x_{ij}^k) \end{aligned} \quad \begin{aligned} &\forall i \in V_0 \\ &\forall j \in V_{N+1}' \\ &i \neq j \\ &\forall k \in K \end{aligned} \tag{2.5b}$$

$$\begin{aligned} Q - y_j^k - [\Phi_1 + \Phi_2(u_i^k + m)]t_{ij} \geq \\ (-Q - [\Phi_1 + (C+m)\Phi_2]t_{ij})(1 - x_{ij}^k) \end{aligned} \quad \begin{aligned} &\forall i \in F' \\ &\forall j \in V_{N+1}' \\ &i \neq j \\ &\forall k \in K \end{aligned} \tag{2.5c}$$

$$\begin{aligned} u_i^k - u_j^k - q_j \geq (-C - q_j)(1 - x_{ij}^k) \end{aligned} \quad \begin{aligned} &\forall i \in V_0' \\ &\forall j \in V_{N+1}' \\ &i \neq j \\ &\forall k \in K \end{aligned} \tag{2.5d}$$

$$u_i^k - u_j^k - q_j \leq (C - q_j)(1 - x_{ij}^k) \qquad \begin{array}{l} \forall i \in V_0' \\ \forall j \in V_{N+1}' \\ i \neq j \\ \forall k \in K \end{array} \tag{2.5e}$$

$$(2.1\text{h}) - (2.1\text{n})$$

$$0 \leq u_j^k \leq C \qquad \begin{array}{l} \forall j \in V_0' \\ \forall k \in K \end{array} \tag{2.5f}$$

2.2 The Adaptive Large Neighborhood Search

The main idea of ALNS is to repeatedly destroy and repair a solution. This should be done in such a way that the solution improves over time. In the context of the EVRP, a solution corresponds to planned routes for the electric vehicles. As for the destroy and repair operators, they are called *remove* and *insert operators* for the EVRP since they are used to remove and insert customers and charging stations from/into the routes.

Which nodes to remove and where they should be reinserted can be decided in different ways, depending on the remove and insert operator used. In ALNS there should be multiple operators to choose from in each iteration; which one to use is decided by a so called selection operator. What makes the algorithm adaptive is that for each remove and insert operator, the probability of being selected is impacted by its performance in previous iterations.

The repaired solution is used as the starting point in the next iteration if *accepted*, which is decided by an acceptance criterion. If not accepted, the new solution is discarded and the previous one is retained. The algorithm terminates once a termination criterion has been reached.

Algorithm 1 shows an example of a typical ALNS structure, not specific to the EVRP. The algorithm is inspired by articles by Windras Mara [4] and Voigt [16] with some adjustments and clarifications. The parameters and variables used in Algorithm 1 are defined as in Table 2.2.

The algorithm utilizes Simulated Annealing (SA) to help avoid local minima [4]. This is done by accepting higher cost solutions with the probability

$$P = e^{-\frac{f(S') - f(S)}{T}}. \tag{2.6}$$

The system is initialized with a high temperature, T, which slowly decreases until the system "freezes". The idea is that the algorithm should explore in the beginning and then slowly reach a steady state [17].

Algorithm 1 Typical structure of ALNS

Require: Ω^-, Ω^+, η_s, α, T_0 and an initial solution S_0,

1: $S^* \leftarrow S_0$, $S \leftarrow S_0$, $i \leftarrow 1$, $T \leftarrow T_0$
2: Initialize w_k^-, w_ℓ^+ and calculate $p_k^- = \frac{w_k^-}{\sum_{j=1}^{|\Omega^-|} w_j^-}$ and $p_\ell^+ = \frac{w_\ell^+}{\sum_{j=1}^{|\Omega^+|} w_j^+}$ for $k = 1, 2, \ldots, |\Omega^-|$ and $\ell = 1, 2, \ldots, |\Omega^+|$
3: **while** stop condition is not met **do**
4: `Remove()`, `Insert()` $\leftarrow \theta(\Omega^-, \Omega^+)$
5: $S' \leftarrow$ `Insert`(`Remove`(S, C))
6: **if** $f(S') < f(S)$ **or** `rand`$[0, 1] < P$ **then**
7: $S \leftarrow S'$
8: **end if**
9: **if** $f(S') < f(S^*)$ **then**
10: $S^* \leftarrow S'$
11: **end if**
12: **if** $i \mod \eta_s = 0$ **then**
13: Update w_k^-, w_ℓ^+ and calculate p_k^- and p_ℓ^+ as in step 2
14: **end if**
15: $T \leftarrow \alpha T$, $i \leftarrow i + 1$
16: **end while**

Table 2.2: Names and descriptions of all parameters and variables used in Algorithm 1.

Name	Description
f	Objective function.
C	Set of all objects that may be removed from/inserted in the solution.
S	Feasible solution. For a graph problem it could consist of lists of nodes or a logical matrix describing which edges are used.
S_0	Feasible initial solution.
S'	New feasible solution.
S^*	Best solution found so far.
Ω^-	Set of remove operators.
Ω^+	Set of insert operators.
θ	Operator that selects which remove and insert operators to use from Ω^- and Ω^+, respectively.
w_k^-	Weight of remove operator, $k \in \Omega^-$, must be non-negative.
w_ℓ^+	Weight of insert operator, $\ell \in \Omega^+$, must be non-negative.
p_k^-	Probability of remove operator $k \in \Omega^-$ being selected.
p_ℓ^+	Probability of insert operator $\ell \in \Omega^+$ being selected.
η_s	Updating period for the weights.
T_0	Initial temperature used in simulated annealing, must be positive.
T	Current temperature used in simulated annealing, must be non-negative.
α	Cooling rate used in simulated annealing, $\alpha \in (0, 1)$.

3

The ALNS algorithm

In this chapter the design of ALNS is presented, which includes the different variational components in the algorithm. The remove and insert operators are described in Sections 3.2 and 3.3, the selection operator in Section 3.1, and the acceptance and termination criteria in Sections 3.5 respective 3.6. The ALNS in this report follows the structure shown in Algorithm 1.

3.1 Operator selection

Each remove and insert operator are assigned a weight. In every iteration, one of each operator type (remove and customer insert) are selected independently using Roulette Wheel Selection, with the probability of selection proportional to their respective weight. How the weights are initialized and updated is described in Section 3.4. Let Ω be a set of operators (i.e., removal or customer insert), then the probability, p_k, of selecting operator $k \in \Omega$ is determined using the formula

$$p_k = \frac{w_k}{\sum_{j=1}^{|\Omega|} w_j}, \tag{3.1}$$

where w_k is the corresponding weight.

3.2 Remove operators

The decision of which operators to implement was guided by the recommendations and rankings shown in [16]. The ranking is for different versions of the VRP and therefore some modifications are made to fit our problem. In addition, some simplifications were needed due to time limitations.

In general, each operator is given a remove proportion parameter, μ, used for calculating the target number of nodes to remove. The proportion is of the total number of node visits in all the routes excluding the depot. This means that charging stations that occur multiple times in the routes can be removed more than once, since each visit to the charging stations is considered independently. However, this is not the case for Shaw removal (see Section 3.2.6) since it is calculated apriori, that is without considering the solution. Instead, it is the proportion of the number of customers and charging stations in the problem. More nodes than the target number may be removed due to the design of some of the remove operators.

All the operators work iteratively and does not terminate until the target number of nodes has been removed. Before termination, all routes only containing the start

and end depot are emptied (meaning the depots are removed) and if a charging station is visited multiple times in a row, all except one occurrence are removed. This is because there is never a reason to go back immediately to the same charging station.

3.2.1 Random node removal

All visits to nodes in the solution routes—except for the start and end depot—are given equal probabilities to be removed. Then charging stations and customers are randomly chosen and removed. If a charging station is to be removed and it is visited in multiple routes, it is removed exactly once from a single route. This can be seen as each visit to the charging station being treated as an individual node that can be removed.

3.2.2 Random route removal

Given a solution, this operator randomly selects routes to empty with equal probability.

3.2.3 Worst costs node removal

Let $\Delta c_{i,j}$ denote the difference in the objective value when removing the node in position j from route i. The depot is ignored since it cannot be removed. With the objective function being the total traveled distance, we can deduce that $\Delta c_{i,j} \leq 0$. The first node to remove is then obtained by solving

$$\underset{i,j}{\operatorname{argmin}} \; c_{i,j} \tag{3.2}$$

which gives the route and position index of the node visit with the highest cost. The node is removed and the removal costs, $\Delta c_{i,j}$, for the modified route are recalculated. This procedure is repeated by solving (3.2), removing the node with the highest cost and then recalculating the costs for the given route.

3.2.4 Worst costs route removal

With the given objective function, the worst cost route removal corresponds to removing the routes with the longest distance traveled.

3.2.5 Shortest route removal

In this context, the shortest route is the one containing the fewest number of nodes, and the actual distance of traveling the route is not considered. The operator removes all nodes from the shortest route until it terminates.

3.2.6 Shaw removal

In Shaw removal—described in [18]—the nodes that are to be removed are determined by a relatedness factor with respect to a randomly selected node, called the *reference node*. The relatedness factor is a score of how similar the customers and charging stations are to the reference node. In this report, the similarity is based solely on distance. This means that the nodes to be removed are the customers and charging stations closest to the reference node. If a charging station is removed, it is taken out of all the routes in the solution.

3.3 Insert operators

As for the remove operators, the choice of which insert operators to implement was aided by the ranking in [16]. There are two types of insert algorithms, those that insert customers and those that insert charging stations. For this project there are four customer insert operators and one charging station insert operator. The latter is used to ensure that all battery constraints are fulfilled in the other insert operators.

In each iteration of ALNS, the list of customers removed by the remove operator is given to the chosen insert operator. If at least one customer cannot be inserted into the solution due to constraints being broken, the operator has failed to find a solution. This is bound to happen sometimes, in which case the ALNS algorithm will fall back to its last accepted solution.

In each customer insert operator, after all customers have been inserted, the routes that have not been modified are processed again to check if the time and battery constraints are fulfilled. This is because charging stations in these routes may have been removed causing the route to be infeasible.

3.3.1 Charging station insert

The general idea of the charging station insert algorithm is to find the *first battery-negative node*, select a number of charging stations, investigate their feasibility upon inserting them into the routes and insert the "best" one. The first battery-negative node is defined as the first node in the route that the vehicle cannot reach due to insufficient battery.

The possible insertion actions are identified during a backwards traversal of the route, starting from the first battery-negative node, see Algorithm 2. At each step, up to three charging stations are considered: one closest to the arc between two consecutive nodes in the route, and one closest to each of the two nodes forming the arc (denoted n_1 and n_2). Here, n_1 is the node closer (in the route) to the first negative node. Duplicate charging stations are removed if they are considered for the same position, which at each step is the one between n_1 and n_2. The number of steps to traverse back is denoted as k_{cs}.

For each remaining candidate, it is only recorded if the vehicle has enough battery to reach it. The insertion actions that could cause loops (such as consecutively visiting the same charging station or going back and forth between the same charging stations without intermediate customer visits) are discarded. The latter is for

simplicity not included in Algorithm 2.

Algorithm 2 Finding charging station insertions

```
ranking ← 1
current_node ← the first battery-negative node
possible_insertions ← [ ]

for i in 1:k_cs do
    previous_node ← node before current_node in route

    cs1 ← Find the charging station closest to current_node
    cs2 ← Find the charging station closest to the arc between
          previous_node and current_node
    cs3 ← Find the charging station closest to previous_node
    cs_list ← [cs1, cs2, cs3]
    Remove duplicates from cs_list

    for cs in cs_list do
        if cs ≠ previous_node AND cs ≠ current_node AND there
           is enough battery to reach cs then
            ind_in_route ← index of current_node in route
            append(possible_insertions, (cs, ind_in_route,
                ranking))
            ranking ← ranking + 1
        end if
    end for
    current_node ← previous_node
end for

return possible_insertions
```

Each recorded insertion is assigned an incremental ranking starting from one, with lower values indicating earlier discovery and a generally closer proximity to the first negative node. The ranking, $r_{i,j}$ is used to assign a score, $s_{i,j}$, that is calculated using the formula

$$s_{i,j} = \gamma_1 r_{i,j} + \gamma_2 \frac{\Delta c_{i,j}}{M_\mathrm{d}} + \gamma_3 f_{i,j} \tag{3.3}$$

where

- $\Delta c_{i,j}$ is the change in objective value after the insertion,
- M_d is the largest distance between any two nodes in the problem, and
- $f_{i,j}$ is a binary variable that has the value one if the action of inserting charging station i in the jth position in the route ensures enough battery to reach the first negative node.

The parameters γ_1, γ_2 and γ_3 are the corresponding weights for the ranking, objective difference and feasibility, which decide the impact of each component in the score.

The insertion action with the lowest score is performed and kept if the time window constraints are fulfilled. Otherwise, the insertion with the second lowest score is considered. This continues until a charging station is inserted. If the battery constraints are fulfilled the algorithm terminates, otherwise the new first negative node is found, and the procedure is repeated. If none of the insertions are feasible with regards to the time constraints, the algorithm has failed, and the route cannot be made feasible with regards to the battery constraints. The algorithm is clarified in Algorithm 3.

Algorithm 3 Charging station insert algorithm

Require: A battery infeasible route

while the route is infeasible with regards to battery constraints **do**
    Find the first negative node.
    `insertions` ← Find all charging station insertions using Algorithm 2
    Calculate `scores` for each insertion in `insertions` using (3.3).
    Sort `insertions` by their corresponding score in ascending order

    for `insertion` in `insertions` **do**
        if the insertion is feasible with regards to the time constraints **then**
            Insert charging station in route
            `break`
        end if
    end for

    if No charging station has been inserted **then**
        Algorithm has failed
        `break`
    end if
end while

Greedy insert

The basic greedy algorithm for the VRP is described in [5], beginning with the calculation of insertion costs for each customer in each route and position, ignoring those not feasible due to time and weight constraints. The customer with the lowest insertion cost is inserted in its best route and position. The costs are then recalculated for the modified route and the new customer with the lowest insertion cost is inserted. This is repeated until all customers have been inserted. Routes must satisfy the time and weight constraints after customer insertions, otherwise the node may not be inserted there.

Since the EVRP has battery constraints not present in the VRP, only checking time and weight constraints is not sufficient. However, simply ignoring insertions that lead to battery infeasibility (similar to what is done to time and weight) is not suitable, since such infeasibilities can be remedied through the insertion of charging station stops. Therefore, modifications were made to allow for multiple insertion

tries if the battery constraints are not fulfilled. In these cases, the charging station insert operator described in Section 3.3.1 is used to investigate if one or more charging stations can be added to make the solution feasible with regards to the battery constraints. If the charging station insert operator was unsuccessful, the customer is not inserted and is prohibited from being so in any position in the route in future iterations. However, if the battery constraints are fulfilled, or it can be remedied with charging station insertions, the customers and necessary charging stations are inserted. The algorithm continues by recalculating the costs and choosing the next customer to insert.

The greedy insert operator is used to generate an initial solution by using it on empty routes with the goal to insert all customers.

3.3.2 Random insert

The random insert algorithm works by randomly choosing a customer from the customer list and inserting it in the route and position that increases the objective value the least. If it cannot be inserted due to time, weight or battery constraints it is inserted in its second best position. This is repeated until a successful insertion has been found but it is limited to k_{random} attempts, otherwise the insert algorithm failed.

If the battery constraint is not fulfilled, the charging station insert algorithm from Section 3.3.1 is used to determine if the customer can be inserted with some additional charging stations. If possible, the battery constraints are deemed as fulfilled and the customer, together with the required charging stations, are inserted in their respective positions. The algorithm then continues by randomly choosing the next customer from the list and inserting it using the same procedure.

3.3.3 Highest position regret insert

The highest position regret heuristic [19] uses a so called *regret value* to decide the insertion order. The regret value, R_i, for customer i is computed using the formula

$$R_i = \sum_{h=2}^{k} (\Delta c_i^h - \Delta c_i^1) \tag{3.4}$$

where Δc_i^h is the increase of the objective value when inserting the customer in its hth best feasible position. For the position to be considered, the insertion must fulfill the weight and time constraints but not the battery constraints as this may be remedied by inserting charging stations.

The customer with the highest regret value is inserted in its best position if the battery constraints are fulfilled. If not, the charging station insert operator will be used to investigate potential charging station insertions. If the route can be made feasible with regard to the battery constraints, then the customer and its corresponding charging stations are inserted, otherwise the customer is tested in its second best position. This continues until the customer has been inserted in a feasible position, or all the k best positions have been tested. If the customer is inserted, the regret values need to be recalculated. The process is then repeated for

the next customer with the highest regret value. The algorithm terminates when there are no more customers to insert. If any of the customers could not be inserted, the algorithm has failed to find a new feasible solution.

3.3.4 Highest route regret insert

Highest position route regret, described in [5], operates similarly to the highest position regret heuristic. The difference is that the Δc_i^h in the regret value formula (3.4) is defined as the increase of the cost function when inserting customer i in its hth best *route* compared to inserting in its hth best *position*. Instead of testing the customer with the highest regret value in its k best positions, the k best routes are considered, and the insertion tries are attempted in the best position in each route. Other than that, both of the regret algorithms follow the same procedure.

3.4 Weight updates

The operator weights are all initialized to a value of one, and depending on the operators' performance the weights are increased or decreased. The update policy used is adapted from [5], where the weights are updated once every 100th iteration using the formula

$$w_{i,j+1} = w_{i,j}(1 - r) + r\frac{\pi_i}{\theta_i}. \tag{3.5}$$

Here $w_{i,j}$ is the weight of operator i after the jth update, π_i is the *score* of operator i and θ_i is the number of times it has been used since the last update. The parameter $r \in [0, 1]$ regulates how quickly the weights are updated.

The score for each operator is initialized to zero and is then incremented depending on the performance of the operator. There are three different cases where the score is updated:

- when the new solution is better than the best solution found so far,
- when the new solution is accepted and has a lower cost than the previously accepted solution, and
- when the new solution is accepted despite having a higher cost than the last accepted solution.

The increment sizes for the different cases are denoted σ_1, σ_2 and σ_3 respectively. Unlike in the update scheme in [5], previous solutions are not remembered when calculating the operator scores.

3.5 Acceptance criterion

The acceptance criterion determines whether or not a new solution should be accepted and used as the starting solution in the next iteration. The Metropolis criterion is commonly used in ALNS, as it reduces the risk of becoming trapped in local optima [4]. The criterion states that a new solution is immediately accepted

if it is better than the best found so far. Otherwise, simulated annealing is applied, as described in (2.6), allowing a worse solution to be accepted with the varying probability P.

The annealing temperature, T_i is updated according to the formula

$$T_{i+1} = \alpha T_i, \tag{3.6}$$

for each iteration i, where α is the cooling factor. The initial temperature, T_0, is obtained as described in [5]. There, the value of T_0 is set such that a new solution with z times higher objective value is accepted with a probability of 50% (e.g., for $z = 0.05$, the probability P of accepting a 5% worse solution is 50%).

3.6 Termination criterion

ALNS is a heuristic method, thus it is not guaranteed to reach an optimum. Therefore, it is essential to use a suitable termination criterion. In this work, ALNS is set to terminate after a given time limit, to make the results comparable to solutions generated by other solvers. The time limit is determined based on the number of customers in the data instance. The time limits used are presented together with the results in Section 4.

4
Tests and Results

In this section, tests and results showcasing the performance of the ALNS implementation are presented. The programming language used is Julia and the code can be found in [20].

4.1 Evaluation data

The data used in this report is introduced in [13] and is commonly used by researchers for solving the EVRP (for example, in [21], [22] and [23]). The data includes both smaller and larger data files each containing 5, 10, 15 or 100 customers. It also includes information about available charging stations. The instances are divided into three categories based on their customer distribution: random (`r`), clustered (`c`), and randomly clustered (`rc`). All instances starting with `r1`, `c1` or `rc1` have tighter time windows than the ones starting with `r2`, `c2` or `rc2`.

In this project the ALNS algorithm is evaluated on 5, 15 and 100 customers with three data instances for each customer distribution type. The average running time and the objective value are then reported.

The dataset contains the following information:

- location, time window, service time and demand for each node including the depot,
- average vehicle velocity,
- battery capacity,
- load capacity,
- average energy consumption rate, and
- average inverse recharging rate.

However, the data does not contain the number of available vehicles since it is part of the objective function in the article where it was introduced. Therefore, the available number of vehicles is based on the results presented in [13] but somewhat arbitrarily inflated to ensure a feasible initial solution. The data does not contain any travel distances, therefore the Euclidean distance between nodes based of their geographical coordinates was used. The travel times are calculated from the distances using the average velocity.

For the C-EVRPTW with both full and partial charging, the energy consumption is calculated as the average energy consumption rate multiplied by the distance

between nodes. As mentioned in Section 2.1.3 there are additional parameters (Φ_1, Φ_2 and m) required to model the load dependent discharging. As Φ_1 and Φ_2 are not included in the data, they are calculated according to (2.3) and (2.4) with the constant values set as in [15]. This gives $\Phi_1 = 0.07509$ and $\Phi_2 = 0.0005103$. For the mass of the vehicle, m, a reasonable value for the mass of a car was used, rather than that of a truck, as the energy consumption and battery capacity data in [13] is more in line with that of a car. The value chosen was 1579 kg, and was taken simply as a reasonable mass for a typical car [24].

As an external reference, the best objective value for each selected data instance found in [13] is used. The values are generated for a model similar, but not identical, to the C-EVRPTW in this report, meaning that comparisons are not entirely fair. However, the same data is used as in this report enabling the value to be used as an approximate guess of the actual objective value.

4.2 Parameter tuning

ALNS relies on several parameters that can be tuned to enhance its performance, see Table 4.1. In this project, the values presented in [5] are used to approximate reasonable starting values for the parameters. For the parameters not present in [5] (k_{cs}, γ_1, γ_2, γ_3, k_{random} and k), the starting values are set arbitrarily. The procedure for the parameter tuning is inspired by [5] and starts by allowing the first parameter to vary while the others remain fixed. ALNS is then run three times for each variant of the EVRP (C-EVRPTW, C-EVRPTW with the partial charging policy and LD-EVRPTW) on the data instance `c101` with a time limit of one minute. The value that shows the best average objective value within a reasonable number of iterations performed is selected and used when tuning the next parameter. The iteration count is considered to reach a balance between efficiency and solution quality. The result of the parameter tuning for each problem variant is found in Table 4.2.

4.3 Evaluation using a commercial solver

For the purpose of both testing our model formulations and generating reference values for the ALNS evaluation, the commercial solver Gurobi is used [25]. The models are formulated as Mixed-Integer Linear Programming (MILP) problems, which Gurobi is specialized in solving. It is an exact solver, meaning that it uses proven methods for finding optima [25]. However, due to the computational complexity of the problem and time constraints, it is not possible to allow the solver to run until full completion for all cases. Consequently, the same time limit is set for both the ALNS and Gurobi runs. Additionally, Gurobi is given a number of charging station copies to allow for the multiple charging station visits. The values of the time limits and the number of copies depends on the size of the problem, see Table 4.3.

For each data instance used in the performance evaluations, ALNS is run with three different random seeds on each of the three EVRP formulations. For the instances with five customers, Gurobi is run three times and its running time aver-

Table 4.1: Names and descriptions of all parameters that are tuned for each problem variant.

Variable name	Description
μ	The proportion of nodes to remove in each iteration.
γ_1, γ_2 γ_3	Weights controlling the impact of the ranking, cost and feasibility to the score in the charging station insert operator.
k_{cs}	Number of steps to traverse in the charging station insert operator.
k_{random}	Number of insertion tries in random insert before terminating.
k	Parameter used for the regret heuristics.
r	The reaction factor deciding how quickly the weights react to updates.
σ_1, σ_2, σ_3	Score increments used for the weight updates.
z	The parameter used for calculating the initial temperature in the simulated annealing.
α	Cooling rate used in the simulated annealing.

aged to compensate for the fluctuations in time on such small instances, while for the other data instances it is only run once. Before any evaluations, a warm-up of five minutes is performed.

4.4 Result comparisons on smaller datasets

In this section, the performance analysis of the ALNS algorithm on instances with five and 15 customers is presented.

4.4.1 Performance evaluation on the C-EVRPTW

The results obtained by solving (2.1) for the smaller instances are presented in Table 4.4. For the datasets with five customers, the MILP approach proved optimality in most cases. In all except one of these, ALNS also found the same solution but in shorter time than Gurobi. Gurobi was unable to prove optimality within the set time limit for any of the datasets with 15 customers. Nevertheless, the difference in objective value between the solutions found by ALNS and Gurobi is less than 1% in all cases. Also for these instances, ALNS found its best solution faster than the MILP approach.

To further evaluate the performance of the ALNS algorithm, several visualizations were generated, including figures of the objective value over time, the operator weights across iterations, and the initial and final solution routes. Note that the weights are updated every 100th iteration, and to improve the clarity of the visualizations, only a subset of all iterations are shown in the figures. Example figures of the objective value over time for two instances, `rc103C15` and `r104C5`, are

Table 4.2: Parameter values after tuning for each problem variant. Here C-EVRPTW with PCP refers to the C-EVRPTW with the partial charging policy.

Parameter	**Parameter value**		
	C-EVRPTW	C-EVRPTW with PCP	LD-EVRPTW
μ	0.25	0.25	0.20
k_{cs}	3	5	4
γ_1	0.60	1.20	0.60
γ_2	0.60	1.00	0.60
γ_3	0.60	0.20	0.80
k_{random}	5	5	5
k_{regret}	3	3	2
α	0.99975	0.99975	0.99975
σ_1	19	31	22
σ_2	13	19	16
σ_3	13	22	13
z	0.05	0.05	0.07
r	0.30	0.30	0.30

Table 4.3: Time limits and the number of charging station copies used when generating the results for data with different numbers of customers. The charging station copies are used in the model formulation for the MILP.

Number of customers	**Time limit (s)**	**Number of copies**
5	60	5
15	300	7
100	600	15

shown in Figures 4.1 and 4.2. For instance `rc103C15`, additional related figures are provided in Figures 4.3 – 4.7. The corresponding figures for instance `r104C5` can be found in Appendix A.

Comparing the figures for the routes of the initial solution in Figure 4.3, the MILP-generated solution presented in Figure 4.4, and the ALNS-generated solution presented in Figure 4.5, differences in the routes can be observed. As both ALNS and the MILP approach use the same initial solution, they will start with the same objective value at time zero, as shown in Figure 4.1. With time, there will be a gap between the two as they find better solutions. The objective for the ALNS solution decreases quickly to its lowest objective, which can also be seen in Table 4.4, whereas the objective value for the MILP solutions decrease more slowly.

Analyzing the changes in the weights for the insert operators in Figure 4.6 no clear pattern emerges, and the usage of different operators fluctuates over time. The removal operator weights shown in Figure 4.7 indicate that Shaw Removal and Random Removal performed well for this particular instance. However, this is not a consistent trend, and it varies depending on the instance and the random seed used in the implementation.

In Figure 4.2, which corresponds to an instance with 5 customers, the objective

Table 4.4: The average objective value, f, and the average time, t, corresponding to when these solutions were found for different instances when solving the C-EVRPTW. The parameter n_{v} is the number of available vehicles. Opt. is set to "Yes" if the solution generated using the MILP approach is proven to be optimal and "No" otherwise. The value Δf is the relative difference between the objective value from the MILP and ALNS, calculated as $\Delta f = \frac{f_{\text{ALNS}} - f_{\text{MILP}}}{f_{\text{MILP}}}$. Ref. is a reference value for a similar model taken from [13]. If ALNS and the MILP gives different average best objective values, the lowest one is marked in bold.

		ALNS		**MILP**				
Dataset	n_{v}	f (km)	t (s)	f (km)	t (s)	Opt.	Δf (%)	Ref.
c103C5	2	165.67	0.003	165.67	0.43	Yes	0.00	176.05
c206C5	2	245.11	0.003	**236.58**	1.58	Yes	3.61	242.55
c208C5	2	158.48	0.001	158.48	2.42	Yes	0.00	158.48
r104C5	3	136.69	0.007	136.69	0.95	Yes	0.00	136.69
r105C5	3	156.08	0.0003	156.08	1.09	Yes	0.00	156.08
r202C5	2	128.78	0.003	128.78	0.94	Yes	0.00	128.78
rc105C5	3	238.05	0.003	238.05	6.07	No	0.00	241.30
rc108C5	3	253.93	0.0007	253.93	23.86	No	0.00	253.93
rc208C5	2	167.98	0.01	167.98	2.13	Yes	0.00	167.98
c103C15	5	**371.70**	0.44	374.12	187.56	No	-0.65	384.29
c202C15	4	**399.27**	0.38	399.49	296.79	No	-0.06	383.62
c208C15	4	301.98	0.02	**300.55**	4.79	No	0.48	300.55
r102C15	8	**424.20**	0.08	427.35	197.23	No	-0.74	413.93
r202C15	4	**358.00**	0.16	359.08	288.77	No	-0.30	358.00
r209C15	3	293.20	0.13	293.20	98.45	No	0.00	313.24
rc103C15	7	**397.67**	4.37	401.19	226.36	No	-0.88	397.67
rc108C15	5	370.25	0.005	370.25	1.30	No	0.00	370.25
rc202C15	4	**394.39**	0.48	397.20	33.87	No	-0.71	394.39

values of the solutions generated by ALNS and the MILP approach follow similar trends. Comparable patterns were observed for the other small instances, where both methods produced closely aligned results. Similar observations were also made for the figures corresponding to the other instances, and there was no observed difference depending on the type of customer distribution.

4.4.2 Performance evaluation on the C-EVRPTW with the partial charging policy and the LD-EVRPTW

The performance of ALNS and Gurobi on the C-EVRPTW with the partial charging policy and the LD-EVRPTW are presented in Tables 4.5 and 4.6. The problem formulations and models solved are the ones presented in Sections 2.1.2 and 2.1.3 respectively. The results are similar to those of the previous section for the full charging policy and example figures are therefore displayed in Appendix A.

Comparing ALNS to the MILP approach on the smaller data instances shows that the MILP in general outperforms ALNS on the datasets with 5 customers for

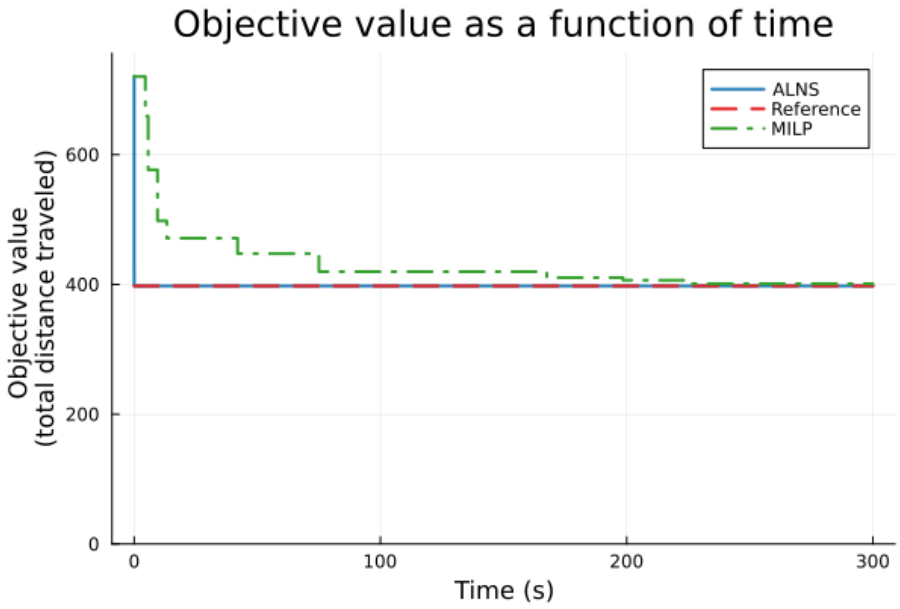

Figure 4.1: Objective value over time for the instance `rc103C15` when solving the C-EVRPTW using the ALNS method and the MILP approach, along with a reference value representing the best known objective reported in [13]. Note that the reference value is for a similar and not identical model to any of the ones included in this project, and should only be used as an approximate guess.

all three variants of the EVRP. Meanwhile, ALNS has the advantage considering both the lowest objective value and the time it took to find the solution for the instances with 15 customers, see Tables 4.4, 4.5 and 4.6. In the cases that the MILP outperforms ALNS the largest relative difference seen on the smaller data instances is 6.19%.

4.5 Performance comparisons of ALNS on larger datasets

In this section, the performance of ALNS on data instances with 100 customers will be evaluated.

4.5.1 Experiment results for the C-EVRPTW

The results from evaluating ALNS on the large data instances is displayed in Table 4.7. As can be seen, Gurobi was not able to find an optimal solution within the set time limit for any of the instances. For all instances but one, Δf is negative, which means that ALNS found a better solution within the allotted time than the MILP approach.

Figures 4.8 and 4.9 show the weights as a function of the iteration for the operators and Figure 4.10 shows the objective values over time for the instance `rc203`. For the corresponding figures illustrating the routes for the initial solution, the ALNS

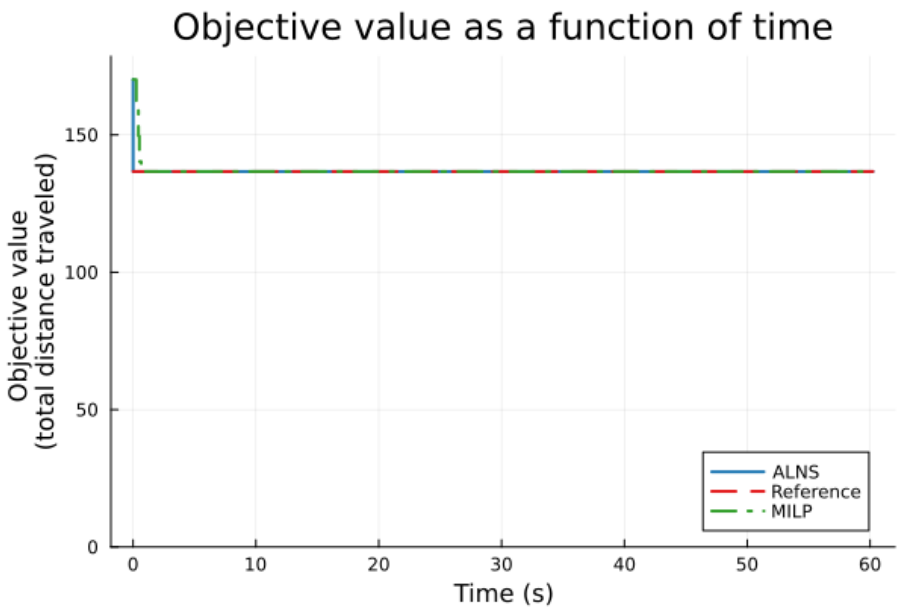

Figure 4.2: Objective value over time for the instance `r104C5` when solving the C-EVRPTW using ALNS and the MILP approach, along with a reference value representing the best known objective reported in [13]. Note that the reference value is for a similar model and not the an identical one as used in this project, and should only be used as an approximate guess.

solution and the MILP solution, see Appendix A. Some examples of 30 minute runs are presented and discussed in Appendix B.

4.5.2 Experiment results for the C-EVRPTW with the partial charging policy and LD-EVRPTW

The objective values obtained for the C-EVRPTW with the partial charging policy are presented in Table 4.8. The corresponding results for the LD-EVRPTW, presented in Table 4.9, shows that ALNS managed to find a better solution than the initial one for all instances. Meanwhile, the MILP approach did not manage to do so for three of the instances. The other results for the performance evaluation were similar to those presented for the C-EVRPTW with the full charging policy, see Section 4.5.1.

For all three EVRP variants, ALNS outperforms the MILP approach on all but one data instance with 100 customers, as can be seen in Tables 4.7, 4.8 and 4.9. The MILP approach also seems to struggle to find a solution within the time limit. This can be seen since the times are 0.00 on multiple occasions indicating that there is either a very quick solution generated and then not improved, or it is the value of the initial solution.

By comparing the Δf values for the different sizes of the data instances, it can be observed that the difference between the objective value obtained by the MILP and ALNS is greater for those with 100 customers.

Table 4.5: The average objective value, f, and the average time, t, corresponding to when these solutions were found for different instances when solving the C-EVRPTW with the partial charging policy. The parameter n_{v} is the number of available vehicles. Opt. is set to "Yes" if the solution generated using the MILP approach is optimal and "No" otherwise. The value Δf is the relative difference between the objective value from the MILP and ALNS, calculated as $\Delta f = \frac{f_{\text{ALNS}} - f_{\text{MILP}}}{f_{\text{MILP}}}$. Ref. is a reference value for a similar model taken from [13]. If ALNS and the MILP gives different average best objective values, the lowest one is marked in bold.

		ALNS		**MILP**				
Dataset	n_{v}	f (km)	t (s)	f (km)	t (s)	Opt.	Δf (%)	Ref.
c103C5	2	170.54	0.0003	**165.67**	1.56	Yes	2.94	176.05
c206C5	2	243.83	0.004	**236.58**	4.89	Yes	3.07	242.55
c208C5	2	158.48	0.003	158.48	0.89	Yes	0.00	158.48
r104C5	3	136.69	0.005	136.69	0.50	Yes	0.00	136.69
r105C5	3	156.08	0.0003	156.08	0.88	Yes	0.00	156.08
r202C5	2	128.78	0.001	128.78	0.91	Yes	0.00	128.78
rc105C5	3	238.05	0.002	**233.77**	7.81	No	1.83	241.30
rc108C5	3	253.93	0.00	253.93	8.06	No	0.00	253.93
rc208C5	2	167.98	0.003	167.98	3.07	Yes	0.00	167.98
c103C15	5	373.03	1.91	**369.32**	212.00	No	1.01	384.29
c202C15	4	389.37	0.12	**369.56**	30.14	No	5.36	383.62
c208C15	4	303.40	0.10	**300.55**	16.09	No	0.95	300.55
r102C15	8	**419.75**	5.62	429.09	45.79	No	-2.18	413.93
r202C15	3	382.87	0.29	**363.78**	169.17	No	5.25	358.00
r209C15	4	293.20	0.14	293.20	222.78	No	0.00	313.24
rc103C15	7	**397.67**	1.17	445.10	286.79	No	-10.66	397.67
rc108C15	5	**378.36**	0.10	408.89	226.06	No	-7.47	370.25
rc202C15	4	394.39	2.30	394.39	42.61	No	0.00	394.39

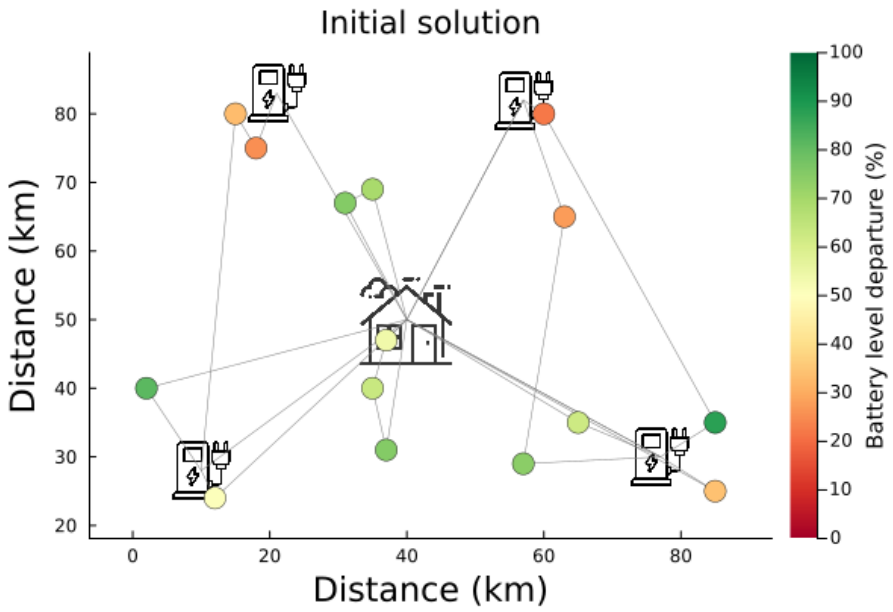

Figure 4.3: Initial solution routes for the C-EVRPTW for the instance `rc103C15`.

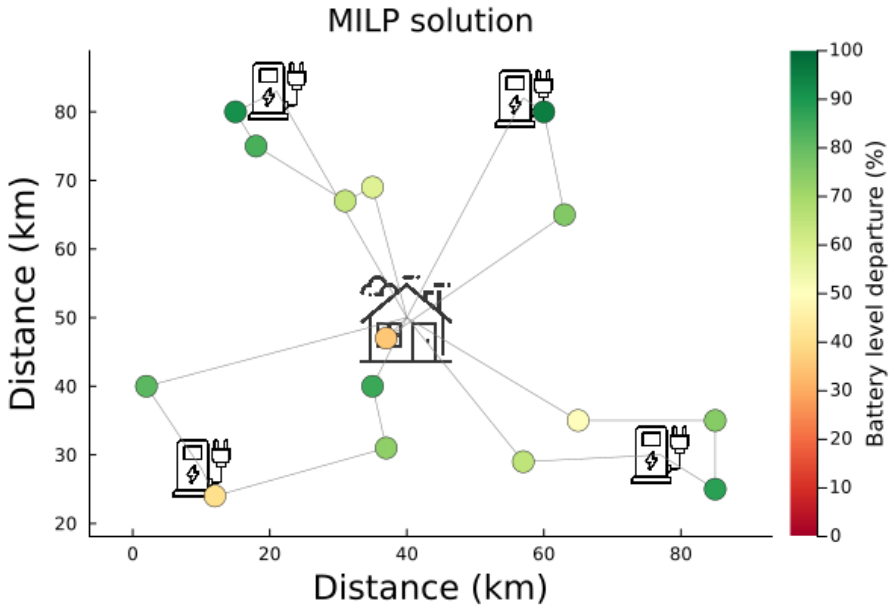

Figure 4.4: Solution routes for the C-EVRPTW generated by the MILP approach for the instance `rc103C15`. This solution is not proven to be optimal.

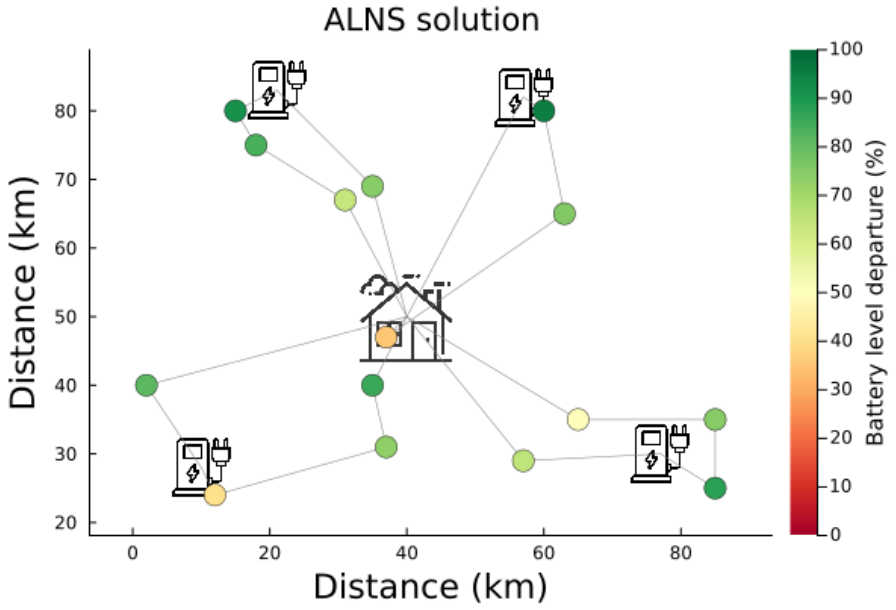

Figure 4.5: Solution routes for the C-EVRPTW generated by ALNS for the instance `rc103C15`.

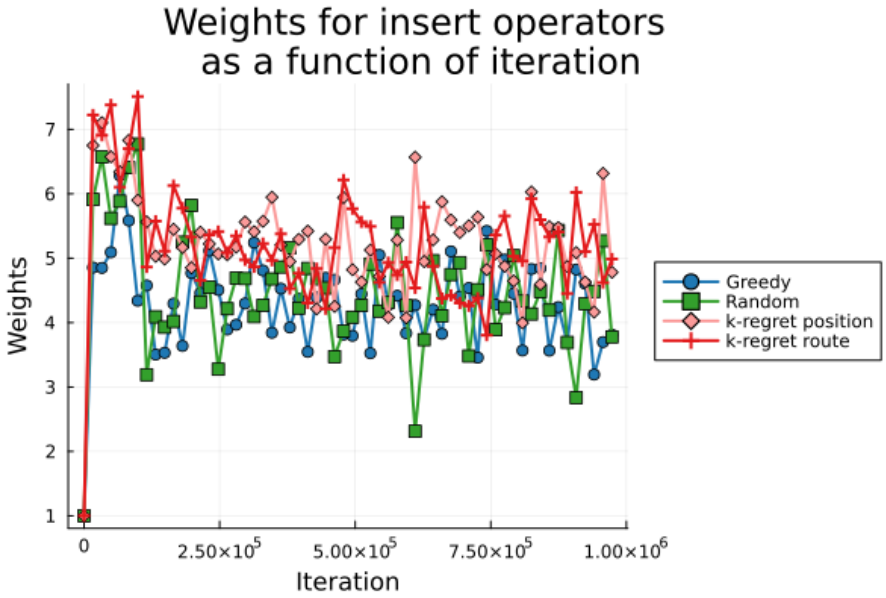

Figure 4.6: Weights over iterations for the insert operators for the C-EVRPTW for the instance `rc103C15`.

Figure 4.7: Weights over iterations for the remove operators for the C-EVRPTW for the instance `rc103C15`

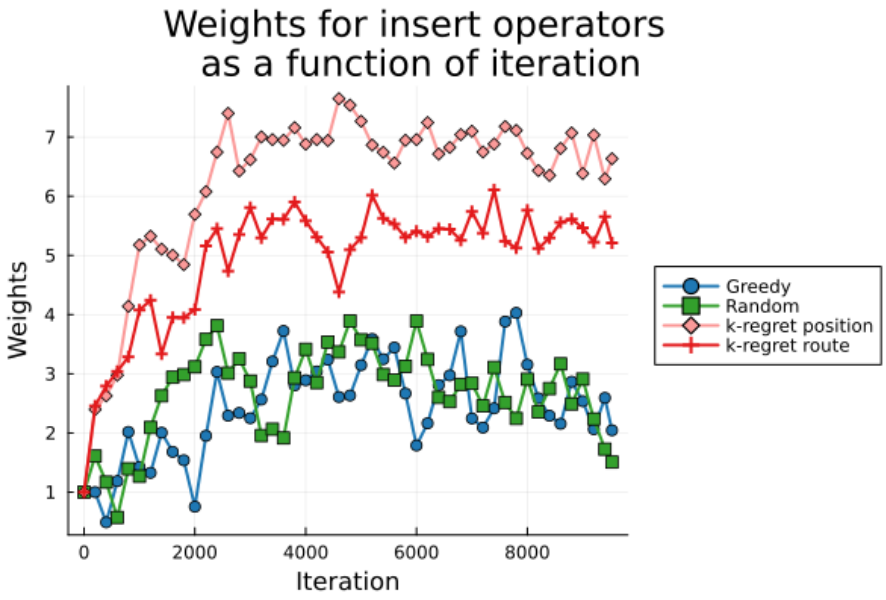

Figure 4.8: Weights over iterations for the insert operators for the C-EVRPTW for the instance `rc203`.

Table 4.6: The average objective value, f, and the average time, t, corresponding to when these solutions were found for different instances when solving the LD-EVRPTW. The parameter n_{v} is the number of available vehicles. Opt. is set to "Yes" if the solution generated using the MILP approach is optimal and "No" otherwise. The value Δf is the relative difference between the objective value from the MILP and ALNS, calculated as $\Delta f = \frac{f_{\mathrm{ALNS}} - f_{\mathrm{MILP}}}{f_{\mathrm{MILP}}}$. Ref. is a reference value for a similar model taken from [13]. If ALNS and the MILP gives different average best objective values, the lowest one is marked in bold.

		ALNS		**MILP**				
Dataset	n_{v}	f (km)	t (s)	f (km)	t (s)	Opt.	Δf (%)	Ref.
c103C5	2	168.84	0.0007	**161.26**	0.82	Yes	4.70	176.05
c206C5	2	226.89	0.003	**221.98**	6.86	Yes	2.21	242.55
c208C5	2	158.20	0.00	158.20	1.30	Yes	0.00	158.48
r104C5	3	136.69	0.002	136.69	0.68	Yes	0.00	136.69
r105C5	3	156.08	0.0007	156.08	1.70	Yes	0.00	156.08
r202C5	2	126.78	0.002	126.78	0.63	Yes	0.00	128.78
rc105C5	3	233.90	0.00	**230.65**	38.95	No	1.41	241.30
rc108C5	3	249.94	0.0003	249.94	1.50	No	0.00	253.93
rc208C5	2	167.98	0.008	167.98	7.83	Yes	0.00	167.98
c103C15	5	**371.61**	1.62	373.54	293.10	No	-0.52	384.29
c202C15	4	**399.27**	1.98	410.63	228.00	No	-2.77	383.62
c208C15	4	**298.41**	0.08	298.41	101.75	No	-3.33	300.55
r102C15	8	**405.15**	3.18	421.86	166.48	No	-3.96	413.93
r202C15	3	**366.86**	0.62	411.85	262.62	No	-10.93	358.00
r209C15	4	294.31	0.29	277.15	203.34	No	6.19	313.24
rc103C15	7	**390.20**	0.09	398.39	121.78	No	-2.06	397.67
rc108C15	5	**378.36**	0.27	396.99	275.24	No	-4.96	370.25
rc202C15	4	412.19	0.64	**392.20**	81.48	No	5.10	394.39

Table 4.7: The average objective value, f, and the average time, t, corresponding to when these solutions were found for different instances when solving the C-EVRPTW. The parameter n_v is the number of available vehicles. Opt. is set to "Yes" if the solution generated using the MILP approach is optimal and "No" otherwise. The value Δf is the relative difference between the objective value from the MILP approach and ALNS, calculated as $\Delta f = \frac{f_{\text{ALNS}} - f_{\text{MILP}}}{f_{\text{MILP}}}$. Ref. is a reference value for a similar model taken from [13]. If ALNS and MILP gives different average best objective values, the lowest one is marked in bold.

		ALNS		**MILP**				
Dataset	n_v	f (km)	t (s)	f (km)	t (s)	Opt.	Δf (%)	Ref.
c102	16	**1030.52**	297.92	1702.83	468.27	No	-39.48	1056.47
c107	16	**1033.59**	147.63	1551.89	382.29	No	-33.40	1031.56
c201	8	704.08	67.05	**703.35**	461.47	No	0.10	645.16
r102	25	**1501.46**	149.45	2224.20	0.00	No	-32.49	1495.31
r106	20	**1348.51**	291.49	1897.30	0.00	No	-28.92	1344.66
r201	5	**1166.91**	115.96	1561.41	226.57	No	-25.27	1264.82
rc103	20	**1540.46**	308.17	2276.65	0.00	No	-32.34	1351.15
rc108	17	**1317.51**	476.76	1991.72	495.72	No	-33.85	1209.61
rc203	5	**1007.06**	113.64	1643.92	566.56	No	-38.74	1073.98

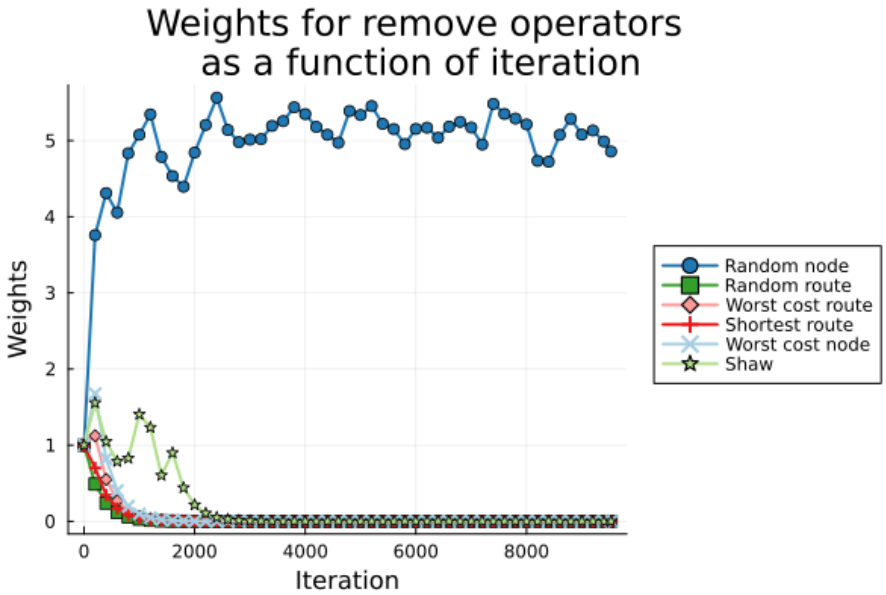

Figure 4.9: Weights over iterations for the remove operators for the C-EVRPTW for the instance `rc203`

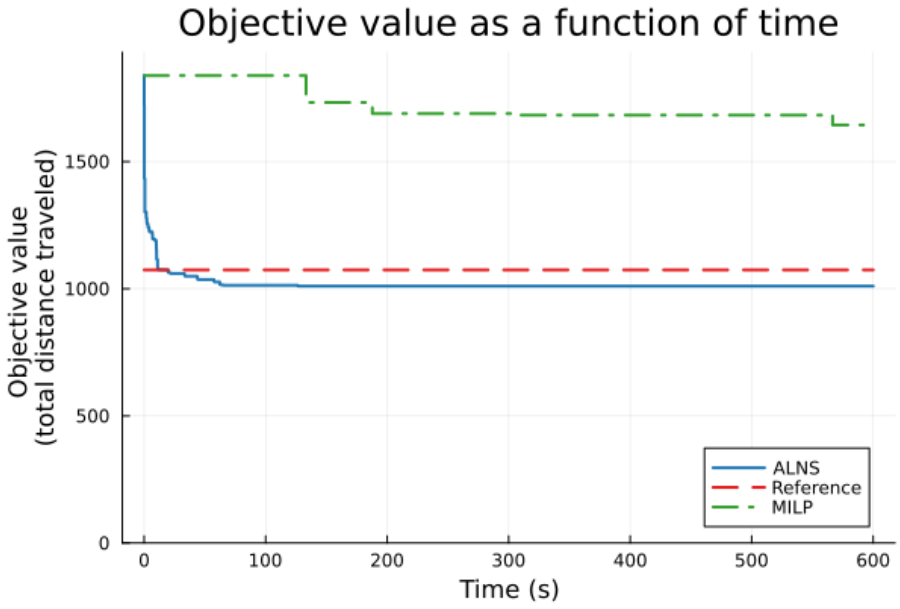

Figure 4.10: Objective value over time for instance `rc203` when solving the C-EVRPTW using the ALNS method and the MILP approach, along with a reference value representing the best known objective reported in [13]. Note that the reference value is for a similar and not identical model to any of the ones included in this project, and should only be used as an approximate guess.

Table 4.8: The average objective value, f, and the average time, t, corresponding to when these solutions were found for different instances when solving the C-EVRPTW with the partial cahrging policy. The parameter n_v is the number of available vehicles. Opt. is set to "Yes" if the solution generated using the MILP approach is optimal and "No" otherwise. The value Δf is the relative difference between the objective value from the MILP and ALNS, calculated as $\Delta f = \frac{f_\mathrm{ALNS} - f_\mathrm{MILP}}{f_\mathrm{MILP}}$. Ref. is a reference value for a similar model taken from [13]. If ALNS and the MILP gives different average best objective values, the lowest one is marked in bold.

		ALNS		**MILP**				
Dataset	n_v	f (km)	t (s)	f (km)	t (s)	Opt.	Δf (%)	Ref.
c102	16	**1033.66**	226.23	1604.62	535.11	No	-35.58	1056.47
c107	16	**1033.15**	212.89	1537.61	464.04	No	-32.81	1031.56
c201	8	639.93	11.71	**629.95**	318.85	No	1.58	645.16
r102	25	**1456.40**	386.90	2134.77	0.00	No	-31.78	1495.31
r106	20	**1329.49**	73.45	1855.50	0.00	No	-28.35	1344.66
r201	5	**1191.39**	206.42	1659.61	199.37	No	-28.21	1264.82
rc103	20	**1406.56**	346.10	2100.63	0.00	No	-33.04	1351.15
rc108	17	**1247.29**	414.57	1762.42	518.15	No	-29.23	1209.61
rc203	5	**1028.85**	187.29	1651.12	317.53	No	-37.69	1073.98

Table 4.9: The average objective value, f, and the average time, t, corresponding to when these solutions were found for different instances when solving the LD-EVRPTW. The parameter n_{v} is the number of available vehicles. Opt. is set to "Yes" if the solution generated using the MILP approach is optimal and "No" otherwise. The value Δf is the difference between the objective value from the MILP and ALNS, calculated as $\Delta f = \frac{f_{\text{ALNS}} - f_{\text{MILP}}}{f_{\text{MILP}}}$. Ref. is a reference value for a similar model taken from [13]. If ALNS and the MILP gives different average best objective values, the lowest one is marked in bold.

		ALNS		**MILP**				
Dataset	n_{v}	f (km)	t (s)	f (km)	t (s)	Opt.	Δf (%)	Ref.
c102	16	**1034.17**	442.06	1685.84	0.00	No	-38.66	1056.47
c107	16	**1033.18**	37.39	1562.01	0.00	No	-33.86	1031.56
c201	8	**743.71**	165.11	875.88	529.05	No	-15.09	645.16
r102	25	**1458.35**	298.97	2177.37	0.00	No	-33.02	1495.31
r106	20	**1303.14**	362.75	1931.49	0.00	No	-32.53	1344.66
r201	5	**1202.74**	18.93	1602.27	337.55	No	-34.94	1264.82
rc103	20	**1465.83**	256.85	2283.78	0.00	No	-35.82	1351.15
rc108	17	**1283.45**	356.91	1925.44	0.00	No	-33.34	1209.61
rc203	5	**1044.32**	76.42	1709.61	593.82	No	-38.91	1073.98

5
Discussion and Conclusion

This chapter presents the discussion and the conclusions drawn in this project. Due to a lack of time, plenty of ideas were not fully explored, instead these are mentioned in Section 5.4 as suggestions for future work.

5.1 Comparing ALNS to the MILP approach

There are several limitations concerning the obtained results and the method used for gathering them. One such limitation for ALNS is that the parameter tuning was performed using one-minute runs, which might make the conditions unfavorable for longer runs. Another limitation, but for the MILP approach, is that the number of charging station copies used to allow multiple visits to charging stations increases the complexity of the problem which in turn reduces the solver's performance. However, if instead there are too few copies, the vehicles may be forced to visit charging stations further away which can increase the objective value unnecessarily. Differences in the number of copies across data instances with different charging stations counts prohibits direct performance comparisons, due to the non-uniform scaling of the problem.

Using Gurobi to generate reference solutions was appropriate for this project since it is a method that is easy to implement and it provides a baseline that the ALNS solutions can be compared to. However, it may not be the most effective software to compare with when considering the execution time since there exist quicker solution methods. This was not an option in this project since they require more time to implement. The comparisons to the reference values from [13] are not reliable since they also minimize the number of vehicles and their problem is not identical to any of the variants considered in this project.

It is important to note that the MILP approach will generate an optimal value given enough time while ALNS cannot guarantee optimality. However, as can be seen in the results, ALNS quickly finds an adequate solution, often within a few seconds and thereafter, the improvements are small to none. Depending on the purpose of solving the EVRP it may be advantageous to consider ALNS. If the optimality is less relevant and the solution speed is of more importance, ALNS seems to be a viable choice. If the purpose is to plan routes long before departure, the execution time becomes less important and an exact solver has an advantage. Despite that, ALNS can reduce the required resources and computation time making it possible to make the route planning more cost efficient. It is also possible to solve much larger problems or multiple instances in parallel with an efficient algorithm.

Comparing the results for the different problem variants, the differences are small, partly due to the similarity between the variants. Analyzing the results for the data instances with 5 customers, there is an indication that ALNS in general performs better on the C-EVRPTW compared to the C-EVRPTW with the partial charging policy and the LD-EVRPTW. For the 15 and 100 customer instances, it is difficult to draw any conclusions on which problem variant ALNS excels, mainly due to the lack of good comparisons. For these, the MILP approach does not reach optimality and, as previously mentioned, the reference value is not comparable. Therefore it is difficult to differentiate between the case where ALNS performs better and the case where MILP performs worse.

Regarding the C-EVRPTW with the partial charging policy, there is not an equal standing when solving it using ALNS, since checking the battery constraints require double the computations. This because the recharging times cannot be decided by going through a route once. Whereas for the LD-EVRPTW, the MILP approach is at a disadvantage since there are multiple constraints added to model the tracking of the weight.

Some conclusions that can be drawn for 15 customers are that ALNS seems to be the most consistent for the C-EVRPTW but performs the best on the LD-EVRPTW, based off the value of Δf. For 100 customers, the differences between the problem variants are quite small, but overall ALNS seemed to perform the best on the C-EVRPTW and the LD-EVRPTW, considering the Δf values. The problem variant where ALNS seems to perform the worst is for the C-EVRPTW with the partial charging policy. Reasons for this is discussed in the next section.

5.2 Challenges of implementing a general algorithm

One of the aims of this project was to investigate the possibility of creating a general implementation of ALNS that could be used for all three selected problem variants. The implementation of the algorithm is general in the sense that all the different problem variants use the same logic, except for the battery level calculations and the energy consumption function used. These parts are easily switched out in the algorithm making it possible to use the same insert and remove operators for all cases. The algorithm is also general in the sense that the objective function can be replaced.

As can be seen by comparing Table 4.4 and 4.5 the performance of ALNS on the C-EVRPTW seems to be better than the performance on the C-EVRPTW with the partial charging policy. This may partly be due to both the design of the charging station insert algorithm and the battery calculations. The charging station insert algorithm does not allow insertions if the time window constraints are unfulfilled and the insertions are done one by one. However, given a feasible route where multiple charging stations have been removed, there is a possibility that the time windows are not fulfilled when replacing only one of them since the recharging times are affected. Regarding the battery calculations, they are calculated based on the assumption that all vehicles arrive with empty batteries to the charging stations and the end

depot (after the initial battery gained at the start depot is depleted). This is not optimal since a vehicle might decide to charge briefly at a charging station while waiting for a time window at a customer to start. These issues are addressed for the MILP approach but not for ALNS leaving it at a slight disadvantage. However, since the algorithm is heuristic it is not guaranteed an optimal solution and therefore such assumptions, if they simplify the problem, may be justified. Important to note is that inserting one charging station at a time causes the same issue when using the full charging policy, but it is more prominent when using the partial charging policy.

Usually, solving the LD-EVRPTW can be challenging since the energy consumption on each arc changes if a node is removed or inserted. None of the insert or remove operator used for this report requires a constant energy consumption on the arcs, making the algorithm effective even for this problem instance. However, as greedy insert is inefficient for larger datasets, the *Earliest Time Window Heuristic* was initially used to create the initial solutions. A description of the Earliest Time Window Heuristic can be found in Appendix C. Due to other requirements for the load-dependent discharging, it had to be exchanged for the more versatile greedy heuristic. This to allow for comparisons between the different problem variants.

5.3 Conclusions

The conclusion is that ALNS quickly generates an adequate solution for all three problem variants (C-EVRPTW, C-EVRPTW with the partial charging policy and LD-EVRPTW). However, for a more robust evaluation it is necessary to have better reference solutions as ALNS is a heuristic method. These reference solutions are also needed to draw conclusions on which problem variant ALNS perform the best. However, it can be seen that ALNS performs slightly worse on the C-EVRPTW with the partial charging policy. Overall, the ALNS algorithm has demonstrated strong performance across all variants of the EVRP studied in this project, as evidenced by the conducted evaluation.

5.4 Future Work

Due to the time limitation, there are unexplored aspects to the project both regarding the problem formulations and the implementation of ALNS. One of the most important is to find solutions to compare with that are more appropriate than those generated by Gurobi. This may be done by finding results from research on more similar models, a faster exact solver or other solution methods. It is also of interest to investigate if there is an appropriate way to find a lower bound on the objective value that can be used as a measure on how close the solutions may be to the optimal value.

To properly evaluate the algorithm it is preferred to run the algorithm for much longer than what was done for this report. There is also a need for a more extensive parameter tuning.

Evaluation on different datasets and possibly even real-world data could be beneficial. As a minimum, all instances included in [13] should be used instead of only a subset.

Regarding the code itself, it is not optimized and there is potential for both parallelization and logical improvements in the algorithm to improve the execution time.

Implementing multiple charging station insert operators as well as customer insert operators could also be advantageous. A suggestion is to implement a greedy insert with noise on all calculated objective values as in [5] to avoid local minima. Diversity in the operators can greatly benefit the solution search. A proper analysis and evaluation of the operators to find which performs the best across the data instances would also be of interest. A brief analysis of the weights was done for this project as well by only considering the figures. However, this can be done more scientifically by for example comparing the average of the weights. Another needed improvement, as previously mentioned, is a more advanced battery and time handling algorithm to address the problems with the partial charging policy. It may also be advantageous to design a charging station insert operator which performs multiple insertions at a time instead of inserting them one by one.

Regarding the problem formulations, it would be interesting to limit the number of available vehicles since the initial solution often require extra vehicles to generate a feasible solution. This could be cause for allowing infeasible initial solutions, designing a more vehicle efficient initial solution algorithm, or changing the objective function to also minimize the number of vehicles used. It is also worth mentioning that this report only considers three different problem variants which could be extended by for example considering inhomogeneous fleets, worker conditions or a limited number of simultaneous visits to charging stations. We see a lot of potential for the algorithm to be extended to various of these other problem variants with minor modifications due to our focus of building a general and reusable implementation.

Bibliography

[1] Intergovernmental Panel on Climate Change (IPCC). Transport. In: Climate Change 2022 - Mitigation of Climate Change: Working Group III Contribution to the Sixth Assessment Report of the Intergovernmental Panel on Climate Change. Cambridge: Cambridge University Press; 2023. p. 1049–160. DOI: 10.1017/9781009157926.012.

[2] Volvo Trucks. Volvo Trucks biggest in electric trucks in Europe and North America. [accessed 2025-05-24]. https://www.volvotrucks.com/en-en/news-stories/press-releases/2025/mar/volvo-trucks-biggest-in-electric-trucks-in-europe-and-north-amer.html.

[3] Osaba E, Yang XS, Del Ser J. Traveling salesman problem: a perspective review of recent research and new results with bio-inspired metaheuristics. In: Nature-Inspired Computation and Swarm Intelligence. 1st ed. Elsevier; 2020. p.135-64. DOI: 10.1016/B978-0-12-819714-1.00020-8.

[4] Windras Mara ST, Norcahyo R, Jodiawan P, Lusiantoro L, Rifai AP. A survey of adaptive large neighborhood search algorithms and applications. Computers & Operations Research. 2022 Oct;146:105903. DOI: 10.1016/j.cor.2022.105903.

[5] Røpke S, Pisinger D. An adaptive large neighborhood search heuristic for the Pickup and Delivery Problem with Time Windows. Transportation Science. 2006 Nov;40(4):455–472. DOI: 10.1287/trsc.1050.0135.

[6] Zhao B, Wu Y. Research on multitrip vehicle routing problem with soft time window based on ALNS algorithm. In: Ninth International Conference on Electromechanical Control Technology and Transportation (ICECTT 2024). Proc SPIE. 2024 Aug 28;13251:1325143. DOI: 10.1117/12.3039452.

[7] Yilmaz Y, Kalayci CB. Variable Neighborhood Search Algorithms to Solve the Electric Vehicle Routing Problem with Simultaneous Pickup and Delivery. Mathematics. 2022;10(17):3108. DOI: 10.3390/math10173108.

[8] Nie ZH, Yang Q, Zhang E, Liu D, Zhang J. Ant Colony optimization for Electric Vehicle Routing Problem with Capacity and Charging Time Constraints. In: 2022 IEEE International Conference on Systems, Man, and Cybernetics (SMC); 2022 Oct; Prague, Czech Republic. IEEE; 2022. p. 480-5. DOI: 10.1109/SMC53654.2022.9945248.

[9] Li C, Zhu Y, Lee KY. Route Optimization of Electric Vehicles Based on Reinsertion Genetic Algorithm. IEEE Transactions on Transportation Electrification. 2023 Sep; 9(3):3753–68. DOI: 10.1109/TTE.2023.3237964

[10] Shaw P. Using Constraint Programming and Local Search Methods to Solve Vehicle Routing Problems. In: Maher M, Puget JF, editors. Principles and practice of constraint programming – CP98. Lecture Notes in Computer Science. Vol. 1520. Berlin: Springer; 1998. p.417-31. DOI: 10.1007/3-540-49481-2_30.

[11] Lin J, Zhou W, Wolfson O. Electric Vehicle Routing Problem. Transportation Research Procedia. 2016 Jan;12:508-21. DOI: 10.1016/j.trpro.2016.02.007

[12] Kancharla SR, Ramadurai G. Electric vehicle routing problem with non-linear charging and load-dependent discharging. Expert Systems with Applications. 2020 Dec;160:113714. DOI: 10.1016/j.eswa.2020.113714

[13] Schneider M, Stenger A, Goeke D. The Electric Vehicle-Routing Problem with Time Windows and Recharging Stations. Transportation Science. 2014;48(4):500–20.

[14] Kucukoglu I, Dewil R, Cattrysse D. The electric vehicle routing problem and its variations: A literature review. Computers & Industrial Engineering. 2021 Nov;161. DOI: 10.1016/j.cie.2021.107650.

[15] Xu W, Zhang C, Cheng M, Huang Y. Electric Vehicle Routing Problem with Simultaneous Pickup and Delivery: Mathematical Modeling and Adaptive Large Neighborhood Search Heuristic Method. Energies. 2022 Dec;15(23):9222. DOI: 10.3390/en15239222.

[16] Voigt S. A review and ranking of operators in adaptive large neighborhood search for vehicle routing problems. European Journal of Operational Research. 2024 May;322(2):357–375. DOI: 10.1016/j.ejor.2024.05.033.

[17] Kirkpatrick S, Gelatt CD, Vecchi MP. Optimization by Simulated Annealing. In: Fischler MA, Firschein O, editors. Readings in Computer Vision: Issues, Problems, Principles, and Paradigms; Elsevier; 1987. p. 606–15. DOI: 10.1016/B978-0-08-051581-6.50059-3

[18] Demir E, Bektaş T, Laporte G. An adaptive large neighborhood search heuristic for the Pollution-Routing Problem. European Journal of Operational Research. 2012 Dec;223(2):346–59. DOI: 10.1016/j.ejor.2012.06.044

[19] Hemmelmayr VC, Cordeau JF, Crainic TG. An adaptive large neighborhood search heuristic for Two-Echelon Vehicle Routing Problems arising in city logistics. Computers & Operations Research. 2012 Dec;39(12):3215–28. DOI: 10.1016/j.cor.2012.04.007

[20] Gustavsson R, Svenningsson ES. optimizing-the-evrp-using-alns-master-thesis. GitHub; 2025 [accessed 2025-06-12]. `https://github.com/EllinorAndRegina/optimizing-the-evrp-using-alns-master-thesis`

[21] Kancharla SR, Ramadurai G. An Adaptive Large Neighborhood Search Approach for Electric Vehicle Routing with Load-Dependent Energy Consumption. Transportation in Developing Economies. 2018 Oct;4:10. DOI: 10.1007/s40890-018-0063-3.

[22] Zhou Y, Kong L, Wang H, Cai Y, Liu S. A local search with chain search path strategy for real-world many-objective vehicle routing problem. Complex & Intelligent Systems. 2025 Apr;11:199. DOI: 10.1007/s40747-025-01825-9.

[23] Akbay MA, Blum C, Kalayci CB. Application of Adapt-CMSA to the Electric Vehicle Routing Problem with Simultaneous Pickup and Deliveries. In: Computer Aided Systems Theory – EUROCAST 2024. Cham: Springer; 2025. p. 90-106. (Lecture Notes in Computer Science; vol. 15172). DOI: 10.1007/978-3-031-82949-9_9

[24] Volvo Cars. Volvo V70 (2012) - Technical Data. Gothenburg: Volvo Cars; [accessed 2025-06-25]. https://www.media.volvocars.com/global/en-gb/models/v70-2008-2016/2012/specifications

[25] Gurobi Optimization, LLC. Gurobi Optimizer Reference Manual. 2024. [accessed 2025-06-25]. https://www.gurobi.com

[26] Wang H, Cheu RL. Operations of a Taxi Fleet for Advance Reservations Using Electric Vehicles and Charging Stations. Transportation Research Record, 2013;2352(1):1-10. DOI: 10.3141/2352-01

A

Additional figures

This appendix provides additional example figures that supplement those presented in Section 4. The figures for each data instance are organized into separate sections, each named after the corresponding instance.

A.1 Instance `r104C5` – C-EVRPTW with the full charging policy

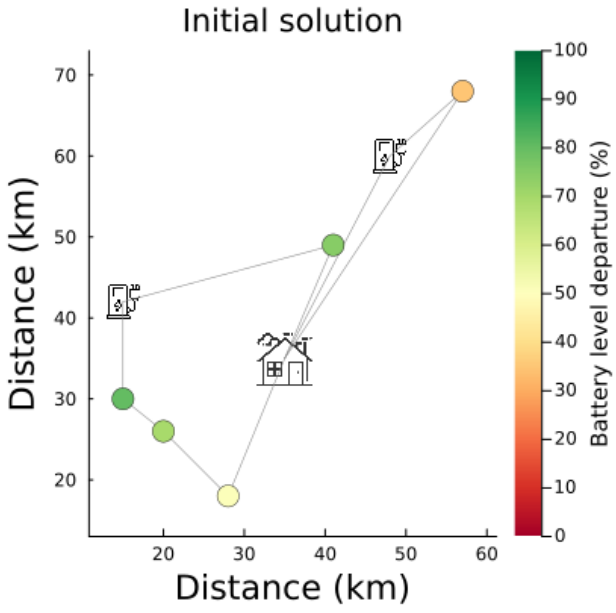

Figure A.1: Initial solution routes for the C-EVRPTW for the instance `r104C5`.

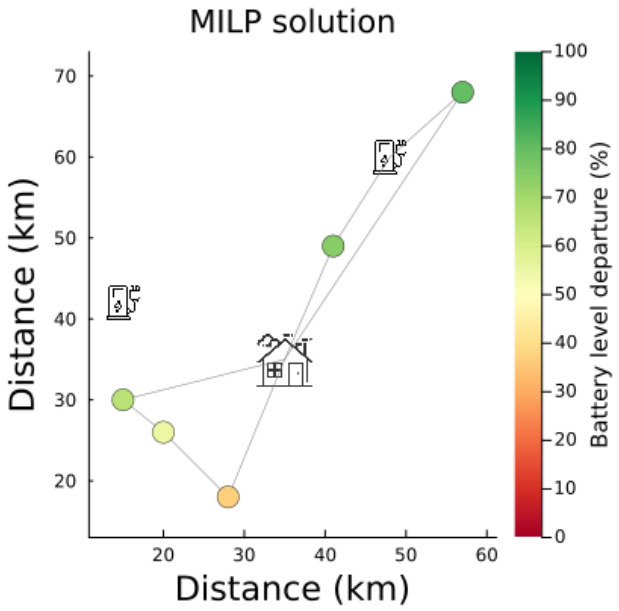

Figure A.2: Solution routes for the C-EVRPTW generated by the MILP approach for the instance `r104C5`.

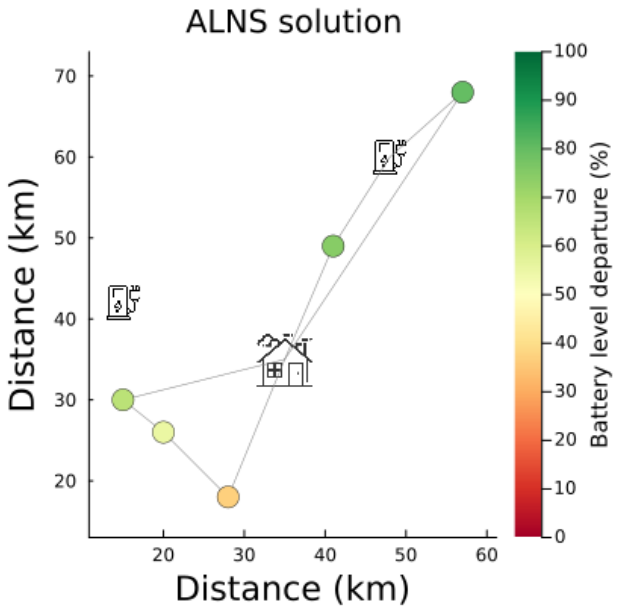

Figure A.3: Solution routes for the C-EVRPTW generated by ALNS for the instance `r104C5`.

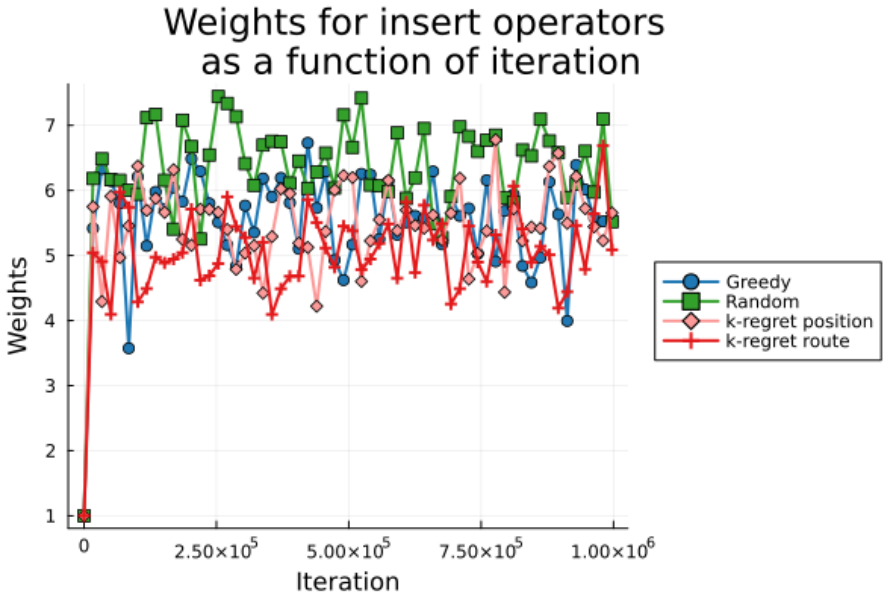

Figure A.4: Weights over iterations for the insert operators for the C-EVRPTW for the instance `r104C5`.

Figure A.5: Weights over iterations for the remove operators for the C-EVRPTW for the instance `r104C5`

A.2 Instance `rc103C15` – C-EVRPTW with the partial charging policy

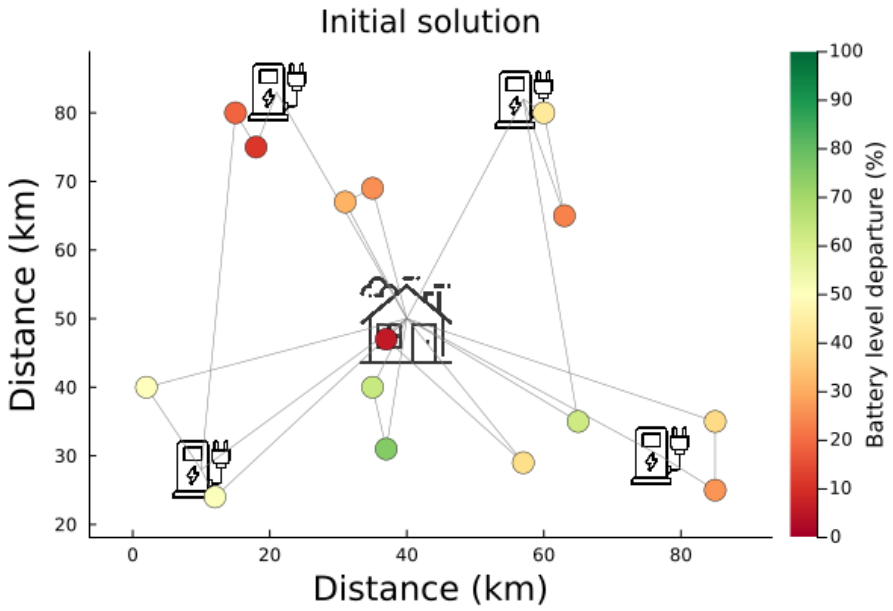

Figure A.6: Initial solution routes for the C-EVRPTW with the partial charging policy for the instance `rc103C15`.

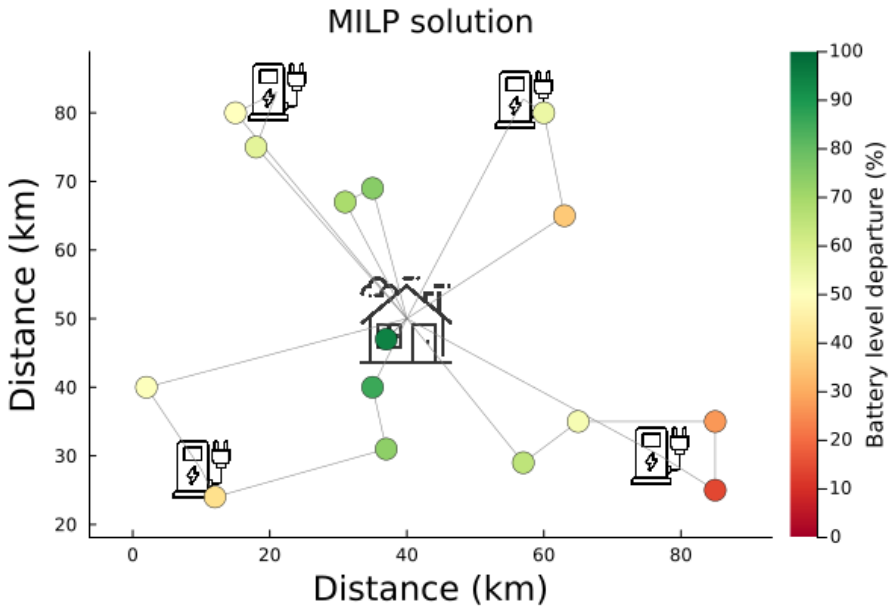

Figure A.7: Solution routes for the C-EVRPTW with the partial charging policy generated by the MILP approach for the instance `rc103C15`.

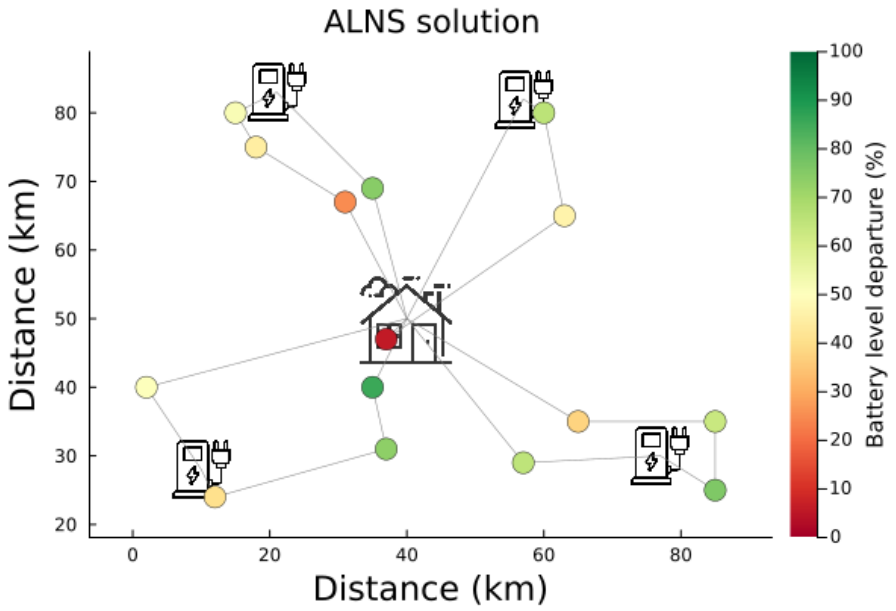

Figure A.8: Solution routes for the C-EVRPTW with the partial charging policy generated by ALNS for the instance `rc103C15`.

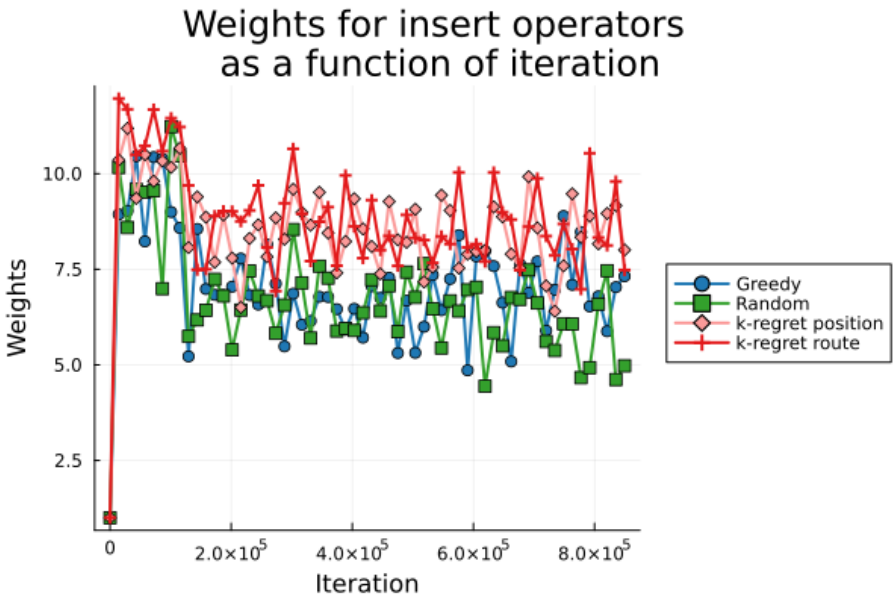

Figure A.9: Weights over iterations for the insert operators for the C-EVRPTW with the partial charging policy for the instance `rc103C15`.

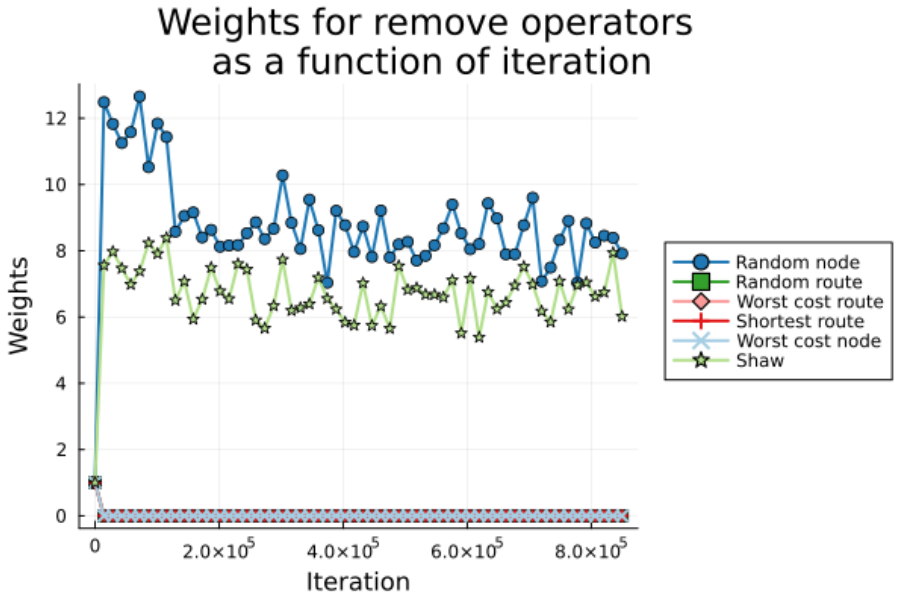

Figure A.10: Weights over iterations for the remove operators for the C-EVRPTW with the partial charging policy for the instance `rc103C15`

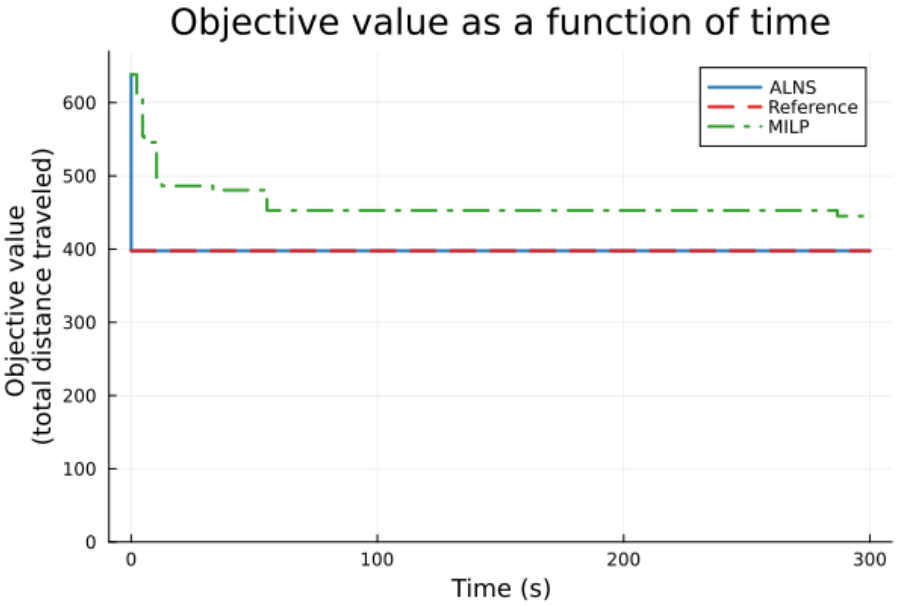

Figure A.11: Objective value over time for instance `rc103C15` when solving the C-EVRPTW with the partial charging policy using ALNS and the MILP approach, along with a reference value representing the best known objective reported in [13]. Note that the reference value is for a similar and not identical model to any of the ones included in this project, and should only be used as an approximate guess.

A.3 Instance `rc103C15` – LD-EVRPTW

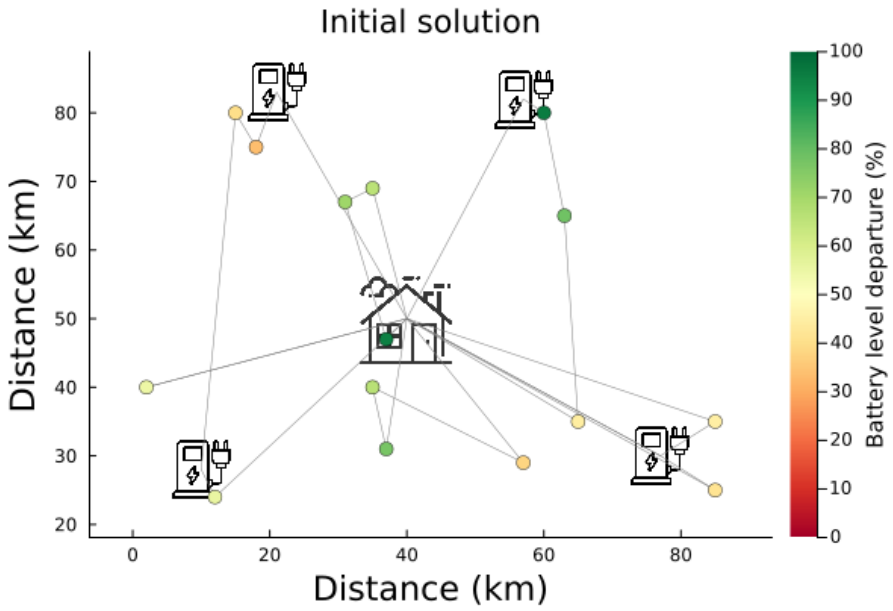

Figure A.12: Initial solution routes for the LD-EVRPTW for the instance `rc103C15`.

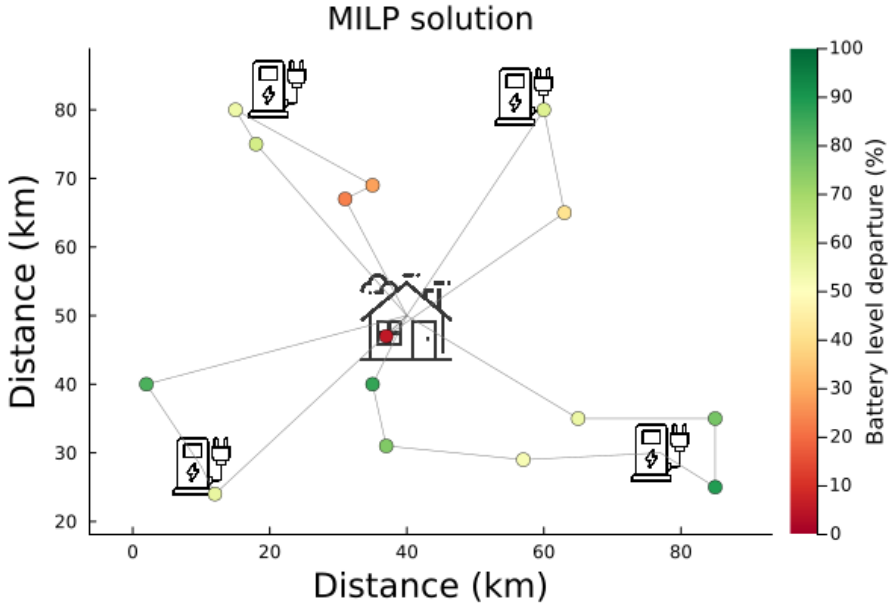

Figure A.13: Solution routes for the C-EVRPTW generated by the MILP approach for the instance `rc103C15`.

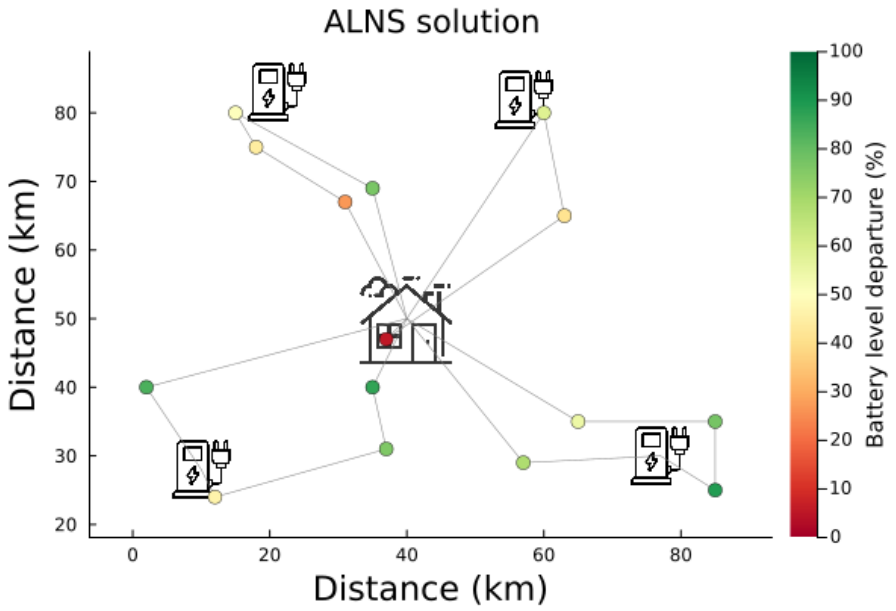

Figure A.14: Solution routes for the LD-EVRPTW generated by the ALNS method for the instance `rc103C15`.

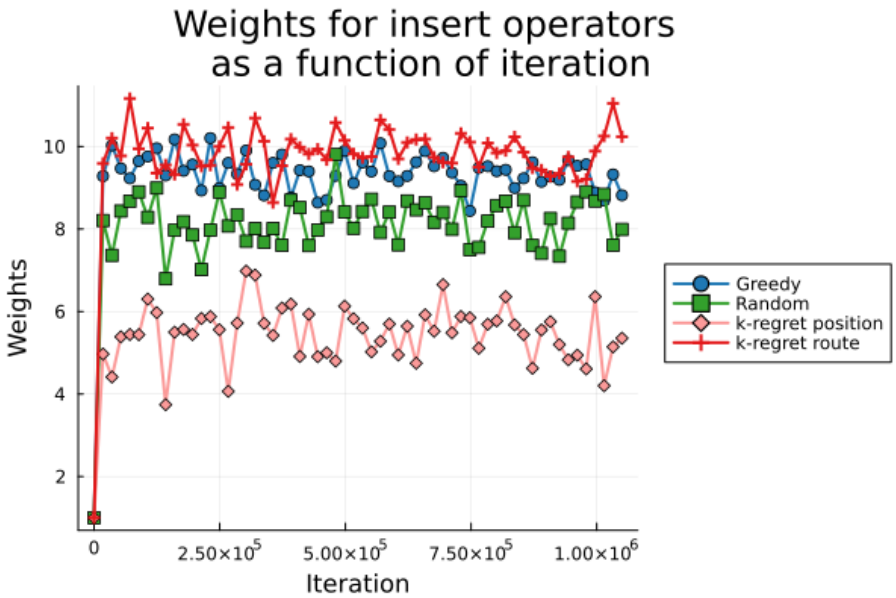

Figure A.15: Weights over iterations for the insert operators for the LD-EVRPTW for the instance `rc103C15`.

Figure A.16: Weights over iterations for the remove operators for the LD-EVRPTW for the instance `rc103C15`

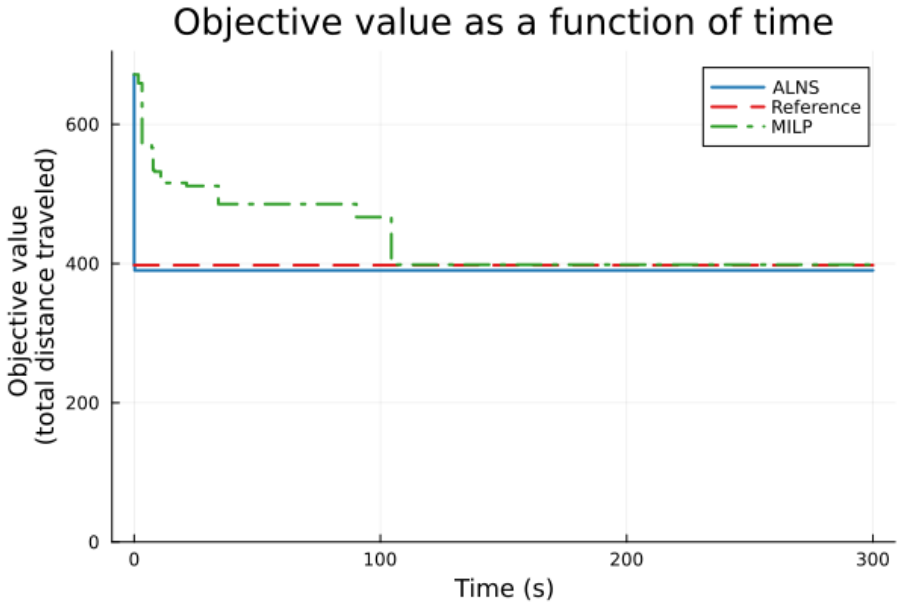

Figure A.17: Objective value over time for instance `rc103C15` when solving the LD-EVRPTW using ALNS and the MILP approach, along with a reference value representing the best known objective reported in [13]. Note that the reference value is for a similar and not identical model to any of the ones included in this project, and should only be used as an approximate guess.

A.4 Instance `r209C15` – LD-EVRPTW

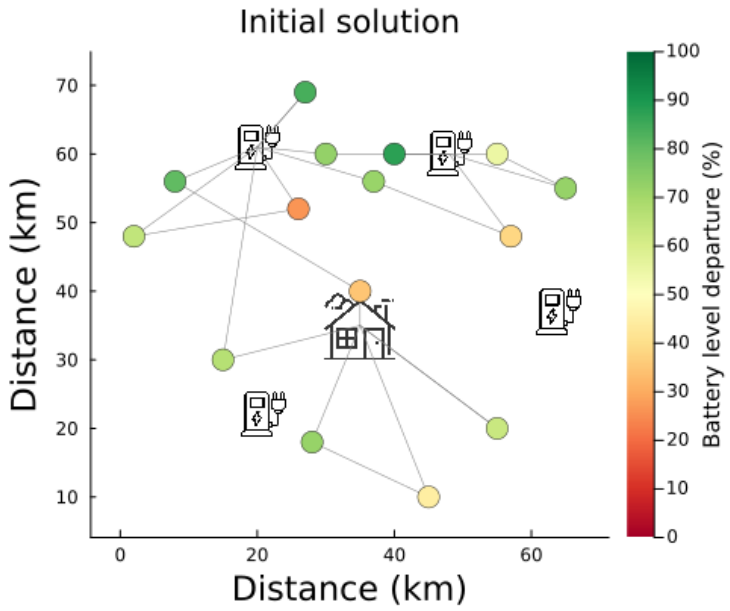

Figure A.18: Initial solution routes for the LD-EVRPTW on the instance `r209C15`.

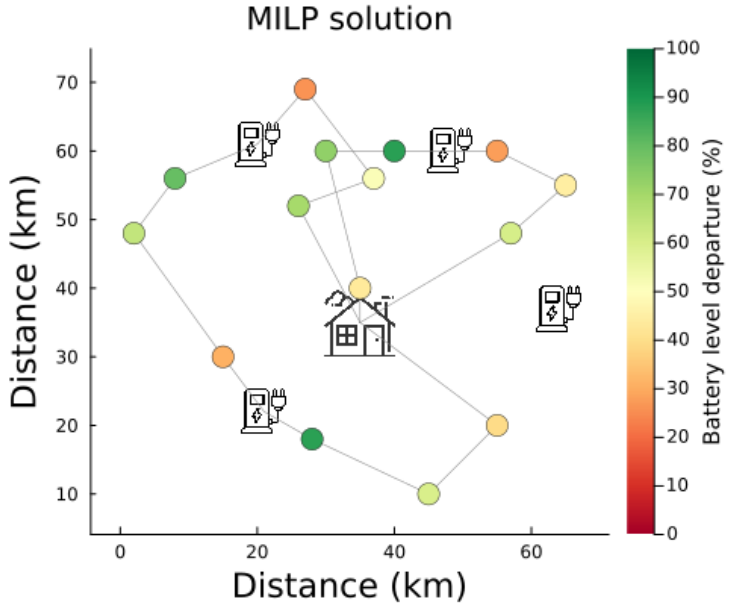

Figure A.19: Solution routes for the LD-EVRPTW generated by the MILP approach for instance `r209C15`.

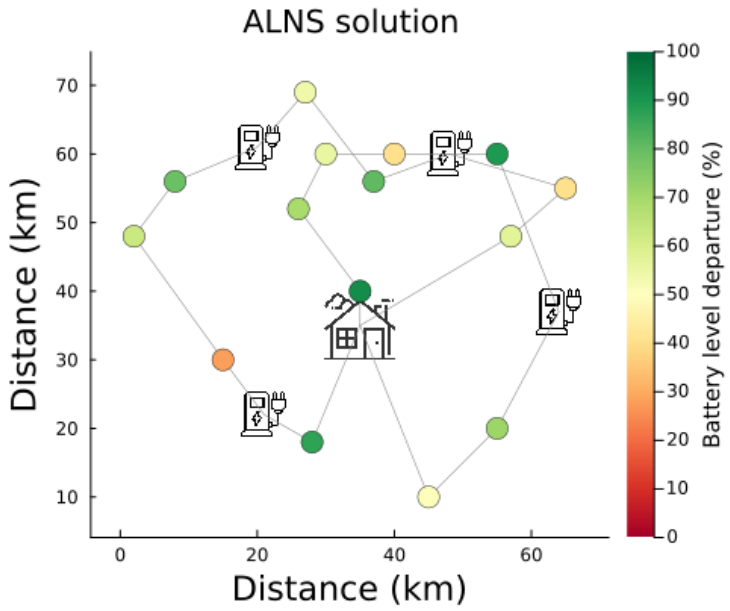

Figure A.20: Solution routes for the LD-EVRPTW generated by the ALNS method for instance `r209C15`.

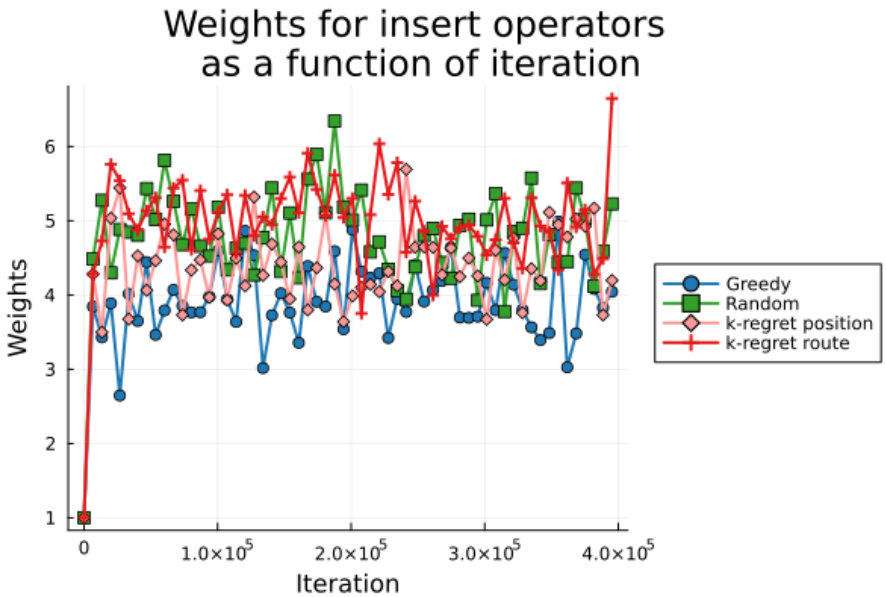

Figure A.21: Weights over iterations for the insert operators for the LD-EVRPTW for the instance `r209C15`.

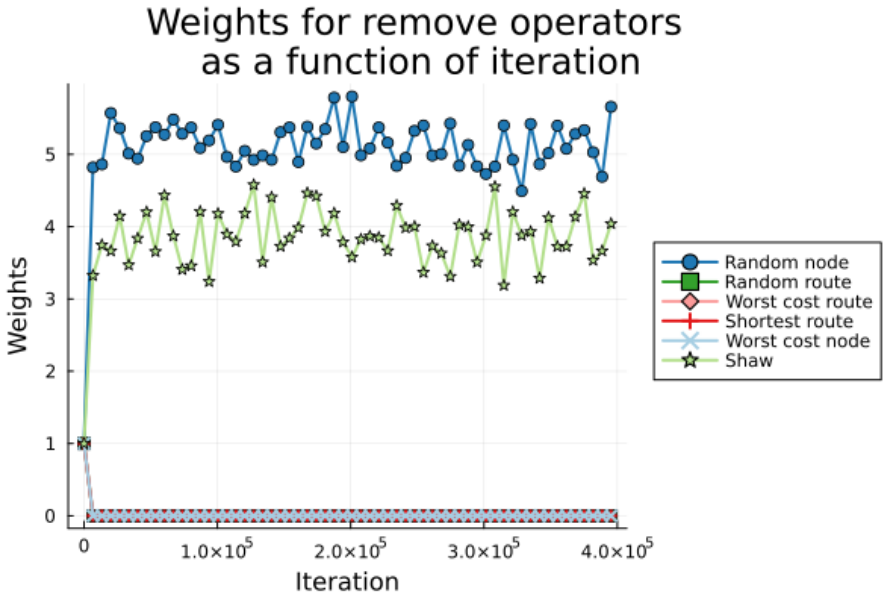

Figure A.22: Weights over iterations for the remove operators for the LD-EVRPTW for the instance `r209C15`

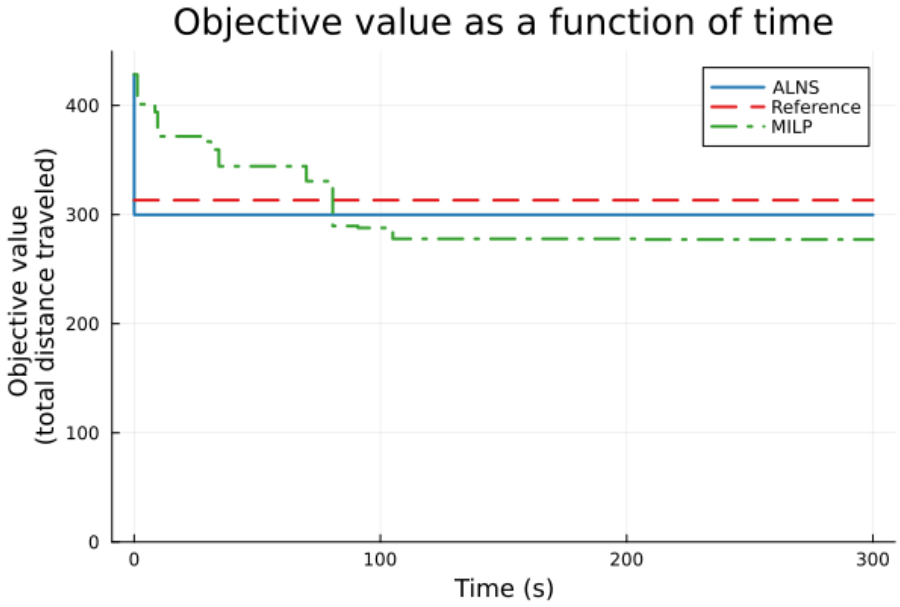

Figure A.23: Objective value over time for instance `r209C15` when solving the LD-EVRPTW using ALNS and the MILP approach, along with a reference value representing the best known objective reported in [13]. Note that the reference value is for a similar and not identical model to any of the ones included in this project, and should only be used as an approximate guess.

A.5 Instance `rc203` – C-EVRPTW with the full charging policy

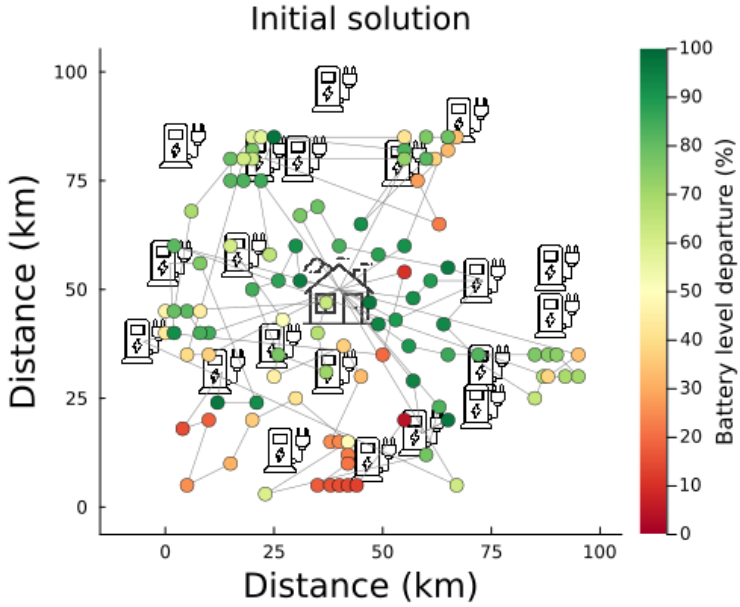

Figure A.24: Initial solution routes for the C-EVRPTW for the instance `rc203`.

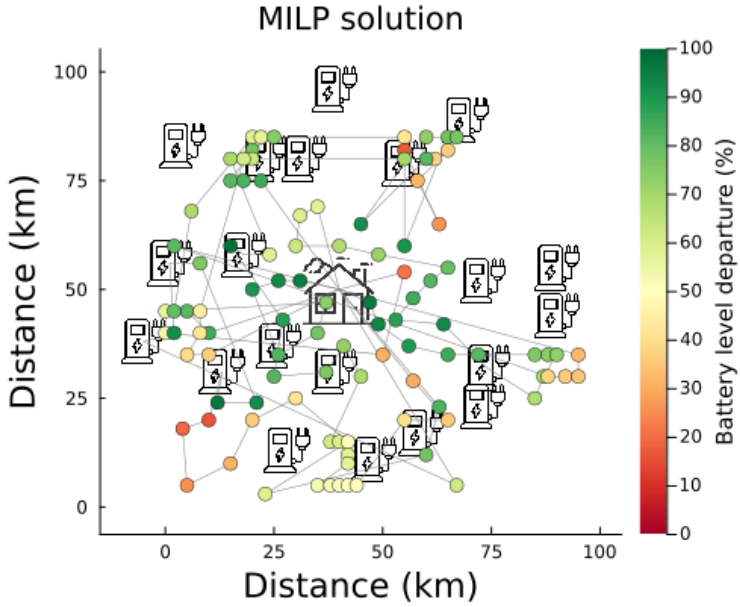

Figure A.25: Solution routes for the C-EVRPTW generated by the MILP approach for the instance `rc203`.

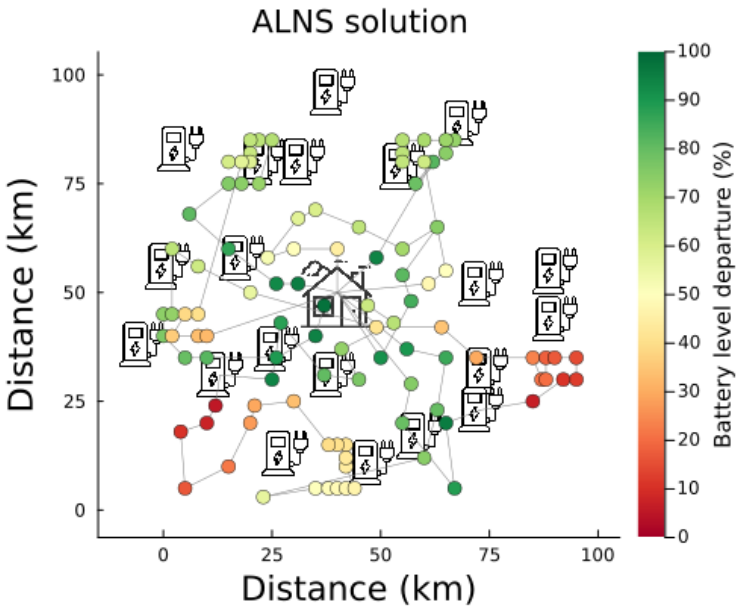

Figure A.26: Solution routes for the C-EVRPTW generated by the ALNS method for the instance `rc203`.

B

ALNS and MILP performance evaluations for longer runs

For the larger datasets, evaluating each instance for 10 minutes is not enough. Due to time restrictions, this was unavoidable and instead three 30 minute runs are showcased, one for each customer distribution type for the C-EVRPTW with the full charging policy. After a 5 minute warm-up, each data instance is run once using the MILP and once using ALNS. The results can be found in Table B.1. The first data instance is chosen such that the MILP approach generates a better solution if given 10 minutes (see Table 4.7) to investigate if ALNS is able to outperform it given more time. The remaining two data instances are randomly chosen from the examined data instances with 100 customers such that one of them had randomly distributed customers and the other had randomly clustered customers. For a fair comparison to the previously obtained results for the 10 minutes runs, the same number of charging station copies was used in the MILP formulation, that is 15.

Table B.1: The average objective value f and the time corresponding to when this solution was found t, for different instances, applied on the C-EVRPTW. The parameter n_{v} is the number of available vehicles. Opt. is set to "Yes" if the solution generated using the MILP approach is optimal and "No" otherwise. The value Δf is the difference between the objective value from the MILP and ALNS, calculated as $\Delta f = \frac{f_{\mathrm{ALNS}} - f_{\mathrm{MILP}}}{f_{\mathrm{MILP}}}$. Ref. is a reference value for a similar model taken from [13]. If ALNS and the MILP gives different average best objective values, the lowest one is marked in bold.

		ALNS		**MILP**				
Dataset	n_{v}	f (km)	t (s)	f (km)	t (s)	Opt.	Δf (%)	Ref.
c201	8	704.08	59.42	**703.35**	463.23	No	0.10	645.16
r102	25	**1489.76**	1697.91	2224.20	0.00	No	-33.02	1495.31
rc203	5	**1028.37**	1581.48	1559.84	1269.91	No	-34.07	1073.98

The objective value as a function of time is visualized in Figures B.1, B.2 and B.3. For the data instance `c201`, it can be seen both from Table B.1 and Figure B.1 that the extra time given does not improve the results since the lowest values are reached within 10 minutes. For the other two data instances, there is small or no improvements for both the MILP approach and ALNS when giving the algorithms more time. This might not be true for all the data instances, but there is enough support for the decision to evaluate the algorithms only on the first 10 minutes.

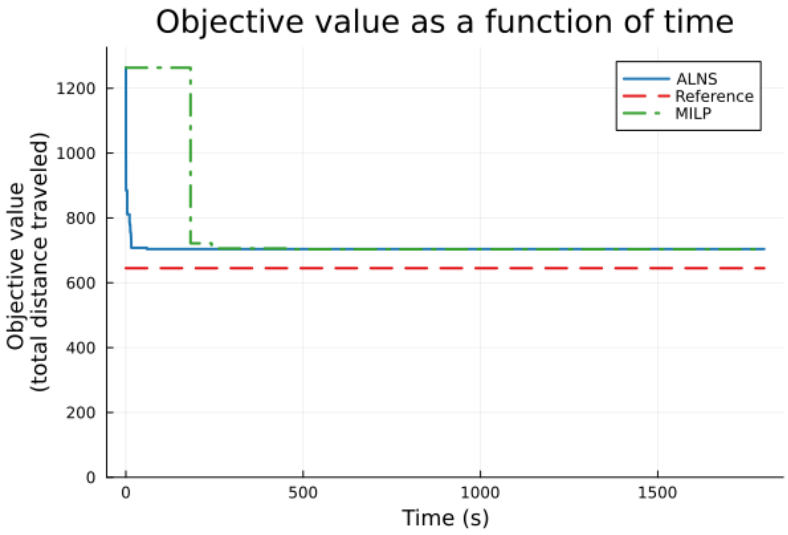

Figure B.1: Objective value over time for instance `c201` when solving the C-EVRPTW using the ALNS method and the MILP approach, along with a reference value representing the best known objective reported in [13]. Note that the reference value is for a similar and not identical model to any of the ones included in this project, and should only be used as an approximate guess.

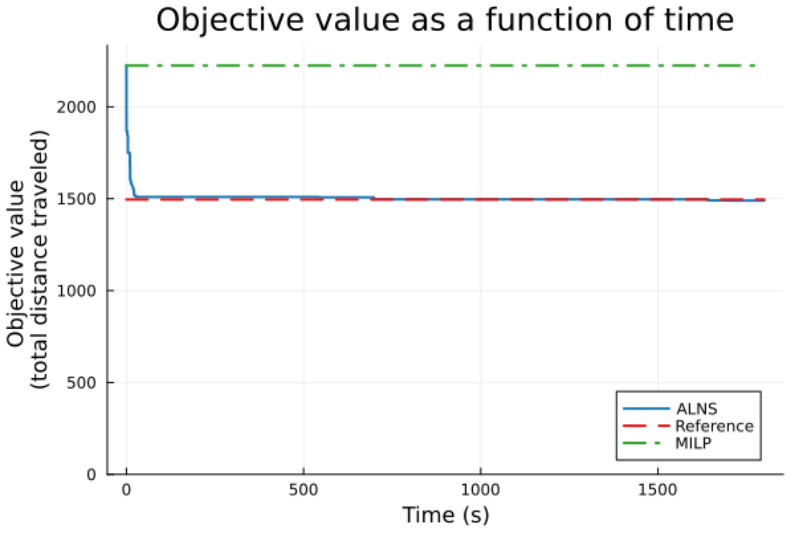

Figure B.2: Objective value over time for instance `r102` when solving the C-EVRPTW using the ALNS method and the MILP approach, along with a reference value representing the best known objective reported in [13]. Note that the reference value is for a similar and not identical model to any of the ones included in this project, and should only be used as an approximate guess.

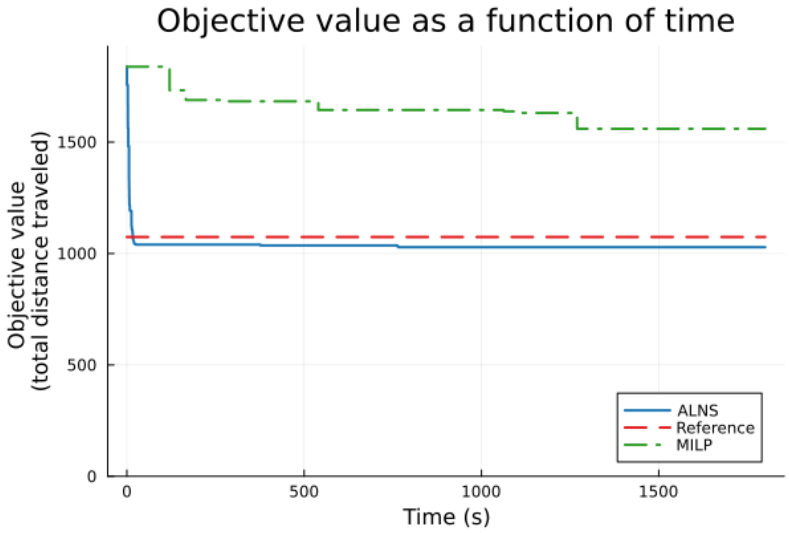

Figure B.3: Objective value over time for instance `rc203` when solving the C-EVRPTW using the ALNS method and the MILP approach, along with a reference value representing the best known objective reported in [13]. Note that the reference value is for a similar and not identical model to any of the ones included in this project, and should only be used as an approximate guess.

C

Alternative initial solution algorithm

The algorithm for the Earliest Time Window heuristic is based on the description presented in [26]. In the implementation developed for this project, customers can be sorted by the start or end of their time windows, or by their demand. Other than this sorting step, the algorithmic steps remain the same. It begins with empty routes and sequentially inserts customers in the order determined by the sorted list. Before a customer is inserted (at the end of the route) the algorithm checks feasibility with respect to time windows, vehicle battery capacity, and weight limits. If it is not possible to insert the customer into the current route, it tries to insert it into the next available route, continuing this process until all routes have been tested. If necessary, charging stations are added to maintain feasibility considering the energy constraints.

Charging stations are inserted using a simplified version of the algorithm described in Section 3.3.1, which considers only the charging station closest to the previous node. If the insertion violates time windows or if the battery is insufficient to reach the charging station, the insertion is deemed unsuccessful and the algorithm terminates. If the insertion succeeds, the process continues until all battery constraints are satisfied across all routes. A failure occurs if either the time windows are violated or there is insufficient battery to reach the nearest charging station. In such cases, the customer responsible for the infeasibility cannot be inserted, and no initial solution can be constructed.

DEPARTMENT OF DEPARTMENT OF MATHEMATICAL SCIENCES
CHALMERS UNIVERSITY OF TECHNOLOGY
Gothenburg, Sweden
www.chalmers.se

CHALMERS
UNIVERSITY OF TECHNOLOGY