
State of Charge-Induced Battery Deterioration in the Electric Vehicle Routing Problem with Time Windows and Partial Recharge Strategies

Egle Sakalauskaite (621819)

A large, stylized, dark blue cursive logo of the word "Erasmus" in a script font.

Supervisor:	BTC van Rossum
Second assessor:	Name of your second assessor
Date final version:	16th June 2024

The views stated in this thesis are those of the author and not necessarily those of the supervisor, second assessor, Erasmus School of Economics or Erasmus University Rotterdam.

Abstract

As the use of electric vehicles becomes increasingly popular, considering their limitations has become a relevant field to study. *Electric Vehicle Routing Problem with Time Windows and Partial Recharge* (EVRPTW-PR) is a variant of *Vehicle Routing Problem with Time Windows* (VRPTW) in which we construct routes for a fleet of electric vehicles (EVs) and seek better solutions by allowing partial recharging to occur. EVs are known for their limited driving range and the need to recharge at special service stations. Studies show that the battery *State of Charge* (SoC) at which recharging occurs can have detrimental effects on the battery's lifespan, leading to premature battery replacement costs in the future if harmful charging patterns are often repeated. Therefore, our goal is to incorporate these costs into the existing solution methods of EVRPTW-PR to evaluate the possible financial loss in cases where they were ignored. We define the problem as a mixed Integer linear program (MILP) and solve it using the *Adaptive Large Neighborhood Search* (ALNS) algorithm, which repeatedly destroys and repairs existing solutions by adding and removing location visits. Our results show that total costs are reduced in most tested instances when incorporating battery degradation into the problem compared to the solutions obtained by minimizing the total distance traveled only.

1 Introduction

The global sales of electric vehicles (EVs) have increased from 320 thousand in 2014 to 14 million in 2023, rapidly replacing conventional vehicles by reaching a market share of 16% in the previous year. It is estimated that by the year 2045, all newly sold vehicles will be zero-emission (Irle, 2024). This trend will continue to impact climate change positively. According to the European Environment Agency (EEA, 2024), as of 2022, domestic transportation has still contributed 23.7% of all greenhouse gas emissions in the EU. However, several challenges are faced when using electric vehicles: the limitations in battery capacity have shortened the driving range of EVs compared to conventional vehicles, and in many areas, finding a charging station at reasonable proximity is a challenge. Additionally, complete recharging takes a few hours, which is much longer than a gas refill (Pasha et al., 2024). Regardless, we should remember that the advantages of using battery-powered vehicles significantly outweigh the drawbacks. Therefore, it is crucial to analyze the negative aspects to understand their impacts thoroughly: this will help address these challenges more effectively and improve the overall efficiency and sustainability of using EVs.



Electric vehicles are increasingly used in practical applications such as public transport and delivery services; therefore, finding ways to reduce costs and improve efficiency is more important than ever. *Electric Vehicle Routing Problem* (EVRP) is an extension of well-known *Vehicle Routing Problem* (VRP), in which EVs are used instead of traditional combustion engine vehicles. The goal is to determine the most efficient routes for a fleet of electric vehicles to serve several customers. In most cases, the efficiency of the routes is measured by the total driving distance and duration, but other factors may be regarded, such as the total number of vehicles used. Recharging or battery swapping is a crucial element in such problems, which presents an extra level of complexity than VRP.

In this paper, we explore the variant of the EVRP presented by Keskin and Çatay (2016),

known as the *Electric Vehicle Routing Problem with Time Windows and Partial Recharge* (EVRPTW-PR). The objective is to route a homogeneous fleet of EVs while minimizing the total driving distance. In addition to the usual EVRP constraints, such as vehicle loading and battery capacity, customer service is restricted by delivery time windows. Since the charging duration is proportional to the battery level increase, allowing partial recharging improves flexibility in visiting customers before their deadlines. Consequently, the authors propose that vehicles be fully charged when leaving the depot. If a vehicle is recharged at least once during its route, it returns to the depot with the battery empty, as any additional recharging is time-inefficient.

However, there is scientific proof that allowing the battery to (nearly) fully discharge and charging it to full capacity can negatively affect its performance (BU, 2023). Using the battery at a state of charge (SoC) between 65 and 75% preserves its capacity the most. However, this is not very practical, especially considering EVs, since it would significantly reduce the already limited driving distance between charges. The best trade-off between preserving the battery capacity and autonomous use seems to be the strategy that keeps the battery SoC between 25 and 85%. According to König et al. (2021), batteries contribute to a large share of the total price of EVs. Therefore, failing to preserve it can lead to substantial financial losses.

To the best of our knowledge, battery deterioration has yet to be explored well in the context of EVRP. We aim to improve on this by first replicating the results of Keskin and Çatay (2016) of solving EVRPTW-PW without considering battery deterioration. Then, we will expand on their research by introducing battery deterioration costs in obtaining the solutions and make a comparative analysis to examine the impact battery deterioration has on the quality of this problem's solution. When comparing with solutions in which battery degradation costs were only accounted for after the solution was obtained, we obtained better results in most tested instances if these costs were considered while obtaining the solution.

This paper is organized as follows: First, Section 2 gives an overview of the development of EVRP and some existing problem extensions. Second, Section 3 describes EVRPTW-PR in detail by presenting a MILP formulation and extends it by including penalties for violating the recommended SoC. Next, Section 4 presents the algorithm of ALNS together with all insertion and removal operations it uses in this paper. Section 5 compares our findings with the original paper and the difference in results once the effects of SoC are considered. Finally, Section 6 gives a summary of our findings and discusses future research opportunities.

2 Literature review

The *Vehicle Routing Problem* (VRP) was first proposed by Dantzig and Ramser (1959). VRP focuses on finding the shortest route while delivering goods to numerous customers by a fleet of vehicles. Since then, many variations of the problem have been explored, incorporating different constraints that can better reflect the complexity of cases in real life. Particularly, Solomon (1987) has addressed the scheduling restrictions in VRP when the deliveries could only be performed during specific time windows. leading to the introduction of *Vehicle Routing Problem with Time Windows* (VRPTW).

By the late 20th century, concerns about climate change grew and began to be reflected in operations research studies that followed. Conrad and Figliozzi (2011) presented us with a

Recharging Vehicle Routing Problem (RVRP), in which vehicles are allowed to recharge at the customer locations. Their approach not only minimized the total costs but also the number of vehicles required. Just as in Keskin and Çatay (2016), the delivery time window restrictions were also considered, and partial recharge was optional, but only to a fixed level, such as up to 80% of the battery's capacity. However, in practical scenarios, vehicle recharging is typically conducted at locations other than customer sites. A good implementation of this can be found in a publication of Wang and Cheu (2013), in which operations of the EV taxi fleet were explored, focusing only on the pre-booked trips. The authors used a two-phase heuristic approach by first constructing an initial solution using one of the nearest-neighbor, sweep, and earliest-time window insertion heuristics and refining it using tabu search. Additionally, they proposed three distinct recharging strategies, each offering different driving ranges, and evaluated these against a full charging scheme.

The study of Schneider, Stenger and Goeke (2014) introduced the *Electric Vehicle Routing Problem with Time Windows* (EVPTW) to address new environmental regulations that promote using electric vehicles for last-mile deliveries. To solve this problem, the authors developed a hybrid heuristic combining variable neighborhood search with tabu search techniques. It is evident that this article was an inspiration for Keskin and Çatay (2016). However, there was one key difference: Schneider et al. (2014) did not address the possibility of partial recharging. Since charging EVs takes time, partial recharging could be the key to more efficient schedules. This idea is backed up by Felipe, Ortuño, Righini and Tirado (2014). In their research, the authors explored a variation of a *Green Vehicle Routing Problem* (GVRP) with electric vehicles. The study considers not only the optimal routing schedule but also the specific amounts of energy to be recharged and the charging technology employed. A constructive and deterministic local search heuristics integrated within a non-deterministic Simulated Annealing framework was proposed. It was demonstrated that partial recharges can lead to considerable cost and energy savings and assist in maintaining feasibility in more complex scenarios.

After publishing Keskin and Çatay (2016), the authors continued their research on EVRPTW with the ALNS approach by exploring a different challenge that electric vehicles face. Due to long charging durations, some vehicles must wait at the recharging stations to use the facility (Keskin, Laporte & Çatay, 2019). This may introduce additional costs if driver salary is involved and affect the routes' feasibility when delivery time windows are present. Since these waiting times are dynamic in the real world, the time window constraints are relaxed to allow late arrivals. However, these scenarios result in additional penalty costs. Another drawback of electric vehicles is that their energy consumption rate fluctuates with the load they carry (Rastani & Çatay, 2021). According to the article, if the effects of load weight are ignored, the total operating costs might increase to up to 31%. Therefore, incorporating these concerns regarding electric vehicles into the EVRP formulations allows a more accurate representation of real-life problems and better results.

To the best of our knowledge, little research has been done regarding the effects of SoC on batteries in EVRP, but some studies explore this. Cataldo-Díaz, Linfati and Escobar (2022) incorporate this by first restricting the battery depletion below 25% of its capacity. Later, charging the battery to the full capacity is also constrained, allowing the vehicles only to maintain

their battery in the recommended interval between 25% and 85%, which limits the autonomy of the vehicle to only 60%. They investigate the results of the widely used data generated by Schneider et al. (2014) by solving MILP to optimality. While such a strict approach makes a great effort to prevent SoC from contributing to battery deterioration, it highly shortens EVs' already limited driving distance. In cases where time windows of deliveries must be considered, this could greatly affect the feasibility region, resulting in more vehicles needed and larger overall driving distances.

The adverse effects of both high and low SoC on battery wear were also explored by F. Guo, Zhang, Huang and Huang (2022). The battery degradation costs were included in the Location Routing Problem (LRP) objective function to determine the optimal locations of recharging stations. The *Wear Cost* (WC) they used was inspired by the work of Han, Han and Aki (2014). To calculate these costs, some information on the batteries is required: the purchase cost of the battery, the cycle life and depth of discharge (DOD), and the achievable cycle count (ACC).

A different strategy to consider battery deterioration in electric vehicles was proposed by İslim and Çatay (2024). This recent study introduced *Electric Traveling Salesman Problem with Time Windows and Battery Degradation* (ETSPYW-BD). They used battery wear cost as described by Han et al. (2014) and solved the problem with two approaches: they constructed feasible customer visit sequences with *Variable Neighborhood Search* (VNS) and then determined exact charging quantities by solving a *Fixed-Tour Vehicle-Charging Problem with battery degradation* (FTVCP-BD). Their results showed that considering these costs can improve the solution by up to 11%.

Inspired by these works, we wish to contribute to bridging the gap between the research in EVRP and battery degradation by incorporating the effects of State of Charge into our models.

3 Problem description

EVRPTW-PR can be defined as follows: N customers with known demands are served by a homogeneous fleet of EVs with fixed load capacities. Each customer has a predetermined delivery time window and service duration, while the vehicles have limited driving ranges and, therefore, may need to recharge at one of the charging stations. The charging process takes a certain amount of time that depends on the desired increase of the battery level. All vehicles start and end their route at the depot, and partial recharging is allowed in this particular variant of the EVRPTW problem. Each EV departs from the depot with the battery fully charged, and if it gets charged at least once, it returns to the depot with a battery fully depleted. Otherwise, it may return with any battery level remaining. The goal is to construct vehicle routes that minimize the total distance traveled while satisfying all the customers' demands without violating any of the restrictions described above.

The mathematical formulation of the problem is as presented by Keskin and Çatay (2016). Let $V = U \cup R'$ be the set of all vertices, where $U \in \{1, \dots, N\}$ represents the customers, and R' is the set of dummy vertices generated to permit several visits to each vertex in the set of R of recharging stations. The depot is denoted by both 0 and $N + 1$. All routes must begin at 0 and end at $N + 1$. For clarity of the constraints, we also introduce $R'_0 = R \cup \{0\}$

$V_0 = V \cup \{0\}$, $V_{N+1} = V \cup \{N+1\}$ and $V_{0,N+1} = V \cup \{0\} \cup \{N+1\}$. The set of arcs $A = \{(i, j) | i, j \in V_{0,N+1}, i \neq j\}$ represents the traveling between any two locations. Therefore, the problem can be summarized by a complete directed graph $G = (V_{0,N+1}, A)$.

Each electric vehicle has a fixed loading capacity C and battery capacity Q . The battery gets recharged at a charging rate of g . Furthermore, let d_{ij} and t_{ij} be the travel distance and travel time between locations $i, j \in V_{0,N+1}$ respectively. When a vehicle travels the arc $(i, j) \in A$, it consumes $h \cdot d_{ij}$ of its remaining battery, where h is the battery consumption rate.

Every customer $i \in U$ has a positive demand q_i ($q_i = 0 \forall i \notin U$), service time s_i ($s_i = 0 \forall i \notin U$) and a time window $[e_i, l_i]$. The service of a customer i cannot begin earlier than e_i or later than l_i , but it may end later.

There are several decision variables to consider. In order to track the vehicle routes, we introduce x_{ij} , which is equal to 1, if arc $(i, j) \in A$ is traversed, and 0 otherwise. Additionally, for every location $i \in V$, we define τ_i , the arrival time to the location, and u_i , the remaining cargo level upon arrival. Since partial recharging is allowed, we have two sets of decision variables to track the battery state of charge: y_i and Y_i , which track the battery state upon arrival to and departure from location i , respectively.

The problem can be defined as a mixed-integer program as follows:

$$\min \sum_{i \in V_0, j \in V_{N+1}, i \neq j} d_{ij} x_{ij} \quad (1)$$

$$\sum_{j \in V_{N+1}, i \neq j} x_{ij} = 1 \quad \forall i \in U \quad (2)$$

$$\sum_{j \in V_{N+1}, i \neq j} x_{ij} \leq 1 \quad \forall i \in R' \quad (3)$$

$$\sum_{i \in V_0, i \neq j} x_{ij} - \sum_{i \in V_{N+1}, i \neq j} x_{ji} = 0 \quad \forall j \in V \quad (4)$$

$$\tau_i + (t_{ij} + s_i)x_{ij} - l_0(1 - x_{ij}) \leq \tau_j \quad \forall i \in U \cup \{0\}, \forall j \in V_{N+1}, i \neq j \quad (5)$$

$$\tau_i + t_{ij}x_{ij} + g(Y_i - y_i) - (l_0 + gQ)(1 - x_{ij}) \leq \tau_j \quad \forall i \in R', \forall j \in V_{N+1}, i \neq j \quad (6)$$

$$e_i \leq \tau_i \leq l_i \quad \forall i \in V_{0,N+1} \quad (7)$$

$$0 \leq u_j \leq u_i - q_i x_{ij} + C(1 - x_{ij}) \quad \forall i \in V_0, \forall j \in V_{N+1}, i \neq j \quad (8)$$

$$0 \leq u_0 \leq C \quad (9)$$

$$0 \leq y_j \leq y_i - (h \cdot d_{ij}) + Q(1 - x_{ij}) \quad \forall i \in U, \forall j \in V_{N+1}, i \neq j \quad (10)$$

$$0 \leq y_j \leq Y_i - (h \cdot d_{ij}) + Q(1 - x_{ij}) \quad \forall i \in R'_0, \forall j \in V_{N+1}, i \neq j \quad (11)$$

$$y_i \leq Y_i \leq Q \quad \forall i \in R'_0 \quad (12)$$

$$x_{ij} \in \{0, 1\} \quad \forall i \in V_0, \forall j \in V_{N+1}, i \neq j \quad (13)$$

The objective function 1 aims to minimize the total distance traveled. The connectivity of the customers and recharging stations are ensured by constraints 2 and 3, respectively. Constraints 4 are for flow conservation, while constraints 5 and 6 enforce the time feasibility between two visits when leaving from a customer location (or the depot) and a recharging station separately. The time window restrictions are represented by constraints 7. Additionally, constraints 5 - 7 work as a sub-tour elimination. Demand satisfaction of all customers is ensured by constraints

8 and 9. Furthermore, constraints 10 - 12 track the battery charge and stop it from dropping below 0. Finally, constraints 13 define decision variables as binary.

3.1 Battery degradation costs

When battery degradation (BD) is considered, the problem formulation remains almost identical, as stated above. However, two new types of decision variables are introduced, and the objective function is expanded to include the additional BD costs. İslim and Çatay (2024) suggest that an efficient way to track battery degradation costs is to account for them while the vehicle is charging. Therefore, we introduce non-negative decision variables z_i^L and z_i^H , representing the energy units charged in station $i \in R'$ in intervals $0 - lb\%$ and $ub - 100\%$, respectively. For simplicity, we define $LB = \frac{lb*Q}{100}$ and $UB = \frac{ub*Q}{100}$.

$$z_i^L \geq LB - y_i \quad \forall i \in R' \quad (14)$$

$$z_i^H \geq Y_i - UB \quad \forall i \in R' \quad (15)$$

$$z_i^L \geq 0 \quad \forall i \in R' \quad (16)$$

$$z_i^H \geq 0 \quad \forall i \in R' \quad (17)$$

BD of charging at the depot still needs to be accounted for. Since the previous formulation assumed that all vehicles leave and return to the depot with identical battery charges, the variables Y_0 and y_{n+1} are unsuitable for accurately computing the deterioration costs. Therefore, we introduce the non-negative variable $z_{0,i}^H$, which tracks the units of energy needed to be charged in the depot in the interval $UB - Q$ for the vehicle that starts its route by visiting location $i \in V$. Similarly, $z_{i,(n+1)}^L$ tracks the units of energy needed to be charged in the depot in the interval $0 - LB$ for the vehicle that ends its route by visiting location $i \in V$. Instead of using the battery level of the depot itself, we consider the battery level of the location that is adjacent to it in the route and the distance between it and the depot:

$$z_{0,i}^H \geq y_i + d_{0i} - UB - Q(1 - x_{0i}) \quad \forall i \in V \quad (18)$$

$$z_{0,i}^H \leq y_i + d_{0i} - UB + Q(1 - x_{0i}) \quad \forall i \in V \quad (19)$$

$$z_{i,(n+1)}^L \geq LB - (Y_i - d_{0i}) - Q(1 - x_{i(n+1)}) \quad \forall i \in V \quad (20)$$

$$z_{i,(n+1)}^L \leq LB - (Y_i - d_{0i}) + Q(1 - x_{i(n+1)}) \quad \forall i \in V \quad (21)$$

$$z_{0,i}^H \geq 0 \quad \forall i \in V \quad (22)$$

$$z_{i,(n+1)}^L \geq 0 \quad \forall i \in V \quad (23)$$

For Constraints 18-19, we use the expression $Q(1 - x_{0i})$ to disable them for locations $i \in V$ that are not visited right after leaving the depot. For Constraints 20-21, $Q(1 - x_{i(n+1)})$ works in a similar way for locations $i \in V$ that are not visited right before returning to the depot.

We define \bar{W}^L and \bar{W}^H as the average wear cost per unit energy charged below LB and above UB of the battery capacity, respectively. To keep the objective function consistent, $W^L = \bar{W}^L/c$

and $W^H = \overline{W}^H/c$ are these costs scaled to be proportional to the cost c of traveling one unit of distance. Then, the objective function becomes:

$$\min \sum_{i \in V_0, j \in V_{N+1}, i \neq j} d_{ij} x_{ij} + \sum_{i \in R'} (W^L z_i^L + W^H z_i^H) + \sum_{i \in V} (W^H z_{0,i}^H + W^L z_{i,(n+1)}^L) \quad (24)$$


The first summation is the same as before and accounts for the costs generated by the distance traveled. The second summation represents the battery wear costs when charging at the stations while traveling the route, and the third summation is for the battery wear costs when charging at the depot.

4 Methodology

Keskin and Çatay (2016) propose an *Adaptive Large Neighborhood Search algorithm* to solve the EVRPTW-PR. The ALNS algorithm was introduced by Ropke and Pisinger (2006) and was designed to dynamically adjust and improve the search process by allowing the algorithm to visit infeasible solutions. In doing so, the algorithm is less likely than its predecessor LNS to be stuck in a local minimum. According to the authors, ALNS successfully improved the previously known best solutions for more than 50% of the examined problems. Therefore, ALNS has been widely applied in various routing problems. The following sections follow the methodology used by Keskin and Çatay (2016). Section 4.5 summarizes the adjustments for this approach that would prioritize solutions that keep battery SoC at $LB - UB\%$, which we will refer to as recommended Soc.

4.1 Overview of ALNS approach

4.1.1 ALNS procedure



Algorithm 1 provides the pseudo-code for the ALNS procedure. The first step is constructing the initial solution described in Section 4.1.2. Then, the algorithm runs for a fixed number of iterations, destroying the current solution by removing individual customer or service nodes from a specific route or an entire route from the solution. Then, it repairs the solution by adding charging stations, removing customers from existing routes, or creating new ones. It does so by using four classes of algorithms: *Customer Removal* (CR), *Customer Insertion* (CI), *Station Removal* (SR), and *Station Insertion* (SI).

Each class of algorithms includes a few different approaches, and they are selected based on a probability that corresponds to their past performance in enhancing the solution. The performance of algorithm a is measured by adaptive weight w_a and a score π_a . Initially, all weights are equal, and all scores are set to 0. When an algorithm is used, its score is increased by σ_1 , if the new best solution is found, σ_2 , if a better solution than the previous is found, σ_3 , if a worse solution than the previous is found, but it is accepted using the simulated annealing rule and 0 otherwise. Therefore, higher values of π_a indicate better performance. The performance weights are updated at the end of every segment s which lasts N_C iterations for customer, and N_R iterations for recharging station insertion/removal algorithm a according to the formula: $w_a^{s+1} = w_a^s(1 - \rho) + \rho\pi_a/\theta_a$, where θ_a is the number of times it was used during segment s and

ρ is the roulette wheel parameter. Then, the scores π_a are set to 0, and the probabilities of using each algorithm a in the next segment are determined by normalizing the weights of every algorithm class separately.

A worse than a previous solution can still contribute to the used algorithm's score if it is accepted according to the *Simulated Annealing* (SA) technique: if the solution uses fewer vehicles or the solution uses the same number of vehicles but results in a shorter total distance, then it is accepted. Additionally, suppose the number of vehicles is the same, but the distance is greater. In that case, the solution is accepted with a probability given by $e^{-f(X_{New}-f(X_{Current}))/T}$, where X_{New} and $X_{Current}$ are the new and current best solutions respectively, $f(X)$ is the total distance of solution X and T is the current temperature, which is initially set to T_{init} and is reduced with every iteration by a factor of a cooling rate parameter $0 < \epsilon < 1$. T_{init} is set based on the initial temperature control parameter μ , ensuring that a solution which is $\mu\%$ worse than the initial solution has a 50% chance of being accepted

Algorithm 1 ALNS algorithm

```

1: Generate an initial solution (Algorithm 2)
2:  $j \leftarrow 1$ 
3: while Stop-criterion met do
4:   if  $j \bmod N_{SR} \equiv 0$  then
5:     Select SR algorithm and remove stations
6:     Select SI algorithm and repair solution
7:   else if  $j \bmod N_{RR} \equiv 0$  then
8:     for  $n_{RR}$  iterations do
9:       Select RRR or GRR algorithm and remove customers
10:      Select CI algorithm and repair solution (same as 14 - 17)
11:    end for
12:   else
13:     Select CR algorithm and remove customers
14:     Select CI algorithm
15:     while destroyed solution infeasible do
16:       Use selected CI algorithm to repair solution
17:       Perform Greedy Station Insertion
18:     end while
19:   end if
20:   Using SA criterion to accept/reject the solution
21:    $j \leftarrow j + 1$ 
22:   if  $j \bmod N_C \equiv 0$  then
23:     Update adaptive weights of CR and CI algorithms
24:   else if  $j \bmod N_S \equiv 0$  then
25:     Update adaptive weights of SR and SI algorithms
26:   end if
27: end while

```

4.1.2 Construction of an initial solution

The pseudo-code of the algorithm for constructing an initial solution is presented in Algorithm 2. We start by initializing the current route by selecting the customer closest to the depot and adding them to the route. We proceed by iteratively attempting to add new customers

to the current route until all customers are routed. We start every iteration by determining the insertion costs of each unrouted customer to every possible insert position in the current route. According to the exact insert position, arrival times for all customers can be calculated. Therefore, the insertion is considered possible if the route after insertion does not violate any of the routed customers' time windows and if the loading capacity can still satisfy the additional customer. If at least one of such insertions exists, we choose the customer insertion that results in the shortest total travel distance of the route. Otherwise, we initialize a new route by selecting an unrouted customer closest to the depot, adding them to this route, and proceeding with the next iteration. Suppose the battery at any point of the current route would drop below 0. In that case, we add a visit to the recharging station by performing a *Greedy Station Insertion*, described in Section 4.4.2.

Algorithm 2 Initial solution construction

```

1: Start a new route with the customer closest to the depot
2: while all customers are served do
3:   Calculate insertion costs of all unserved customers to the current route
4:   if no customer can be added then
5:     Start a new route with the unserved customer closest to the depot
6:   else
7:     Select the customer which increases the distance least and make the insertion
8:   end if
9:   if a recharging station is needed then
10:    Perform Greedy Station Insertion
11:   end if
12: end while

```

4.2 Removal algorithms

4.2.1 Customer removal

For *Customer Removal* (CR), we use several algorithms as described by Keskin and Çatay (2016) and inspired by Demir, Bektaş and Laporte (2012) and Emeç, Çatay and Bozkaya (2016). The number of customers removed depends on the total number of customers n_c and is a random number $\gamma \sim \mathcal{U}(\underline{n}_c, \overline{n}_c)$. The removed customers are then inserted into the removal list \mathcal{L} .

Random Removal. Customers are selected randomly with equal probabilities to be chosen.

Worst-Distance and *Worst-Time Removal.* Customers that result in higher costs have a higher probability of being chosen. For *Worst-Distance*, the cost is $d_{ji} + d_{ik}$, where i is the customer under consideration, j and k are their successor and predecessor, respectively. For *Worst-Time*, it is simply $|\tau_i - e_i|$. The customer with $\lfloor |\gamma| \lambda^K \rfloor^{th}$ highest cost is chosen, where $\lambda \in [0, 1]$ is random and $K \geq 1$ is the worst removal determinism factor.

Shaw Removal. This operation aims to remove a set of customers that are considered similar. It starts by randomly selecting customer i . Then, the relatedness measure for all remaining non-removed customers is calculated as $R_{ij} = \theta_1 d_{ij} + \theta_2 |e_i - e_j| + \theta_3 l_{ij} + \theta_4 |q_i - q_j|$. l_{ij} is a binary variable that takes the value -1 if customers i and j are in the same route and 1 otherwise, while $\theta_1 - \theta_4$ are the Shaw parameters. Then, all customers j are ranked according to R_{ij} in

descending order, and $\lfloor |\gamma| \lambda^\eta \rfloor^{th}$ customer is chosen to be removed, where η stands for the Shaw removal determinism factor.

Proximity, Time and Demand-Based Removal. Same as *Shaw Removal* but $\theta_1, \theta_2, \theta_4$ is equal to 1 respectively while all the other Shaw parameters are set to 0.

Zone Removal. Customers belonging to the same Cartesian plane zone get removed at once. First, corner points of the area are determined. Then, customers are divided into n_z zones, a zone is randomly selected and all customers belonging to the zone are removed.

Additionally, *Random Route Removal* and *Greedy Route Removal* as described in Section 4.3 can be selected as CR operations.

Sometimes, removing a customer results in some charging of the EV no longer being necessary. To eliminate this inefficiency, we introduce two more algorithms: *Remove Customer with Preceding Station* (RCwPS) and *Remove Customer with Succeeding Station* (RCwSS), in which not only a customer from the removal list is removed, but also the station that is preceding or succeeding them is if such exists.

4.2.2 Station removal

For every N_{SR} iteration, station removal (SR) is performed, followed by station insertion operation. Similarly to γ in customer removal, σ stations are removed, and this number depends on the number of visits to the stations in the current solution. Again, several different algorithms are used to accomplish this. *Random Station* and *Worst Distance Station Removal* work Similarly as *Random* and *Worst Distance Removal* for CR case. In *Worst-Charge Usage Station Removal*, stations visited with high battery levels get removed, whereas in *Full Charge Station Removal*, stations where the vehicle is charged to its capacity are removed.

4.3 Route removal

In addition to these removal algorithms, the ALNS approach additionally aims to reduce the number of routes by performing RRR or GRR algorithm every N_{RR} iteration for n_{RR} times. These algorithms remove the entire route simultaneously, reducing the number of electric vehicles needed. After, the solution is repaired with CI algorithms (Section 4.4.1).

Random Route Removal (RRR). ω routes are chosen randomly, and all the customers in those routes get removed. The number of routes chosen ω is a random number between 10% and $m_r\%$ of the total number of routes.

Greedy Route Removal (GRR). Similar to RRR, but instead of routes being chosen randomly, ω routes with the fewest customers are selected for removal.

4.4 Insertion algorithms

4.4.1 Customer insertion

After some customers are removed, they must be inserted back into the solution to restore feasibility. The algorithms that do this are the following:

Greedy insertion. Every removed node gets inserted into the cheapest possible position. We do this by computing all possible insertion costs for every customer $i \in \mathcal{L}$ as $d_{ji} + d_{ik} - d_{jk}$,

where j is the preceding and k is the succeeding customer. The customer with the lowest possible insertion cost gets chosen first.

Regret- k Insertion. It works similar to *Greedy Insertion*, except for each customer $i \in \mathcal{L}$ we calculate the difference between the best and k^{th} best insertion costs and insert the customer which has the highest difference between the two first. Our application will use *Regret-2* and *Regret-3*.

Time-Based Insertion. Same as *Greedy Insertion*, but the costs are now calculated according to the increase in route travel time

Zone Insertion. Same as *Time-Based Insertion*, but only a part of all routes in the solution are considered. Just like in *Zone Removal*, the solution is split into zones; one of them gets selected randomly, and only the routes within that zone may be expanded by the insertion.

After a customer insertion, the solution might become infeasible with respect to the battery state of charge. In such case, *Greedy Station Insertion* (See section 4.4.2) is performed.

As mentioned in 3, the vehicle starts the route fully charged, and if the recharging station is visited at least once, it returns to the depot with an empty battery. This rule also influences the recharge amount and, consequently, the battery state of charge. Let us summarize how different scenarios affect these values.

In case the vehicle is recharged only once, then there are two possible outcomes: (i) if a customer is added before the visit to the charging station, this insertion affects only the state of charge upon arrival at the station; (ii) if a customer is added after the visit to the station, the recharge amount is increased so that the EV returns to the depot with an empty battery. On the other hand, if the vehicle is recharged multiple times within the route and (iii) if a customer is added before the visit to the station, the procedure is the same as (i); (iv) if the customer is added after the visit to the first station, the recharge amount at the last station is increased so that the EV returns to the depot with an empty battery. However, if this violates any service time windows, a charge amount may be increased at a previous station instead.

4.4.2 Station insertion

When some charging stations are removed, the vehicle will likely no longer have enough battery charge to reach the next station or the depot. Therefore, we identify the location at which the battery charge drops below 0, and we restore feasibility by adding a visit to the station prior to the route. Remember that a station inserted may differ from the one previously removed, and each charging station may be visited multiple times. These three approaches may insert stations:

Greedy Station Insertion. Similar to *Greedy Insertion*, but instead of checking all possible positions, the trip to the station is added right before arriving at a location where the battery would become negative. If this is unfeasible, insertion in earlier positions is attempted.

Greedy Station Insertion with Comparison. Similar to *Greedy Station Insertion*, but this solution is additionally compared to the one in which we insert a station one position earlier and choose the cheaper option. If both options are infeasible, we apply *Greedy Station Insertion* for earlier positions.

Best Station Insertion. Similar to *Greedy Station Insertion with Comparison*, but not just

two positions prior to the visit to a location in which the battery would become negative are considered, but all positions since the depot or the last visit to the station.

SI approach is reiterated until the battery charge feasibility is restored for the entire solution. If this is impossible, the ALNS algorithm reverts to the last feasible solution and starts a new iteration.

4.5 ALNS adjustments for avoiding battery degradation

The main difference when performing the ALNS approach while considering battery degradation (BD) is that before the shift starts, each vehicle now gets charged at the depot to $ub\%$ of its capacity and is required to return to the depot with $lb\%$ of battery capacity remaining. The vehicle is allowed to charge more at the depot if it's needed before visiting the first customer, and it can violate this recommended SoC while traveling the route. However, the BD costs will be considered together with the driving distance when evaluating the costs of the solution. Figure 1 gives an example of how battery degradation can affect solution quality. The percentage below shows the change in battery level, whereas the numbers above the arrows denote the distances between locations. Assume the battery capacity is 100, the recommended SoC is 25 – 85%, and charging one unit of energy below 25% costs the same as traveling one unit of distance. Figure 1a represents the route before the location is inserted. From Figure 1b, we see how inserting customer **C2** before station **S1** increases the distance by 5 and the battery degradation cost by 5, so the total cost increases by 10. On the other hand, if customer **C2** is inserted after station **S1** as shown in Figure 1c, the distance increases by 7, whereas battery degradation cost stays the same. Even though inserting the customer before the station results in a shorter distance, **the overall cheaper insertion is after the station.**

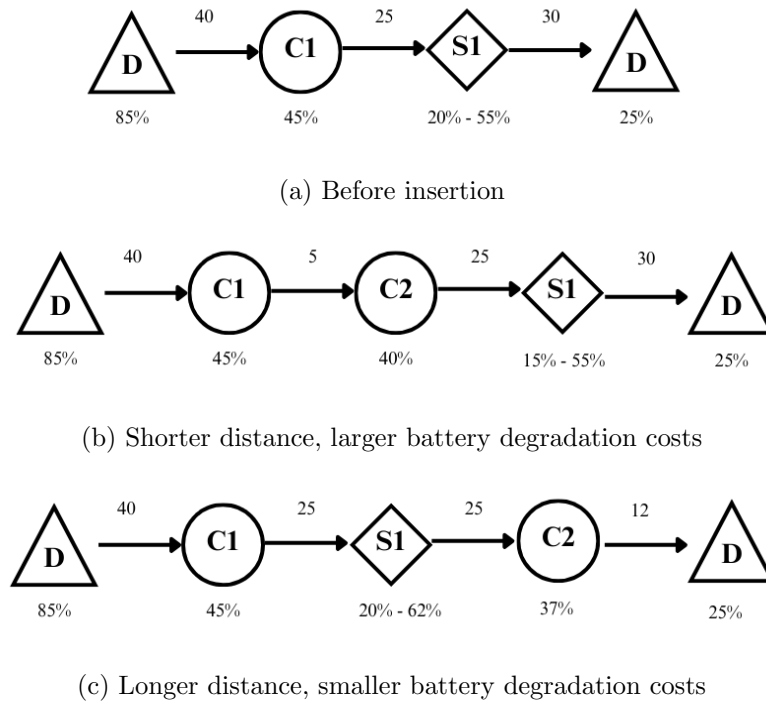


Figure 1: Effects of battery degradation on insertion costs

In order to better avoid battery degradation, we define a new SR operation: *Worst Battery Degradation Station Removal*. It removes the stations that cause the most significant battery degradation costs. The cost for each station can be calculated as $W^L \max(0, LB - y) + W^H \max(0, Y - UB)$. Where y is the battery level upon arrival and Y is the battery level upon departure. This operation is randomly selected as one of the SR operations as described in Section 4.2.2.

To summarize, three fundamental changes are implemented: (1) departure from and return to the depot battery levels correspond to the recommended SoC; (2) **new SR operation added**; (3) battery degradation costs are accounted for when evaluating the quality of a solution.

5 Computational study

In this section, we will present the results of solving the EVRPTW-PR problem in two ways: solving MILP with Gurobi and applying the ALNS approach. First, battery degradation will be ignored when obtaining a solution. Then, we will solve the same problem by including the battery degradation costs to MILP formulation and adjusting the ALNS approach to encourage the battery to be maintained at the recommended level between 25 - 85%. For all solutions, the costs will be calculated to include and exclude the battery degradation costs described in Section 3.1 so we can better compare the results.

The data used in this paper are the widely used instances generated by Schneider et al. (2014), ~~which were obtained through Dominik (2019)~~. The sample has 36 small and 56 large instances, and their name indicates their type. Instances that start with the letter "C" are composed of clustered customers, "R" indicates that customers are randomly distributed, and "RC" is a combination of both. Each type is further divided into two classes, indicated by the first digit in the name after the letter, and both of them differ in vehicle properties and lengths of customer time windows. Large instances include 100 customers and 21 recharging stations, whereas small instances can be split into three equal subsets, each containing 12 problems with 5, 10, and 15 customers randomly drawn from the larger instances. ~~The data does not explicitly provide the distance between locations. Instead, it gives the coordinates.~~ In our study, we will assume the distance to be Euclidean.

We will use all 36 small instances; however, we will reduce the number of large instances to 6. This decision was made after observing that our ALNS implementation takes significantly longer to run (on average 164 min to complete 2500 iterations) compared to the results by Keskin and Çatay (2016). For the same reason, the number of iterations performed for large instances was decreased from 25000 to 2500. To avoid complexity, we will not be tuning the parameter values ourselves. Instead, we will use the values obtained by Keskin and Çatay (2016). However, in order to adapt to the decreasing of iterations, we will scale some of them down: the parameters that determine the frequency at which weight updating (N_C and N_S), route removal (N_{RR}) and station removal (N_{SR}) occur were reduced ten times. As a result, the number of times these parts of the ALNS algorithm were executed in a single run stayed the same. However, the customer removal and insertion were repeated approximately ten times less.

All numerical results presented below are obtained by Intel(R) Core(TM) i5-9300H CPU @ 2.40GHz computer with 16 GB of RAM. The MILP formulation is implemented using Java (ver-

sion 21.0.2) and solved using Gurobi Optimization (Gurobi Optimization, 2023). The runtime limit is set to 7200 seconds. On the other hand, ALNS is implemented using Python 3.12.

5.1 EVRPTW-PR without battery degradation

5.1.1 Numerical results for small-size instances

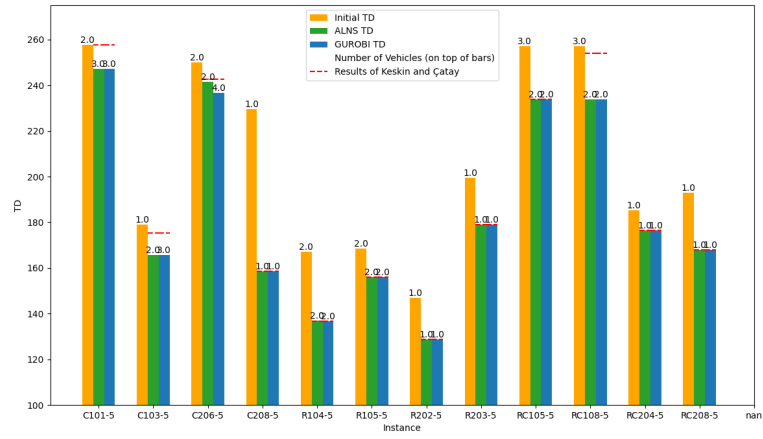
Figure 3 summarizes the solutions for instances with 5, 10, and 15 customers, respectively. The findings are grouped per instance. All bars represent the total distance (TD). In each group, the TD of the initial solution constructed by Algorithm 2 is represented in orange, the best TD obtained through ALNS is in green, and the blue bar corresponds to the best TD obtained with Gurobi. The number on top of each bar is the number of vehicles used in that solution, while the dashed red lines are the corresponding TD result of Keskin and Çatay (2016). Note that we used Gurobi, and the authors used CPLEX, but we will ignore this difference since both optimization tools are similar. As we can see, the results are highly correlated, but there are a few outliers: in some cases, both our ALNS and Gurobi approach outperformed the results of Keskin and Çatay (2016). Since we obtained the data only through a third party, this could be explained by the sample data having minor differences in those instances. In other cases, Gurobi provides nearly identical results, which was expected since most of the instances are solved to optimality, and the remaining ones were terminated after reaching the runtime limit with a relatively low solution gap. Most ALNS cases have the same or highly similar results. However, as the instance size grows, the number of instances that performed worse in ALNS increases. This could be explained by some of our insertion algorithms being stricter at rejecting non-feasible solutions in some cases. Regardless, our ALNS algorithm could significantly decrease the initial solution’s costs. On average, the initial solution was improved by 10.1% for instances with 5, 20.6% for instances with 10, and 25.1% for instances with 15 customers. All exact values and the runtime for each instance can be found in Appendix A Table 3.

5.2 Numerical results for large-size instances

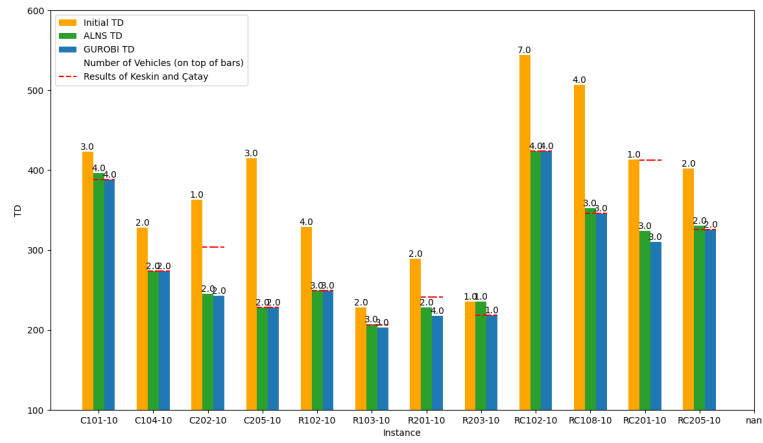
Table 1 summarizes the results of some of the large-size instances obtained by running the ALNS algorithm for 2500 iterations. We first compare the solution’s total distance (TD) with the one obtained initially by Algorithm 2. Our algorithm improved the solution by up to 18.5% and, on average, by 10.7%. $\Delta\%$ shows the difference between our best-found solution and the one obtained by Keskin and Çatay (2016). Our solutions for the large-size instances are still approximately 25% worse. This was expected since we ran the algorithm for ten times less iterations. Additionally, the parameters used were tuned for performing the algorithm for 25000 iterations; therefore, re-tuning them might give better results.

5.3 EVRPTW-PR with battery degradation

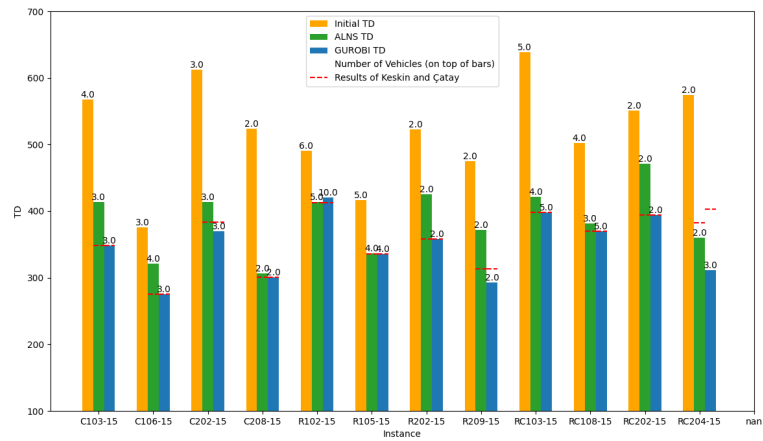
Battery degradation is a complicated process, and the exact cost of it heavily depends on numerous aspects, such as the price and design of the battery, and even other factors that also cause battery degradation, such as the temperature the battery operates in or the voltage of charging (J. Guo, Li, Pedersen & Stroe, 2021). This means many assumptions on these attributes would



(a) 5 customers



(b) 10 customers



(c) 15 customers

Figure 2: Total distance comparison

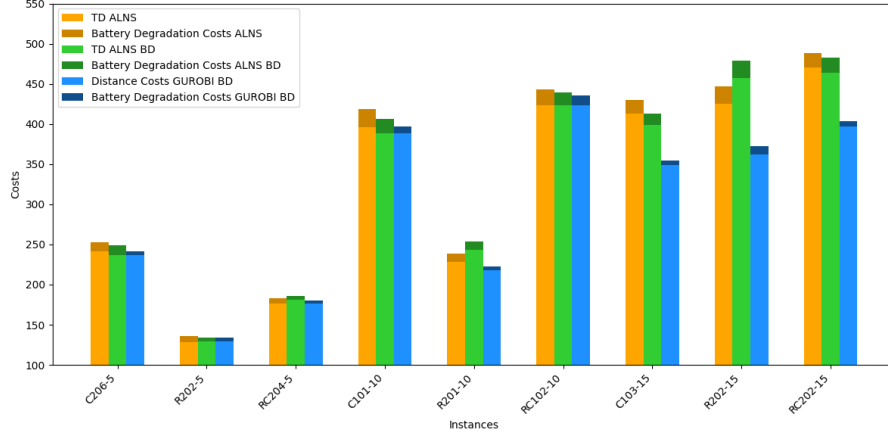
instance	vehicles init	TD init	vehicles 2500	TD 2500	Improvement %	$\Delta\%$	runtime
C101-100	14	1541.01	13	1321.33	14.26	-20.44	7384.94
C201-100	5	1214.59	5	989.78	18.51	-36.35	12688.81
R101-100	22	2190.20	23	1996.07	8.86	-16.77	7709.26
R201-100	4	1845.23	6	1701.66	7.78	-25.60	12015.11
RC101-100	19	2222.86	20	2198.11	1.11	-23.35	6293.93
RC201-100	4	2312.84	8	1999.49	13.55	-27.64	13086.93
Average					10.68	-25.03	9863.16

Table 1: Numerical results for the large-size instances after 2500 ALNS iterations

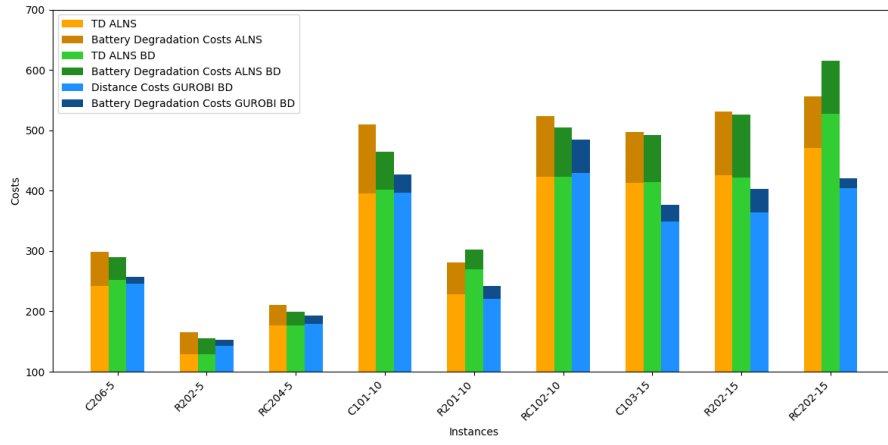
need to be made to evaluate the BD costs for each SoC interval. To make matters even worse, our objective needs to have these costs scaled to the cost of traveling one unit of distance, which may also vary depending on the vehicle’s size, energy prices, and so on. To simplify things, we will try three different pairs of low SoC and high SoC costs: $W^L = \{0.1, 0.5, 2.5\}$, $W^H = \{0.2, 1, 5\}$. The decision was made to make W^H twice as large as W^L since high SoC is considered worse for battery lifespan than low SoC (J. Guo et al., 2021). We also choose the recommended SoC interval to be 25-85% as advised by BU (2023).

5.3.1 Numerical results for small-size instances

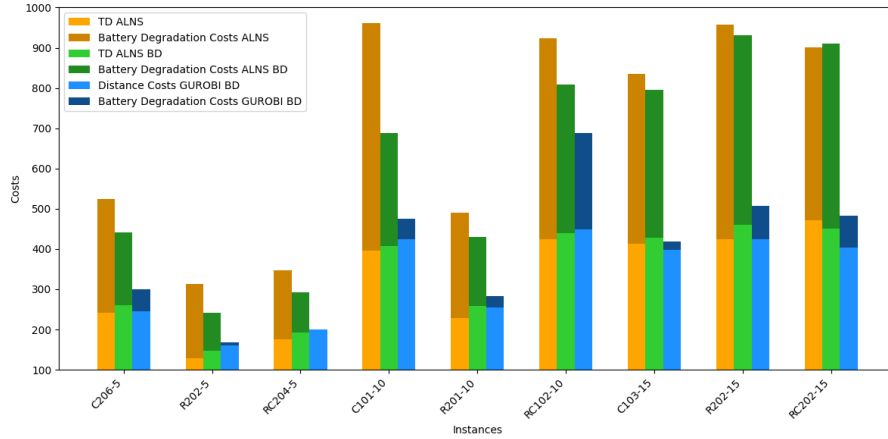
Performing computational study with every instance for each pair of W^L and W^H demands lots of runtime. Therefore we randomly select 9 out of 36 small instances in such a way that instance types ("c", "r" and "rc"), as well as the number of customers in an instance (5, 10 and 15) are evenly distributed. Figures 3a - 3c present results with consideration of 3 different battery degradation cost rates: low BDcost ($W^L = 0.1$, $W^H = 0.2$), medium BD cost ($W^L = 0.5$, $W^H = 1$) and large BD cost ($W^L = 2.5$, $W^H = 5$) respectively. Each bar shows the total costs of one of the solution types, composed of the distance costs (lighter color) and the battery degradation costs (darker color) per instance. ALNS computed as in Sections 4.1 - 4.4.2, but with battery degradation costs computed for the best-found solution after the algorithm terminates is in orange. We will be treating these results as the benchmark. Green represents ALNS with changes of Section 4.5 implemented, and blue bars are for solutions obtained with Gurobi optimization as described in Section 3.1. A better benchmark to compare the results to might have been the solutions obtained with Gurobi for the model described in Section 3. However, the decision variables y_i and Y_i output have proven unreliable for calculating the battery degradation costs since they do not play a role in the optimization function and are not optimized. We wish to overcome this issue by finding a different approach to approximate the battery degradation costs. However, it has proven challenging due to time window constraints and is left for further research. We observed that the problem has resulted in better overall costs for nearly every instance when considering battery degradation while solving. Solutions obtained with Gurobi while considering battery degradation are always better than the benchmark. ALNS with battery degradation, in most instances, performed better than the benchmark; however, the improvement was smaller; there was no significant improvement for the small BD costs, only an average of 1% improvement for the medium BD costs and 12.6% improvement for the large BD costs. This makes sense because from the results in Section 5.1, we saw that even though our ALNS algorithm can



(a) $W^L = 0.1, W^H = 0.2$



(b) $W^L = 0.5, W^H = 1$




(c) $W^L = 2.5, W^H = 5$

Figure 3: Total costs with battery degradation comparison

improve the initial solution, it still lacks performance compared with Gurobi. Additionally, the parameters of ALNS might need to be re-tuned to obtain better performance. The exact result values can be found in Appendix A Tables 4 - 6

We also observed that raising the battery degradation costs relative to the distance costs has no apparent effect on the number of visits to the source. Table 2 shows almost no correlation

between the low or high SoC price and the average number of visits to the source. This suggests that the better timing of visiting the source allows the vehicle to stay at a recommended SoC interval longer, not the frequency of source visits. This different timing of these visits seems to result in less optimal driving distance but lesser BD costs.




	$W^L = W^H = 0$	$W^L = 0.1, W^H = 0.2$	$W^L = 0.5, W^H = 1$	$W^L = 2.5, W^H = 5$
ALNS				
C5	2.33	3.33	3.67	3.33
C10	3.67	6.67	7.00	6.00
C15	7.67	8.33	9.67	8.67
Gurobi				
C5	5.00	3.00	4.00	5.00
C10	5.67	5.33	6.00	5.67
C15	7.67	7.33	8.00	7.67

Table 2: The average number of visits to the source per solution type

6 Conclusion

EVRPTW-PR is a variant of a well-known vehicle routing problem (VRP) with some additional challenges: electric vehicles are being used, which have their own set of limitations as compared to conventional vehicles. Additionally, the customers have strict service time windows that need to be met. **Thankfully**, partial recharging is allowed, which allows more flexibility to visit the customers before their deadlines. In this paper, we first studied the performance of EVRPTW-PR by attempting to replicate the results of Keskin and Çatay (2016) on the widely-used instances generated by Schneider et al. (2014). We were successful at replicating the results obtained by solving MILP problem to optimality quite accurately. However, our ALNS algorithm still lacks some performance, especially regarding runtime. We then expanded on the problem and methodology by including the effects of low-level discharging and high-level recharging on battery degradation and solution costs. Our results show that considering these effects can lead to an improvement of the solution compared to when these costs are accounted for only after the solution is obtained.



For further research, we wish to continue optimizing the performance of our ALNS algorithm to match with Keskin and Çatay (2016) and make further adjustments to it when considering battery degradation costs. It would also be interesting to study how additional factors, such as the different charging voltages, can affect the charging speed and the extra battery degradation they may contribute to.

References

- BU. (2023, Oct). *Bu-808: How to prolong lithium-based batteries*. Retrieved from <https://batteryuniversity.com/article/bu-808-how-to-prolong-lithium-based-batteries>

- Cataldo-Díaz, C., Linfati, R. & Escobar, J. W. (2022, Jan). Mathematical model for the electric vehicle routing problem considering the state of charge of the batteries. *Sustainability*, 14(3), 1645. doi: 10.3390/su14031645
- Conrad, R. G. & Figliozzi, M. A. (2011). The recharging vehicle routing problem. *Proceedings of the 2011 Industrial Engineering Research Conference*.
- Dantzig, G. B. & Ramser, J. H. (1959, Oct). The truck dispatching problem. *Management Science*, 6(1), 80–91. doi: 10.1287/mnsc.6.1.80
- Demir, E., Bektaş, T. & Laporte, G. (2012, Dec). An adaptive large neighborhood search heuristic for the pollution-routing problem. *European Journal of Operational Research*, 223(2), 346–359. doi: 10.1016/j.ejor.2012.06.044
- Dominik, G. (2019). *The electric vehicle-routing problem with time windows and recharging stations*. Retrieved from <https://data.mendeley.com/datasets/h3mrm5dhxw/1>
- EEA. (2024, Apr). *Eea greenhouse gases - data viewer*. Retrieved from <https://www.eea.europa.eu/data-and-maps/data/data-viewers/greenhouse-gases-viewer>
- Emeç, U., Çatay, B. & Bozkaya, B. (2016, May). An adaptive large neighborhood search for an e-grocery delivery routing problem. *Computers amp; Operations Research*, 69, 109–125. doi: 10.1016/j.cor.2015.11.008
- Felipe, , Ortuño, M. T., Righini, G. & Tirado, G. (2014, Nov). A heuristic approach for the green vehicle routing problem with multiple technologies and partial recharges. *Transportation Research Part E: Logistics and Transportation Review*, 71, 111–128. doi: 10.1016/j.tre.2014.09.003
- Guo, F., Zhang, J., Huang, Z. & Huang, W. (2022, Jul). Simultaneous charging station location-routing problem for electric vehicles: Effect of nonlinear partial charging and battery degradation. *Energy*, 250, 123724. doi: 10.1016/j.energy.2022.123724
- Guo, J., Li, Y., Pedersen, K. & Stroe, D.-I. (2021, Aug). Lithium-ion battery operation, degradation, and aging mechanism in electric vehicles: An overview. *Energies*, 14(17), 5220. doi: 10.3390/en14175220
- Gurobi Optimization, L. (2023). *Gurobi optimizer reference manual*. Retrieved from <https://www.gurobi.com>
- Han, S., Han, S. & Aki, H. (2014, Jan). A practical battery wear model for electric vehicle charging applications. *Applied Energy*, 113, 1100–1108. doi: 10.1016/j.apenergy.2013.08.062
- Irle, R. (2024, May). *The electric vehicle world sales database*. Retrieved from <https://ev-volumes.com/>
- Keskin, M., Laporte, G. & Çatay, B. (2019, Jul). Electric vehicle routing problem with time-dependent waiting times at recharging stations. *Computers amp; Operations Research*, 107, 77–94. doi: 10.1016/j.cor.2019.02.014
- Keskin, M. & Çatay, B. (2016, Apr). Partial recharge strategies for the electric vehicle routing problem with time windows. *Transportation Research Part C: Emerging Technologies*, 65, 111–127. doi: 10.1016/j.tre.2016.01.013
- König, A., Nicoletti, L., Schröder, D., Wolff, S., Waclaw, A. & Lienkamp, M. (2021, Feb). An overview of parameter and cost for battery electric vehicles. *World Electric Vehicle*

- Journal*, 12(1), 21. doi: 10.3390/wevj12010021
- Pasha, J., Li, B., Elmi, Z., Fathollahi-Fard, A. M., Lau, Y.-y., Roshani, A., ... Dulebenets, M. A. (2024, Mar). Electric vehicle scheduling: State of the art, critical challenges, and future research opportunities. *Journal of Industrial Information Integration*, 38, 100561. doi: 10.1016/j.jii.2024.100561
- Rastani, S. & Çatay, B. (2021, Nov). A large neighborhood search-based matheuristic for the load-dependent electric vehicle routing problem with time windows. *Annals of Operations Research*, 324(1–2), 761–793. doi: 10.1007/s10479-021-04320-9
- Ropke, S. & Pisinger, D. (2006, Nov). An adaptive large neighborhood search heuristic for the pickup and delivery problem with time windows. *Transportation Science*, 40(4), 455–472. doi: 10.1287/trsc.1050.0135
- Schneider, M., Stenger, A. & Goeke, D. (2014, Nov). The electric vehicle-routing problem with time windows and recharging stations. *Transportation Science*, 48(4), 500–520. doi: 10.1287/trsc.2013.0490
- Solomon, M. M. (1987, Apr). Algorithms for the vehicle routing and scheduling problems with time window constraints. *Operations Research*, 35(2), 254–265. doi: 10.1287/opre.35.2.254
- Wang, H. & Cheu, R. L. (2013, Jan). Operations of a taxi fleet for advance reservations using electric vehicles and charging stations. *Transportation Research Record: Journal of the Transportation Research Board*, 2352(1), 1–10. doi: 10.3141/2352-01
- İslim, R. B. & Çatay, B. (2024, Jun). An effective matheuristic approach for solving the electric traveling salesperson problem with time windows and battery degradation. *Engineering Applications of Artificial Intelligence*, 132, 107943. doi: 10.1016/j.engappai.2024.107943

A Results

Table 3 summarizes the results obtained initially, with ALNS and Gurobi for small instances. TD is the total distance and runtime is given in seconds.

Tables 4 - 6 present the results of both ALNS and Gurobi obtained best solutions when considering battery degradation. #R refers to the number of visits to the source, TD is the total distance, BD is the total battery degradation costs and runtime is given in seconds.

instance	Initial TD	ALNS TD	runtime	Gurobi TD	runtime
C101-5	257.75	247.15	90.78	247.15	0.55
C103-5	178.90	165.67	133.86	165.67	0.08
C206-5	250.09	241.49	183.76	236.58	11.98
C208-5	229.54	158.48	192.04	158.48	1.19
R104-5	167.06	137.01	101.49	136.69	0.39
R105-5	168.47	156.08	93.59	156.08	0.67
R202-5	146.77	128.78	150.18	128.78	0.59
R203-5	199.54	179.06	247.10	179.06	2.22
RC105-5	256.97	233.77	89.67	233.77	86.90
RC108-5	256.97	233.77	90.62	233.77	80.36
RC204-5	185.16	176.39	367.70	176.39	10.50
RC208-5	192.90	167.98	160.74	167.98	0.59
Average					16.33
C101-10	422.72	395.96	515.34	388.25	7200.00
C104-10	328.03	273.93	748.33	273.93	140.61
C202-10	363.02	245.40	1003.04	243.20	1547.10
C205-10	415.31	228.28	582.18	228.28	0.73
R102-10	329.19	249.19	294.11	249.19	4060.14
R103-10	228.00	207.05	290.24	202.85	168.05
R201-10	289.25	228.36	1014.54	217.68	19.35
R203-10	235.15	235.15	3714.24	218.21	512.57
RC102-10	544.19	423.51	201.42	423.51	166.43
RC108-10	506.88	352.09	296.24	345.93	102.98
RC201-10	412.99	324.00	845.21	310.06	120.28
RC205-10	402.25	330.55	890.39	325.98	25.67
Average					1171.99
C103-15	567.96	413.26	1200.22	348.46	7200.00
C106-15	375.93	320.86	822.35	275.13	2.20
C202-15	612.15	413.84	2480.30	369.56	7200.00
C208-15	524.02	306.53	2330.78	300.55	161.75
R102-15	490.56	413.93	586.92	420.61	7200.00
R105-15	417.04	336.94	583.21	336.15	7200.00
R202-15	522.55	425.17	4895.96	358.00	7200.00
R209-15	475.09	371.61	5104.35	293.20	7200.00
RC103-15	638.38	421.72	440.35	397.67	7200.00
RC108-15	502.68	381.28	540.01	370.25	7200.00
RC202-15	551.25	470.63	2647.31	394.39	7200.00
RC204-15	573.97	359.70	5965.29	311.81	7200.00
Average					6013.66

Table 3: Results obtained initially, with ALNS and Gurobi for small instances

instance	ALNS					Gurobi				
	#R	TD	BD	total cost	runtime	#R	TD	BD	total cost	runtime
C206-5	4	236.58	12.06	248.64	472.29	3	236.58	4.44	241.02	138.94
R202-5	3	128.88	4.88	133.76	508.25	2	128.88	4.83	133.71	0.78
RC204-5	3	181.03	5.25	186.27	649.66	4	176.39	3.70	180.10	60.29
C101-10	8	388.25	18.23	406.47	754.66	6	388.63	8.40	397.03	7200.00
R201-10	6	242.86	10.49	253.35	2100.17	4	217.73	5.06	222.79	317.93
RC102-10	6	423.51	15.67	439.18	279.03	6	423.51	12.24	435.75	414.58
C103-15	4	398.67	14.03	412.70	1772.71	6	348.54	5.67	354.20	7200.00
R202-15	12	457.65	21.20	478.85	10368.27	11	362.28	9.66	371.94	7200.00
RC202-15	9	463.78	19.19	482.97	4497.16	5	397.20	6.37	403.57	7200.00

Table 4: Numerical results $W^L = 0.1$, $W^H = 0.2$

instance	ALNS					Gurobi				
	#R	TD	BD	total cost	runtime	#R	TD	BD	total cost	runtime
C206-5	5	252.10	37.97	290.07	483.42	4	246.19	10.76	256.95	459.69
R202-5	3	128.88	26.11	154.99	517.27	4	143.39	9.28	152.67	2.26
RC204-5	3	176.39	22.61	199.00	632.54	4	179.45	13.91	193.36	70.77
C101-10	6	402.25	61.86	464.11	744.75	7	396.18	31.19	427.37	7200.00
R201-10	8	269.70	33.17	302.87	2070.57	5	220.61	21.09	241.69	761.80
RC102-10	7	423.51	81.21	504.72	267.76	6	429.19	55.07	484.27	1426.35
C103-15	7	414.22	77.80	492.01	1766.51	7	348.54	28.35	376.88	7200.00
R202-15	11	421.40	104.71	526.11	10224.46	10	364.49	38.69	403.17	7200.00
RC202-15	11	527.77	87.56	615.33	4384.96	7	404.43	15.71	420.14	7200.00

Table 5: Numerical results $W^L = 0.5$, $W^H = 1$

instance	ALNS					Gurobi				
	#R	TD	BD	total cost	runtime	#R	TD	BD	total cost	runtime
C206-5	4	259.42	182.64	442.06	460.78	5	246.19	53.81	300.00	1191.95
R202-5	3	146.77	95.19	241.96	489.09	6	161.28	7.54	168.82	2.95
RC204-5	3	192.73	99.26	291.99	679.46	4	199.90	1.01	200.92	105.73
C101-10	6	406.68	281.88	688.55	701.28	5	424.62	50.65	475.27	7200.00
R201-10	6	259.04	170.79	429.83	2005.62	7	255.62	27.85	283.48	3269.46
RC102-10	6	440.27	367.58	807.86	292.38	5	448.39	239.77	688.16	7200.00
C103-15	6	428.33	366.94	795.28	1699.99	5	397.72	21.49	419.21	7200.00
R202-15	11	460.41	470.58	930.99	10391.33	10	424.28	82.28	506.55	7200.00
RC202-15	9	451.60	459.59	911.19	8863.39	8	404.43	78.56	482.99	7200.00

Table 6: Numerical results $W^L = 2.5$, $W^H = 5$