Erasmus University Rotterdam
Erasmus School of Economics
Bachelor Thesis International Econometrics and Operation Research

Method Review on Partially Charging Electric Vehicle Routing Problem with Time Windows and Study on Additive Uncertainty to Travel Time

Xiaojing Lu (595182)

Supervisor:	Bart van Rossum
Second assessor:	Ece Karakoyun
Date final version:	1st July 2024

Abstract

This paper explores the Electric Vehicle Routing Problem with Time Windows and Partial Recharge Strategies (EVRPTW-PR) by replicating Mixed Integer Linear Programming (MILP) and examining the Adaptive Large Neighborhood Search (ALNS) algorithm proposed by Keskin and Çatay (2016). We critically assess the suggested algorithm, enhancing it with additional details, and introduce new algorithms to address potential issues encountered during the heuristic process. Simultaneously, we conduct an analysis of the robustness of the ALNS solution by integrating stochastic travel times into the model. This research provides valuable insights into the performance and resilience of routing solutions under conditions of travel time uncertainty.

1 Introduction

With the development of artificial intelligence and car manufacturing, autonomous electric vehicles have gained significant attention. The environmental benefits of replacing conventional fuel vehicles with electric ones, combined with the integration of advanced AI technology, captivate public interest. A prime example of an autonomous vehicle is the Line 500 Park Shuttle in Rotterdam, operating near the metro and bus station of Kralingse Zoom (Picture in Appendix). This shuttle emphasizes both convenience and safety, equipped with sensors that continuously monitor the dynamic environment, ensuring it understands its surroundings at all times. Despite following a specific path, the shuttle's mechanism is designed to stop rather than avoid when encountering unexpected obstacles, such as a nearby car or even a pigeon. Occasionally, the shuttle may suspend operations for up to five minutes due to unforeseen circumstances, causing delays for passengers waiting at the next stop. This scenario inspires an extension to the work by Keskin and Çatay (2016), where all travel times between any pair of different locations are assumed to be predefined and constant before the routing optimization, based on the design of objective function or research directions.

Nevertheless, quite a few authors have explored the research direction of considering travel time uncertainty in the context of the electric vehicle routing problem with time windows and partial charging. In practical scenarios, it is frequently challenging to ensure that scheduled deliveries or services adhere precisely to their timetables, primarily due to delays induced by stochastic factors. In certain sectors, the ability to accommodate a broad range of contingencies is particularly critical. Examples include the transplantation of organs, production lines in high-throughput manufacturing settings, and the delivery of high-value items, where delays can incur significant costs.

This paper focuses on replication of MILP and ALNS algorithm, attempting to solve the problems based on the current benchmarks, as well as conducts a review on the two methods proposed by Keskin and Çatay (2016), adding more details and suggesting solutions to practical potential issues that may appear in replication process. Drawing upon limited existing research on stochastic travel time for EVRPTW-PR, this paper spares space for the impact of additive stochastic variations to predefined planned travel times. Specifically, it examines how fluctuations in travel time influence the feasibility of solutions generated by Adaptive Large Neighborhood Search (ALNS). This study aims to provide a deeper understanding of the extent

to which travel time variability can affect the robustness and reliability of these optimization methodologies in logistical planning, and hopefully contributes to sectors where punctuality is critically valued.

For the structure of this paper, Section 3 provides a comprehensive problem description of EVRPTW-PR; Section 4 and 5 review MILP and ALNS methods respectively, suggesting details and new elementary algorithms to address potential issue; Section 6 demonstrates the replication results and Section 7 mainly concerns the travel time stochasticity.

2 Literature

As pioneers in exploring the electric vehicle routing problem with time windows and charging stations (EVRPTW), Schneider, Stenger and Goeke (2014) developed a Mixed Integer Programming (MIP) model capable of solving a basic EVRPTW to optimality. Building on this work, Keskin and Çatay (2016) introduced partial charging (EVRPTW-PR) as a strategy to relax the full charging requirement and improve solutions derived from the EVRPTW. Following their contributions, numerous studies have expanded on Keskin and Çatay (2016)'s work. For example, Keskin, Çatay and Laporte (2021) proposed a simulation-based heuristic to solve EVRPTW-PR with stochastic waiting times at recharging stations. Cortés-Murcia, Prodhon and Afsar (2019) identified satellite customers with stricter service time requirements, complicating the routing schedule. Froger, Jabali, Mendoza and Laporte (2022) considered the finite capacity of charging stations, while Dönmez, Koç and Altıparmak (2022) examined a mixed fleet comprising both electric and fuel vehicles. Additionally, S. Zhang, Chen and Zhang (2019) assumed that customer demand is positively stochastic, adding another layer of complexity to the problem.

In 2003, Foster et al. (2003) conducted a seminal study on the vehicle routing problem (VRP) with stochastic travel times, identifying traffic jams as a primary source of contingency. Taş, Dellaert, Van Woensel and De Kok (2013), which explores VRP with soft time windows and associated service costs. Further developments by Taş, Dellaert, van Woensel and De Kok (2014) consider time-dependent vehicle routing, while Miranda and Conceição (2016) focuses on hard time windows. The comprehensive review by Li, Tian and Leung (2010) synthesizes models and algorithms pertinent to stochastic travel time VRP. Additionally, both Hashemi Doulabi, Pesant and Rousseau (2020) and J. Zhang, Lam and Chen (2013) examine the trade-offs between cost and customer service, specifically within the healthcare sector. This evolving body of work highlights the growing complexity and relevance of addressing stochastic elements in vehicle routing to enhance logistical efficiency and service reliability.

The literature cited above offers a wealth of knowledge and significant insights into the modeling and problem-solving aspects of stochastic travel times. This foundational work is crucial for advancing the research presented in this paper, particularly concerning EVRPTW-PR under stochastic conditions.

3 Problem Description

Fundamentally, the EVRPTW-PR is a variant of the Vehicle Routing Problem (VRP) with additional requirements and constraints imposed by customer demands, vehicle capacity for carrying a certain amount of cargo, and the energy charge level of Electric Vehicles (EVs). Delivery is performed by EVs, where energy consumption is assumed to be linearly dependent on the traveling distance. Additionally, for preliminary convenience, it is assumed that the charging rate of EVs at each station is also linear, while being dependent on time up to full energy level. Furthermore, all stations are homogeneous except for their different locations in the graph, and all EVs are also assumed to be homogeneous, sharing the same cargo capacity, energy capacity, and traveling rate. This is so called homogeneous fleet.

Starting from the basic setup of VRP, in EVRPTW-PR there is also one depot serving as both the starting and ending point, where all EVs depart and return. Each customer in the graph has a specific time window during which an arriving EV can offer service. Arrival outside this time window is considered invalid, but the service time may exceed the end of the window. If an EV arrives earlier than the start of the window, it should wait until the window opens. Inheriting from VRP, EVRPTW-PR ensures that each customer can only be served once, distinctively unlike charging stations, which each EV can visit multiple times without any additional constraints, as long as it is traveling within the time limit of the entire route. To summarize, intuitively the necessary conditions for model formulation can be listed as the bullet points showing below:

- Only one depot, serving as both the starting and ending node, starting from time 0 and ending at the time limit of the graph
- Homogeneous electric vehicles, sharing same cargo capacity, energy capacity, speed, energy consuming rate linearly dependent on distance, and energy charging rate linearly dependent on time; having no customer preference
- Homogeneous stations, offering partially charging service, having infinite capacity for energy charging, same energy charging rate and no vehicle preference, starting from time 0 and ending at the time limit of the graph
- Customers, having personal demand, specific time window for accepting service and different service duration, no preference for any EV
- EVs starting from depot and stop working as long as they back to the depot
- Customer being only visited once while stations any times if possible

As mentioned by Schneider et al. (2014), due to the high acquisition cost of electric vehicles, our primary objective is to minimize the number of EVs. The secondary objective is to minimize the total traveled distance after the EVs have fulfilled all customer requests. This objective hierarchy and strategy fundamentally impacts the Mixed Integer Linear Programming (MILP) formulation and ALNS algorithm design.

4 Mixed Integer Linear Programming

MILP is a powerful tool for explicitly finding the optimal solution, if one exists. Considering Section 3, the MILP model formulation starts as follows. We directly quote the mathematical notations and problem formulation defined by Keskin and Çatay (2016) where the prototype was introduced by Schneider et al. (2014). Let $V = \{1, \ldots, N\}$ be the set of customers and F be the set of recharging stations. Denote set F' containing dummy nodes standing for each visit at a recharging station. Nodes 0 and $N+1$ represent the start and end of the depot. Let V' be the set of nodes where $V' = V \cup F'$. For convenience to express a union including either or both of the depot nodes, a set is indexed with 0 or $N+1$. Hence, $F'_0 = F' \cup \{0\}$, $V'_0 = V' \cup \{0\}$, $V'_{N+1} = V' \cup \{N+1\}$ and $V'_{0,N+1} = V' \cup \{0\} \cup \{N+1\}$. We can now define the problem on a complete directed graph $G = (V'_{0,N+1}, A)$, where the set of arcs is $A = \{(i,j) \mid i,j \in V'_{0,N+1}, i \neq j\}$. Each arc $(i,j) \in A$ with a distance d_{ij} and travel time t_{ij}. Energy consumption rate is h, with each traveled arc consuming $h \cdot d_{ij}$ of the remaining battery. q_i and s_i denote demand and service duration for customer $i \in V$ with a time window $[e_i, l_i]$. All EVs have a cargo capacity C and a battery capacity Q. At a recharging station, the battery is charged at a rate of g. Decision variables τ_i, u_i, and y_i keep track of the service start time, remaining cargo, and battery state of charge upon arrival at a node $i \in V'_{0,N+1}$, respectively. The binary decision variable x_{ij} represents the selection of arc (i,j). EVRPTW-PR allows for partial recharges by introducing another state decision variable Y_i, representing the battery state of charge upon departure from recharge station $i \in F'$. Thanks to Y_i, the partial charge at a station can be quantified. See Appendix B for the list of parameters and decision variables defined in this MILP model. The complete model formulation is demonstrated in Appendix C. The final model comprises four sets of constraints that define the feasible area within which we can search for an optimal solution, if one exists. These include inflow and outflow, time window, cargo, and energy constraints.

4.1 Inflow and Outflow

$$\sum_{j \in V'_{N+1}, i \neq j} x_{ij} = 1 \qquad \forall i \in V \tag{1}$$

$$\sum_{j \in V'_{N+1}, i \neq j} x_{ij} \leq 1 \qquad \forall i \in F' \tag{2}$$

$$\sum_{i \in V'_0, i \neq j} x_{ij} - \sum_{i \in V'_{N+1}, i \neq j} x_{ji} = 0 \qquad \forall j \in V' \tag{3}$$

$$x_{ij} \in \{0,1\} \qquad \forall i \in V'_0, \forall j \in V'_{N+1}, i \neq j \tag{4}$$

Constraints 1 and 2 respectively define the outflow for customers and recharging stations, while constraint 3 ensures the equality of inflow and outflow for each node in the graph, except for the starting and ending points at the depot. For a customer, who can only be visited once, there must be exactly one inflow from the previous node. Since this customer cannot be an endpoint, there must also be a successor node. This situation is addressed by the combination of constraints 1 and 3. Slightly differently, considering the dummies for stations, since an EV can choose whether or not to visit these dummy stations, the outflow or inflow incident to a

station may be less than one. Nevertheless, there should also be equality between inflow and outflow at a station, thus constraints 2 and 3 ensure the validity of station flow. Due to these constraints features, a route starting from the depot has to and will end at the depot, leading to the equality between outflow of depot start and inflow of depot end, which is also beneficial for quantifying the number of vehicles. The alternative to formulate this part is using the inflow constraints.

$$\begin{aligned} &\sum_{j\in V_{n+1}, i\neq j} x_{ij} = 1 \quad \forall i \in V \\ &\sum_{j\in V_{n+1}, i\neq j} x_{ij} \leq 1 \quad \forall i \in F' \end{aligned} \xrightarrow{\textbf{Equivalent}} \begin{aligned} &\sum_{i\in V_{n+1}, i\neq j} x_{ij} = 1 \quad \forall j \in V \\ &\sum_{i\in V_{n+1}, i\neq j} x_{ij} \leq 1 \quad \forall j \in F' \end{aligned}$$

4.2 Time, Cargo and Energy

$$\tau_i + (t_{ij} + s_j)x_{ij} - l_0(1 - x_{ij}) \leq \tau_j \qquad \forall i \in V_0, \forall j \in V'_{N+1}, i \neq j \qquad (1)$$

$$\tau_i + t_{ij}x_{ij} + g(Y_i - y_i) - (l_0 + gQ)(1 - x_{ij}) \leq \tau_j \qquad \forall i \in F', \forall j \in V'_{N+1}, i \neq j \qquad (2)$$

$$0 \leq u_j \leq u_i - q_i x_{ij} + C(1 - x_{ij}) \qquad \forall i \in V'_0, \forall j \in V'_{N+1}, i \neq j \qquad (3)$$

$$0 \leq y_j \leq y_i - (h \cdot d_{ij}x_{ij}) + Q(1 - x_{ij}) \qquad \forall i \in V, \forall j \in V'_{N+1}, i \neq j \qquad (4)$$

$$0 \leq y_j \leq Y_i - (h \cdot d_{ij}x_{ij}) + Q(1 - x_{ij}) \qquad \forall i \in F', \forall j \in V'_{N+1}, i \neq j \qquad (5)$$

For time, cargo, and energy level constraints, they all utilize inequalities to ensure the correct connection during the transfer from one node to the next. For example, if the arc (i, j) is chosen, nodes i and j are related such that cargo is non-increasing, arrival time is increasing, and energy changes depending on the type of the two nodes. When (i, j) is not chosen, another threshold is triggered, requiring that the current cargo, time, or energy level must be lower than the maximum values defined in the graph. This ensures the constraints are either satisfied or effectively relaxed. Since the relationship between two consecutive nodes involves inequalities, the decreases in cargo and energy and the increases in arrival time are not accurately subtracted from the previous node states. Conversely, the state of available cargo space and energy will drop more than they should in reality, while arrival time will pass more than it should. These can be interpreted as "resource control" for cargo and energy, and as waiting period with respect to time, assuming for each customer, the demand may become more, the traveling time may increase and more energy is required to go there. In other words, if under resource control the route remains still feasible, then it must also be feasible in reality when all resources are available and every action is performed immediately. This explanation validates the constraints modeling and highlights the inconsistencies happening in cargo, time, or energy states when analyzing the values of decision variables in a feasible solution.

4.3 Objective Functions

The primary objective is to minimize the number of EVs, while Keskin and Çatay (2016) and Schneider et al. (2014) do not provide an explanation on how to initially achieve the minimum number of EVs. In this section, we suggest three strategies for attaining the minimum vehicle

count and achieving the minimum total travel distance with the fewest vehicles.

4.3.1 Minimal Vehicles Objective Function

$$\min \quad \sum_{j \in V'} x_{0j}$$

The first strategy concerns the change of objective function to minimize the number of routes using identical constraints to solve the problem under the fewest vehicle condition. Since for each route there is only one EV performing service to the customers throughout the path, minimizing the outflow of the starting depot is equivalent to minimizing the number of EVs running in the graph. Based on this fact, the objective function $\sum_{j \in V'} x_{0j}$ can also be converted to $\sum_{i \in V'} x_{i(N+1)}$, where 0 and $N+1$ represent depot start and depot end, respectively, since the outflow of depot start and inflow of depot end are equal. Note that theoretically, a route could start from the depot and then immediately return to it. However, in practice, this is not meaningful and, for the sake of optimization, such route arrangement will never occur due to an infinitely high cost margin. Denote ω as the minimum number of EVs found by the model. Then, a new constraint can be added to the original formulation to restrict the number of EVs used. $\sum_{j \in V'} x_{0j} = \omega$ can also be transformed to $\sum_{i \in V'} x_{i(N+1)} = \omega$

4.3.2 Additive Route Cost

The additive route cost strategy includes the acquisition cost of each vehicle running on a route, which can be also interpreted as operational cost of opening a new route. Instead of performing MILP model optimization twice to get the minimal number of operated EVs first, this method can solve the problem to optimality with the fewest number of EVs. Define the cost of each vehicle or opening route as c; then the new mixed objective function is illustrated as follows.

$$\min \quad \sum_{i \in V'_0, j \in V'_{N+1}, i \neq j} d_{ij} x_{ij} + c \sum_{j \in V'} x_{0j} \tag{1}$$

Since the optimizer considers both the route cost and total distance, we should prioritize the cost part by assigning a relatively large value to the cost scalar c to ensure the optimization process concentrates on minimizing the number of routes first. Theoretically, c should be greater than the total traveled distance or, better, of a greater magnitude. This procedure can be done by first estimating the total distance, taking the average distance $\bar{d}$ between two nodes, and then multiplying this average value by the number of customers and dummy stations $|V'|$ to get the primarily estimated total traveled distance $\bar{d}|V'|$.

4.3.3 Iterative Feasibility Check

In contrast to the minimal vehicles objective function method, which relies on an additional optimization model to find the minimum number of vehicles, using an iterative feasibility check can be more efficient in some instances. Before optimization, we add the constraint $\sum_{j \in V'} x_{0j} = 1$ into the original model and run it. If there is no feasible solution, we increase the value by 1

until a feasible solution is found. This is the forward iteration of the feasibility check. In some cases, backward iteration of the feasibility check could also be used, where initially we solve the problem without any EVs number constraint to get a feasible solution. Calculate the number of routes, reduce the number by 1 and add it as constraint into the program until infeasibility occurs, but it may be more time-consuming since infeasibility is typically easier to identify than solving a programming model to optimality. The pseudo code is shown as follows.

Algorithm 1 Forward Iteration Feasibility Check

```
Start a new MILP model
Set ω as the number of routes
Assign ω = 1
Add ω = 1 into the model and update
while no feasible solution found do
    ω += 1
    Update the model
end while
return ω =0
```

4.4 Formulation Details

4.4.1 Subtour Elimination

As we can see from the MILP model, when defining a parameter or decision variable involving two nodes, it is explicitly specified that $i \neq j$. This not only means that a directed arc or related parameter is meaningful when it points from one node to another distinct location, but it also theoretically helps eliminate self-to-self subtours in EVRPTW-PR. Moreover, the arrival time decision variable and time window constraints automatically eliminate subtours among customers and dummy stations. This can be shown using the contradiction cycle below. Assume that $C1$ as a customer is the starting point of this route, then denote the arrival time as τ_1, τ_2 and τ_3 corresponding respectively to $C1$, $C2$ and $C3$, where $\tau_1 \leq \tau_2 \leq \tau_3$. However as $C1$ is also the successor of $C3$, then there must be $\tau_3 \leq \tau_1$, which contradicts the assumption that $C1$ is the starting point. The whole proof could be completed by mathematical induction to the global graph.

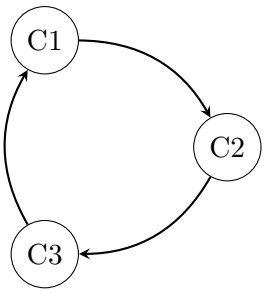


Figure 1: Contradiction Cycle Subtour

4.4.2 Dummies of Stations

Dummies of stations are useful in MILP modeling as they help avoid inner cycles in a route. In theory, we can and should set the number of dummies as high as possible, since it cannot be predicted how many times each station will be visited in the optimal solution. However, because the VRP is an NP-hard problem with a time complexity of $O(n^k)$, where n is the number of nodes in the graph and k is a positive scalar, setting a large number of dummies for each station will increase the running time polynomially in the worst case. Faced with this situation, making number of dummies as a parameter to tune seems viable. The other tip for MILP model formulation is that when creating binary decision variables x_{ij}, we could choose either to not generate such binary variables, or to set the distance as infinity between two dummies of a same station. One reason is that this formulation effectively decreases the total number of arcs in a graph, leading to a reduction in complexity. Additionally, setting no arcs between dummies of the same station prevents cross-travel between them, as shown in Figure 2 where $D1 \in F'$ and $D2 \in F'$ are dummies of station $S0 \in F$. Cross-travel—traveling consecutively between dummies of a station—is inefficient and increases the complexity of the feasible area search. Furthermore, cross-travel is equivalent to, and can be replaced by, visiting only one of the dummies. By doing this, the final solution output is easier to explain. Note that the cross-travel does not impact the optimal solution, only the value assignment of the decision variables.

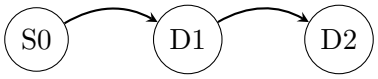


Figure 2: Dummies Cross-Travel

4.4.3 Optimization

It can be observed that at the depot, there are also decision variables τ_i, u_i, y_i and Y_i to track arrival time, remaining cargo capacity, arrival energy level, and departure energy level. The state variables at the depot can be interpreted as shared variables for all homogeneous EVs in the graph, meaning that when they arrive at or depart from the depot, their states in these four aspects are the same. This is achieved through resource control and waiting time, as mentioned in Section 4.2, combined with the fact that there is no cost for changes in time, cargo, or energy level. Based on this, the optimizer tends to set the decision variables as far as possible from one limit to another bound. This is why, in an optimal solution obtained by the Gurobi Optimizer, the departure energy at the depot is always fully equal to Q and arrival energy finally back is zero. This is useful later in the ALNS algorithm when defining a feasibility checker using the MILP model.

5 Adaptive Large Neighbourhood Search

Although current optimizers in the business market are powerful enough to derive optimal solutions based on the EVRPTW-PR model, this VRP variant remains an NP-hard problem.

This means that as the number of nodes in the graph increases, the solving time will increase polynomially. For this reason, the MILP model is considered a tentative approach to finding a good solution. Additionally, MILP model optimization will become more complex given the preference of fewer EVs in the routes operation. For the purpose of achieving a relatively good solution in a short time, Adaptive Large Neighbourhood Search (ALNS) was first introduced by Ropke and Pisinger (2006) on the foundation of Large Neighbourhood Search (LNS) innovated from the work of Shaw (1998).

The proposed ALNS heuristic includes four algorithm classes: Customer Removal (CR), Customer Insertion (CI), Station Removal (SR), and Station Insertion (SI). After constructing an initial feasible solution, the ALNS iteratively improves it until a stopping condition is met, normally using an iteration limit. At each iteration, the current feasible solution is partially destroyed by removing customers, stations, or both (a pair of customer and station connected to each other), and repaired by heuristically inserting them into existing or new routes.

Algorithms are dynamically selected based on adaptive weights w and scores π, which are updated according to their performance. Scores increase by σ_1 for new best solutions, σ_2 for improvements, and σ_3 if accepted via simulated annealing. Weights are updated using $w^a_{s+1} = w^a_s(1-\rho) + \rho\pi_a/\theta_a$, where ρ is the roulette parameter, θ_a is usage count, and π_a is the score. Probabilities for the next segment are calculated as $P^a_{s+1} = \frac{w^a_{s+1}}{\sum_{n=1}^{m} w^n_{s+1}}$.

The simulated annealing approach accepts a new solution if it has fewer vehicles or a shorter total distance. If the number of vehicles is equal but the distance is longer, the solution is accepted with probability $e^{-\frac{(f_{\text{new}} - f_{\text{current}})}{T}}$, where T is the temperature, initially set to T_{init} and decreased by $T = T \times \epsilon$, where $\epsilon \in (0, 1)$ serves as a decay factor. T_{init} is based on a control parameter μ to ensure a solution $\mu\%$ worse than the initial is accepted with 50% probability. Based on this, formula for computing T_{init} is as follow, where log is natural.

$$T = \frac{0.01 \cdot \mu \cdot f_{current}}{\log(2)}$$

Since the elementary ALNS algorithms and operators are comprehensively described in the work of Keskin and Çatay (2016), this paper will not cover those details. Instead, it introduces new algorithms to address potential issues. Additionally, reviews and observations of the ALNS procedure proposed by Keskin and Çatay (2016) are provided. See Section 5.4 for more details.

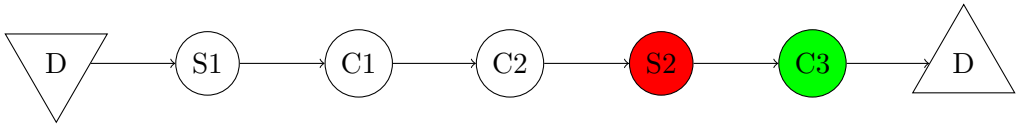
5.1 MILP Feasibility Checker

In heuristic processes, the feasibility check is crucial as it effectively filters out infeasible solutions, allowing focus on identifying the best feasible heuristic solution. In the context of EVRPTW-PR, there are four key constraints discussed in Section 4.2: time, energy, cargo, and flow. During the heuristic process, each customer is inserted into a route once, ensuring flow constraints are met. However, as cargo demand for each customer is predefined, cargo capacity becomes the most stringent constraint. After serving a certain number of customers, no additional customers can be added if the remaining cargo capacity cannot meet the total demand.

Unlike cargo constraints, time and energy constraints offer a broader plane for feasible combination searches. Typically, time and energy feasibility checks are conducted independently

to assess extremities in scenarios. For time, we assume an EV bypasses stations without charging, maximizing available travel time to its limits. If this relaxed scenario fails to meet time constraints, no less relaxed configuration will either. For energy, assuming time constraints are met, we focus solely on distance, arrival energy, and departure energy. An EV is assumed to charge to full capacity at each station regardless of time. If this approach fails to satisfy energy constraints, then the current route configuration is deemed infeasible. Based on this extreme case check, we define the two checkers Greedy Time and Greedy Energy.

It is all good if Greedy Time and Energy could validate the infeasibility of a route, however, the dilemma occurs when both of them signal that this route set is feasible in extreme scenario, since the time and energy extreme feasibility will not necessarily ensure the actual feasibility, as an example shown below.

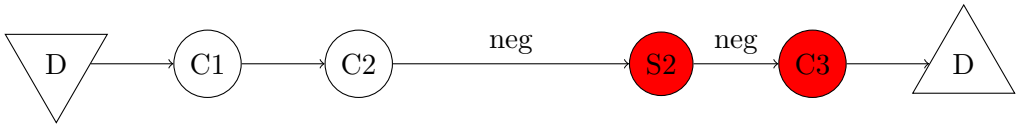


For green customer C3 and red station S2, imagine Greedy Time and Greedy Energy both endorse their own feasibility on this route, while for Greedy time the arrival time on C3 is just the end of time window and the charging duration at S2 is 0. This route is eventually highly probably infeasible since the EV will spend time charging at S2.

To address this challenge, we propose using an Adapted MILP as the time-energy feasibility checker. In this context, the MILP is employed solely to verify the feasibility of constraints interacting with each other. As mentioned, the cargo constraints are straightforward to check, and the flow constraint is automatically satisfied due to the heuristic approach. The Adapted MILP checker includes only the original time and energy constraints, with the objective function set to zero. This strategy allows the MILP model to focus solely on feasibility without optimization, as the optimal objective is already predetermined. The Adapted MILP is highly accurate, considering all time and energy constraints simultaneously, and efficient, as all decision variables are continuous. Therefore, this Adapted MILP check essentially functions as a linear programming model, avoiding the flaw generated by Greedy Time and Energy. See Adapted MILP in Appendix D.

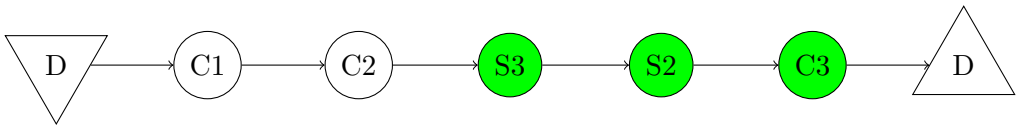
5.2 Greedy Station Insertion - Station Negative

Explicitly defined by Keskin and Çatay (2016), Greedy Station Insertion (GSI) identifies the first customer in the route where an EV arrives with a negative battery level. It then inserts the "best" station (the one that minimally increases the distance) on the arc between that customer and the previous one. If this insertion is not possible, earlier arcs are similarly attempted. However, this algorithm will ignore some potentially complicated insertion problems. Consider the graph below.



By utilizing Greedy Energy checker, we can have a brief overview of the location where the negative battery level occurs. In the graph it assumed that C3 is the first customer at which the running EV arrives with a negative energy level. Confronting this situation, Greedy Station Insertion (GSI) does not function properly to tackle this challenge. Firstly, the arrival energy at S2 is already negative before the vehicle reaching customer C3, which causes infeasibility starting from the station. Secondly, GSI only concerns the station insertion between customers, while neglecting any possible insertion between a station and a customer.

Instead of finding the first customer with negative battery level, we propose to change to the first negative node, could either be a station or customer. This is so called Greedy Station Insertion - Station Negative (GSI-SN). After this insertion, the route is more likely to be repaired since we consider every previous arc between two nodes, not only between two customers.

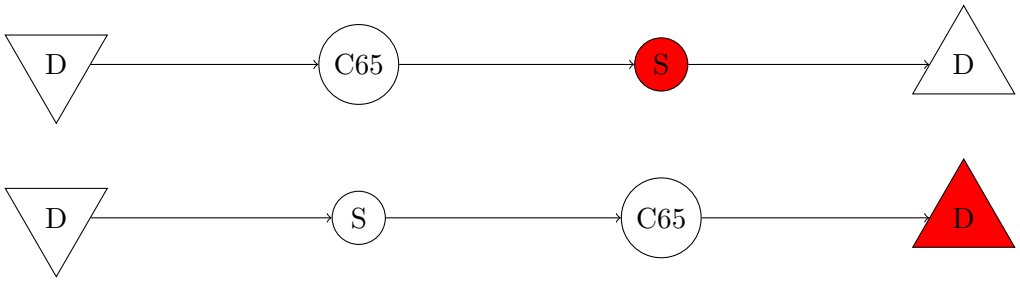


Algorithm 2 Find Minimum Cost Route with Station Insertion

```
0: Find the first negative node index j
0: for each station s in all stations do
0:   for k = 0 to j − 1 do
0:     if insertion of s in (j − k − 1, j) is feasible then
0:       Find the minimum cost route return route
0:     else
0:       Continue
0:     end if
0:   end for
0: end for
0: return the previous route =0
```

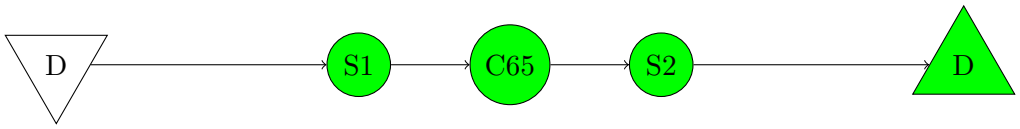
5.3 Supplement Station Insertion - Multiple Stations

The other flaw of the SI algorithms proposed in the paper by Keskin and Çatay (2016) is that they only consider inserting one station into the current route per iteration. If this fails to repair the route, we revert to the previous feasible solution and start the next iteration. During the ALNS heuristic procedure, this won't impact the process significantly since we can always return to a feasible solution. However, this will greatly affect the creation of the initial solution. Consider customer C65 from instance R103_21. According to Ropke and Pisinger (2006), the motivation to include Regret-k for CI is that Greedy Customer Insertion will leave the "hard" customers to the end, given that they have high distance cost from other customers and basically impossible to be inserted into a route to still maintain feasible. They are really outsiders in the graph. From the suggested initial solution create process, Greedy Customer Insertion based on the distance cost is applied. Below is the example of C65.



This is the iteration visualization of GSI-SN, inserting only one station to the first route in different arcs, attempting to make it feasible. Unfortunately, all current SI algorithms will fail due to their feature that for each iteration they only insert one station to see if the route becomes feasible, and return to the previous solution if they fail.

For solving this issue, we propose Supplement Station Insertion (SSI) which iteratively inserts multiple stations one by one to the current route until the route becomes feasible or the count threshold is triggered. The mechanism of SSI is simple: every time it inserts one station and keep the current infeasible route and continue to insert. Theoretically this algorithm is way more useful in initial solution search since it can repair any destroyed route. Given the situation that if SSI is not employed, C65 will be left in the removal list forever and there is always a preliminary route frame $[Depot, Depot]$ in the iteration waiting for any possible customer or station insertion. The route is repaired by SSI. C65 customer is particular as it requires two stations charging for an EV instead of only one station for this route.



Algorithm 3 Supplement Station Insertion

0: $Route$
0: $Count \leftarrow 0$, $Threshold \leftarrow N$
0: Find the first negative node j
0: $New\_Route \leftarrow$ infeasible
0: **while** New_Route is infeasible **do**
0: $\quad Count \leftarrow Count + 1$
0: $\quad$ **if** $Count = Threshold$ **then return** $Route$
0: $\quad$ **end if**
0: $\quad New\_Route \leftarrow$ GSI-SN($Route$)
0: **end whilereturn** ω =0

5.4 Reviews and Observations

- Although this paper uses the tuning parameters proposed by Keskin and Çatay (2016), we do not suggest to tune the parameters since this can be viewed as manipulation of data or data mining, which helps find a particular pattern to get a better solution on the data sets provided. This would lead to overfitting and losing generalization for other instances.

- In each iteration, a feasible solution is destroyed and repaired. If the new solution becomes infeasible, we revert to the previous feasible solution. Thus, each iteration, including the initial one, must maintain feasibility.

- The algorithms (operators) weights are updated solely after a specific segment, which means that for each iteration within the segment, we only calculate the scores and count the numbers one algorithm is called.

- Simulated annealing approach can be viewed as a tool to include stochasticity, in order to jump out of maybe the current local minimum area to the other feasible searching areas.

- For zone removal, this paper suggests to divide the whole graph into a grid with equal squares based on the range of X-axis and Y-axis. Given the zone number parameter proposed by Keskin and Çatay (2016), the grid is a 5×5 area with 25 squares. And this algorithm tends to remove all customers in the zone randomly selected instead of γ customers.

- Remove Customer with Preceding Station and Remove Customer with Succeeding Station serve as filter to process the destroy route by checking there is preceding or succeeding charging stations. These two filters work independently. For example, denote X as as basic CI, then there are three to be selected in each iteration or fewer according to the optimization process preference: X, $X - RCwPS$ and $X - RCwSS$.

- When performing CI algorithms that involve ordering the selection cost, customers are removed one by one. This means that for each removal, the order is recalculated based on the remaining selections. The algorithm performs each removal iteratively rather than ordering and removing all at once. This is applied to all ordering customer removal algorithms.

- Shuffling the order of the removal list in the process of initial solution heuristic will not change the result if any greedy CI algorithm is applied all the way.

6 Computational Results

Here we quote the data description from the paper of Schneider et al. (2014). This data set includes 56 large instances with 100 customers and 21 recharging stations and 36 smaller instances derived from the larger ones with 5, 10, and 15 customers, categorized into three classes based on customer distribution: clustered, random, and a mix of both. For the large instances, recharging stations are strategically placed—one at the depot and others randomly, ensuring all customers are reachable with at most two recharging stops. Small instances are created by selecting customers randomly from the large instances to form new, smaller instances.

For MILP solver, Gurobi Optimizer version 11.0.2 build v11.0.2rc0 (win64 - Windows 10.0 (19045.2)). CPU model: Intel(R) Core(TM) i7-10510U CPU @ 1.80GHz. Thread count: 4 physical cores, 8 logical processors, using up to 8 threads. For ALNS the programming language is Python, version 3.11 in virtual environment.

For the hyper parameters defined in ALNS, we directly applied the fine-tuned parameters by Keskin and Çatay (2016) which yields the smallest result variance. For more details of the parameters, see paper of Keskin and Çatay (2016). In the meantime, due to the inefficient running of the program for C100 instances and to maintain consistency across all instances, we have set the iteration parameters as follows: N = 2500, NC = 200, NS = 400, NRR = 200, NSR = 100, nRR = 50.

6.1 Small Instances

As we can observe from Table 1, all objective values derived from the MILP are smaller than those obtained by ALNS, for three main reasons. First, the objective function of the MILP is solely set to minimize the total travel distance, disregarding the total number of vehicles. This effectively relaxes the constraints on limited vehicle numbers. Second, the MILP model is capable of solving the problem to optimality, given sufficient time, denoting the objective value obtained by MILP as $f*$ and f by ALNS. Theoretically there is $f* \leq f$. Third, due to computational limitations, we have set the maximum number of iterations to 2500. This constraint hinders the optimizer's ability to converge to a global minimum, potentially trapping it in local minima or providing insufficient iterations to reach the global optimum. This could explain the high average difference ratio between the two methods. Another observation is that the solving time for MILP, with a few exceptions, is significantly better than that for ALNS. This discrepancy is likely due to differences in data structure, programming language, algorithm selection, and coding complexity. Further investigations are warranted to explore these aspects.

6.2 Large Instances

Due to computation capacity, the feasible solutions presented in Table 2 for ALNS derived by this paper are initial solutions. This explains the huge gap between the total travel distances for ALNS and ALNS reference provided by Keskin and Çatay (2016). In terms of number of vehicles, they do not differ much since route removal algorithms iteratively attempt to reduce the total number of routes.

7 Travel Time Uncertainty

In the replication of MILP and ALNS, we assume a perfect world where all EVs are homogeneous and all predefined parameters remain fixed throughout the entire traveling process until reaching the time limit. These assumptions simplify modeling and problem-solving by eliminating any uncertainty or stochastic random events occurring during service, travel, and charging. In reality, with high-quality regulation control, process monitoring, and service quality assurance, the charging and service offerings are industrially efficient and adhere strictly to automated processes and norms, making the charging and service requirements as accurate as possible. Furthermore, the time window is planned or decided before routing begins and is a range,

Table 1: Detailed Comparison of MILP and ALNS Solutions

	MILP			ALNS			ALNS Reference			
N	T	V	t	T	V	t	r	T	V	t
c101C5	247.15	3	0.13	257.75	2	11.39	0.04	257.75	2	0.03
c103C5	165.67	3	0.05	175.37	1	31.88	0.06	175.37	1	0.05
c206C5	236.58	4	0.34	242.56	1	27.43	0.02	242.56	1	0.07
c208C5	158.48	2	0.59	158.48	1	13.25	0	158.48	1	0.06
r104C5	136.69	2	0.1	167.06	2	9.65	0.18	136.69	2	0.04
r105C5	156.08	2	0.08	168.47	2	12.45	0.07	156.08	2	0.04
r202C5	128.78	1	0.11	128.78	1	13.57	0	128.78	1	0.08
r203C5	179.06	1	0.24	179.06	1	58.86	0	179.06	1	0.1
rc105C5	233.77	3	0.53	241.89	3	16.04	0.03	233.77	2	0.03
rc108C5	253.93	2	1.63	316.51	3	25.33	0.2	253.93	2	0.04
rc204C5	176.39	2	0.53	176.39	1	56.65	0	176.39	1	0.08
rc208C5	167.98	1	0.51	167.98	1	64.14	0	167.98	1	0.07
c101C10	388.25	4	5.26	395.57	3	367.15	0.02	388.25	3	0.1
c104C10	273.93	2	2.92	306.45	2	1690.42	0.11	273.93	2	0.17
c202C10	243.2	2	0.78	301.62	2	1380.79	0.19	304.06	1	0.2
c205C10	228.28	2	0.25	258.66	2	437.75	0.12	228.28	2	0.16
r102C10	249.19	3	1.54	308.82	4	234.73	0.19	249.19	3	0.11
r103C10	202.85	3	15.44	209.47	2	264.76	0.03	206.12	2	0.17
r201C10	217.68	3	1.04	241.51	1	673.93	0.1	241.51	1	0.21
r203C10	218.21	1	2.92	218.21	1	1635.43	0	218.21	1	0.62
rc102C10	423.51	4	1.4	447.1	4	235.38	0.05	423.51	4	0.09
rc108C10	345.93	3	2.08	384.72	3	348.97	0.1	345.93	3	0.09
rc201C10	310.06	4	0.48	412.86	1	590.04	0.25	412.86	1	0.17
rc205C10	325.98	3	0.9	399.97	2	620.77	0.18	325.98	2	0.19
c103C15	348.46	3	451.01	379.16	3	840.4	0.08	348.46	3	0.23
c106C15	275.13	3	2.18	367.41	3	439.52	0.25	275.13	3	0.15
c202C15	369.56	3	32.97	505.59	3	1535.46	0.27	383.62	2	0.29
c208C15	300.55	2	5.04	305.8	2	802.58	0.02	300.55	2	0.26
r102C15	412.78	6	3600.06	480.29	6	154.51	0.14	412.78	5	0.12
r105C15	336.15	6	46.98	347.88	4	132.29	0.03	336.15	4	0.09
r202C15	358	2	794.95	386.86	2	1548.07	0.07	358	2	0.51
r209C15	293.2	2	37.71	344.31	2	1466.09	0.15	313.24	1	0.92
rc103C15	397.67	4	2107.72	486.94	5	167.91	0.18	397.67	4	0.12
rc108C15	370.25	3	3507.98	440.16	4	206.21	0.16	370.25	3	0.15
rc202C15	394.39	2	14.35	556.15	2	1218.88	0.29	394.39	2	0.31
rc204C15	310.58	2	3600.06	428.22	3	2980.26	0.27	382.22	1	1.35
Average	273.18	2.72	395.58	313.72	2.36	564.25	0.11	282.14	2.06	0.21

Table 2: Detailed Comparison of ALNS Large Instances

name	ALNS		ALNS ref		name	ALNS		ALNS ref		
c101_21	1780.92	15	1051.23	12	r112_21	1648.49	15	1017.31	11	
c102_21	1547.25	13	1034.24	11	r201_21	1816.75	4	1266.06	3	
c103_21	1619.92	14	973.39	10	r202_21	1696.93	4	1052.32	3	
c104_21	1326.28	12	886.72	10	r203_21	1414.23	4	895.54	3	
c105_21	1684.67	14	1037.78	11	r204_21	1235.69	3	780.14	2	
c106_21	1544.70	13	1024.18	11	r205_21	1636.56	4	987.36	3	
c107_21	1794.43	14	1058.11	10	r206_21	1637.27	4	922.70	3	
c108_21	1764.57	14	1033.50	10	r207_21	1317.25	3	846.59	2	
c109_21	1628.75	13	960.03	10	r208_21	1079.84	3	736.12	2	
c201_21	1043.62	5	629.95	4	r209_21	1653.61	4	868.95	3	
c202_21	1237.39	6	629.95	4	r210_21	1395.12	4	843.36	3	
c203_21	1308.16	5	629.95	4	r211_21	1313.18	3	862.56	2	
c204_21	1199.25	4	629.95	4	rc101_21	2288.36	19	1684.84	16	
c205_21	1427.20	6	629.95	4	rc102_21	2127.49	18	1155.50	14	
c206_21	1401.23	5	629.95	4	rc103_21	2195.65	17	1329.58	13	
c207_21	1569.17	6	629.95	4	rc104_21	1932.38	15	1202.93	11	
c208_21	1362.52	6	629.95	4	rc105_21	2146.69	18	1458.49	14	
r101_21	2175.54	22	1661.33	18	rc106_21	2047.74	17	1422.96	13	
r102_21	2082.61	21	1461.48	16	rc107_21	2142.89	17	1261.03	12	
r103_21	1827.28	18	1262.75	13	rc108_21	2056.34	16	1185.68	11	
r104_21	1683.60	16	1078.99	11	rc201_21	2363.89	5	1446.84	4	
r105_21	1890.51	18	1373.94	15	rc202_21	2179.41	4	1416.96	3	
r106_21	1903.29	18	1310.46	13	rc203_21	1871.58	4	1069.27	3	
r107_21	1482.86	14	1118.91	12	rc204_21	1646.77	4	887.76	3	
r108_21	1712.44	16	1031.14	11	rc205_21	1991.29	5	1262.22	3	
r109_21	1837.58	17	1201.04	13	rc206_21	2027.16	4	1213.89	3	
r110_21	1703.75	15	1112.80	11	rc207_21	1811.71	4	993.49	3	
r111_21	1682.62	15	1084.13	12	rc208_21	1552.28	4	839.71	3	

providing a buffer for some arrival delays. However, travel time is significantly, frequently and unexpectedly impacted by uncertainty due to the dynamic environment of the route and other agents involved. Without ideal regulation and monitoring of the travel process, and considering the high costs associated with managing travel uncertainty—such as building exclusive private routes for EVs or imposing impractical restrictions on other agents—controlling stochasticity in travel time is impractical.

Based on this reality, this paper analyzes the robustness of ALNS, testing the stability of a predefined scheduling plan in the face of stochasticity in the routing process. We adopt the mathematical notation as defined in the works of (Schneider et al., 2014) and (Keskin & Çatay, 2016). We denote $t_{ij}^{'}$, a random variable, as the actual travel time between location i and j, where $i \neq j$. For clarity and further definition, we introduce another random variable, ϵ, representing the stochastic component added to the predefined constant travel time t_{ij} between locations i and j. The relationship is described as follows:

$$t_{ij}^{'} = t_{ij} + \epsilon \tag{2}$$

$$\epsilon \sim F_{\epsilon} \tag{3}$$

In typical analyses, it is common to assume that the stochastic variable ϵ, which represents travel time variability, follows a normal distribution with parameters mean $\mu = 0$ and standard deviation σ. This assumption is logical under conventional traffic conditions where no extraordinary events influence driving times.

7.1 Upper Bound

Given the assumption that the uncertainty in travel times is additive, a brutal strategy for managing this uncertainty involves fitting a distribution to the observed data of travel time variances. Once the appropriate distribution is identified, the corresponding quantile at a specified confidence level can be added to the planned travel times. This adjusted figure represents the worst-case scenario within the bounds of the chosen confidence level. This approach ensures that we can estimate, with a defined probability, the likelihood that delivery to each customer will occur within their designated time windows. Concurrently, this method aims to minimize the total travel distance, thus optimizing the route efficiency under the constraints of uncertainty and service time requirements. This strategy allows for the balancing of reliability in meeting time windows with the goal of maintaining route compactness and efficiency. For example, we can replace the original travel time in the MIP formulation with $t_{ij}^{'}$.

$$\tau_i + (t_{ij}^{'} + s_i)x_{ij} - l_0(1 - x_{ij}) \leq \tau_j \quad \forall i \in V_0, \forall j \in V_{n+1}, i \neq j \tag{1}$$

$$\tau_i + t_{ij}^{'}x_{ij} + g(Y_i - y_i) - (l_0 + gQ)(1 - x_{ij}) \leq \tau_j \quad \forall i \in F', \forall j \in V_{n+1}, i \neq j \tag{2}$$

While this intuitive method presents significant drawbacks, it assumes uncertainty in every arc between locations i and j, substantially affecting the normal and practical operations of electric vehicles (EVs). More critically, since time stochasticity is factored in while the personal time

windows for the clients remain unchanged, adding a high quantile of time stochasticity to the planned travel time may effectively bypass some customer time windows. This approach could lead to scenarios where we inadvertently overlook certain customers in the network, which is particularly problematic in sensitive areas such as healthcare. Therefore, a more appropriate method for evaluating time stochasticity needs to be considered.

7.2 Parameter Estimation

As normal distribution is applied to simulate the additive time stochasticity, it is crucial to get the mean and standard deviation of the time stochastic factor ϵ. Denote μ as the mean and σ as the standard deviation, where ϵ follows a normal distribution. Since the underlying distribution for this times stochastic factor is normal, the good features are that the sample mean is Maximum Likelihood Estimator (MLE), which is unbiased and consistent when the sample size converges to infinity; the sample variance is also an MLE given the estimation formula, being consistent and convergent to the population variance when sample size is increasing. The length of the time window is crucial for deciding whether an arrival delay would affect much or not the feasibility of the whole route, since the higher the value of the time window length, the more buffer it provides for the service delay. Under this context, we propose the variance of time window as the variance of the stochastic factor.

$$\epsilon \sim \mathcal{N}(\mu, \sigma^2), \quad \bar{t} = \frac{1}{N}\sum_{i \neq j} t_{ij}, \quad \hat{\sigma}^2 = \frac{1}{n}\sum_{i}((l_i - e_i)) - (\bar{l - e}))^2$$

where N is the cardinality of the arcs set, n is the number of customers. The estimated distribution of ϵ is

$$\epsilon \sim \mathcal{N}(\bar{t}, \hat{\sigma}^2)$$

Note that ϵ is a distributional estimator of the travel time between two locations, it is, in theory, not a proper stochastic factor for simulating the delay or earlier arrival at a customer or station, since it is not practical to assume when an emergency, accident or something unforeseen occurring during the travel, the time stably changes on the base of a positive sample mean plus the stochasticity derived from the variance. Facing this to better simulate the stochastic process, we denote the additive time stochastic factor ϵ' as the term to introduce uncertainty.

$$\epsilon' \sim \mathcal{N}(0, \hat{\sigma}^2)$$

7.3 Simulation Methodology

$$t'_{ij} = t_{ij} + \epsilon'(t'_{ij} \geq 0) \tag{4}$$

This equation serves as the data processing formula by adding the stochastic time factor. For this factor, its distribution is normal with a mean of zero, so in the process of drawing a sample, both negative and positive values can appear. In practice, positive values added to travel time indicate delays caused by stochastic events hindering punctual arrival. Conversely, negative values can be interpreted as the EV speeding up due to an emergency request or other reasons.

And since the negative traveling time does not make sense, we only keep the stochastic travel time if it is non-negative Since normality is generated for each directed arc's traveling time, we have $t'_{ij} \neq t'_{ji}$ This makes sense because the different stochasticities between moving from A to B and moving from B to A should be taken into consideration. Typically, (i, j) and (j, i) do not share the same source of uncertainty, neither the same additive stochasticity.

As mentioned before, it is unrealistic to add stochasticity to each arc since uncertainty occurs sporadically. However, because we cannot determine the distribution of the dynamic environment, we should set specific parameters when adding stochasticity to travel time. The objective is to simulate "accidental" events occurring in the routing process. Set p as the probability parameter to determine for each directed arc how likely an uncertainty will appear, and if it appears, the time factor is added to the fixed traveling time. By testing the robustness of the ALNS solution derived from all predefined parameters, Monte Carlo simulation is suggested here to give a feasibility ratio as the performance metrics.

7.4 Computational Results

We propose three pairs of p and n, (0.1, 1), (0.05, 0.5), (0.01, 0.1) which denote the probability that an event occurs for any arc during the routing process, and adjustment multiplier to standard deviation scaling how much affection the uncertainty would bring to the travel time. The N represents the number of iterations in the Monte Carlo simulation is set to 1000. As seen in Table 3 which demonstrates all types of tendency by representative instances, generally speaking, when the probability and affection scale decrease, the feasibility rate tends to increase, while for some instances, no matter how the pair of parameters is, they remain the same value of 1 or 0, showing different sensitivity to the stochasticity added to the travel time. It suggests that we may rearrange the scheduling of those instances which are more sensitive to small uncertainty as the customer visit plan is made too intensive, as well as considering the trade-off between the cost of opening new routes and utility to deliver on time.

8 Conclusion

This paper replicates the work done by Keskin and Çatay (2016), concentrating on MILP and ALNS heuristic methods in order to achieve good quality feasible solutions to the benchmark instances. Comparisons made between MILP and ALNS result illustrate that the two methods own different advantages facing various instances - MILP derives the solution accurately and quickly given the data size being relatively small and ALNS outperforms MILP instead to provide a feasible solution iteratively on large data set. Meanwhile, reviews on the ALNS algorithms are done with new algorithms proposed to address potential issues during the process of ALNS solution search. Finally the current paper conducts an investigation on the stochastic elements added to the traveling time, studying how to simulate the time uncertainty and proposing that rescheduling or further investigation should be performed on the instances which are sensitive to traveling time stochasticity. Further researches on new more efficient algorithms of ALNS and traveling time stochasticity simulation are needed for future practical purpose.

Table 3: Feasibility Results

height**Name**	**0.1 and 1**	**0.05 and 0.5**	**0.01 and 0.1**
c101C5	0.366	0.98	1
c103C5	0.005	0.015	0.077
c206C5	0.047	0.089	1
c208C5	1	1	1
c101C10	0.221	0.923	1
c104C10	0	0.002	0.358
c202C10	0	0.003	0.082
c205C10	1	1	1
c103C15	0	0.001	0.083
c106C15	0	0.001	0.413
c202C15	0.001	0.001	0.001
c208C15	1	1	1
c101_21	0	0.011	0.533
c108_21	0	0	0.103
c202_21	0	0	0.002
c203_21	0	0	0
c204_21	0	0	0
c205_21	1	1	1
c206_21	0	0	0.016
c207_21	0	0	0.001
c208_21	1	1	1
r101_21	1	1	1
r206_21	0	0	0
r207_21	0	0	0

References

Cortés-Murcia, D. L., Prodhon, C. & Afsar, H. M. (2019). The electric vehicle routing problem with time windows, partial recharges and satellite customers. *Transportation Research Part E: Logistics and Transportation Review*, *130*, 184–206.

Dönmez, S., Koç, Ç. & Altıparmak, F. (2022). The mixed fleet vehicle routing problem with partial recharging by multiple chargers: Mathematical model and adaptive large neighborhood search. *Transportation Research Part E: Logistics and Transportation Review*, *167*, 102917.

Foster, G., Gandrabur, S., Langlais, P., Plamondon, P., Russell, G. & Simard, M. (2003). Statistical machine translation: Rapid development with limited resources. In *Proceedings of MT Summit IX* (pp. 110–119). New Orleans, USA.

Froger, A., Jabali, O., Mendoza, J. E. & Laporte, G. (2022). The electric vehicle routing problem with capacitated charging stations. *Transportation Science*, *56*(2), 460–482.

Hashemi Doulabi, H., Pesant, G. & Rousseau, L.-M. (2020). Vehicle routing problems with synchronized visits and stochastic travel and service times: Applications in healthcare. *Transportation Science*, *54*(4), 1053–1072.

Keskin, M. & Çatay, B. (2016). Partial recharge strategies for the electric vehicle routing problem with time windows. *Transportation research part C: emerging technologies*, *65*, 111–127.

Keskin, M., Çatay, B. & Laporte, G. (2021). A simulation-based heuristic for the electric vehicle routing problem with time windows and stochastic waiting times at recharging stations. *Computers & Operations Research*, *125*, 105060.

Li, X., Tian, P. & Leung, S. C. (2010). Vehicle routing problems with time windows and stochastic travel and service times: Models and algorithm. *International Journal of Production Economics*, *125*(1), 137–145.

Miranda, D. M. & Conceição, S. V. (2016). The vehicle routing problem with hard time windows and stochastic travel and service time. *Expert Systems with Applications*, *64*, 104–116.

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

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

Shaw, P. (1998). Using constraint programming and local search methods to solve vehicle routing problems. In *International conference on principles and practice of constraint programming* (pp. 417–431).

Taş, D., Dellaert, N., Van Woensel, T. & De Kok, T. (2013). Vehicle routing problem with stochastic travel times including soft time windows and service costs. *Computers & Operations Research*, *40*(1), 214–224.

Taş, D., Dellaert, N., van Woensel, T. & De Kok, T. (2014). The time-dependent vehicle routing problem with soft time windows and stochastic travel times. *Transportation Research Part C: Emerging Technologies*, *48*, 66–83.

Zhang, J., Lam, W. H. & Chen, B. Y. (2013). A stochastic vehicle routing problem with travel time uncertainty: trade-off between cost and customer service. *Networks and Spatial*

Economics, *13*, 471–496.

Zhang, S., Chen, M. & Zhang, W. (2019). A novel location-routing problem in electric vehicle transportation with stochastic demands. *Journal of Cleaner Production*, *221*, 567–581.

A Abbreviations

- AI: Artificial Intelligence
- ALNS: Adaptive Large Neighborhood Search
- CI: Customer Insertion
- CR: Customer Removal
- EV: Electric Vehicle
- EVRPTW-PR: Electric Vehicle Routing Problem with Time Windows and Partial Recharging
- GSI: Greedy Station Insertion
- GSI-SN: Greedy Station Insertion - Station Negative
- LNS: Large Neighborhood Search
- MILP: Mixed Integer Linear Programming
- MIP: Mixed Integer Programming
- RCwPS: Remove Customer with Proceeding Station
- RCwSS: Remove Customer with Succeeding
- SI: Station Insertion
- SR: Station Removal
- SSI: Supplement Station Insertion Station
- VRP: Vechile Routing Problem

B MILP Parameters

Variable	Parameter Definitions of the EVRPTW-PR MILP Model
$0, N+1$	Depot instances
F'	Set of visits to recharging stations, dummy vertices of set of recharging stations F
F'_0	Set of recharging visits including depot instance 0
V	Set of customers $V = \{1, \ldots, N\}$
V_0	Set of customers including depot instance 0
V'	Set of customer vertices including visits to recharging stations, $V' = V \cup F'$
V'_0	Set of customers and recharging visits including depot instance 0: $V'_0 = V' \cup \{0\}$
V'_{N+1}	Set of customers and recharging visits including depot instance $N+1 : V'_{N+1} = V' \cup \{N+1\}$
$V'_{0,N+1}$	Set of customers and recharging visits including depot instances 0 and $N+1 : V'_{0,N+1} = V' \cup \{0\} \cup \{N+1\}$
d_{ij}	Distance between vertices i and j
t_{ij}	Travel time between vertices i and j
C	Vehicle capacity
g	Recharging rate
h	Charge consumption rate
Q	Battery capacity
q_i	Demand of vertex i, 0 if $i \notin V$
e_i	Earliest start of service at vertex i
l_i	Latest start of service at vertex i
s_i	Service time at vertex i ($s_0, s_{N+1} = 0$)
τ_i	Decision variable specifying the time of arrival at vertex i
u_i	Decision variable specifying the remaining cargo on arrival at vertex i
y_i	Decision variable specifying the remaining battery capacity on arrival at vertex i
Y_i	Decision variable specifying the remaining battery capacity on departure at vertex i
x_{ij}	Binary decision variable indicating if arc (i, j) is traveled

C MILP Full Model

$$\min \sum_{i \in V_0', j \in V_{N+1}', i \neq j} d_{ij} x_{ij} \tag{1}$$

subject to

$$\sum_{j \in V_{N+1}', i \neq j} x_{ij} = 1 \qquad \forall i \in V \tag{2}$$

$$\sum_{j \in V_{N+1}', i \neq j} x_{ij} \leq 1 \qquad \forall i \in F' \tag{3}$$

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

$$\tau_i + (t_{ij} + s_j) x_{ij} - l_0(1 - x_{ij}) \leq \tau_j \qquad \forall i \in V_0, \forall j \in V_{N+1}', i \neq j \tag{5}$$

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

$$e_j \leq \tau_j \leq l_j \qquad \forall j \in V_{0,N+1}' \tag{7}$$

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

$$0 \leq u_0 \leq C \tag{9}$$

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

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

$$y_i \leq Y_i \leq Q \qquad \forall i \in F_0' \tag{12}$$

$$x_{ij} \in \{0, 1\} \qquad \forall i \in V_0', \forall j \in V_{N+1}', i \neq j \tag{13}$$

D Adapted MILP Feasibility Checker

$$0 \tag{1}$$

subject to

$$\tau_i + (t_{ij} + s_j) x_{ij} - l_0(1 - x_{ij}) \leq \tau_j \qquad \forall i \in V_0, \forall j \in V_{N+1}', i \neq j \tag{5}$$

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

$$e_j \leq \tau_j \leq l_j \qquad \forall j \in V_{0,N+1}' \tag{7}$$

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

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

$$y_i \leq Y_i \leq Q \qquad \forall i \in F_0' \tag{12}$$

(5)

E Images

Figure 3: https://www.transdev.nl/nl/onze-routes/vervoersgebieden/parkshuttle-rivium

Figure 4: https://corporate.ret.nl/nieuws/aanpassingen-dienstregeling-bus-vanaf-30-januari