Erasmus University Rotterdam
Erasmus School of Economics
Bachelor Thesis Analytics and Operations Research in Logistics

The Electric Vehicle-Routing Problem with Time Windows and Recharging Stations using Non-Linear Recharging

Daniël Andreas Alblas (562278)

Supervisor:	dr. R. Spliet
Second assessor:	M. Wagenvoort
Date final version:	28th June 2023

Abstract

In this study, we analyse the performance of the VNS/TS hybrid meta heuristic introduced in Schneider et al. (2014) for the purpose of electric vehicle routing problems with time windows and recharging. We extend their work to include non-linear recharging. We find that the runtime performance of the meta heuristic is heavily dependent on how well the code implements portions not discussed as part of the heuristic itself, leading to a massive worsening of efficiency. Additionally, we find that non-linear recharging makes a majority of solutions found using linear recharging infeasible, as well as a necessity to adapt the heuristic further to find feasible solutions in a reasonable amount of runs.

1 Introduction

Green logistics, which takes into account environmental and social factors (Sbihi & Eglese, 2010), is becoming an ever more important topic in a world faced with increasingly many issues surrounding emission pollution, and congestion. Interest in green logistics by companies is driven by a combination of factors, including, among others, an increase in energy and material costs, regulations necessitating sustainable logistics, growing demand for greener ways, and the possibility to be a pioneer in employing new green operations leading to an improvement in market position (see, e.g. Kleindorfer et al., 2005; Dekker et al., 2012; McKinnon et al., 2015).

Green logistics activities are mainly directed at transportation among others, due to its negative influences in the shape of noise, emission pollution, and congestion (Dekker et al., 2012). For instance, domestic transport was to blame for more than 20% of greenhouse gas (GHG) emissions in the European Union (EEA, 2023) in 2021. The European Commision aims to reduce total GHG emissions by 55% compared to the numbers of 1990 (EC, 2023), with emission regulations becoming more common.

Customarily, vehicle routing problems (VRPs) represent the distribution tasks of logistics companies. These problems seek to minimize factors such as transportation costs, number of vehicles, or distance traveled. Two extensions which have become prominent are the capacitated VRP, which incorporates vehicles with a finite freight capacity (see, e.g. Toth & Vigo, 2002), and VRP with time windows (VRPTW), where locations have to be reached between two points in time (see, e.g. Gendreau & Tarantilis, 2010). VRPs can also be extended to include green considerations. One way is to include emission costs as part of the objective function of models, which allows a trade-off to occur between economic and sustainable goals (see, e.g. Figliozzi, 2010; Bektaş & Laporte, 2011). Another direction is to consider alternative fuel vehicles that pollute less. One example is electric commercial vehicles (ECVs). ECVs have as of recently become more wide-spread due to governmental incentives (Pelletier et al., 2016). To transition from conventional vehicles to these ECVs, efficient route-planning techniques, which take into account the characteristics of ECVs, are required. The range of ECVs is possibly insufficient for classical delivery routes (see, e.g. Feng & Figliozzi, 2013; Botsford & Szczepanek, 2009; Tredeau & Salameh, 2009). While reducing the number of customer deliveries per vehicle is an unfruitful direction, visiting recharging stations could be promising. Recharging station visits would need to be incorporated in models to avoid long detours, which may be especially prevalent when there are a small number of stations.

This paper will follow the procedure by Schneider et al. (2014) to model a VRP that includes practical restrictions mentioned before. Namely, capacity restrictions, as well as time windows, which are common among the small-package shipping sector. The latter is influenced by the introduction of recharging times, which depend on the battery charge upon arrival at a station, and take a notable amount of time, which impacts route planning. The problem incorporates two main elements of employing ECVs in a real world scenario: (1) a reduced operating range, and (2) the ability for vehicles to increase their range by refueling at recharging stations. However, certain simplifications are made to other real world complexities. The effect of payloads on routing costs are disregarded, grading is not accounted for, i.e., flat terrain is assumed, and travel speeds are assumed to both be given and constant. These characteristics affect both conventional vehicles and ECVs. However, they may be more prevalent for the latter, as these factors all directly or indirectly influence the need to visit recharging stations as well as the recharging time. Finally, recharging is always assumed to be performed until battery capacity is full, and is assumed to be linear in time. In reality, it has been shown by Marra et al. (2012) that the time to charge increases at the last 10-20% of the battery capacity.

This electric vehicle-routing problem with time windows and recharging stations (E-VRPTW) first aims to minimize the number of used vehicles and then total traveled distance. The complexity of this problem seems to render exact solutions inadequate in order to solve realistically-sized problem instances (Baldacci et al., 2012). For this purpose, a hybrid meta heuristic is employed which combines a Variable Neighborhood Search (VNS) heuristic and a Tabu Search (TS) method used for the intensification phase of this VNS. Two sets of benchmark instances introduced by Schneider et al. (2014) will be used. Namely, results found using a set of small-sized instances will be compared with known results through exact solution method as well as prior results found using the heuristic in Schneider et al. (2014) to assess performance, as well as a big-sized instance closer to reality, which also be compared to the results found in Schneider et al. (2014).

We extend the E-VRPTW by including an approximation of nonlinear charging, using two piecewise linear functions, to assess the difference in performance and solutions when charging is either slowed down or sped up due to the difference in charging time compared to the small data instances of in Schneider et al. (2014).

When comparing the performance for the small instances, we find the same results. Running large instances, we, for the most part, find worse results. This could be attributed to the fact that a lesser number of trials are ran due to a limitation of time. Another explanation is due to the possible misinterpretation of methodology in Schneider et al. (2014). We find a worse runtime performance for both cases. This likely has to do with a worse implementation of code due to unmentioned aspects in Schneider et al. (2014). Finally, when testing out two different cases of non-linear recharging, we find that for over half of the tested instances, the results for linear recharging cannot be used. Additionally, we find that the hybrid heuristic needs to be altered further to obtain feasible results in a reasonable amount of runs.

The remaining part of our paper is organised as follows: In Section 2 we perform a literature review relating to our problem. Section 3 describes our specific problem together with assumptions and simplifications, as well as notation used in the paper. In Section 4 we describe the

hybrid VNS/TS solution method as well as extension, followed by Section 5 where we describe the used problem instances further as well as results of the methodology. Ultimately, in Section 6 we discuss the implications of our results, the limitations of our study, and opportunities for further research.

2 Literature review

Green logistics and E-VRPTW, as previously discussed, are closely linked to one another. A multitude of its aspects are featured in past papers (Kleindorfer et al., 2005; Srivastava, 2007; McKinnon et al., 2015). More specifically, the role of Operations Research within the domain of green logistics has been reviewed by Sbihi and Eglese (2010) and Dekker et al. (2012). An even further specialization was carried out by Bektaş et al. (2019), focusing on just the green transportation angle, a subset of green logistics.

In the last decade multiple green VRP models have been discussed. One VRP variant has been studied by Gonçalves et al. (2011) with pickup and delivery, and a fleet composing a mix of ECVs and conventional vehicles. The time to recharge an ECV is calculated by considering the possible travel range of just one battery charge, and the total distance traveled. Here they disregard the location of charging stations. Conrad and Figliozzi (2011) limit recharging possibilities to certain customer locations. Time windows are included, and charge consumption is assumed to be proportional to distance traveled.

Erdoğan and Miller-Hooks (2012) introduced the green VRP (G-VRP). This model considers alternative fuel vehicles that can refuel during a route, where again consumption is proportional to distance traveled. Time windows and capacity restrictions are not accounted for in this model. Two heuristics are introduced that can solve large G-VRP instances. One is the Density-Based Clustering Algorithm, where routes are constructed after clustering. Another is the Modified Clarke and Wright Savings heuristic, which creates routes through inserting fuel stations, merging feasible routes based on savings, and removal of stations that are redundant. Both heuristics perform well based on numerical experiments.

This model was extended by Ángel Felipe et al. (2014) to allow partial recharging based on different charging infrastructures. Here an algorithm based on an exhaustive local search was found to work best for instances with up to 200 customers, whereas an algorithm based on Simulated Annealing performed better for larger instances. Two more models that consider mixed fleets of conventional vehicles and ECVs are detailed by Sassi et al. (2015) and Goeke and Schneider (2015). The former allows for partial charging, whereas the latter does not, but instead considers realistic charge consumption based on speed, vehicle mass and gradient.

While many papers assume constant driving time, Demir et al. (2014) investigated the relationship between fuel consumption and driving time. As fuel consumption goes up, the driving time goes down, meaning their minimization is in clash. Their model introduced a bi-objective Pollution-Routing Problem (PRP) where both elements are minimized. Another angle is to optimize the routing plan as well as the charging station locations, which was done by Li-ying and Yuan-bin (2015). Specifically, different charging technologies are considered.

The E-VRPTW model introduced by Schneider et al. (2014) has been extended by Ding et al. (2015) to include partial charging as well as a pickup and delivery policy. Bruglieri et al.

(2015) has studied a variant of the model where battery charging level is a decision variable. Desaulniers et al. (2016) has also extended E-VRPTW but instead considered four different charging strategies including partial recharges. J. Lin et al. (2016) has studied E-VRP with a heterogeneous fleet of ECVs as well as the effect of vehicle load on battery consumption as was done in Goeke and Schneider (2015).

Schiffer and Walther (2017) studied an E-VRPTW with partial recharging where ECVs can charge at any location in the network. Schiffer and Walther (2018) studied a location routing problem with facilities for the purpose of intermediate stops in routes to keep vehicles from being nonfunctional. Here they use an adaptive large neighborhood search, enhanced by local search and dynamic programming components, which produces competitive results applicable to real life situations.

Finally, Hiermann et al. (2019) introduced an E-VRP with a mix of conventional, plug-in hybrid, and electric vehicles. Plug-in hybrid vehicles have the ability to omit going to a charging station by swapping to fossil fuels.

3 Problem description

We first define the parameters used for the E-VRPTW model. For this purpose, a route is defined as a path of vertices that start at depot vertex 0 and end at depot vertex $N+1$, which are the same depot. We have the set of customers $V = \{1,...,N\}$, and a set F' with dummy vertices, created such that vehicles can visit the same charging station in the set F multiple times. An apostrophe indicates a union with the set F', e.g. $V' = V \cup F'$. Subscripts are used to indicate whether a set includes the depot vertex at the start or the one at the end, or both, e.g. $V_0 = V \cup \{0\}$, $V_{N+1} = V \cup \{N+1\}$, $V'_{N+1} = V' \cup \{N+1\}$. The capacity of an ECV is C, whereas the battery capacity is Q of which the linear recharging rate is denoted with g. The demand at vertex i is q_i, assumed to be 0 if $i \notin V$. The distance to travel between vertices i and j is indicated with d_{ij}, and the corresponding time with t_{ij}; the battery charge that is consumed to travel this arc can be calculated with $h \cdot d_{ij}$, with h the rate of charge consumption. The earliest possible service start time and latest possible service start time at vertex i are denoted with e_i and l_i respectively. The service time necessary at vertex i is s_i (s_0, $s_{N+1} = 0$).

The vehicles used are assumed to be homogeneous. Vertices $i \in V'_{0,N+1}$ have a positive demand q_i that has to be delivered by one vehicle. Additionally, vertices $i \in V'_{0,N+1}$ all have time windows $[e_i, l_i]$, in which service must be started, though not ended, as service takes s_i time. As service may only start from e_i onwards for vertex i, a waiting time may be incurred. When a recharging station is visited by an ECV, the battery is recharged back to the capacity Q. In the initial case, at linear rate g, but in the non-linear cases with piecewise linear functions. In any case, the charging time depends on the remaining charge when visiting a recharging station.

The algorithm is to decide what routes are utilized. The objective of the E-VRPTW is multi-levelled. First, as done frequently in the context of VRPTW (e.g. Bräysy and Gendreau (2005)), the number of vehicles used are minimized, meaning a superior solution is one with fewer vehicles, after which this number is used to minimize the total distance traveled. Given the fact that ECVs have high acquisition costs, this hierarchical structure seems even more applicable in this context than one with conventional vehicles.

4 A Hybrid Meta Heuristic VNS/TS Solution Method

A hybrid meta heuristic, combining VNS and TS, is employed to obtain solutions. The performance of this hybrid has been proven in the past on both routing and other related problems (see, e.g. Melechovskỳ et al., 2005; Tarantilis et al., 2008). VNS, introduced by Mladenović and Hansen (1997), is a metaheuristic that performs local search on neighborhoods that increase over time, for the purpose of efficiently exploring the solution space while not getting stuck in local optima. TS, which was introduced by Glover and Laguna (1998), is a metaheuristic that guides a local search heuristic to perform searches through the solution space both economically and effectively. Starting from a solution, the best nontabu move is performed during each tabu iteration. A memory structure, namely a tabu list, is used to prevent the heuristic from cycling solutions.

As part of the algorithm, a preprocessing step is used to remove infeasible solutions, after which an initial solution, S, is generated, using a given number of vehicles m, as described in Section 4.1. Infeasible solutions are allowed during the search, but are assessed using a penalizing cost function (see Sections 4.2 and 4.3). As a first phase, feasibility is explored. After no feasible solutions are found after η_{feas} iterations, the maximum number of vehicles m is increased by one, and a new solution is generated using the steps described in Section 4.1. Once a feasible solution has been found, η_{dist} iterations are performed to decrease the traveled distance. Our search iterations are guided by a VNS component that is described in 4.3, which conducts a random perbutation move on the current solution using VNS neighborhood $\mathcal{N}_\kappa$, which is used as initial solution for the TS phase. After η_{tabu} iterations, the newly retrieved solution is compared with the current solution, and either accepted or rejected using an acceptance criterion, based on simulated annealing (SA).

4.1 Generating an Initial Solution

First, we perform a series of preprocessing steps to remove infeasible arcs in our list of possible arcs for the generation an initial solution. Namely, an arc (v, w) can be eliminated from the list if any of the following conditions hold:

$$v, w \in V \wedge q_v + q_w > C, \tag{1}$$

$$v \in V_0', w \in V_{N+1}' \wedge e_v + s_v + t_{vw} > l_w, \tag{2}$$

$$v \in V_0', w \in V' \wedge e_v + s_v + t_{vw} + s_w + t_{wN+1} > l_0, \tag{3}$$

$$\begin{aligned} &v, w \in V \wedge \forall j \in F_0', \\ &i \in F_{N+1}' : h \cdot (d_{jv} + d_{vw} + d_{wi}) > Q. \end{aligned} \tag{4}$$

Equations 1-3 are widely adapted preprocessing steps based on infeasibility regarding capacity and time windows. Equation 4 is specific to our problem, and specifically refer to arcs where there is not enough charge to both travel the arc, as well as travelling from that arc to the depot or a charging station, based on the battery capacity.

The initial solution is then constructed similarly to the approach introduced in Cordeau et

al. (2001). Here, all customers are sorted in nondecreasing order based on the angle between the depot, a randomly chosen point between the upper and lower bounds of all location coordinates, and the customer itself. These customers are then iteratively inserted beginning at the start of the list into the currently active route causing a minimal increase in the traversed distance, until an insertion would entail a violation of the capacity or battery capacity. If this were the case, a new active route is activated until m routes are activated in total. After this point, all remaining customers are inserted into route m, no matter if a violation occurs, still following the principle of increasing traveled distance as little as possible.

A battery capacity violation is asserted under the assumption that recharging is impossible, as an initial solution only includes the customers and a depot at the start and end of routes. While time window feasibility is not guaranteed, a customer u may only be inserted between vertices i, j if $e_i \leq e_u \leq e_j$. However, feasibility is guaranteed for capacity and battery capacity constraints in all but the last route.

4.2 The Generalized Cost Function

Infeasible solutions are allowed during our VNS/TS search process. For that reason, solutions are assessed using the following generalized cost function

$$f_{\text{gen}}(S) = f(S) + \alpha P_{\text{cap}}(S) + \beta P_{\text{tw}}(S) + \gamma P_{\text{batt}}(S) \tag{5}$$

where $f(S)$ indicates the traveled distance across all routes in solution S, $P_{\text{cap}}(S)$ the capacity violation, $P_{\text{tw}}(S)$ the time window violation, and finally $P_{\text{batt}}(S)$ the battery capacity violation. Here, α, β, and γ are weights for these penalties, which are initialized as α_0, β_0, and γ_0, but dynamically increased or decreased between lower ($\alpha_{\min}$, $\beta_{\min}$, $\gamma_{\min}$) and upper bounds ($\alpha_{\max}$, $\beta_{\max}$, $\gamma_{\max}$). While not made clear explicitly in Schneider et al. (2014), an increase by a factor δ occurs to a weight if the respective constraint violation was a positive number for η_{penalty} consecutive iterations of the algorithm, and likewise a division by the same factor occurs if the respective violation was zero η_{penalty} consecutive iterations. This alternating of weight size is to balance between diversification and intensification.

We now describe how to efficiently calculate the constraint violations. A vehicle route r is defined as a sequence of locations, i.e., customers, depots, charging stations, $\langle v_0, v_1, \ldots, v_n, v_{N+1} \rangle$, where v_0 and v_{N+1} represent the depot (see, e.g. Nagata et al., 2010). A solution S is defined as $S = \{r_k, k = 1, \ldots, m\}$ with m routes. Vert(r) characterizes the vertex set of a route r. Using this, the capacity violation of route r can be calculated thusly

$$P_{\text{cap}}(r) = \max\Big(\sum_{v \in \text{Vert}(r)} q_v - C, 0\Big)$$

With which the total capacity violation can be calculated by simply summing up the individual violations of each route in the set:

$$P_{\text{cap}}(S) = \sum_{k=1}^{m} P_{\text{cap}}(r_k)$$

To attain constant time $\mathcal{O}(1)$ calculations of capacity violations for any solution obtained through neighborhood operations of our TS component described in Section 4.3, we save forward and backward capacity requirements for each vertex (see, e.g. Kindervater & Savelsbergh, 1997; Ibaraki et al., 2005). This is the amount of capacity that is required to traverse from the starting depot to a given vertex and the amount of capacity that is required to traverse from the vertex to the end depot, respectively. When merging two partial routes to form a new route, we sum the forward capacity requirements of the first partial route, the backwards capacity requirements of the second partial route, and the demand of the first location at the latter partial route. If instead, we add a vertex between two partial routes, we additionally add the demand of that vertex. If a vertex is moved using an intraroute move, no change occurs.

We define two variables for each vertex in a route $r = \langle v_0, v_1, \ldots, v_n, v_{N+1} \rangle$ for the purpose of calculating the battery capacity violations. Namely, $\Upsilon^{\rightarrow}_{v_i}$, the battery charge required to traverse from the most recent recharging station visit or depot to vertex v_i, and $\Upsilon^{\leftarrow}_{v_i}$, the battery charge required to traverse from vertex v_i to the next recharging station or depot:

$$\Upsilon^{\rightarrow}_{v_i} = \begin{cases} h \cdot d_{v_{i-1}v_i} & \text{if } v_{i-1} \in F'_0, \\ \Upsilon^{\rightarrow}_{v_{i-1}} + h \cdot d_{v_{i-1}v_i} & \text{otherwise} \end{cases} \quad i = 1, \ldots, n+1$$

$$\Upsilon^{\leftarrow}_{v_i} = \begin{cases} h \cdot d_{v_i v_{i+1}} & \text{if } v_{i+1} \in F'_{N+1}, \\ \Upsilon^{\leftarrow}_{v_{i+1}} + h \cdot d_{v_i v_{i+1}} & \text{otherwise} \end{cases} \quad i = 0, \ldots, n$$

These variables are used to define the battery capacity violation of a route r. We sum up the violations at each charging station and at the ending depot:

$$P_{\text{batt}}(r) = \sum_{v_i \in \text{Vert}(r) \cap F'_{N+1}} \max(\Upsilon^{\rightarrow}_{v_i} - Q, 0)$$

Evidently, we can also calculate any battery capacity violations obtained through neighborhood operations in Section 4.3 in constant time $\mathcal{O}(1)$. For this, we calculate the change in violation at the first stations or depot after the points of vertex insertion, removal, or merging. This change can be calculated using the forwards and backwards battery charge requirements of prior locations, depending on whether a charging station was inserted between the relevant station and the one prior to it, and the changed distances and charge consumption rate.

For the purpose of calculating time window violations, we adapt the methods of Nagata et al. (2010), which were enhanced by Schneider et al. (2013). This approach is based on the idea of time travel being possible, i.e., the latest feasible arrival time at preceding (violating) vertices is assumed when calculating the violation at a certain vertex. By doing this, time window violations are not propagated to the vertices of a route following a location with a violation. Additionally, this approach allows us to calculate violations in constant time $\mathcal{O}(1)$ for interroute moves for most classical neighborhood structures. Specifically, using stored penalties, we can calculate in constant time the time window violation of a route $r_1 = \langle v_0, \ldots, u, w, \ldots, v_{N+1} \rangle$ made

up of two partial routes $\langle v_0,\ldots,u\rangle$ and $\langle w,\ldots,v_{N+1}\rangle$, or a route $r_2 = \langle v_0,\ldots,u,v,w,\ldots,v_{N+1}\rangle$ created by inserting a vertex v between the two partial routes. This is done using forward and backward time window penalty slacks that are stored for each interim solution.

While not directly mentioned in Schneider et al. (2014), we adapt the time-window handling approach to not include just service time, but also recharging time. If a location is a charging station, and thus not a customer, the time spent at the location is instead taken as the charging time at the location. This is calculated using the corresponding recharge function and the minimum between the battery capacity and the battery charge spent to travel from the previous charging station or depot to the relevant charging station. We take this minimum to not also include battery charge violations within the calculation of time window violations. Due to the inclusion of recharges, constant time calculations as described previously are not possible if a recharging station is included in the earlier mentioned partial path $\langle w,\ldots,v_{N+1}\rangle$. If recharging station z is included, i.e., $\langle w,\ldots,z,z+1,\ldots,v_{N+1}\rangle$, the slack variables have to be calculated again by traversing the partial route $\langle w,\ldots,z,z+1\rangle$ for the first route, and $\langle v,\ldots,z,z+1\rangle$ for the second. A recalculation is not warranted if a recharging station is included in the partial route $\langle v_0,\ldots,u\rangle$ or if the vertex v in r_2 is a recharging station.

4.3 The Variable Neighborhood Search

We mainly use our VNS phase to diversify our search. We shake the current solution using neighborhood structure $\mathcal{N}_\kappa$ in our set of predefined κ-neighborhood structures. Every neighborhood structure is defined by means of the cyclic-exchange operator (Thompson & Orlin, 1989). Here, sequences of locations of arbitrary length are transferred between arbitrary differing routes in solution S. Our κ-neighborhood structures are defined according to the number of affected routes $\#Rts$, and $\Lambda_{\max}$, the maximum length of a sequence of locations. All structures are shown in Table 1. The translocation chain length is randomly selected from the interval $[0, \min(\Lambda_{\max}, n_k)]$ where n_k indicates the number of customers and recharging stations in route r_k, and the starting vertex is chosen randomly among customers and recharging stations that allow the fulfillment of the translocation chain length. Before we generate a random point, we reduce κ to one if the number of routes in solution S is smaller than $\#Rts$, and only modify S in our VNS phase if the number of routes or vehicles m is larger than one.

Table 1: All κ-neighborhood structures for the VNS

κ	$\#Rts$	$\Lambda_{\max}$	κ	$\#Rts$	$\Lambda_{\max}$	κ	$\#Rts$	$\Lambda_{\max}$
1	2	1	6	3	1	11	4	1
2	2	2	7	3	2	12	4	2
3	2	3	8	3	3	13	4	3
4	2	4	9	3	4	14	4	4
5	2	5	10	3	5	15	4	5

After our VNS phase, we apply Tabu Search to enhance the randomly generated solution S', obtaining solution S''. We then compare the solution S'' found at the end of the TS to the current solution S. We use an acceptance criterion based on the metaheuristic SA (Kirkpatrick et al., 1983) instead of only accepting improving solutions as to diversify our search further. If a

solution improves the current solution, we always accept it. Otherwise, we accept it according to probability $e^{(-(f(S'')-f(S)))/T}$. After obtaining our first initial solution, we initialize temperature T in a way such that a total traveled distance $f(S'')$ which is Δ_{SA} worse than $f(S)$ of the initial solution, is accepted 50% of the time. This means that at the beginning, a worse solution is often accepted, intended to diversify the search. Then, after the feasibility phase is over, we decrease temperature T after each VNS iteration by a constant such that it is below 0.0001 during the last 20% of iterations to decrease total traveled distance. Once T becomes negative, we only accept improving solutions, achieving an intensification at each iteration to improve distance.

4.4 The Tabu Search

The Tabu Search component starts with the solution S' generated by the random perturbation move of the VNS phase. It lasts for η_{tabu} iterations. In each iteration, a solution S^* is generated by applying the best non tabu move in the list of moves generated by applying the following neighborhood operators on the current interim solution: 2-opt*, relocate, exchange, and the problem-specific operator stationInRe introduced in Schneider et al. (2014). A move with a lower cost based on the generalized cost function is superior, but a move that reduces the number of vehicles is chosen over one that does not. While not explained in Schneider et al. (2014), we will assume a move reduces the number of vehicles m if it creates a feasible solution where a route is changed such that it only contains charging stations or depot vertices.

The 2-opt* neighborhood operator was proposed by Potvin and Rousseau (1995) specifically for VRPTW unlike the initially introduced 2-opt operator (S. Lin, 1965). Here, two new routes are created by linking the first half of one route, with the second half of another, and vice versa, and thus avoiding route direction reversal. This operator is defined for interroute moves, and we allow for moves to insert and remove any arcs, including recharging stations. The relocate neighborhood operator (Savelsbergh, 1992) moves a vertex from one position to another, albeit in the same route or another. This operator is defined for both intra- and interroute moves, and is applied to both customers and recharging stations. The exchange neighborhood operator (Savelsbergh, 1992) exchanges the location of two vertices. This operator is applied to both intra- and interroute moves, but we do not allow the swapping of a customer with a recharging station, or two recharging stations with one another. Finally, stationInRe (Schneider et al., 2014) either adds a charging station v prior to vertex w, or if it already exists prior to w, removes it.

An arc ε which is removed from a solution through the application of a move is set as tabu, i.e., we do not allow the reinsertion of this arc into a specific part of the solution for a given number of tabu iterations, the tabu tenure. Each move, we randomly draw this tabu tenure ϑ from the interval $[\vartheta_{\min}, \vartheta_{\max}]$ for each arc ε. As station visits strongly affect charge levels as well as time windows due to the time it takes to recharge, we define the tabu attribute $(\varepsilon, k, \mu, \zeta)$. This attribute forbids the insertion of arc ε in route r_k, between station or depot $\mu, \zeta \in F_{0,N+1}$. This way we allow the reinsertion of the arc in other parts of the route or solution. If a move would lead to a feasible solution with a better result according to our definition compared to our current solution S, we allow the insertion of an arc even if it were tabu. Each time we restart the Tabu Search phase, we clear our list of tabu arcs.

We further diversify the search by utilizing a continuous diversification mechanism based

on Cordeau et al. (2001). For this purpose, we characterize solutions through vertex-based attributes (u, k, μ, ζ). A solution S is made up out of the attribute set $B(S) = \{u, k, \mu, \zeta\}$ where customer or station u in route r_k is positioned between station or depot $\mu, \zeta \in F_{0,N+1}$. Per Tabu Search phase, we save the frequency $p_{uk\mu\zeta}$ that an attribute has been added to an interim solution. If solution S^* would be worse than the previous solution in the Tabu Search phase, we penalize the generalized cost function for the purpose of the comparison to other moves with

$$P_{\text{div}}(S^*) = \lambda_{\text{div}} \cdot f(S^*) \cdot \sqrt{|V^*|m} \sum_{(u,k,\mu,\zeta) \in B(S^*)} p_{uk\mu\zeta}$$

where λ_{div} is the diversification factor to control the extent of diversification. We use a scaling factor $f(S^*) \cdot \sqrt{|V^*|m}$ to provide a relation between the penalization and the traveled distance as well as instance size ($|V^*|$ indicates the number of customers and recharging station visits in solution S^*). Through this diversification, we explore possibilities of using other stations and other positions for customers and stations in a route.

4.5 Non-Linear Recharging

The recharging function used in Schneider et al. (2014) is linear in time. However, as previously stated, this is not aligned with reality, where time to charge increases at the last 10-20% of the battery capacity (Marra et al., 2012). Montoya et al. (2017) extended the classical E-VRP by considering a non-linear charging function and obtained results using an Iterated Local Search with Variable Neighbourhood Descent, and improves the solutions using a heuristic concentration where a set partitioning problem is solved by using a pool of routes found previously. We intend to expand Schneider et al. (2014)'s hybrid heuristic for the E-VRPTW of VNS and TS, instead of descent, where a worse solution can be accepted for further VNS/TS iterations. For this purpose, we allow for non linear recharge functions when calculating the time window slacks in 4.2

5 Results

All experiments are run on an AMD Ryzen 7 5800X Processor clocked at 3.8 GHz computer with 32 GB RAM, running Windows 10 Professional. We implement the VNS/TS as single thread code in Java. Our machine has a clock speed that is 42% higher than the one used to obtain run time results in Schneider et al. (2014). We will therefore assume that the run time results would decrease by 30% if ran again using our system for the point of comparison.

5.1 Parameter Settings

Our parameter settings are the same as those of Schneider et al. (2014). While not elaborated upon in Schneider et al. (2014), we take the minimum number of routes found using VNS/TS in Schneider et al. (2014) for a given instance as the number of routes m at the beginning of a search for that instance. These settings are used for all following experimental studies.

5.2 Test instances

We use two sets of benchmark instances introduced in Schneider et al. (2014) (see Goeke, 2019). The first set is made up out of 56 large instances, with 100 customers per instance, as well as 21 recharging stations. The second set includes 36 small instances, with 12 instances of 5, 10, and 15 customers each. Within both sets, there are three classes based on geographical distribution of customer locations, namely random customer distribution (R), clustered customer distribution (C), and a mixture of the two (RC). Additionally, the sets have two groups, namely group one with a short scheduling horizon, i.e., more vehicles are generally required to serve customers compared to those in group two, which have a long scheduling horizon. A large instance will be denoted first by their class, then group, and finally an instance number. As an example, the eighth large instance of class R and group 1 will be denoted as R108. As the small instances were constructed from the large instances, they will be denoted by adding an identifier of the customer size after the respective altered large instance, e.g. R108-5 was constructed from R108 and has 5 customers.

When considering non linear recharging functions, we adapt the instances as follows. Montoya et al. (2017) presents the reader with three piecewise linear approximation functions based on a slow, normal, and fast recharging time based on real life data. For simplicity, we keep one mode of recharging per experiment, namely normal recharging, but scale their functions based on the given instance.

There are multiple ways to approximate recharging to a linear function. We will consider two options, and obtain piecewise functions based on these. First, we will consider the assumption that the first 0-85% of battery recharging continues for the remaining 15%. This is an optimistic interpretation as a full recharge would be quicker than in reality. Second, we will consider that a full recharge does take the same amount of time as with non-linear recharging. This is a pessimistic approximation, as for a good portion of time, the recharge rate would be faster in reality, and only slow down near the end.

Based on this, we use two variations of piecewise linear functions. Case (1) where 0-85% of battery capacity is charged at the point where linear recharging would be too, but the time to 95% and time to 100% are scaled based on the percentage difference in time breakpoints of the normal recharging in Montoya et al. (2017). The original breakpoint to get to 85% battery capacity is 61% of time; to 95% battery capacity is 76% of time, and to 100% of battery capacity is 100% of time. Therefore, in case (1), the jump from 85% battery capacity to 95% should lead to an increase of a factor $76/61 = 1.25$ in time, which would be $85\% \cdot 1.25 = 105.56\%$ of the original time we would expect to be fully charged. Similarly, in case (2), we assume that recharging from 0-100% takes the original amount of time, to 95% would be 76% of time, and to 85% would be 61% of total time. Figure 1 depicts the different function types that will be used. The charging time from any battery level can be calculated by taking the difference between the time the battery level is at 100% and the time the battery would be at the given battery level using the different functions.

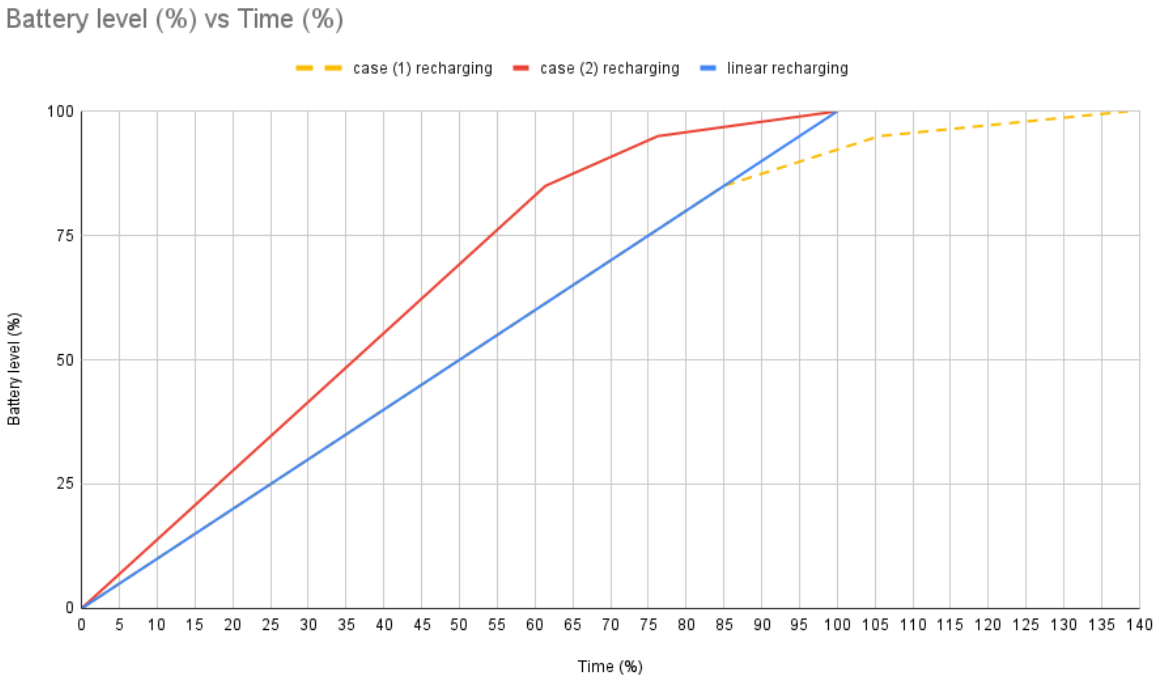

Figure 1: Linear and nonlinear recharging

5.2.1 Performance on small instances

We compare our results of using the VNS/TS heuristic on the small instances with results found by Schneider et al. (2014) using both CPLEX and the VNS/TS heuristic. Table 2 summarizes the results. Columns m and f show the best solutions found within 10 runs, and $t(s)$ shows the runtime in seconds to obtain this solution. The gaps $\Delta_{\text{CPLEX}}f(\%)$ and $\Delta_{2014}f(\%)$ denote the percentage difference in results compared to the results of CPLEX and VNS/TS in Schneider et al. (2014) respectively. While for RC108-5 Schneider et al. (2014) reported one vehicle found using both CPLEX and the VNS/TS heuristic, Erdelić et al. (2019), Andelmin (2014), and Keskin and Çatay (2016) all reported 2 vehicles. For our comparison, we consider the number of reported vehicles with the correction for RC108-5. This leads to us having no gap in the number of employed vehicles, which are therefore not reported.

As can be seen in the table, we find no difference in the number of vehicles or distance compared to Schneider et al. (2014). Even for RC204-15, the only instance for which the VNS/TS hybrid found a better result than CPLEX, we find the same results.

Regarding time, while we cannot make a completely fair assessment, as the processors used are different, we can see that the heuristic is slower in our experiments than in Schneider et al. (2014), whereas based on the processors, the opposite would be expected. This likely has to do with a worse implementation of the algorithm. It cannot be said for certain why this is the case, as the methodology of Schneider et al. (2014) was replicated. Therefore, it is logical to assume that parts not explained in the prior mentioned paper were implemented differently in Java. For example, the frequency of attributes is reproduced completely for routes that are changed in the tabu phase, whereas this could potentially be restricted to the attributes that would have

Table 2: Comparison of results of the VNS/TS for small instances to Schneider et al. (2014)

Instance	m	f	$\Delta_{\text{CPLEX}} f(\%)$	$\Delta_{2014} f(\%)$	$t(s)$
C101-5	2	257.75	0.00	0.00	3.75
C103-5	1	176.05	0.00	0.00	1.33
C206-5	1	242.56	0.00	0.00	2.98
C208-5	1	158.48	0.00	0.00	1.82
R104-5	2	136.69	0.00	0.00	2.28
R105-5	2	156.08	0.00	0.00	2.70
R202-5	1	128.78	0.00	0.00	2.06
R203-5	1	179.06	0.00	0.00	3.00
RC105-5	2	241.30	0.00	0.00	4.16
RC108-5	2	253.93	0.00	0.00	11.91
RC204-5	1	176.39	0.00	0.00	2.44
RC208-5	1	167.98	0.00	0.00	1.78
C101-10	3	393.76	0.00	0.00	19.34
C104-10	2	273.93	0.00	0.00	15.73
C202-10	1	304.06	0.00	0.00	13.22
C205-10	2	228.28	0.00	0.00	16.20
R102-10	3	249.19	0.00	0.00	15.17
R103-10	2	207.05	0.00	0.00	12.48
R201-10	1	241.51	0.00	0.00	12.71
R203-10	1	218.21	0.00	0.00	12.65
RC102-10	4	423.51	0.00	0.00	12.49
RC108-10	3	345.93	0.00	0.00	14.94
RC201-10	1	412.86	0.00	0.00	14.74
RC205-10	2	325.98	0.00	0.00	20.15
C103-15	3	384.29	0.00	0.00	44.27
C106-15	3	275.13	0.00	0.00	34.55
C202-15	2	383.62	0.00	0.00	49.10
C208-15	2	300.55	0.00	0.00	39.94
R102-15	5	413.93	0.00	0.00	43.11
R105-15	4	336.15	0.00	0.00	38.91
R202-15	2	358.00	0.00	0.00	50.40
R209-15	1	313.24	0.00	0.00	33.06
RC103-15	4	397.67	0.00	0.00	42.17
RC108-15	3	370.25	0.00	0.00	32.79
RC202-15	2	394.39	0.00	0.00	50.82
RC204-15	1	384.86	-5.61	0.00	35.42
Avg.			-0.16	0.00	19.85

Note. m *is the number of vehicles,* f *the traveled distance.* $t(s)$ *is the runtime in seconds.*

actually changed. Additionally, a less efficient usage of data structures may have caused an expensive increase in running time. Any inefficiency compared to the code of Schneider et al. (2014) has an explosive effect on the running time and performance of the heuristic due to the many iterations. This is not a fault of the hybrid-heuristic, but instead a failure in our coding quality. Nonetheless, even if we decreased the time required in CPLEX by the prior mentioned 30% for all instances, the hybrid heuristic still finds results faster for all but instances R104-5, R105-5, R202-5, and C205-10. In the grand scheme of things, these instances remain quite small, which could also indicate that the finding of initial solutions is done inefficiently, as that becomes a larger part of the running time in smaller instances if feasibility is found in a similar amount of time. Based on these results, it can be concluded that while the algorithm is potentially very efficient, even the finer, and unmentioned, details heavily affect its performance.

5.2.2 Performance on large instances

We also compare our results of using the VNS/TS heuristic on the results retrieved for large instances using the VNS/TS hybrid in Schneider et al. (2014). However, the prior mentioned difference in runtime performance has an explosive effect on the runtime for the larger instances. While in Schneider et al. (2014) the runtime changes from an average of 5.03 seconds for the small instances to 16.22 minutes for the large instances, which is 193 times as much as the prior runtime, the runtime changes from an average of of 19.85 seconds for the small instances to 2.95 hours in the the large instances using our code, which is 535 times as much than in the small instances. This disproportionate increase can be explained through both increases in number of customers, and overall number of routes, being handled worse due to worse code implementation.

Due to this fact, the number of trials we run need to be reduced, as it would take an enormous amount of time to run each instance 10 times. Therefore, we compare one run for each instance to the 10 runs of the VNS/TS heuristic in Schneider et al. (2014). Results for this can be found in Table 3. At the bottom of the table, the total difference in number of vehicles, average difference in traveled distance, and average runtime are denoted.

The conclusions that can be drawn here are regrettably rather limited. Any difference could in theory be blamed on a difference in number of runs, as the results can differ between results due to the random elements within the algorithm, and results of individual runs are not reported. In a limited number of instances, i.e., C101, C201, C203, C204, C205, C206, C207, and C208, the results are the same, which shows that our interpretation of the algorithm is able to find the same results as in Schneider et al. (2014) for at least some instances. In one instance, R211, we find a better result than Schneider et al. (2014) using VNS/TS, but worse than the best known solution found in the same paper. This suggests that 10 runs do not guarantee that the heuristic finds the best result it could.

In all other cases, we either find worse results, or no feasible results at all. The latter of which can be recognized by a lack of noted traveled distance; the heuristic was able to leave the feasibility phase, but it did not finish with a feasible solution. There are two explanations that could the reason for these differences, or potentially a combination. One, as mentioned previously, is the number of runs. The initially generated solution, as well as the perturbation move, and acceptance criterion, includes elements which are random. This leads to the possibility

Table 3: Comparison of results of the VNS/TS for large instances to Schneider et al. (2014)

Instance	m	f	Δ_m	$\Delta_f(\%)$	t(hour)
C101	12	1,053.83	0	0.00	1.64
C102	11	1,075.51	0	1.74	1.95
C103	11	1,002.01	1	-3.80	5.34
C104	10	1,013.25	0	3.31	1.93
C105	11	1,086.50	0	1.03	1.56
C106	11	1,066.60	0	0.83	1.64
C107	11	1,060.73	0	2.83	1.42
C108	11	1,063.81	1	-3.32	4.81
C109	11	1,080.60	0	2.73	4.18
C201	4	645.16	0	0.00	1.43
C202	4	646.52	0	0.21	1.18
C203	4	644.98	0	0.00	1.16
C204	4	636.43	0	0.00	1.00
C205	4	641.13	0	0.00	1.03
C206	4	638.17	0	0.00	1.13
C207	4	638.17	0	0.00	0.96
C208	4	638.17	0	0.00	1.14
R101	19	N/A	1	N/A	12.01
R102	18	N/A	2	N/A	14.03
R103	14	N/A	1	N/A	11.04
R104	12	N/A	1	N/A	3.99
R105	16	N/A	2	N/A	6.52
R106	14	1,349.64	1	0.37	4.84
R107	13	1,229.00	1	6.45	3.34
R108	12	N/A	1	N/A	4.44
R109	13	1,295.58	1	0.12	6.00
R110	12	1,138.01	1	-0.48	5.75
R111	13	N/A	1	N/A	5.19
R112	11	N/A	0	N/A	2.43
R201	3	1300.31	0	2.81	0.98
R202	3	N/A	0	N/A	1.12
R203	3	917.08	0	0.46	1.83
R204	2	811.42	0	2.64	1.36
R205	3	1,018.13	0	2.98	1.00
R206	3	940.31	0	1.63	0.98
R207	2	904.00	0	6.01	1.02
R208	2	738.26	0	0.23	0.94
R209	3	899.90	0	3.16	1.07
R210	3	861.48	0	1.70	1.10
R211	2	853.94	0	-1.42	1.01
RC101	17	N/A	1	N/A	4.44
RC102	15	1,584.97	0	1.95	2.40
RC103	13	N/A	0	N/A	3.31
RC104	12	1,238.66	1	-0.85	4.29
RC105	14	1,485.70	0	0.16	2.70
RC106	13	N/A	0	N/A	3.84
RC107	12	1,282.21	0	0.50	2.49
RC108	12	N/A	1	N/A	4.71
RC201	4	1,464.13	0	1.17	0.98
RC202	3	1,461.41	0	3.43	0.88
RC203	3	1,095.69	0	1.61	1.03
RC204	3	949.05	0	6.73	1.03
RC205	3	N/A	0	N/A	4.03
RC206	3	1,247.14	0	4.70	0.97
RC207	3	1,049.93	0	5.47	1.05
RC208	3	899.70	0	7.35	0.91
Sum/Avg.			18	1.50	2.94

Note. m is the number of vehicles, f the traveled distance, Δ_m the gap in number of vehicles, and $\Delta_f(\%)$ in the traveled distance, both compared to the results of the VNS/TS heuristic in Schneider et al. (2014). $t(hour)$ is the runtime in hours.

of differences between runs, which could be to blame for the difference in results. Second, the difference in results could be explained due to the differences in methodology. While we retrieved the same results for the small instances, we cannot conclude that our methodology is wholly the same as in Schneider et al. (2014).

Elements of Schneider et al. (2014) could have been misinterpreted. First, we decided to increase our violation weights after η_{penalty} consecutive iterations where the respective constraint was violated, and decreased if the respective constraints where met for η_{penalty} iterations. Another way their description could be interpreted is that while an increase happens after η_{penalty} consecutive iterations, a decrease happens whenever the constraints are met. This could lead us to including less diversification within our procedure, which could lead to worse results using the same parameter settings. Another point of confusion was found regarding the intensification of results using the search temperature. Schneider et al. (2014) described the temperature being below 0.0001 during the last 20% of iterations, however, it is unclear what iterations means in this context. The amount of feasibility iterations is not defined from the start, meaning the total number of iterations is neither. We therefore decided to interpret this as distance iterations, which are known from the start, but another meaning could lead to a once again misshapen balance between intensification and diversification. Finally, in Schneider et al. (2014), the generalized cost function includes $P_{\text{div}}(S)$, but is only described when comparing tabu moves within one tabu iteration, which is how we have used this diversity penalty. However, the way the paper is written, it could be that even across iterations this diversity penalty should have been included, which would lead to iterations succeeding those including the penalty becoming penalized less often, as they do not deteriorate the current solution as much. This would greatly affect which moves are deemed superior, and would lead to less diverse moves. Generally, it seems that there could be a difference within the balance of intensification and diversification.

5.2.3 Effect of non-linear recharging on small instances

We will research how transforming different interpretations of a linear recharging approximation to functions closer to reality affect the results, mainly in number of vehicles required, costs, and heuristic performance. In either case it would seem that longer recharges will be preferred compared to the original linear situation, as short recharges take a relatively longer amount of time. Additionally, in the prior mentioned case (1), due to the longer time required to fully recharge, we expect time windows to be harder to meet than in case (2), which could lead to a bigger increase in number of vehicles or distance due to new detours.

Due to the slow performance running large instances compared to Schneider et al. (2014) as mentioned previously, we also limit ourselves in the research the effect of non-linear recharging. Namely, by only investigating the effect on the small instance. Table 4 depicts the best feasible results in 10 runs using the VNS/TS hybrid with the altered instances with non-linear recharging. At the bottom of the table the total difference in number of vehicles, average difference in traveled distance, and average runtime are are once again denoted.

Multiple phenomena can be recognized within these results. First, we can see that some results remain the same as in the linear case. This can occur when solutions are not very sensitive to delays, i.e., there are gaps of time between the arrival to a destination, and the

latest possible service start time. Alternatively, recharging is not very prevalent in a route or solution. However, arguably the biggest characteristic of these results is the opposite: 52 out of 72 results do not match with the linear cases, either in number of vehicles, or distance. This is to say that using the same routing plan in a situation where recharging is not linear, like in reality, may lead to the inability to meet time windows. This is an important consideration if the algorithm were to be used for real life scenarios. However, there is of course still the limitation that the algorithm assumes charging always needs to be concluded at 100% of the battery capacity, which is an overly restrictive constraint.

Additionally, we can see a difference in performance of the algorithm when ran using linear recharging and non-linear recharging. First, when the number of vehicles does not remain the same, the runtime noticeably exceeds that of the original. This has to do with the fact that the feasibility phase is extended, as a feasible solution may no longer be able to be found for the starting value m. This points to the fact that the starting m has a strong effect on runtime. Furthermore, we can see that for some instances of the two cases, no feasible solution is returned whatsoever in any run, which is why no traveled distance is denoted. While we can say that a feasible solution was found during the process for the given number of vehicles, as the algorithm remains in the feasibility phase until that is the case, the final solution included violation penalties. This implies that while not using the best parameter settings may not have a detrimental effect on the VNS/TS performance with linear recharging (see Schneider et al., 2014), it may in the non-linear case. This suggests that parameter tuning would have to be redone for each case following the steps described in Schneider et al. (2014) using the large instances. However, this does not guarantee feasible results for a final solution. Due to the random nature of the acceptance criterion, and tabu phase, which allow for solutions to become worse for the sake of diversity, a feasible solution may not be rediscovered after exiting the feasibility phase. A potential extension to the algorithm could be a sort of 'fall back' to feasible solutions found prior, if no feasible solution is found for a certain amount of iterations in the distance phase. Another way to steer away from results with big violation penalties is to compare the generalized cost functions in the acceptance criterion, instead of the distances, with a correction to counteract the difference in penalty factors.

Table 4: Results of the VNS/TS for small instances with two cases of non-linear recharging

	Case 1					Case 2				
Instance	m	f	Δ_m	$\Delta_f(\%)$	$t(s)$	m	f	Δ_m	$\Delta_f(\%)$	$t(s)$
C101-5	3	247.15	1	-4.11	8.27	2	268.1	0	4.02	2.61
C103-5	2	165.67	1	-5.90	3.41	1	N/A	0	N/A	0.64
C206-5	1	265.97	0	9.65	2.50	1	254.13	0	4.77	2.69
C208-5	1	164.34	0	3.70	1.55	1	164.34	0	3.70	1.63
R104-5	2	136.69	0	0.00	2.04	2	136.69	0	0.00	2.09
R105-5	3	182.92	1	17.20	8.54	3	182.92	1	17.20	8.63
R202-5	1	128.78	0	0.00	2.07	1	128.78	0	0.00	2.09
R203-5	1	179.06	0	0.00	3.01	1	179.06	0	0.00	2.96
RC105-5	3	238.05	1	-1.35	11.00	2	243.06	0	0.73	4.12
RC108-5	3	316.51	1	24.64	20.45	2	253.93	0	0.00	11.17
RC204-5	1	176.39	0	0.00	2.44	1	176.39	0	0.00	2.41
RC208-5	1	167.98	0	0.00	1.81	1	167.98	0	0.00	1.83
C101-10	4	N/A	1	N/A	59.43	3	410.28	0	4.20	19.26
C104-10	2	293.64	0	7.20	13.58	2	287.83	0	5.07	14.45
C202-10	2	250.83	1	-17.51	55.44	1	N/A	0	N/A	15.38
C205-10	2	290.90	0	27.43	14.36	2	290.90	0	27.43	14.66
R102-10	4	262.92	1	5.51	49.09	3	N/A	0	N/A	17.72
R103-10	2	210.60	0	1.71	12.96	2	209.67	0	1.27	11.55
R201-10	2	228.36	1	-5.44	55.32	2	228.36	1	-5.44	56.15
R203-10	1	N/A	0	N/A	14.70	1	N/A	0	N/A	14.59
RC102-10	5	436.05	1	2.96	38.74	5	436.05	1	2.96	45.39
RC108-10	4	395.45	1	14.32	50.97	3	415.40	0	20.08	16.4
RC201-10	2	333.29	1	-19.27	58.47	2	333.29	1	-19.27	60.77
RC205-10	2	325.98	0	0.00	20.63	2	325.98	0	0.00	21.24
C103-15	3	429.44	0	11.75	41.56	3	391.45	0	1.86	39.31
C106-15	3	328.92	0	19.55	26.38	3	310.52	0	12.86	31.25
C202-15	3	403.63	1	5.22	185.94	2	415.11	0	8.21	51.31
C208-15	2	300.55	0	0.00	35.00	2	300.55	0	0.00	38.03
R102-15	5	459.43	0	10.99	58.54	5	438.16	0	5.85	52.28
R105-15	4	402.95	0	19.87	44.03	4	378.89	0	12.71	42.23
R202-15	2	367.85	0	2.75	54.04	2	367.85	0	2.75	52.61
R209-15	1	N/A	0	N/A	36.85	1	313.24	0	0.00	35.94
RC103-15	4	397.67	0	0.00	32.66	4	397.67	0	0.00	38.12
RC108-15	4	408.30	1	10.28	123.04	3	387.61	0	4.69	32.73
RC202-15	2	394.39	0	0.00	53.58	2	394.39	0	0.00	51.76
RC204-15	1	407.18	0	5.80	40.27	1	N/A	0	N/A	42.91
Sum/Avg.			14	4.45	34.52			4	3.73	23.86

Note. m *is the number of vehicles,* f *the traveled distance,* Δ_m *the gap in number of vehicles, and* $\Delta_f(\%)$ *in distance, both compared to the linear case.* $t(s)$ *is the runtime in seconds.*

6 Conclusion

In this paper, we analyse the performance of the VNS/TS hybrid meta heuristic for the purpose of electric vehicle routing problems with time windows and recharging, as well as an extension which includes non-linear recharging. We find that while for the tested small data instances we find the same results using the meta heuristic, the runtime performance is a lot worse. The same trend can be noticed in the runtime performance for the large instances that are tested. This likely has to do with a worse code implementation in unmentioned respects in Schneider et al. (2014). Running large instances, we for the most part find worse results, however, this could be attributed to the fact that a lesser number of trials are ran. Alternatively, this may have to do with the misinterpretation of methodology in the prior mentioned paper. Finally, when testing out two different cases of non-linear recharging, we find that for over half of the tested instances, the results for linear recharging cannot be used. Additionally, we find that the hybrid heuristic cannot be used to obtain feasible results directly just by implementing non-linear recharging. While Schneider et al. (2014) mentions that parameter settings likely would not have a detrimental effect on the VNS/TS performance with linear recharging, running the problem with non-linear recharging, feasible results are not found within 10 runs in multiple instances.

For future research, one could extend the model further to include 'fall backs' to feasible solutions if a feasible solution is not found for a certain amount of iterations. Alternatively, when moving from one solution to another, the acceptance of a new solution could be changed to be affected by how much the solutions violate feasibility, to steer towards solutions that are closer to feasibility. Finally, the model could be extended to include partial recharging, as this has a bigger impact when considering non-linear recharging due to its slowing down near the end of a recharge.

References

Andelmin, J. (2014). *Optimal routing of electric vehicles* (Unpublished master's thesis). Aalto University.

Baldacci, R., Mingozzi, A. & Roberti, R. (2012). Recent exact algorithms for solving the vehicle routing problem under capacity and time window constraints. *European Journal of Operational Research*, *218*(1), 1–6.

Bektaş, T., Ehmke, J. F., Psaraftis, H. N. & Puchinger, J. (2019). The role of operational research in green freight transportation. *European Journal of Operational Research*, *274*(3), 807-823. Retrieved from `https://www.sciencedirect.com/science/article/pii/S0377221718305241` doi: https://doi.org/10.1016/j.ejor.2018.06.001

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

Botsford, C. & Szczepanek, A. (2009). Fast charging vs. slow charging: Pros and cons for the new age of electric vehicles. In *International battery hybrid fuel cell electric vehicle symposium* (pp. 1–9).

Bräysy, O. & Gendreau, M. (2005). Vehicle routing problem with time windows, part i: Route construction and local search algorithms. *Transportation science*, *39*(1), 104–118.

Bruglieri, M., Pezzella, F., Pisacane, O. & Suraci, S. (2015). A variable neighborhood search branching for the electric vehicle routing problem with time windows. *Electronic Notes in Discrete Mathematics*, *47*, 221-228. Retrieved from `https://www.sciencedirect.com/science/article/pii/S1571065314000717` (The 3rd International Conference on Variable Neighborhood Search (VNS'14)) doi: https://doi.org/10.1016/j.endm.2014.11.029

Conrad, R. G. & Figliozzi, M. A. (2011). The recharging vehicle routing problem. In *Proceedings of the 2011 industrial engineering research conference* (Vol. 8).

Cordeau, J.-F., Laporte, G. & Mercier, A. (2001). A unified tabu search heuristic for vehicle routing problems with time windows. *Journal of the Operational research society*, *52*(8), 928–936.

Dekker, R., Bloemhof, J. & Mallidis, I. (2012). Operations research for green logistics–an overview of aspects, issues, contributions and challenges. *European journal of operational research*, *219*(3), 671–679.

Demir, E., Bektaş, T. & Laporte, G. (2014). The bi-objective pollution-routing problem. *European Journal of Operational Research*, *232*(3), 464-478. Retrieved from `https://www.sciencedirect.com/science/article/pii/S0377221713006486` doi: https://doi.org/10.1016/j.ejor.2013.08.002

Desaulniers, G., Errico, F., Irnich, S. & Schneider, M. (2016). Exact algorithms for electric vehicle-routing problems with time windows. *Operations Research*, *64*(6), 1388–1405.

Ding, N., Batta, R. & Kwon, C. (2015). Conflict-free electric vehicle routing problem with capacitated charging stations and partial recharge. *SUNY, Buffalo*.

EC. (2023). *2030 climate target plan, (2023).* Retrieved from `https://climate.ec.europa.eu/eu-action/european-green-deal/2030-climate-target-plan_en`

EEA. (2023). *European environment agency: Greenhouse gas data—emissions share by sector in eu27, (2021).* Retrieved May 3rd 2023, from `https://www.eea.europa.eu/data-and`

-maps/data/data-viewers/greenhouse-gases-viewer

Erdelić, T., Carić, T., Erdelić, M. & Tišljarić, L. (2019). Electric vehicle routing problem with single or multiple recharges. *Transportation Research Procedia*, *40*, 217–224.

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

Feng, W. & Figliozzi, M. (2013). An economic and technological analysis of the key factors affecting the competitiveness of electric commercial vehicles: A case study from the usa market. *Transportation Research Part C: Emerging Technologies*, *26*, 135–145.

Figliozzi, M. (2010). Vehicle routing problem for emissions minimization. *Transportation Research Record*, *2197*(1), 1–7.

Gendreau, M. & Tarantilis, C. D. (2010). *Solving large-scale vehicle routing problems with time windows: The state-of-the-art.* Cirrelt Montreal.

Glover, F. & Laguna, M. (1998). *Tabu search.* Springer.

Goeke, D. (2019). *E-vrptw instances.* (Version 1) [Data set]. doi: https://doi.org/10.17632/h3mrm5dhxw.1

Goeke, D. & Schneider, M. (2015). Routing a mixed fleet of electric and conventional vehicles. *European Journal of Operational Research*, *245*(1), 81-99. Retrieved from https://www.sciencedirect.com/science/article/pii/S0377221715000697 doi: https://doi.org/10.1016/j.ejor.2015.01.049

Gonçalves, F., Cardoso, S. R., Relvas, S. & Barbosa-Póvoa, A. (2011). Optimization of a distribution network using electric vehicles: A vrp problem. In *Proceedings of the io2011-15 congresso da associação portuguesa de investigação operacional, coimbra, portugal* (Vol. 2011, pp. 18–20).

Hiermann, G., Hartl, R. F., Puchinger, J. & Vidal, T. (2019). Routing a mix of conventional, plug-in hybrid, and electric vehicles. *European Journal of Operational Research*, *272*(1), 235–248.

Ibaraki, T., Imahori, S., Kubo, M., Masuda, T., Uno, T. & Yagiura, M. (2005). Effective local search algorithms for routing and scheduling problems with general time-window constraints. *Transportation science*, *39*(2), 206–232.

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.

Kindervater, G. & Savelsbergh, M. (1997). *Local search in combinatorial optimization, chapter vehicle routing: handling edges exchanges windows.* John Wiley & Sons, Chichester, USA.

Kirkpatrick, S., Gelatt Jr, C. D. & Vecchi, M. P. (1983). Optimization by simulated annealing. *science*, *220*(4598), 671–680.

Kleindorfer, P. R., Singhal, K. & Van Wassenhove, L. N. (2005). Sustainable operations management. *Production and Operations Management*, *14*(4), 482-492. Retrieved from https://onlinelibrary.wiley.com/doi/abs/10.1111/j.1937-5956.2005.tb00235.x doi: https://doi.org/10.1111/j.1937-5956.2005.tb00235.x

Lin, J., Zhou, W. & Wolfson, O. (2016). Electric vehicle routing problem. *Transportation research procedia*, *12*, 508–521.

Lin, S. (1965). Computer solutions of the traveling salesman problem. *Bell System Technical Journal*, *44*(10), 2245–2269.

Li-ying, W. & Yuan-bin, S. (2015). Multiple charging station location-routing problem with time window of electric vehicle. *Journal of Engineering Science & Technology Review*, *8*(5).

Marra, F., Yang, G. Y., Træholt, C., Larsen, E., Rasmussen, C. N. & You, S. (2012). Demand profile study of battery electric vehicle under different charging options. In *2012 ieee power and energy society general meeting* (pp. 1–7).

McKinnon, A., Browne, M., Whiteing, A. & Piecyk, M. (2015). *Green logistics: Improving the environmental sustainability of logistics*. Kogan Page Publishers.

Melechovskỳ, J., Prins, C. & Calvo, R. W. (2005). A metaheuristic to solve a location-routing problem with non-linear costs. *Journal of Heuristics*, *11*, 375–391.

Mladenović, N. & Hansen, P. (1997). Variable neighborhood search. *Computers & operations research*, *24*(11), 1097–1100.

Montoya, A., Guéret, C., Mendoza, J. E. & Villegas, J. G. (2017). The electric vehicle routing problem with nonlinear charging function. *Transportation Research Part B: Methodological*, *103*, 87–110.

Nagata, Y., Bräysy, O. & Dullaert, W. (2010). A penalty-based edge assembly memetic algorithm for the vehicle routing problem with time windows. *Computers & operations research*, *37*(4), 724–737.

Pelletier, S., Jabali, O. & Laporte, G. (2016). 50th anniversary invited article—goods distribution with electric vehicles: review and research perspectives. *Transportation science*, *50*(1), 3–22.

Potvin, J.-Y. & Rousseau, J.-M. (1995). An exchange heuristic for routeing problems with time windows. *Journal of the Operational Research Society*, *46*(12), 1433–1446.

Sassi, O., Cherif, W. R. & Oulamara, A. (2015). Vehicle routing problem with mixed fleet of conventional and heterogenous electric vehicles and time dependent charging costs. *International Journal of Mathematics and Computer Science*.

Savelsbergh, M. W. (1992). The vehicle routing problem with time windows: Minimizing route duration. *ORSA journal on computing*, *4*(2), 146–154.

Sbihi, A. & Eglese, R. W. (2010). Combinatorial optimization and green logistics. *Annals of Operations Research*, *175*, 159–175.

Schiffer, M. & Walther, G. (2017). The electric location routing problem with time windows and partial recharging. *European journal of operational research*, *260*(3), 995–1013.

Schiffer, M. & Walther, G. (2018). An adaptive large neighborhood search for the location-routing problem with intra-route facilities. *Transportation Science*, *52*(2), 331–352.

Schneider, M., Sand, B. & Stenger, A. (2013). A note on the time travel approach for handling time windows in vehicle routing problems. *Computers & Operations Research*, *40*(10), 2564–2568.

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

Srivastava, S. K. (2007). Green supply-chain management: a state-of-the-art literature review.

International journal of management reviews, *9*(1), 53–80.

Tarantilis, C. D., Zachariadis, E. E. & Kiranoudis, C. T. (2008). A hybrid guided local search for the vehicle-routing problem with intermediate replenishment facilities. *INFORMS Journal on computing*, *20*(1), 154–168.

Thompson, P. M. & Orlin, J. B. (1989). *The theory of cyclic transfers* (Working Paper). MA: Operations Research Center, MIT.

Toth, P. & Vigo, D. (2002). *The vehicle routing problem*. SIAM.

Tredeau, F. P. & Salameh, Z. M. (2009). Evaluation of lithium iron phosphate batteries for electric vehicles application. In *2009 ieee vehicle power and propulsion conference* (pp. 1266–1270).

Ángel Felipe, Ortuño, M. T., Righini, G. & Tirado, G. (2014). 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. Retrieved from `https://www.sciencedirect.com/science/article/pii/S1366554514001574` doi: https://doi.org/10.1016/j.tre.2014.09.003

A Programming code

There are five classes that are used in the programming code. These classes are: **Main**, **GenerateFunctionValues**, **Attribute**, **TabuAttribute**, and **TxtReader**. The bulk of the implementation is part of **Main**, whereas the other classes are mostly helper classes.

TxtReader is a class that is called very early within **Main**. It reads the file of the specified instance using a scanner, and includes a multitude of get-methods to obtain the necessary data. **Attribute** and **TabuAttribute** work similarly. Both are records that allow for the holding and comparison of (tabu) attributes. These are used as part of the algorithm for the P_{div} function and the tracking of tabu moves respectively. Their **equal** methods are overridden, as we only care whether their values are the same.

Main works as follows: options are chosen to decide which type of instance(s) to run. Then, **main** starts a loop for the given instance(s), and runs each one for the given amount of runs, as well as clearing the output file beforehand. Using **TxtReader**, the necessary data is retrieved, and variables are set to their starting values. After which, an initial solution is generated with **generateInitialSolution()**.Here, the **GenerateFunctionValues** class is called for each route to calculate the function values in the solution. Then, the search process starts. A neighboring solution is generated using **generateRandomPoint(solution, neighborhoodStructure)** with a given solution, and integer representation of a neighborhood structure. After this, a tabu search is started using the neighboring solution with **applyTabuSearch(neighboringSolution)**. This method calls **applyTabuSearchIteration(solution)** with a solution that changes each tabu iteration. This method itself calls a method for each different type of move in the tabu phase: **applyTabuSearchIterationTwoOptStar(solution)**, **applyTabuSearchIterationRelocate(solution)**, **applyTabuSearchIterationStationInRe(-solution)**, and **applyTabuSearchIterationExchange(solution)**. Each of these four method returns the best move for the given move type, and of the four moves, the best one is selected. This move is then actually applied using **applyBestMove(bestMove, solution)**, and function values are recalculated for the changed routes using **GenerateFunctionValues**. This method also updates the attribute set using **generateAttributeSet(solution)**, and the new attribute addition frequency using **updateAttributeAdditionFrequency(newAttributeSet, attributeAdditionFrequency)**. Finally, it returns the new and updated solution.

After the tabu phase is done for an iteration, **acceptSA()** is called to compare the current solution with the solution retrieved via the tabu search. If the new solution is accepted, function values are updated. If the algorithm is still in the feasibility phase, we check if the solution is feasible, and if so exit the feasibility phase, otherwise we increase the number of vehicles used and generate a new initial solution if the maximum feasibility iterations are met. If we are not in the feasibility phase, the temperature is cooled.

After an iteration, we update the weights for the violations of the generalized cost function using **dynamicPenaltyUpdate(functionPcapS, functionPbattS, functionPtwS)** depending on how often they were consecutively met or not met.

After the feasibility phase is exited, and all iterations are done, we use the **txtWriter** method to write the results of that run in a text file, and continue with either the next run, next instance, or exit the code, depending on the current progress.

B Supplementary material

Table 5: Symbols for parameters together with their definitions

Symbol	Definition
0, $N+1$	Vertices of same depot
F'	Set of dummy vertices of set of recharging stations F
F'_0	Set of dummy recharging stations including depot instance 0: $F' \cup \{0\}$
V	Set of customers $V = \{1,...,N\}$
V_0	Set of customers including depot 0
V'	Set of customers including visits to recharging stations: $V' = V \cup F'$
V'_0	Set of customers, visits to recharging stations, and depot 0: $V'_0 = V \cup \{0\}$
V'_{N+1}	Set of customers, recharging visits, and depot instance $N+1$: $V'_{N+1} = V' \cup \{N+1\}$
$V'_{0,N+1}$	Set of customers, recharging visits, and depot instances 0 and $N+1$: $V'_{0,N+1} = V'_{N+1} = V' \cup \{0\} \cup \{N+1\}$
d_{ij}	Distance to travel between vertices i and j
t_{ij}	Time to travel between vertices i and j
C	Capacity of ECV
g	Recharging rate
h	Rate of ECV charge consumption
Q	Capacity of ECV battery
q_i	Demand at vertex i, assumed to be 0 if $i \notin V$
e_i	Earliest possible service start time at vertex i
l_i	Latest possible service start time at vertex i
s_i	Service time necessary at vertex i (s_0, $s_{N+1} = 0$)