Bachelor Thesis
Econometrie en Operationele Research

Erasmus School of Economics

Date final version: July 5, 2020

Recharge strategies for the Electric Vehicle Routing Problem with Time Windows

Name student: Anne de Vries
Student ID number: 483897

Supervisor: Mathijs van Zon
Second assessor: Albert Wagelmans

Abstract

The Electric Vehicle Routing Problem with Time Windows and Recharging Stations (EVRPTW) is a generalization of the Vehicle Routing Problem with Time Windows (VRPTW) in which the vehicles are electric and are able to recharge at recharging station. The problem introduces new challenges like limited driving range, scare recharge infrastructure, and charge dependent recharge times. This paper aims to adapt a metaheuristic for the EVRPTW, the Variable Neighbourhood Search/Tabu Search (VNS/TS) metaheuristic, to the extended problem in which partial recharge is allowed (EVRPTW-PR). We find that on average, routes can be made 0.73% shorter with introducing recharge strategies and a new operator.

Contents

1 Introduction

The field of green logistics has the objective of increasing the sustainability of production and distribution by taking into account the social and environmental effects (Sbihi & Eglese (2010)). Transportation is an important issue in green logistics due to its high emission of greenhouse gas, for example in the United States, where 28% of the national greenhouse gas is due to the transportation sector (US EPA (2009)).

Using Electric Vehicles (EV's) instead of conventional vehicles is one way for suppressing these negative effects (Hall & Lutsey (2018)). The use of these vehicles imposes new restrictions on logistics, like limited driving range, scarce recharging infrastructure and charge dependant recharge times (Xiao et al. (2019)). These restrictions are studied in the Electric Vehicle Routing Problem with Time Windows and Recharging Stations (EVRPTW), which is introduced by Schneider et al. (2014). The EVRPTW concerns serving dispersed customers from a depot. These customers have specified time windows in which they have to be served. For serving these customers, a set of routes which start and end at a depot have to be made. The electric vehicles that will perform these routes are homogeneous and have a limited load capacity and limited battery capacity. Recharging the battery while on route is possible at certain charging stations. Recharges are always done until maximum battery capacity is reached.

One way of solving the EVRPTW is with heuristics. Schneider et al. (2014) proposed a Variable Neighbourhood Search/Tabu Search (VNS/TS) metaheuristic and show that it has a strong performance. Keskin & Çatay (2016) point out that requiring full recharge limits the efficiency and practically of solutions. Therefore they generalize the problem to allowing partial recharges and propose an Adaptive Large Neighbourhood Search algorithm which allows for partial recharge. However, this heuristic has a weaker performance compared to the VNS/TS metaheuristic of Schneider et al. (2014).

This paper aims to combine the computational efficiency of the VNS/TS heuristic and the practical efficiency of allowing partial recharge. To this aim, we will first implement the VNS/TS algorithm and compare the performance of our implementation with the results in Schneider et al. (2014). We will then explore different ways of relaxing the full recharge requirement within the VNS/TS algorithm. These are allowing to charge to 100% and 70% of the battery capacity and only charging the amount necessary to reach the next charging station on the route. For this the benchmark EVRPTW instances made by Schneider et al. (2014) will be used. The results of this paper suggest that by implementing recharge strategies and a new operator, VNS/TS is able to increase performance. On problem sets of 100 customers, the saving average distance of the routes can be decreased by 0.73%.

The remainder of this paper is structured as follows. In Section 2 a summary of the literature and theoretical background will be provided. Then, in Section 3 the problem is described more extensively and an exact model for solving it is introduced. Following this is Section 4, which introduces our VNS/TS. The results of this heuristic will be compared to results of models and of other heuristics in Section 5. Lastly, we summarize our findings in Section 6.

2 Literature review

The Vehicle Routing Problem (VRP) is a generalization of the Traveling Salesman Problem and considers finding an optimal set of routes that start and finish at one or multiple warehouses which covers a set of customer nodes. The VRP is introduced by Dantzig & Ramser (1959) and has become a well-known and well-studied combinatorial optimization problem. Since then, many generalizations of the problem have been introduced, for example allowing vehicles to have a maximum capacity in the Capacitated VRP (CVRP), allowing multiple vehicles to fulfill the demand of a single customer in the Split Delivery VRP (SDVRP), allowing vehicles to perform multiple trips in the Vehicle Routing Problem with Multiple Trips (VRPMT), and allowing time windows to be specified in which customers' demand has to be satisfied in the Vehicle Routing Problem with Time Windows (VRPTW). For a summary of the VRP and its generalizations, we refer to Gayialis et al. (2019).

In more recent years, various generalizations of the VRP also considered the environmental impact of transportation. This impact has been considered by incorporating, amongst other variables, the cost of greenhouse gasses in the objective function in the Pollution-Routing Problem (PRP) by Bektaş & Laporte (2011). The impact has also been considered in Green Vehicle Routing Problem (G-VRP), proposed by Erdoğan & Miller-Hooks (2012). In the G-VRP, the vehicles are alternative fuel vehicles which need to replenish its fuel. This can be done at a (scarce) number of alternative fuel stations and is assumed to take a fixed amount of time. The case of refueling times being dependent on the amount of fuel charged is researched by Conrad & Figliozzi (2011). The problem they study was named the Recharging VRP (RVRP) but is also called the Electric VRP (EVRP). In the EVRP refueling could be done at certain customer nodes and is assumed to be done until a full charge is reached.

Schneider et al. (2014) introduced the Electric Vehicle Routing Problem with Time Windows and Recharging Stations (EVRPTW), which will also be considered in this paper. In this problem, a fleet of homogeneous, capacitated EV's with limited battery capacity needs to service a set of customer nodes within specified time windows by being assigned to routes starting and ending at a depot. The EV's are allowed to charge at multiple charging stations, which do not need to be at customer nodes. In this problem, charging is always done until the battery is full. Moreover, it is assumed that charging times are linearly dependent on the amount of fuel charged. They simplify the real world by assuming flat terrain, assuming speeds along edges to be known and constant, and neglecting the influence of payload on fuel efficiency. The primary objective of the problem is minimizing the number of EV's needed and the secondary objective is minimizing total distance traveled. In the paper, a mathematical model for the problem is provided. This model makes use of dummy variables for recharging stations to allow multiple recharges at the same station. Because the problem is a generalization of the VRP, and because the VRP is known to be NP-hard, see Lenstra & Kan (1981), the EVRPTW is NP-hard as well. Desaulniers et al. (2016) proposed an exact branch-price-and-cut algorithm for the problem. If shorter computational times are preferred, heuristics are an option. Schneider et al. (2014) proposed a hybrid metaheuristic which they call VNS/TS. This heuristics is a combination of Variable

Neighbourhood Search (VNS) and Tabu Search (TS). Furthermore, it makes use of the acceptance criterion of Simulated Annealing (SA). They show by testing on numerous instances that the VNS/TS heuristic has a strong performance.

Since its introduction, the EVRPTW has been further researched. Some examples of which will be named here. Goeke & Schneider (2015) considers the case having a mixed fleet of EV's and internal combustion commercial vehicles and they model the energy consumption rate instead of assuming it to be fixed. Xiao et al. (2019) analyzes the effects of modeling the energy consumption rate and making speeds across arcs a decision variable. The possibility of time dependent waiting times at the charging station is considered in Keskin et al. (2019). This paper also allows customers to be served late at the cost of a monetary penalty. Their total cost consists of costs incurred due to vehicles, drivers, energy, and late penalties. A distinction between Full Recharge (FR) and Partial Recharge (PR) is made by Keskin & Çatay (2016). That paper considers the EVRPTW-PR case and develops a Adaptive Large Neighbourhood Search (ALNS) algorithm for solving it. Zuo et al. (2019) also looks at PR case and relaxes the assumption of a constant charging rate by modeling it with a concave nonlinear charging function.

3 Problem description

The EVRPTW-PR concerns a set of customers, each with a known demand, delivery time window, and service duration. These customers are to be served by a set of homogeneous electric vehicles with known load and battery capacities. The routes for these vehicles have to start and end at a known depot. The vehicles leave with a full battery and the charge in the battery decreases proportionally to the distance traveled. To avoid negative charge, vehicles are able to charge at a number of charging stations, where the battery is charged proportionally to the time spend charging. Contrary to the EVRPTW, vehicles do not have to attain full charge before continuing the route. The primary objective of the problem is to minimize the number of vehicles required to satisfy all customers. The secondary objective is to minimize the total distance traveled by those vehicles.

As is common in VRP modeling techniques, several real-world characteristics are simplified. The effect of payload on route cost is neglected and the terrain is assumed to be flat. The distance between vertices is the Euclidean distance and the speed over these arcs is given and constant. Furthermore, recharging is assumed to be linear, whereas in the real-world the last percentages of charge take more time. For a more extensive overview of the assumptions, we refer to Schneider et al. (2014).

3.1 Mathematical formulation

Now we will introduce the mathematical formulation for the problem. The model is based on the one used in Keskin & Çatay (2016). We will follow the notation used there, which in turn follows the notation in Schneider et al. (2014).

Let the set $V = \{1, ..., N\}$ represent the customer nodes and F denote the set of charging stations. To allow multiple visits to charging stations, dummy variables are generated for these nodes. These dummy variables are in the set F'. The depot is represented by two nodes, namely 0 and $N+1$. These nodes are at the same location but 0 represents the start, and $N+1$ the end of the routes. The set $V' = V \cup F'$ is the set of all customer nodes and charging station dummies. If a depot representing node is added to the set, this node is subscriped to the set, for example $F'_0 = F' \cup \{0\}$, $V_{N+1} = V \cup \{N+1\}$, and $V'_{0,N+1} = V \cup F' \cup \{0\} \cup \{N+1\}$.

The problem is defined on a complete directed graph $G = (V'_{0,N+1}, A)$, where $A = \{(i,j) | i, j \in V'_{0,N+1}, i \neq j\}$ is the set of arcs. Each arc is associated with an distance d_{ij} and a travel time t_{ij}. The charge required to travel an arc is $h \cdot d_{ij}$, where h denotes the charge consumption rate. Each vertex in $V'_{0,N+1}$ is assigned a non-negative demand q_i. For each $i \notin V$, $q_i = 0$. Moreover, for all nodes a time window $[e_i, l_i]$ is given, where e_i denotes the earliest time service to node i can start, and l_i the latest time service can start. The latter means that service can end after l_i, but only when it started before that. All nodes in $V_{0,N+1}$ have a fixed service time s_i. For 0 and $N+1$ this time is zero. For the recharge stations, the recharge time is linearly dependant on the amount charged by a factor g, the recharge rate.

The set of homogeneous vehicles is located at the depot. Each vehicle has a maximum load capacity of C and a maximum battery capacity of Q.

The model makes use of five decision variables. The variable x_{ij} equals one when arc a_{ij} is traveled and zero otherwise. The time service to node i started is denoted as τ_i. The remaining load in the vehicle after servicing a node i is denoted u_i. The charge of the vehicle servicing i is denoted as y_i. For each recharge station i, a the variable Y_i is defined as the amount of energy charged.

This problem has two objectives. The primary objective is minimizing the number of vehicles used. the secondary is minimizing the total distance traveled. We will first present the model minimizing the primary objective and then explain how this model has to be altered to optimize the secondary objective. The mathematical model for the primary objective, Model 1, is as follows:

$$\min \quad \sum_{i \in V'_{N+!}} x_{0i} \tag{1}$$

$$\text{s.t.} \quad \sum_{j \in V'_{N+1},\ i \neq j} x_{ij} = 1 \qquad \forall i \in V \tag{2}$$

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

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

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

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

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

$$0 \leq u_j \leq u_i - q_i x_{ij} + C(1 - x_{ij}) \qquad \forall i \in V'_0,\ \forall j \in V'_{N+1},\ i \neq j \tag{8}$$

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

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

$$0 \leq y_j \leq Q - h d_{ij} x_{ij} + Q(1 - x_{ij}) \qquad i = 0,\ \forall j \in V'_{N+1} \tag{11}$$

$$0 \leq y_j \leq Y_i - h d_{ij} x_{ij} + Q(1 - x_{ij}) \qquad \forall i \in F',\ \forall j \in V'_{N+1},\ i \neq j \tag{12}$$

$$y_i \leq Y_i \leq Q \qquad \forall i \in F' \tag{13}$$

$$x_{ij} \in \{0, 1\} \qquad \forall i \in V'_0,\ \forall j \in V'_{N+1},\ i \neq j \tag{14}$$

The objective function is defined by (1). Constraints (2) and (3) ensure the connectivity of customer and charging nodes respectively. The flow conservation at those nodes is handled by (4). Constraint (5) and (6) ensure the time feasibility of routes. The time window restriction is modeled by Constraint (7). Moreover, these three constraints together eliminate sub-tours. Constraint (8) and (9) enforce the load capacity restriction. The next four constraints deal with the charge restrictions. Constraint (10) handles the charge when leaving customer nodes and Constraint (11) that for leaving the starting depot, where the charge is assumed to be at the maximum battery capacity. Constraint (12) handles the charge after leaving recharge charging stations, where the range of the amount charged is restricted in (13). Lastly, constraint (14) defines the range of the binary decision variable.

As mentioned before, the model has a secondary objective. To optimize this, we formulate a second model similar to the first one. Let m be the minimal number of vehicles used, as found by Model 1. In addition to Constraints (2) to (14), the following objective and constraint make up the model for optimizing the secondary objective, Model 2:

$$\min \qquad \sum_{i \in V'_0,\ j \in V'_{N+1},\ i \neq j} d_{ij} x_{ij} \tag{15}$$

$$\text{s.t.} \qquad \sum_{i \in V'_{N+!}} x_{0i} = m \tag{16}$$

Where (15) is the objective function of minimizing the total travel distance and Constraint (16) ensures that the primary objective, the number of vehicles used, is not worsened.

To improve the efficiency of the models, we added several preprocessing constraints. These constraints can be found in Appendix A.

4 Methodology

As a solution method for the EVRPTW-PR, we use a Variable Neighbourhood Search/Tabu Search (VNS/TS) metaheuristic. As the name suggests, this heuristic is a combination of a VNS as proposed by Mladenović

& Hansen (1997) and a TS as proposed by Glover (1989). Hybrid combinations of the two heuristics have been successfully implemented for VRP's, for an example see Tarantilis et al. (2008). The one used in this research is based on, but not identical to, the VNS/TS proposed by Schneider et al. (2014).

The metaheuristic works as follows. An initial set of routes is generated. These routes do not need to be feasible but have to cover all customer nodes. Section 4.1 describes how these routes are generated. The initial solution will be improved in the following iterations. Solutions are compared based on a generalized cost function. This means that infeasible solutions are allowed but are penalized for breaking constraints. The generalized cost function is explained in detail in Section 4.2. In each iteration, the solution is altered based on a neighbourhood structure. This is called shaking and is described in Section 4.3. The altered solution is then improved by a Tabu Search, which is explained in Section 4.4. If the resulting solution is better than the solution at the start of the iteration, the resulting solution is chosen to start the next iteration with. If it is not better, it has a probability of being chosen based on an SA criterion, which is detailed in 4.3. Section 4.5 specify the different recharge strategies.

The heuristic starts in a feasibility phase. During this phase, a new route is added to the solution after each η_{feas} iterations. This phase is terminated as soon as a feasible solution was found. After the phase, the total distance of the solution is improved for η_{dist} iterations. In Algorithm 1, the pseudo code for the algorithm can be found.

Algorithm 2: An overview of the VNS/TS metaheuristic.

```
N_k ← the set of VNS neighbourhood structures
S ← a generated initial solution
k ← 1
i ← 0
feasibilityPhase ← true
while feasibilityPhase ∨ (¬feasibilityPhase ∧ i <η_dist) ∨ S is not feasible
 do
    S' ← shake S using N_k
    S'' ← apply η_tabu iterations of Tabu Search on S''
    if S'' is accepterd over S by the SA criterion then
        S ← S''
        k ← 1
    else
        k ← (k modulo k_max) + 1
    end
    if feasibilityPhase then
        if S is not feasible then
            if i = η_feas then
                Add a vehicle to S
                i ← −1
            end
        else
            feasibilityPhase ← false
            i ← −1
        end
    end
    i ← i + 1
end
```

4.1 Initial solution

The initial solution is creating in a way similar to the initial solution in Cordeau et al. (2001). To start, a random point on the plane is chosen. Next, the customer nodes are sorted based on the angle the line segment between the node and the depot makes with the line segment between the depot and the selected random point. The first route is opened, initially from the start depot to the end depot, making the number of routes k equal to 1. The first unassigned customer is added to the open route between two successive nodes such that the increase in route length is minimized. If this insertion either increases the length of the route to above the distance a vehicle can travel on one fully charged battery or makes the total demand on the route exceed the load capacity, the node is added to a newly opened route instead, increasing k by one. This is only done if the number of routes k is still lower than a predetermined maximum number of routes m. This process is repeated for all customer nodes. The result is m routes of which the first $m-1$ satisfy the battery and the capacity restriction without recharging. For these routes the time window constraints may be broken. For the last route, all these restrictions may be broken.

The value of m is based on a simple lower bound on the number of vehicles based on the total demand and vehicles capacity. This lower bound is identical to the lower bound used for Model 1, as described in Appendix A.

4.2 Generalized cost function

The VNS/TS search aims to improve the initial solution on the basis of the generalized cost function $f_{gen}(S)$. The formula for this function is as follows:

$$f_{gen}(S) = f(S) + \alpha P_{cap}(S) + \beta P_{tw}(S) + \gamma P_{batt}(S). \tag{17}$$

In this formula, $f(S)$ denotes the total distance traveled. $P_{cap}(S)$, $P_{tw}(S)$, and $P_{batt}(S)$ respectively denote the violation of the load capacity, time window, and battery range restrictions. These violations are scaled by penalty factors α, β, and γ, respectively. These factors are initialized at (α_0, β_0, γ_0) but after $\eta_{penalty}$ successive iterations of breaking the respective restriction, the factor is multiplied by a predetermined δ. Conversely, after $\eta_{penalty}$ successive iterations of satisfying the respective restriction, the factor is divided by δ. The value of the factors are bounded by (α_{min}, β_{min}, γ_{min}) and (α_{max}, β_{max}, γ_{max}).

We will now describe how the value of the violations are determined. For this, a route r is defined as a sequence of nodes v_i starting at v_0 and ending at v_{n+1}, both representing the depot. The function $Vert(r)$ returns the set of nodes in route r.

The value of $P_{cap}(S)$ is the sum of the capacity penalties $P_{cap}(r_k)$ of the individual routes that make up S. $P_{cap}(r_k)$ is calculated as follows:

$$P_{cap}(r_k) = -min(C - \sum_{v \in Vert(r)} q_v, 0).$$

The value of $P_{tw}(S)$ is based on the notion of time travel. Nagata et al. (2010) introduced this approach and later Schneider et al. (2013) improved it. Time travel entails that if a customer node i is served late, it is assigned a penalty but handled as if the vehicle serviced i at its due time. This prevents the penalty to be carried over to all consecutive nodes and thereby possibly penalizing efficient segments of a route.

The time travel approach also has advantages in computational time for inter-route moves. This is achieved by storing forward and backward time window penalty slacks. Using these, some insertions of a node, some removals of a node, and some merging of two partial routes can be calculated in $\mathcal{O}(1)$. This is not possible when a charging station is present in the route segment after the operator because the charging time depends on the battery charge, which in turn depends on the traveled distance. In this case, the penalty slacks have to be recalculated for the nodes between the operation and the charging station, for the charging station itself, and the node following the charging station.

The calculation of $P_{batt}(S)$ is based on the variables $\Upsilon^{\rightarrow}_{v_i}$ and $\Upsilon^{\leftarrow}_{v_i}$. These variables denotes the charge required to travel from previous non-customer node to v_i and the charge required to reach the next charging station or depot from node v_i, respectively. The value of these variables is calculated as follows:

$$\Upsilon^{\rightarrow}_{v_i} = \begin{cases} hd_{v_{i-1}v_i} & \text{if } v_{i-1} \in F'_0, \\ \Upsilon^{\rightarrow}_{v_{i-1}} + hd_{v_{i-1}v_i} & \text{otherwise,} \end{cases} \qquad i = 1, ..., n+1;$$

$$\Upsilon^{\leftarrow}_{v_i} = \begin{cases} hd_{v_i v_{i+1}} & \text{if } v_{i+1} \in F'_{n+1}, \\ \Upsilon^{\leftarrow}_{v_{i+1}} + hd_{v_i v_{i+1}} & \text{otherwise,} \end{cases} \qquad i = 0, ..., n.$$

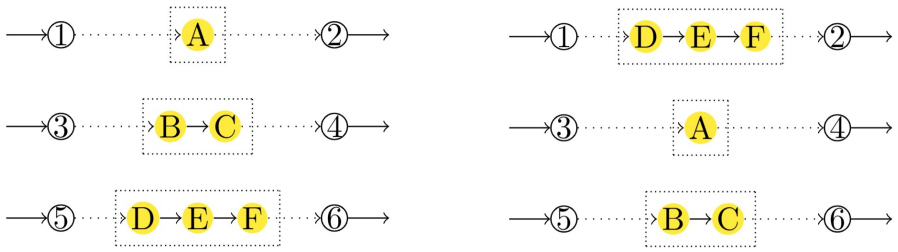
Using this variable, the battery penalty of a route $P_{batt}(r)$ is calculated as follows:

$$P_{batt}(r) = \sum_{v_i \in Vert(r)} max(\Upsilon^{\rightarrow}_{v_i} - E_{cs}, 0),$$

where E_{cs} is the amount of charge the vehicle had when departing from the previous non-customer node. The exact value of this variable will be explained later. The total battery penalty is of a solution is the sum of the battery penalties of the routes in the solution. The variable $\Upsilon^{\leftarrow}_{v_i}$ is slack which is used to evaluate TS operators.

4.3 Variable Neighbourhood Search component

For understanding the VNS component of the VNS/TS metaheuristic, we first sketch how a general standalone VNS algorithms works. Such an algorithm makes use of a predetermined set of neighbourhood structures. Each iteration is started by a shaking phase in which the current best solution S is altered using the neighbourhood structure $\mathcal{N}_k$ to a neighbouring solution S'. This is called a perturbation move. This solution is improved using a greedy search to a local optimum S''. If S'' is better than S, it is taken as the new S and the neighbourhood is reset to $\mathcal{N}_1$. If this is not the case, the initial S is kept and a the next iteration will be



(a) Before the cyclic exchange. (b) After the cyclic exchange.

Figure 1: An example of a cyclic-exchange move with $\#Rts = 3$ and $\Lambda_{max} = 3$.

started with neighbourhood $\mathcal{N}_{k+1}$, which is larger than $\mathcal{N}_k$

In this VNS/TS this procedure is altered. The greedy search is replaced by a Tabu Search, which will be described in the next section. The acceptance criterion is changed to the acceptance criterion of SA. This criterion helps diversifying the search. Furthermore, the neighbourhood structure will be specified.

The criterion of SA entails that a new but worse solution S'' has a probability of being accepted. This probability is dependent on the temperature T of the iteration and on the value of the current solution S as in the formula:

$$e^{\frac{-f(S'')-f(S)}{T}}.$$

The initial temperature T_0 is set such that a solution with a generalized cost which is δ_{SA} higher than the cost of the initial solution, has a 50% of being accepted as new solution. However, every iteration after this the temperature T is decreased by multiplying it with a cooling factor t, $0 < t < 1$.

The neighbourhood structure of the VNS are defined as cyclic-exchange operators, which are introduced by Thompson et al. (1989). A cyclic-exchange entails swapping sequences of customers between two or more routes. In the neighbourhood structure $\mathcal{N}_k$ of the VNS/TS are characterized by two parameters, namely $\#Rts$ and Λ_{max}. $\#Rts$ denotes the number of routes involved in the exchange. For each of these routes a randomly chosen sequence of nodes is selected. These sequences have a length which is randomly chosen on the interval $[0, min(\Lambda_{max}, n_k)$. These sequences then swap place in the routes. An example of a cyclic-exchange move is provided in Figure 1. The k-neighbourhood structures we used are identical to the ones in Schneider et al. (2014) and are shown in Table 1.

k	$\#Rts$	Λ_{max}	k	$\#Rts$	Λ_{max}	k	$\#Rts$	Λ_{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

Table 1: An overview of the k-neighbourhood structures defined by the number of routes involved $\#Rts$ and the maximum number of translocated nodes Λ_{max} .

4.4 Tabu Search component

A general Tabu Search explores a function without getting stuck in local optima. This is done by declaring moves tabu. If a move is tabu, it is forbidden to perform this move for a few iterations. In a Tabu Search, solutions are altered by means of a neighbourhood structure. The operation generating the neighbouring solution with the lowest generalized cost that does not involve a tabu move is performed. This is done even if that generalized cost is higher than the cost of the original solution. The moves of the performed operation will then be added to the tabu list for a number of iterations to avoid moving back in the next few iterations. The combination of forbidding moves and accepting worse solutions avoids getting stuck in local optima.

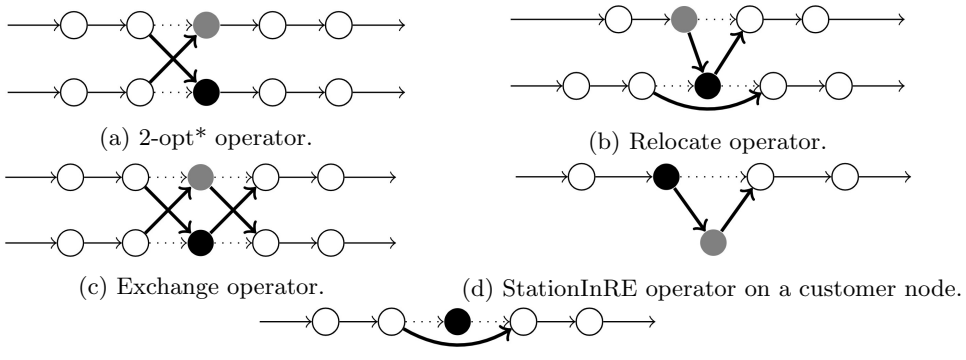
In the VNS/TS, the TS replaces the greedy heuristic. In this case the neighbourhood for the order of the nodes in the route is defined by four operators, namely 2-opt*, relocate, exchange, and stationInRe. Additionally, we will define a fifth operator we will call changeStrategy. This operator changes the amount of energy charged at charging stations. The first four operators change the order of the nodes in the routes. For these operators, examples of these changes are shown in Figure 2.

The 2-opt* operator is proposed by Potvin & Rousseau (1995). It is adaptation to the time window case of the 2-opt operator by Lin (1965). It functions by making a cut in two routes. Then the first half of first route is connected to the second half of second route and vice versa.

The relocate operator takes a single node from a route and inserts it in a different route or in the same route at a different place. This operator is introduced by Savelsbergh (1992). In its implementation in the VNS/TS algorithm, the operator is allowed to relocate charging stations.

Savelsbergh (1992) also introduced the exchange operator. This operator swaps the position of two nodes within one route or between two routes. This operator is not allowed to move charging stations.

The operator stationInRe is introduced by Schneider et al. (2014). If this operator is called on a charging station node, it removes this node from the route. If it is called on an other node type, it inserts a charging station after the node with the default charging strategy of full recharge.



(a) 2-opt* operator.

(b) Relocate operator.

(c) Exchange operator.

(d) StationInRE operator on a customer node.

(e) StationInRE operator on a charging station.

Figure 2: Examples of the operators that change the order of nodes in a route. The operator is called on the black nodes. Possible grey nodes are the second node the operator used. Dashed edges are those which are removed and thick edges are those which are inserted.

The operator changeStrategy can only be called on charging station. It switches the current recharge strategy applied at this station to an other recharge strategy. The recharge strategies will be explain in the next section. For this operator, no tabu's are implemented.

To increase computational speeds, the amount of moves to be evaluated can be lowered by requiring that at least one arc that will be inserted is promising. This is called sparsification. In Schneider et al. (2014), this is achieved by performing the operators on generator arcs, which are introduced in Toth & Vigo (2003). Our implementation is based on nodes instead of arcs, therefore we assign two sets to each node. The first is a set of the g_{cost} closest customer nodes and the second one the g_{cs} closest charging stations. Operators are only called on a node in combination with a node from one of these sets, the starting depot, or the ending depot.

The tabu list contains all moves which are declared tabu for an iteration. A tabu move is defined as an attribute (ξ, k, μ, ζ). It means that the insertion of arc ξ into route r_k between node μ and ζ is prohibited. The number of iterations a move is prohibited is called the tabu tenure. The tenure ϑ is randomly drawn on the interval $[\vartheta_{min}, \vartheta_{max}]$. The VNS move is also added to the tabu list. If an aspiration criterion is met, the tabu status of an arc is ignored. In this heuristic the aspiration criterion is met if the resulting solution is feasible. The number of iterations TS is performed in a VNS iteration is η_{tabu}.

4.5 Recharge strategies

Generalizing the problem from EVRPTW to EVRPTW-PR introduces the decision of how much should be charged at a charging station. Because a decision variable for the amount charged, as the Y_i in the model, does not fit in the VNS/TS framework, we define two recharge strategies and the operator changeStrategy, the latter of which has already been introduced.

Recharge strategies determine the amount of energy that will be charged. Because there can be leftover charge y_i when arriving at a charging station, the amount of energy charged is calculated as $E_{cs} - y_i$. Here E_{cs} denotes the desired amount of charge when leaving the charging station. The value of this variable depends on the recharge strategy at the charging station.

The first recharge strategy, which we will call Level Recharge (LR), entails charging up to a certain level. These levels are defined as a percentage of the total battery capacity Q. In this way, FR, requiring full charge when leaving a charging station like in the EVRPTW, can be reformulated as LR-100. We will also experiment with other strategies as LR-75 and LR-50. For a LR-x, E_{cs} is determined as $x \cdot Q$. Keskin & Çatay (2016) experimented with a similar approach to recharging. They called this partial recharge schemes but used the same scheme for every charging station. By allowing different occurrences of charging stations to have different recharge strategies, we possibly allow for better solutions.

The second recharge strategy entails recharging just enough for reaching the next charging station or the ending depot. We will call this strategy Required recharge (RR). The amount of energy charged E_{cs}, is determined by the formula $E_{cs} = min(\Upsilon^{\leftarrow}_{v_i}, Q)$.

Recharge strategies are linked to a specific occurrence of a charging station. This means that when a recharge station is relocated, the recharge strategy is also carried over. It also implies that if a recharge station is used multiple times to recharge, each occurrence has its own recharge strategy. Only the stationInRe operator can insert a new occurrence of a charging station and when it does, it gives the station the LR-100 recharge strategy. Only the changeStrategy operator can change the strategy.

5 Computational experiments

To assess the performance of our metaheuristic we perform multiple computational experiments. For this we will use the EVRPTW data sets which are described in Section 5.1. All experiments were performed on a laptop computer equipped with an Intel Core i7 processor clocked at 2.2GHz with a turbo of 3.1 GHz and with 8GB memory, running Windows 10 Home. The code was executed using Eclipse IDE for Java developers version 2020-3 using Java jdk 14. The implementation of the models made use of library of the commercial solver CPLEX Studio 12.10.

An overview of the parameters used for the metaheuristic is given in Table 2. These values are based on those used in Schneider et al. (2014). The values of new or undefined parameters are based on preliminary tests. During these tests we found that increasing the number of Tabu Search iterations η_{tabu} was beneficial for optimizing the primary objective, especially for smaller instances. Therefore we chose to use $\eta_{tabu} = 150$ for these small instances. For the larger instances we used the $\eta_{tabu} = 100$, like in Schneider et al. (2014), to decrease run time.

General		VNS		Penalties		TS	
g_{cost}	8	η_{feas}	500	$(\alpha_0, \beta_0, \gamma_0)$	10	ϑ_{min}	15
g_{cs}	3	η_{dist}	200	$(\alpha_{min}, \beta_{min}, \gamma_{min})$	0.5	ϑ_{max}	30
		δ_{SA}	0.08	$(\alpha_{max}, \beta_{max}, \gamma_{max})$	5000	η_{tabu}	100/150
		t	0.9	δ	1.2		
				$\eta_{penalty}$	2		

Table 2: An overview of all the values of the parameters used.

5.1 Data

The data we will use is the set of benchmark EVRPTW instances as available on Goeke (2019). These data instances are made by Schneider et al. (2014) when introducing the EVRPTW problem and are based on benchmark instances for the VRPTW problem proposed by Solomon (1987). The data set contains 36 small instances and 56 large instances. The small instances have 5, 10 or 15 customer nodes and between 2 and 8 charging stations. The large instances have 100 customers and 21 charging stations each. In all instances, one of the charging stations is at the depot.

The instances can be divided into multiple classes. These classes are based on distribution of the nodes and the length of the scheduling horizon. The distribution of nodes divides the data into classes R, C, and RC for respectively random distribution, clustered distribution, or a mix of both. A short scheduling horizon,

requiring more vehicles to be feasible, is denoted by a 1 and a long scheduling horizon by a 2. The resulting six classes therefore are: R1, C1, RC1, R2, C2 and RC2. Within a class, the instances differ on the basis of time window density and width. For each instance the fuel tank capacity, vehicle load capacity, the inverse fueling rate, the fuel consumption rate, and the average velocity are given. The latter two are equal to one for simplicity's sake.

For more details about these instances, we refer to Schneider et al. (2014).

5.2 Full recharge performance comparison

In this section, the performance of our VNS/TS will be compared to the performance of the VNS/TS of Schneider et al. (2014) and the results of their model for the small instance problem. To distinguish it from ours, we will refer to their VNS/TS in this section as SSG, after the three authors. Schneider et al. (2014) used CPLEX to solve the 36 instances using an time-limit of 7200 seconds. Within this time-limit, 25 of these instances were solved to optimally. For the remaining 11 an upper-bound was found. The SSG metaheuristic results are the best results found in ten runs. For 35 instances, the SSG heuristic found a solution equal to the solution of the model. In the remaining instance, the heuristic found a solution lower than the upper-bound by the model. The results are shown in Table 3.

Table 3 also displays the performance of our VNS/TS. For this heuristic too, the results shown are the best of 10 runs. For 35 instances, the primary objective equals that of the solution by the model. For these instances the gap in distance with model solution is displayed. This reveals that in 28 cases the solutions were equal and in 6 cases the heuristic solution was worse. For one instance, the distance was less than that of the model solution. The average gap is 1.20%.

For the last instance, *rc108C5*, one extra vehicle was used. Despite thorough examination into this case, no cause was found. We suspect that a restriction might be implemented too strict. Our main suspect for this is the time window restriction, because the total distance is the same for both solutions. An other, less likely, possibility is that minor rounding errors native to java accumulate and distort calculations.

We suspect the cause of the aforementioned vehicle deviation also might be the cause of some of the other solutions that require more traveling. We observe that the deviations tend to occur for the larger instances. They also seem to happen less for the clustered distribution instances. More pronounced is that they happen more for class 2 instances, those with longer planning horizons. These observations lead us to conclude that deviations are more likely for longer routes. This strengthens our believe that the cause is in the calculation of time window restriction.

Run times cannot be compared directly since the Schneider et al. (2014) results are obtained on a different computer. However, it is save to say that the SSG implementation is faster. The run time of our implementation seems to be more dependent on the number of routes in the final solution. This leads us to believe that in the SSG implementation the savings per operator were saved across TS iterations, if the route are not changed. However, because the exact implementation of SSG is unknown this is not certain. We did

	FR CPLEX			SSG			VNS/TS			
Instance	$\#Veh$	TD	$t(s)$	$\#Veh$	TD	$t(s)$	$\#Veh$	TD	$t(s)$	ΔTD
c101C5	2	257.75	81	2	257.75	0.21	2	257.75	45.29	0.00%
c103C5	1	176.05	5	1	176.05	0.12	1	176.05	9.02	0.00%
c206C5	1	242.55	518	1	242.55	0.14	1	242.56	20.12	0.00%
c208C5	1	158.48	15	1	158.48	0.11	1	158.48	17.18	0.00%
r104C5	2	136.69	1	2	136.69	0.13	2	136.69	60.12	0.00%
r105C5	2	156.08	3	2	156.08	0.11	2	156.08	72.56	0.00%
r202C5	1	128.78	1	1	128.78	0.11	1	128.78	10.21	0.00%
r203C5	1	179.06	5	1	179.06	0.15	1	179.06	7.03	0.00%
rc105C5	2	241.30	764	2	241.30	0.14	2	241.30	79.96	0.00%
rc108C5	1	253.93	311	1	253.93	0.17	2	253.93	80.53	
rc204C5	1	176.39	54	1	176.39	0.15	1	176.39	11.19	0.00%
rc208C5	1	167.98	21	1	167.98	0.13	1	167.98	10.50	0.00%
c101C10	3	393.76	171	3	393.76	0.77	3	399.31	124.28	1.41%
c104C10	2	273.93	360	2	273.93	0.95	2	273.93	20.21	0.00%
c202C10	1	304.06	300	1	304.06	0.71	1	304.06	36.23	0.00%
c205C10	2	228.28	4	2	228.28	0.49	2	228.28	320.59	0.00%
r102C10	3	249.19	389	3	249.19	0.65	3	249.19	483.03	0.00%
r103C10	2	207.05	119	2	207.05	0.72	2	207.05	74.44	0.00%
r201C10	1	241.51	177	1	241.51	0.78	1	254.32	25.39	5.30%
r203C10	1	218.21	573	1	218.21	0.71	1	246.87	7.64	13.13%
rc102C10	4	423.51	810	4	423.51	0.69	4	423.51	739.34	0.00%
rc108C10	3	345.93	39	3	345.93	0.90	3	345.93	403.95	0.00%
rc201C10	1	412.86	7200	1	412.86	0.90	1	412.86	49.47	0.00%
rc205C10	2	325.98	399	2	325.98	0.81	2	325.98	333.55	0.00%
c103C15	3	384.29	7200	3	384.29	15.37	3	384.29	84.64	0.00%
c106C15	3	275.13	17	3	275.13	14.94	3	275.13	188.56	0.00%
c202C15	2	383.62	7200	2	383.61	13.41	2	383.62	113.56	0.00%
c208C15	2	300.55	5060	2	300.55	11.08	2	300.55	415.24	0.00%
r102C15	5	413.93	7200	5	413.93	19.55	5	419.94	161.69	1.45%
r105C15	4	336.15	7200	4	336.15	13.35	4	336.15	232.37	0.00%
r202C15	2	358.00	7200	2	358.00	13.17	2	358.00	211.68	0.00%
r209C15	1	313.24	7200	1	313.24	13.73	1	378.96	16.50	20.98%
rc103C15	4	397.67	7200	4	397.67	14.62	4	397.67	811.15	0.00%
rc108C15	3	370.25	7200	3	370.25	12.92	3	370.25	76.17	0.00%
rc202C15	2	394.39	7200	2	394.39	12.74	2	405.53	77.46	2.82%
rc204C15	1	407.45	7200	1	384.86	15.57	1	394.94	16.84	-3.07%
Avg.			2483.25			5.03			151.32	1.20%

Table 3: A comparison of FR results obtained by Schneider et al. (2014) with CPLEX and their VNS/TS (denoted as SSG) and our VNS/TS on the small EVRPTW instances.

not implement this feature in our implementation.

5.3 Required recharge on small instances

We will now turn to PR case and start by comparing FR and RR strategies for the VNS/TS metaheuristic. We will compare this with the two models run by CPLEX. The models had a time limit of 5200 seconds each. The standard number of dummy variables was two, but for the instances *rc201C10* and *rc204C15* three were required for the optimal solution. The time limit prevented finding an optimal solution for 9 instances, which are therefore upper-bounds.

Because the RR requires backward computation instead of forward, like for both FR and LR, a fast implementation using slacks requires new functions. Due to time restrictions, we did not implement RR with slacks. Instead, we assess the performance of RR using a slower version of the program that completely calculates all costs for all solutions after each operator. In this version, RR is the standard and only possible

strategy at recharge stations. Due to the slower implementation, we were only able to run small instances. We will also only compare based on one run. Nonetheless, we think this short inquiry in RR is valuable because it provides insights. Moreover, we would like to note that implementing RR with slacks is possible and does not require any new slacks.

The results are shown in Table 4. From it we see that the FR recharge strategy outperforms RR in 22 instances. For 11 of these this was also in the number of vehicles. In 13 instances RR was better, one of these was in the number of vehicles. The remaining instance yielded the same solution for both strategies. Besides generally outperforming RR in results, LR-100 was also faster as it was 80 seconds quicker on average. The cause of this time difference is that in the current implementation for RR, time window penalties are calculated in $\mathcal{O}(n^2)$ instead of $\mathcal{O}(n)$.

It might feel counter-intuitive that RR performs worse in some instances. To illustrate how this can be, we created an example which is displayed in Figure 3. We assume that $Q = 91$, $h = 1$, and that, due to the time windows, node 1 has to be served before node 3. Let the early time window of node 2 be such that, irrespective of the amount charged at CS1, the vehicle needs to wait before starting its service. Due to this vehicles following FR or RR arrive at CS2 at the same time. The vehicle following FR, however, has 51 instead of zero units of charge left. Due to this, the RR vehicle will have to stay charging for longer. If both vehicles would then travel to CS3, charge the amount dictated by the recharge strategy and than service node 2 before going to the depot, the vehicle following RR would arrive at node 2 20 time units later. If the due time of this node passed in these 20 time units, the vehicle would have to go to node 3 directly after CS2, increasing the total route length by 11 units of distance.

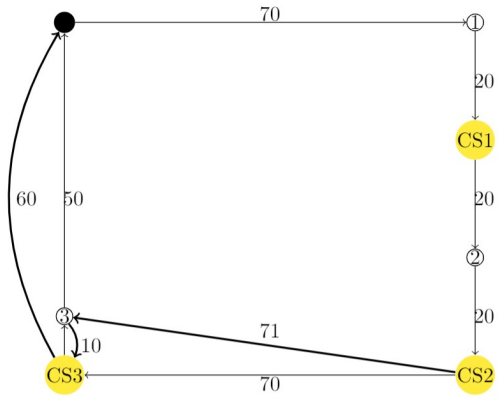


Figure 3: An example of how RR can increase travel distance. The black, white, and yellow nodes are the depot, the customers, and the charging stations, respectively. The labels of the arcs represent both the travel time and the charge required to travel the arc. The thick arcs represent the division a RR following vehicle could be forced to make.

Comparing the RR results to the CPLEX FR results in the previous section shows that RR was able to improve some solutions. This is more frequent for the instances containing more customers. A similar result was found in Keskin & Çatay (2016). They ascribe this to the small instances being too restrictive in size

to allow alternative routes. These things combined, indicate that RR can be a useful addition in a mixed recharge strategy framework. However, this does require a faster implementation.

	PR CPLEX				FR/LR-100				RR			
Instance	$Veh	TD	t(s) 1	t(s) 2	$\#Veh$	TD	t(s)	ΔTD	$\#Veh$	TD	t(s)	ΔTD
c101C5	2	257.75	0.9	0.3	2	277.28	11.6	7.58%	2	304.28	15.4	15.29%
c103C5	1	175.37	0.4	0.3	1	184.50	2.6	5.21%	1	184.50	2.9	4.95%
c206C5	1	242.56	0.1	0.2	1	242.56	2.6	0.00%	2	253.69	13.0	
c208C5	1	158.48	0.2	0.7	1	158.48	2.2	0.00%	2	164.34	9.7	
r104C5	2	136.69	2.8	0.1	2	136.69	8.7	0.00%	2	137.01	9.4	0.23%
r105C5	2	156.08	0.6	0.1	2	156.28	12.0	0.13%	2	186.98	14.4	16.53%
r202C5	1	128.78	0	0.1	1	128.78	2.1	0.00%	2	128.78	10.4	
r203C5	1	179.06	0.1	0.1	1	179.06	2.8	0.00%	3	229.47	39.0	
rc105C5	2	233.77	18.2	0.5	2	241.30	22.4	3.22%	2	247.55	13.8	5.57%
rc108C5	2	253.93	11.5	0.3	2	253.93	12.7	0.00%	2	262.27	13.6	3.18%
rc204C5	1	176.39	0.1	0.3	1	176.39	2.7	0.00%	2	185.16	10.6	
rc208C5	1	167.98	0.1	0.2	1	167.98	2.1	0.00%	2	186.41	11.5	
c101C10	3	388.25	4273.5	2.1	3	524.35	134.0	35.05%	4	543.10	327.2	
c104C10	2	273.93	5400.3	2	2	424.82	46.2	55.08%	2	353.71	56.3	22.56%
c202C10	1	304.06	2.4	12.6	1	312.91	13.2	2.91%	2	308.05	59.5	
c205C10	2	228.28	27.5	0.2	2	283.72	47.0	24.29%	2	281.46	48.2	18.89%
r102C10	3	249.19	21.6	0.4	4	337.19	237.9		4	295.25	231.6	
r103C10	2	206.12	2966	7.5	2	233.02	56.9	13.05%	2	242.08	60.4	14.85%
r201C10	1	241.51	0.9	2.2	2	280.32	67.0		2	373.76	67.0	
r203C10	1	218.21	0.2	1.1	1	247.93	13.1	13.62%	2	251.84	56.2	
rc102C10	4	423.51	63.5	0.6	4	499.25	415.4	17.88%	4	429.19	512.5	1.32%
rc108C10	3	345.93	1671.3	0.7	3	370.14	137.0	7.00%	3	393.17	162.2	12.02%
rc201C10	1	412.86	422.6	5400.1	2	379.52	67.7		2	440.58	73.6	
rc205C10	2	325.98	115.8	0.5	2	399.46	63.2	22.54%	2	592.60	69.7	44.99%
c103C15	3	348.46	5400.1	145.1	4	648.08	782.2		4	55.23	716.9	
c106C15	3	275.13	2201.4	0.7	4	516.30	819.1		4	529.70	634.0	
c202C15	2	383.62	5400.5	40	2	549.69	140.0	43.29%	2	623.38	213.4	38.46%
c208C15	2	300.55	5400.2	1.8	2	430.27	112.4	43.16%	2	288.10	111.1	-4.32%
r102C15	5	412.78	5537.3	1349.7	9	612.10	3475.3		7	617.26	2238.8	
r105C15	4	336.15	5481.7	36.7	6	590.87	1696.3		7	478.04	1901.3	
r202C15	2	358	5400.6	32.4	2	593.91	136.7	65.90%	2	608.09	241.9	41.13%
r209C15	1	313.24	209.7	51.6	2	925.85	141.4		2	554.29	156.7	
rc103C15	4	397.67	5428.7	357.3	5	514.05	1171.1		5	619.19	1154.1	
rc108C15	3	370.25	5400.5	144	5	584.61	1148.1		5	463.72	1416.5	
rc202C15	2	394.39	5400.1	4.2	2	551.73	132.0	39.89%	2	546.71	164.0	27.86%
rc204C15	2	310.57	5400.1	1107.2	1	420.66	22.2		2	476.06	139.4	34.76%
Avg.								15.99%				17.55%

Table 4: A comparison of different recharge strategies for the VNS/TS heuristic with a PR CPLEX solution for the small instances.

5.4 Mixed level recharge strategies on small instances

Next, we assess the performance of the LR recharge strategy. More precise, we will compare a mixed recharge strategy of LR-100 & LR-75 (Combination A) to a mixed recharge strategy of LR-100 & LR-50 (Combination B). The results are based on one run of the fast implementation. The results will be compared to the same PR model results from the previous section. Table 5 provides an overview of the results.

The Combination A appears to be outperforming Combination B. Both use an extra vehicle compared to the solution of the model for one instance. Furthermore, Combination A improves upon the lower-bound for the number of the model once. For the remaining cases, the number of vehicles is equal to the CPLEX solution. The same distance was found for 21 and 19 instances by Combination A and B, respectively. The

gap for Combination A is also lower with 3.03% compared to 5.55%. Only in run time is Combination B better with a 4 seconds shorter average run time. This is even more remarkable considering that Combination B calculated one route more. No logical explanation for this was found.

	PR CPLEX		LR-100 & LR-75 (A)				LR-100 & LR-50 (B)			
Instance	$\#Veh$	TD	$\#Veh$	TD	$t(s)$	ΔTD	$\#Veh$	TD	$t(s)$	ΔTD
c101C5	2	257.75	2	257.75	9.6	0.00%	2	257.75	8.6	0.00%
c103C5	1	175.37	1	176.05	2.7	0.39%	1	176.05	2.4	0.39%
c206C5	1	242.56	1	242.56	2.3	0.00%	1	242.56	2.0	0.00%
c208C5	1	158.48	1	158.48	2.0	0.00%	1	158.48	1.7	0.00%
r104C5	2	136.69	2	136.69	7.0	0.00%	2	136.69	6.0	0.00%
r105C5	2	156.08	2	156.08	8.3	0.00%	2	156.08	8.2	0.00%
r202C5	1	128.78	1	152.59	2.0	18.49%	1	152.59	1.8	18.49%
r203C5	1	179.06	1	179.06	2.6	0.00%	1	179.06	2.4	0.00%
rc105C5	2	233.77	2	243.09	9.6	3.99%	2	241.30	9.3	3.22%
rc108C5	2	253.93	2	253.93	8.9	0.00%	2	253.93	9.1	0.00%
rc204C5	1	176.39	1	176.39	2.5	0.00%	2	185.16	7.0	
rc208C5	1	167.98	1	167.98	2.0	0.00%	1	200.18	2.0	19.17%
c101C10	3	388.25	3	399.31	51.8	2.85%	3	420.06	43.3	8.19%
c104C10	2	273.93	2	273.93	24.9	0.00%	2	279.93	19.2	2.19%
c202C10	1	304.06	1	351.57	11.7	15.62%	1	304.06	8.9	0.00%
c205C10	2	228.28	2	228.28	40.0	0.00%	2	228.28	30.9	0.00%
r102C10	3	249.19	3	249.19	53.7	0.00%	3	249.19	46.6	0.00%
r103C10	2	206.12	2	210.39	27.2	2.07%	2	210.39	23.3	2.07%
r201C10	1	241.51	1	254.32	9.7	5.30%	1	241.51	9.5	0.00%
r203C10	1	218.21	1	247.00	9.8	13.19%	1	340.02	10.3	55.82%
rc102C10	4	423.51	4	423.51	87.2	0.00%	4	428.93	96.0	1.28%
rc108C10	3	345.93	3	345.93	48.9	0.00%	3	345.93	45.0	0.00%
rc201C10	1	412.86	1	420.57	11.5	1.87%	1	412.86	9.9	0.00%
rc205C10	2	325.98	2	325.98	41.8	0.00%	2	325.98	39.5	0.00%
c103C15	3	348.46	3	387.42	42.3	11.18%	3	384.29	38.2	10.28%
c106C15	3	275.13	3	275.13	72.5	0.00%	3	275.13	63.5	0.00%
c202C15	2	383.62	2	383.62	57.7	0.00%	2	383.62	51.6	0.00%
c208C15	2	300.55	2	300.55	51.8	0.00%	2	300.55	47.7	0.00%
r102C15	5	412.78	5	425.26	156.4	3.02%	5	430.24	144.2	4.23%
r105C15	4	336.15	4	336.15	80.7	0.00%	4	356.12	69.7	5.94%
r202C15	2	358.00	2	358.00	75.2	0.00%	2	394.72	66.3	10.26%
r209C15	1	313.24	2	293.20	71.6		1	378.96	16.6	20.98%
rc103C15	4	397.67	4	397.67	78.1	0.00%	4	397.67	74.9	0.00%
rc108C15	3	370.25	3	378.36	57.6	2.19%	3	378.36	42.5	2.19%
rc202C15	2	394.39	2	483.82	88.2	22.68%	2	470.89	66.1	19.40%
rc204C15	2	310.57	1	418.63	18.7		2	341.58	51.8	9.98%
Avg.					36.9	3.03%			32.7	5.55%

Table 5: A comparison of the performance of a LR-100 & LR-75 mixed recharge strategy and a LR-100 & LR-50 mixed recharge strategy to the PR model on the small EVRPTW instances.

5.5 Mixed level recharge strategies on large instances

The last computational experiment investigates the performance of the heuristic using a partial recharge strategy to using full recharge on the large instances. We will compare a full recharge strategy to a mixed strategy of LR-100 & LR-75. In Table 6 the results based on one run are shown. As mentioned before, for these runs value for the parameter η_{tabu} is lowered from 150 to 100.

The results favor the mixed strategy. In 13 instances, the methods did not find the same number of routes. However, the total number of routes over all the instances is the same for both methods. For the instances that the methods found the same number of routes, the mixed strategy decrease the average route

distance by 0.73%. Because FR has a operator less to compute, it is faster. Due to this, the mixed strategy was on average 1.3 minute slower than the average of 21.9 minutes required by FR.

On these instances, Keskin & Çatay (2016) saw a larger improvement when generalizing the EVRPTW to PR-EVRPTW in their ALNS. There, the total number of vehicles decreased by 9. On the other instances, the total distance decreased by 1.64%. Their larger improvement is explained by that the ANLS decides the amount charged completely free, instead of by recharge strategies. This gap can be partially reduced by combining more recharge strategies. When comparing their results to ours, we find that they use a total of 44 vehicles less. On the instances where the number of vehicles match with our FR result, their distance decrease is 14.62%. This is explained by the completely free recharge amount, the decreased number of TS iterations, and the deviations described in Section 5.2.

	FR/LR-100		LR-100 & LR-75				FR/LR-100		LR-100 & LR-75		
Inst.	$\#Veh$	TD	$\#Veh$	TD	ΔTD	Inst.	$\#Veh$	TD	$\#Veh$	TD	ΔTD
c101	13	1059.524	12	1154.48		r112	12	1126.833	11	1165.63	
c102	12	1157.781	12	1054.37	-8.93%	r201	4	1418.64	4	1405.30	-0.94%
c103	11	1046.654	11	1219.14	16.48%	r202	3	1278.969	3	1267.96	-0.86%
c104	11	1274.44	10	1088.16		r203	3	1015.083	3	1093.81	7.76%
c105	12	1398.407	12	1030.13	-26.34%	r204	2	884.7612	2	932.50	5.40%
c106	12	1045.563	12	1105.44	5.73%	r205	3	1274.947	3	1225.48	-3.88%
c107	11	1237.932	12	1029.95		r206	3	1085.802	3	1054.54	-2.88%
c108	11	1145.066	12	1152.26		r207	3	902.0282	3	908.41	0.71%
c109	11	1118.977	11	1228.33	9.77%	r208	2	829.0458	2	877.17	5.80%
c201	4	673.1388	4	662.52	-1.58%	r209	3	1017.354	3	1074.42	5.61%
c202	4	656.718	4	665.52	1.34%	r210	3	1047.177	3	1019.58	-2.63%
c203	4	901.1314	4	648.04	-28.09%	r211	3	878.3616	3	910.12	3.62%
c204	4	920.9186	4	792.04	-13.99%	rc101	17	1910.823	18	1871.48	
c205	4	641.1283	4	673.11	4.99%	rc102	16	1687.883	15	1756.38	
c206	4	643.4497	4	676.04	5.06%	rc103	14	1420.297	14	1418.17	-0.15%
c207	6	713.7961	5	654.67		rc104	12	1298.111	12	1258.97	-3.02%
c208	4	756.3962	4	689.85	-8.80%	rc105	16	1575.13	16	1693.69	7.53%
r101	20	1803.931	20	1740.45	-3.52%	rc106	15	1549.848	15	1537.85	-0.77%
r102	18	1625.732	18	1551.39	-4.57%	rc107	13	1481.899	13	1368.03	-7.68%
r103	15	1399.676	15	1388.99	-0.76%	rc108	14	1277.416	12	1316.15	
r104	12	1202.038	12	1170.92	-2.59%	rc201	5	1520.294	5	1677.55	10.34%
r105	16	1595.159	16	1537.10	-3.64%	rc202	4	1451.605	4	1642.21	13.13%
r106	15	1466.928	15	1341.69	-8.54%	rc203	3	1296.734	3	1355.09	4.50%
r107	12	1194.718	13	1276.12		rc204	3	1074.939	3	1017.55	-5.34%
r107	12	1192.156	12	1139.33	-4.43%	rc205	4	1341.315	5	1372.02	
r109	13	1423.544	14	1296.90		rc206	3	1478.852	3	1578.92	6.77%
r110	12	1188.025	13	1181.24		rc207	3	1216.837	3	1069.65	-12.10%
r111	13	1253.132	13	1207.61	-3.63%	rc208	3	960.0773	3	1091.66	13.71%
Avg.											-0.73%

Table 6: A comparison of the performance of full recharge and a LR-100 and LR-75 mixed recharge strategy on the large EVRPTW instances.

6 Conclusion

In this paper, we aimed to allow for partial recharge in the VNS/TS framework. We found that there is no straightforward way to do this in the framework. By introducing different recharge strategies and a new operator, we were able to improve performance. These effects were mainly noticeable in the distance traveled. For large instances, this decrease was 0.73%

Our VNT/TS did not outperform the ALNS of Keskin & Çatay (2016). However, the combination of a strong performance of the VNS/TS by Schneider et al. (2014) and results of this paper leads us to believe an improved implementation of the VNS/TS with recharge strategies could rival the results of the ALNS.

The implementation is the major limitation of this research. Due to an unknown cause, not all optimal results were found on the small instances. Moreover, the implementation could have been faster by storing the cost savings of operators across iterations. Furthermore, the implementation could have been closer to the well performing VNS/TS of Schneider et al. (2014) by implementing diversification penalties and generator arcs. An improved implementation also should include a tabu structure for the recharge strategies. An other limitation of the research is the idealizing of the real world in the problem.

Lifting these limitations are opportunities for further research. A better implementation could clarify whether the VNS/TS framework is suited for solving the partial recharge case. Other assumptions as homogeneous fleet, neglecting effects of payloads, and known and fixed speeds could be lifted too.

References

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

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

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

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.

Erdoğan, S., & Miller-Hooks, E. (2012). A green vehicle routing problem. *Transportation Research Part E: Logistics and Transportation Review*, *48*(1), 100–114.

Gayialis, S. P., Konstantakopoulos, G. D., & Tatsiopoulos, I. P. (2019). Vehicle routing problem for urban freight transportation: A review of the recent literature. In *Operational research in the digital era–ict challenges* (pp. 89–104). Springer.

Glover, F. (1989). Tabu search—part i. *ORSA Journal on computing*, *1*(3), 190–206.

Goeke, D. (2019, Apr). *Mendeley data - e-vrptw instances.* Retrieved from `https://data.mendeley.com/datasets/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.

Hall, D., & Lutsey, N. (2018). Effects of battery manufacturing on electric vehicle life-cycle greenhouse gas emissions.

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., Laporte, G., & Çatay, B. (2019). Electric vehicle routing problem with time-dependent waiting times at recharging stations. *Computers & Operations Research*, *107*, 77–94.

Lenstra, J. K., & Kan, A. R. (1981). Complexity of vehicle routing and scheduling problems. *Networks*, *11*(2), 221–227.

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

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

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.

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.

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*(1), 159–175.

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.

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

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., et al. (1989). The theory of cyclic transfers.

Toth, P., & Vigo, D. (2003). The granular tabu search and its application to the vehicle-routing problem. *Informs Journal on computing*, *15*(4), 333–346.

US EPA, E. (2009). Inventory of US greenhouse gas emissions and sinks: 1990–2007. *EPA 430-R-09-004*.

Xiao, Y., Zuo, X., Kaku, I., Zhou, S., & Pan, X. (2019). Development of energy consumption optimization model for the electric vehicle routing problem with time windows. *Journal of Cleaner Production*, *225*, 647–663.

Zuo, X., Xiao, Y., You, M., Kaku, I., & Xu, Y. (2019). A new formulation of the electric vehicle routing problem with time windows considering concave nonlinear charging function. *Journal of Cleaner Production*, *236*, 117687.

A Preprocessing

To increase the efficiency of the models, we formulated additional constraints which do not impede with the optimal solutions. For this we define the set $F'_i \subset F'$, which is the set of dummy nodes for each node $i \in F$. For both models, we used the following restrictions:

$$x_{i,0} = 0 \qquad \forall i \in V'_{N+1} \tag{18}$$

$$x_{N+1,i} = 0 \qquad \forall i \in V'_0 \tag{19}$$

Constraint (18) removes the arcs from any node to the starting depot and constraint (19) removes the arcs from the ending depot to any other node.

We also removed the arc satisfying the preprocessing conditions of Schneider et al. (2014). In other words, the following constraint:

$$x_{i,j} = 0 \tag{20}$$

is added to the model if at least one of the following conditions is met:

$$i,j \in V \wedge q_i + q_j > C, \tag{21}$$

$$i \in V'_0, j \in V'_{N+1} \wedge e_i + s_i + t_{ij} > l_j, \tag{22}$$

$$i \in V'_0, j \in V' \wedge e_i + s_i + t_{ij} + s_j + t_{j,N+1} > l_0, \tag{23}$$

$$i,j \in V \wedge x \in F'_0, y \in F'_{N+1} : h(d_{xi} + d_{ij} + d_{jy} > Q. \tag{24}$$

Condition (21) is based on capacity, Conditions (22) and (23) based on the time window constraint, and Condition (24) based on the limits of the battery.

In the Model 1 we also implemented simple upper and lower bound on the objective function. Let $m_{low} = \lfloor (\sum_{\forall i \in V'_{0,N+1}} q_i)/C \rfloor$ and $m_{high} = |V'_{0,N+1}|$. Then the upper and lower bound constraints are:

$$\sum_{i \in V'_{N+1}} x_{0i} \geq m_{Low} \tag{25}$$

$$\sum_{i \in V'_{N+1}} x_{0i} \leq m_{high} \tag{26}$$