Transportation Research Part C 65 (2016) 111–127

Contents lists available at ScienceDirect

Transportation Research Part C

journal homepage: www.elsevier.com/locate/trc

Partial recharge strategies for the electric vehicle routing problem with time windows

Merve Keskin, Bülent Çatay *

Sabanci University, Faculty of Engineering and Natural Sciences, 34956 Tuzla, Istanbul, Turkey

ARTICLE INFO

Article history:
Received 30 May 2015
Received in revised form 16 January 2016
Accepted 16 January 2016
Available online 10 February 2016

Keywords:
Electric vehicle
Vehicle routing problem with time windows
Adaptive large neighborhood search
Metaheuristics
Partial recharge

ABSTRACT

The Electric Vehicle Routing Problem with Time Windows (EVRPTW) is an extension to the well-known Vehicle Routing Problem with Time Windows (VRPTW) where the fleet consists of electric vehicles (EVs). Since EVs have limited driving range due to their battery capacities they may need to visit recharging stations while servicing the customers along their route. The recharging may take place at any battery level and after the recharging the battery is assumed to be full. In this paper, we relax the full recharge restriction and allow partial recharging (EVRPTW-PR), which is more practical in the real world due to shorter recharging duration. We formulate this problem as a 0–1 mixed integer linear program and develop an Adaptive Large Neighborhood Search (ALNS) algorithm to solve it efficiently. We apply several removal and insertion mechanisms by selecting them dynamically and adaptively based on their past performances, including new mechanisms specifically designed for EVRPTW and EVRPTW-PR. These new mechanisms include the removal of the stations independently or along with the preceding or succeeding customers and the insertion of the stations with determining the charge amount based on the recharging decisions. We test the performance of ALNS by using benchmark instances from the recent literature. The computational results show that the proposed method is effective in finding high quality solutions and the partial recharging option may significantly improve the routing decisions.

1. Introduction

Electric vehicles (EVs) move with electric propulsion and can be used in a variety of transport needs such as public transportation, home deliveries from grocery stores, postal deliveries and courier services, and distribution operations in different sectors. Although EVs enable zero- or low-emission logistics services, operating an EV fleet has several drawbacks such as: (i) low energy density of batteries compared to the fuel of combustion engined vehicles; (ii) limited number of public charging stations; and (iii) long recharging times (Touati-Moungla and Jost, 2012). Battery swap may remedy the last; however, swapping raises additional issues in battery design and compatibility, battery degradation, ownership, and swap station infrastructure. Under these limitations, routing an EV fleet arises as a challenging combinatorial optimization problem in the Vehicle Routing Problem (VRP) literature.

The Electric Vehicle Routing Problem with Time Windows (EVRPTW) was introduced by Schneider et al. (2014) as an extension to the Green Vehicle Routing Problem (GVRP) of Erdoğan and Miller-Hooks (2012). GVRP concerns "green"

* Corresponding author. Tel.: +90 (216) 483 9531; fax: +90 (216) 483 9550.
E-mail addresses: mervekeskin@sabanciuniv.edu (M. Keskin), catay@sabanciuniv.edu (B. Çatay).

http://dx.doi.org/10.1016/j.trc.2016.01.013
0968-090X/

vehicles which run with biodiesel, liquid natural gas, or CNG, and have a limited driving range. Hence, the vehicles may need refueling along their route. Refueling is fast; however, the stations for these fuels are scarce. EVRPTW is a variant of the classical VRPTW where the fleet consists of EVs that may need to visit stations to have their batteries recharged in order to continue their route, as in GVRP. On the other hand, the recharging operation may take a significant amount of time, especially when compared to relatively short refueling times of liquid fuels. Recharging may take place at any battery level and the recharge time is proportional to the amount charged. After the recharge the battery is assumed to be full. The number of stations is usually small and the stations are dispersed in distant locations, which increases the difficulty of the problem. In this paper, we relax the full recharging (FR) restriction and allow partial recharging (PR) which is more practical in the real world due to shorter recharging duration. When the vehicle visits a station near the end of its route, FR may not be needed for the vehicle to return to the depot. A similar situation may exist between two consecutive recharges. Saving from recharging time may allow the vehicle to catch the time window of an otherwise unvisited customer, thus, may improve the solution.

In the PR scheme, the recharge quantity is associated with a continuous decision variable. We refer to this problem as EVRPTW and Partial Recharges (EVRPTW-PR) and formulate it as 0–1 mixed integer linear program. Note that determining the recharge quantities brings significant difficulties to the problem. Since the problem is intractable for large instances, we propose an Adaptive Large Neighborhood Search (ALNS) approach to solve it efficiently. ALNS is based on the destroy-and-repair framework where at each iteration the existing feasible solution is destroyed by removing some customers and recharging stations from their routes and then repaired by inserting the removed customers to the solution along with stations when recharging is necessary. Several removal and insertion algorithms are applied by selecting them dynamically and adaptively based on their past performances**.** The new solution is accepted according to the Simulated Annealing criterion. Our approach combines the removal and insertion mechanisms presented in Ropke and Pisinger (2006a, 2006b), Pisinger and Ropke (2007) and Demir et al. (2012) with some new mechanisms designed specifically for EVRPTW and EVRPTW-PR. Our computational tests show that the proposed ALNS is effective in finding good quality solutions and improves some of the best-known solutions in the literature. Furthermore, our results reveal that the PR scheme may substantially improve the routing decisions.

The contributions of this study can be summarized as follows:

- We extend EVRPTW to a PR scheme, which is more general and practical, and present the mathematical programming formulation of the problem.
- We propose an effective ALNS method to solve the EVRPTW and EVRPTW-PR. The proposed method introduces new removal and insertion mechanisms to tackle the more complex problem structure of VRPs where the fleet consists of EVs.
- We validate the performance of the proposed method using the EVRPTW instances of Schneider et al. (2014) and improve the best-known solutions of four problems.
- We show that the PR scheme may improve the solutions obtained with the FR scheme substantially.

The remainder of the paper is organized as follows: Section 2 reviews the related studies in the literature. Section 3 describes the problem and formulates the mathematical model. The proposed ALNS method is presented in Section 4. Section 5 provides the computational study and discusses the results. Finally, concluding remarks and future research directions are given in Section 6.

2. Related literature

There are relatively few studies on route optimization of AFVs. Artmeier et al. (2010) studied this problem within a graph-theoretic context and proposes extensions to general shortest path algorithms that address the problem of energy-optimal routing. They formalize energy-efficient routing in the presence of rechargeable batteries as a special case of the constrained shortest path problem and present an adaption of a general shortest path algorithm that respects the given constraints. Wang and Shen (2007) developed a model that minimizes the number of tours and total deadhead time hierarchically. The driving range of the vehicle is limited but the charging durations, time windows and vehicle capacities are not considered. A multiple ant colony algorithm was proposed to solve the problem.

Conrad and Figliozzi (2011) introduced the Recharging VRP (RVRP), a new variant of the VRP where the EVs are allowed to recharge at selected customer locations. The model has dual objectives: the primary objective minimizes the number of routes or vehicles whereas the secondary objective minimizes the total costs associated with the travel distance, service time and vehicle recharging. The latter is a penalty cost incurred at each recharge. The EV is charged while servicing the customer and the charging time is constant. The battery level departing from a customer depends on the choice of full charge or partial charging. In the partial charge case the battery is charged to a specified level such as 80% of battery capacity. Conrad and Figliozzi (2011) used an iterative construction and improvement procedure to solve this problem but did not provide its details.

Wang and Cheu (2012) investigated the operations of an electric taxi fleet. Their model minimizes total distance traveled under the recharging constraints and maximum route time. The battery is consumed at a given rate per distance and can be replenished at the recharging stations. Charging times are constant and after charging the battery becomes full. They con-

struct an initial solution using one of the nearest-neighbor, sweep and earliest time window insertion heuristics and improve it using Tabu Search (TS). They also suggested three different recharging plans which provide different driving ranges and compared the results against the full charging scheme.

Omidvar and Tavakkoli-Moghaddam (2012) tackled an AFV routing problem with time-windows and proposed a mathematical model that minimizes total costs related to the vehicles, distance traveled, travel time and emissions. The refueling times are assumed constant. They used the Simulated Annealing (SA) and Genetic Algorithm (GA) approaches and compared their performances. Worley and Klabjan (2012) addressed the problem of locating recharging stations and designing EV routes simultaneously. The objective is to minimize the sum of the travel costs, recharging costs, and costs of locating recharging stations. A solution method was not proposed and left as future work.

Erdoğan and Miller-Hooks (2012) considered the routing of AFVs within the context GVRP and formulated the mathematical model. The model aims at minimizing the total distance traveled where the length of the routes is restricted. Fuel is consumed with a given rate per traveled distance and can be replenished at the alternative fuel stations. Refueling times are assumed to be fixed and after refueling the tank becomes full. The model does not involve time windows and vehicle capacity constraints. Erdoğan and Miller-Hooks (2012) proposed two heuristics to solve the GVRP. The first is a Modified Clarke and Wright Savings (MCWS) algorithm which creates routes by establishing feasibility through the insertion of AFSs, merging feasible routes according to savings values, and removing redundant AFSs. The second is a Density-Based Clustering Algorithm (DBCA) based on the cluster-first and route-second approach. DBCA forms clusters of customers such that every vertex within a given radius contains at least a predefined number of neighbors. Subsequently, the MCWS algorithm is applied to the identified clusters. To test the performance of these two heuristics, they designed two sets of problem instances. The first consists of 40 small-sized instances with 20 customers while the second involves 12 instances with up to 500 customers.

Recently, Felipe et al. (2014) extended GVRP for EVs by allowing partial recharges using multiple technologies, i.e. using different power options. As in GVRP, the problem does not involve time windows but EVs have capacity and total route duration limits. The authors formulated the mathematical programming model and presented constructive and deterministic local search algorithms as well as a metaheuristic extension based on an SA framework. The computational tests on both randomly generated and GVRP data showed that using partial recharge strategies and providing multiple recharge technologies could achieve cost and energy savings and ensure feasibility in some instances.

Schneider et al. (2014) developed a hybrid metaheuristic that combines the Variable Neighborhood Search (VNS) algorithm with TS for solving EVRPTW. They tested the performance of the proposed method on benchmark instances of GVRP and Multi-Depot VRP with Inter-Depot Routes. They also generated new instances for EVRPTW by modifying Solomon (1987) data and reported their results. Desaulniers et al. (2014) tackled the same problem by considering four recharging strategies (single-FR, single-PR, multiple-FR, and multiple-PR) and attempted to solve them optimality using branch-price-and-cut algorithms. Goeke and Schneider (2015) extended EVRPTW to the routing of a mixed fleet of EVs and internal combustion engine (ICE) vehicles. Their objective function consisted of an energy consumption function of speed, gradient, and cargo load distribution, and they proposed an ALNS approach to solve it. Hiermann et al. (2015) addressed the Fleet Size and Mix Vehicle Routing Problem with Time Windows where the fleet consists of EVs. They also implemented an ALNS algorithm equipped with local search and labeling procedures.

3. Problem description and model formulation

EVRPTW-PR concerns a set of customers with known demands, delivery time windows, and service durations. The deliveries are performed by a homogeneous fleet of EVs with fixed loading capacities and limited cruising ranges. While the vehicle is traveling, the battery charge level decreases proportionally with the distance traversed and the vehicle may need to visit a recharging station in order to continue its route. The battery is recharged at any quantity and the duration of the recharge depends on the initial state of battery charge. Unlike EVRPTW where the vehicle departs from the depot/station with full battery and arrives at the depot/station with any state of charge, in EVRPTW-PR the vehicle departs from the depot fully charged but may arrive at/depart from a station with any state of charge and it returns to depot with an empty battery if it has been recharged once during its route.

Fig. 1 illustrates an example involving ten customers (C1–C10), four stations (S1–S4), and the depot (D) which can also be used for recharging. The percentage values along the routes show the battery state of charge when the vehicle arrives at a customer or a station and when it departs from the station after having its battery recharged. EV1 services C1 and C2, returns to the depot without any recharging. EV2 visits S1 after servicing C4 and has its battery recharged before visiting C5 and C3. On the other hand, EV3 is recharged once in S4 and twice in S3. Note that a station can be visited multiple times by the same (see S3) or different vehicles and each station is not necessarily visited (see S2). In what follows, we provide the mathematical model for EVRPTW-PR following the notation and formulation of Schneider et al. (2014).

Let $V = \{1, \ldots, N\}$ denote the set of customers and F denote the set of recharging stations. Since a recharging station may be visited more than once depending on the route structure, we create F' which is the set of dummy vertices generated to permit several visits to each vertex in the set F. Vertices 0 and $N+1$ denote the depot and every route starts at 0 and ends at $N+1$. Let V' be the set of vertices with $V' = V \cup F'$. In order to indicate that a set contains the respective instance of the depot, the set is subscripted with 0 or $N+1$. Hence, $F'_0 = F' \cup \{0\}$, $V'_0 = V' \cup \{0\}$, and $V'_{N+1} = V' \cup \{N+1\}$. Now we can define the problem on a complete directed graph $G = (V'_{0,N+1}, A)$ with the set of arcs $A = \{(i,j) | i,j \in V'_{0,N+1}, i \neq j\}$. Each arc is associated

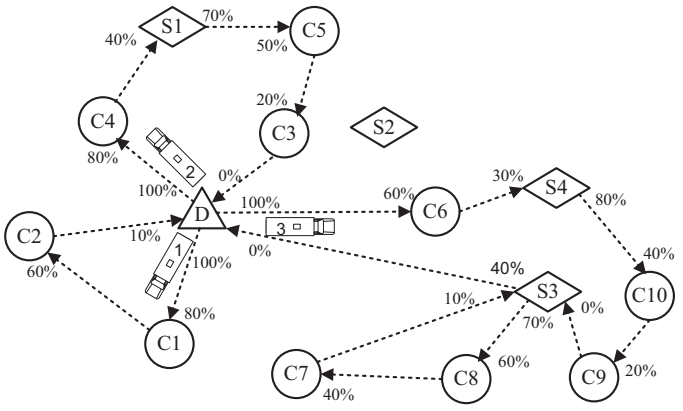


Fig. 1. An illustrative example.

with a distance d_{ij} and travel time t_{ij}. The battery charge is consumed at a rate of h and every traveled arc consumes $h \cdot d_{ij}$ of the remaining battery. Each customer $i \in V'$ has positive demand q_i, service time s_i and time window $[e_i, l_i]$. All EVs have a load capacity of C and a battery capacity of Q. At a recharging station, the battery is charged at a recharging rate of g. The decision variables, τ_i, u_i and y_i keep track of the service starting time, remaining cargo level and battery state of charge on arriving to station $i \in V'_{0,N+1}$, respectively. The binary decision variable x_{ij} takes value 1 if arc (i,j) is traversed and 0 otherwise. In Schneider et al. (2014) the battery is always recharged to full capacity, i.e. the recharge amount is $(Q - y_i)$. EVRPTW-PR allows partial recharges by defining a new decision variable Y_i which represents the battery state of charge on departure from station i.

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

subject to

$$\sum_{j \in V'_{n+1}, i \neq j} x_{ij} = 1 \quad \forall i \in V \tag{2}$$

$$\sum_{j \in V'_{n+1}, i \neq j} x_{ij} \leqslant 1 \quad \forall i \in F' \tag{3}$$

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

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

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

$$e_j \leqslant \tau_j \leqslant l_j \quad \forall j \in V'_{0,n+1} \tag{7}$$

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

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

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

$$0 \leqslant y_j \leqslant Y_i - (h \cdot d_{ij}) x_{ij} + Q(1 - x_{ij}) \quad \forall i \in F'_0, \forall j \in V'_{n+1}, i \neq j \tag{11}$$

$$y_i \leqslant Y_i \leqslant Q \quad \forall i \in F'_0 \tag{12}$$

$$x_{ij} \in \{0, 1\} \quad \forall i \in V'_0, \forall j \in V'_{n+1}, i \neq j \tag{13}$$

The objective function (1) minimizes total distance traveled. Constraints (2) and (3) handle the connectivity of customers and visits to recharging stations, respectively. The flow conservation constraints (4) enforce that the number of outgoing arcs equals to the number of incoming arcs at each vertex. Constraints (5) and (6) ensure the time feasibility of arcs leaving the customers (and the depot), and the stations, respectively. Constraints (7) enforce the time windows of the customers and the depot. In addition, constraints (5)–(7) eliminate the sub-tours. Constraints (8) and (9) guarantee that demand of all customers are satisfied. Constraints (10) and (11) keep track of the battery state of charge and make sure that it is never negative. Constraints (12) determine the battery state of charge after the recharge at a station and make sure that the battery state of charge does not exceed its capacity. Finally, constraints (13) define the binary decision variables.

Proposition 1. *If an optimal solution exists such that an EV leaves the depot with its battery partially charged, i.e.* $Y_0^* < 1$, *then the same EV departing from the depot fully charged is also optimal, i.e.* $Y_0^* = 1$ *is also optimal.*

Proof. Let $Y_0^* < 1$ be optimal. Since fully recharging the battery at the depot does not delay the departure time of the EV, $Y_0^* = 1$ must also be optimal. □

Corollary 1. *If* $Y_0^* < 1$ *is optimal, then the problem has infinite multiple optima.*

Proof. Let $\overline{Y}_0 < 1$ be optimal and $\overline{Y}_0 + \varepsilon \leqslant 1$ not, where ε is a small positive scalar. Then following Proposition 1, multiple optima exist such that $\overline{Y}_0 \leqslant Y_0^* \leqslant 1$. □

Proposition 2. *If an optimal solution exists such that an EV has been recharged at least once and returns to the depot at the end of its route with positive battery state, i.e.* $y_{n+1}^* > 0$, *then its return to the depot with empty battery is also optimal, i.e.* $y_{n+1}^* = 0$ *is also optimal.*

Proof. Let $y_{n+1}^* > 0$ be optimal. Since recharging the battery with less energy at the preceding station does not delay the departure time to cause any time window infeasibility, $y_{n+1}^* = 0$ must also be optimal. □

Corollary 2. *If* $y_{n+1}^* > 0$ *is optimal, then the problem has infinite multiple optima.*

Proof. Let $\bar{y}_{n+1} > 0$ be optimal and $\bar{y}_{n+1} + \varepsilon$ not, where ε is a small positive scalar. Then following Proposition 2, multiple optima exist such that $0 \leqslant y_{n+1}^* \leqslant \bar{y}_{n+1}$. □

Without loss of generality, we assume that an EV departs from the depot with a battery charged in full and returns to the depot with its battery fully consumed if it has been recharged at least once along its route.

4. Solution methodology

We propose an ALNS method to solve the EVRPTW-PR. ALNS was introduced by Ropke and Pisinger (2006a) as an extension of the Large Neighborhood Search (LNS) framework put forward by Shaw (1998). Since local search methods can only make small changes to an existing solution their search space is narrow. Hence, they are unable to move from one promising area to another within the feasible region. To overcome this shortcoming, Ropke and Pisinger (2006a) considered large moves that rearrange up to 40% of the vertices instead of using small moves that relocate or exchange only a few arcs or vertices at each iteration. In a subsequent study, Ropke and Pisinger (2006b) developed a unified ALNS heuristic for a large class of VRP with Backhauls. Pisinger and Ropke (2007) improved this heuristic with additional algorithms and showed that the proposed framework gives competitive results in different VRP variants. Since then, ALNS has been successfully implemented to solve various VRPs, e.g. cumulative capacitated VRP (Ribeiro and Laporte, 2012), pollution-routing problem (Demir et al., 2012), two-echelon VRP (Hemmelmayr et al., 2012), pickup and delivery problems with transshipment (Qu and Bard, 2012) and with vehicle transfers (Masson et al., 2013), VRP with multiple routes (Azi et al., 2014), periodic inventory routing problem (Aksen et al., 2014), and production routing problem (Adulyasak et al., 2014).

4.1. Overview of the proposed ALNS approach

4.1.1. Initial solution construction

The initial solution is obtained by iteratively constructing feasible routes. The route construction begins with the nearest customer to the depot. Then, the insertion costs of all unassigned customers to all possible existing positions in the route are determined respecting the time window constraints, i.e. the insertion of customer i between nodes j and k is calculated as

$d_{ji} + d_{ik} - d_{jk}$ if customers i, j, and k can be visited within their time windows. Next, the best insertion is performed. If no customer can be inserted because of low battery level, we use the Greedy Station Insertion algorithm described in Section 4.2.2 and insert a customer along with a recharging station. In that case, the insertion cost becomes the difference between the total distance of the route after and before the insertions of the customer and the recharging station. When no customer can be inserted to the route due to capacity or time-window constraints the route is finalized and the procedure is repeated by starting with a new route until all customers have been visited. The pseudocode of the initial solution construction procedure is given in Algorithm 1.

Algorithm 1: Initial solution construction

```
Start a new route with the customer closest to the depot
repeat
    Calculate insertion costs of all unserved customers to the current route
    if no customer can be added then
        Start a new route with the unserved customer closest to the depot
    else
        Select the customer which increases the distance least and make the insertion
    end if
    if a recharging station is needed then
        Perform Greedy Station Insertion
    end if
until all customers are served
```

4.1.2. ALNS procedure

The proposed ALNS heuristic includes four classes of algorithms: Customer Removal (CR), Customer Insertion (CI), Station Removal (SR), and Station Insertion (SI). After the initial solution has been constructed, ALNS tries to improve it iteratively until a stopping condition is satisfied. We use an iteration number limit to terminate the heuristic. At each iteration, the existing feasible solution is destroyed by removing some nodes from their routes using a removal algorithm. These nodes consist of customers, recharging stations, or both. The resulting partial solution is then repaired using an insertion algorithm which heuristically inserts removed customers and/or recharging stations (removed or other) to the existing or new routes in an attempt to obtain a better solution than the previous. Several removal and insertion algorithms are applied by selecting them dynamically and adaptively based on a probability calculated using their performances in the previous iterations. In order to calculate the selection probabilities, an adaptive weight w and a score π is assigned to each algorithm. High score corresponds to a successful mechanism and hence the mechanism should be selected with larger probability. Initially, all weights are equal and all scores are 0. In an iteration, if a new best solution has been found, then the scores corresponding to the removal and insertion algorithms which achieved that solution are increased by σ_1. If the algorithms have yielded a better solution than the previous then the scores are increased by σ_2. Finally, if the new solution is worse than the previous but accepted using the simulated annealing rule then the scores are increased by σ_3. The procedure is divided into segments which consist of N_C iterations for customer related mechanisms and N_S for station related mechanisms. At the end of each segment s, the weight of algorithm a is updated using the formula $w_a^{s+1} = w_a^s(1-\rho) + \rho\pi_a/\theta_a$, where ρ is the roulette wheel parameter, θ_a is the number of times it was used during segment s and π_a is the score associated with algorithm a. After updating the weights, the probabilities of the algorithms which will be used in the next segment $(s+1)$ are calculated using the formula $P_a^{s+1} = w_a^s / \sum_{l=1}^{m} w_l^s$ and the scores are reset to zero.

The simulated annealing (SA) approach that accepts or rejects a solution is implemented as follows: if the number of vehicles in the new solution is smaller than that of the current best solution or if they are equal but the total distance of the new solution is shorter then we accept the new solution. On the other hand, we reject the new solution if it requires more vehicles. When the numbers of vehicles are equal but the distance is longer the new solution is accepted with probability $e^{-(f(X_{New})-f(X_{Current})/T}$, where $f(X)$ denotes the total distance of solution X, X_{New} and $X_{Current}$ are the new and current best solutions, respectively, and T is the current temperature. T is initially set to T_{init} and decreased at every iteration using the formula $T = T \times \varepsilon$, where $0 < \varepsilon < 1$ is the *cooling rate* parameter. T_{init} is determined using the *initial temperature control parameter* μ such that a solution which is μ% worse than the initial solution is accepted with probability 0.5.

4.2. Removal algorithms

4.2.1. Customer removal

The current solution is destroyed by removing γ customers from the solution according to different rules and adding them in a removal list $\mathcal{L}$. γ depends on the total number of customers n_c and is determined randomly between $\underline{n_c}$ and $\overline{n_c}$ using a uniform distribution. The removal algorithms are selected in an adaptive manner from the set of algorithms CR.

We utilize *Random*, *Worst-Distance*, *Worst-Time*, *Shaw*, *Proximity-Based*, *Demand-Based*, *Time-Based*, *Zone Removal* algorithms presented in the literature and adopt the Route Removal algorithms presented in Emeç et al. (2016). Random Removal algorithm selects γ customers randomly and removes them from the solution. Worst-Distance Removal algorithm determines customers with high cost, where cost is the sum of distances of the customer from the preceding and succeeding nodes in the route. Then, it removes the customer with $\lfloor|\gamma|\lambda^{\kappa}\rfloor^{th}$ highest cost where $\lambda \in [0,1]$ is a random number and $\kappa \geqslant 1$ is a parameter which introduces randomness in the selection of customers to avoid the selection of the same customers repeatedly and is referred to as *worst removal determinism factor*. Worst-Time Removal algorithm is similar to Worst-Distance Removal algorithm where the cost of customer i is calculated as $|\tau_i - e_i|$.

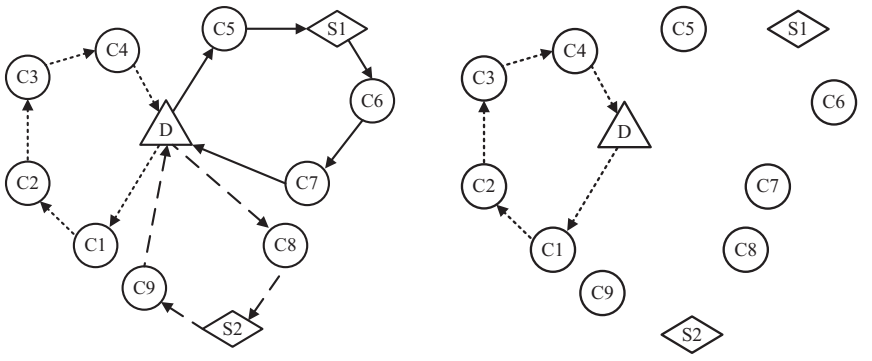
Shaw Removal is designed to remove customers that are similar to each other with respect to several criteria and uses the following relatedness measure: $R_{ij} = \phi d_{ij} + \phi_2|e_i - e_j| + \phi_3 l_{ij} + \phi_4|D_i - D_j|$ where $\phi_1 - \phi_4$ are the Shaw parameters and $l_{ij} = -1$ if i and j are in the same route, and 1 otherwise. Small R_{ij} means high similarity. The algorithm first selects a customer i randomly. Then, it sorts the non-removed customers in the non-decreasing order of their relatedness value with a customer i and chooses the customer listed in position $\lfloor|\gamma|\lambda^{\eta}\rfloor$ where η is a parameter called *Shaw removal determinism factor*. Proximity, Demand and Time-Based Removals are the special cases of *Shaw Removal* where ϕ_1, ϕ_2, ϕ_4 takes value 1, respectively and the other parameters are assumed to be 0. In the *Zone Removal*, the Cartesian coordinate system in which the nodes are located is divided into n_Z many smaller areas that are called as zones. A zone is randomly selected and all the customers in that zone are removed from the solution. More details of these algorithms can be found in Demir et al. (2012).

Random Route Removal (RRR) randomly chooses ω routes and removes all the customers visited in those routes. ω depends on the number of routes in the current solution and is determined randomly between 10% and m_r% of total number of routes. *Greedy Route Removal* (*GRR*) algorithm removes ω routes in a greedy way. ω is determined in the same way as in RRR. The routes are sorted in the non-decreasing order of the number of customers serviced and ω routes are removed starting from the first route in the order. The motivation is to distribute the customers in shorter routes into other existing routes in the solution in an attempt to reduce the number of vehicles. The procedure is illustrated in Fig. 2 for $\omega = 2$.

Note that after a predetermined number of iterations N_{RR} we explicitly perform RRR and GRR for n_{RR} iterations to extensively attempt to reduce the number of vehicles used. RRR and GRR remove the complete routes from the solution. On the other hand, since other removal algorithms remove customers from their routes the battery state, time, and remaining capacity of the EV at its arrival to a node should be updated. Furthermore, some recharges may no longer be necessary and those stations may be removed from the solution. In fact, an EV may visit a recharging station right before or after servicing a customer, and it might be beneficial to remove the customer from the solution with its preceding or succeeding station. So, we introduce the following two operators for the customer removal algorithms in addition to removing customers only (RCO) option:

Remove Customer with Preceding Station (RCwPS): We remove the customer in the removal list along with the preceding station, if any exists. The idea is to eliminate the visit to a station where recharging is not necessarily needed at that battery state if EV no longer visits the removed customer.

Remove Customer with Succeeding Station (RCwSS): We remove the customer in the removal list along with the succeeding station, if any exists. The idea is similar to RCwPS. The recharging may be needed after departing from a customer in order to be able to reach the next customer in the route. In that case, recharging is not necessarily needed at that battery state if the departure customer is removed from the solution and the station can be removed as well.



(a) Current feasible solution (b) Solution after GRR

Fig. 2. Illustration of Greedy Route Removal.

4.2.2. Station removal

The recharging stations are the crucial components of the problem. Hence, removing them or changing their positions in the visit sequence of a route may also improve the solution. So, after a pre-determined number of iterations, an SR (followed by a Station Insertion) procedure is applied. The number of stations to be removed σ is determined in a similar fashion to γ based on the total number of stations in the current station. The *Random Station* and *Worst Distance Station Removal* mechanisms are similar to their customer removal counterparts. We also use the *Worst-Charge Usage Station Removal* which aims at removing the stations visited with high battery levels and *Full Charge Station Removal* which aims at promoting the PR option.

Worst-Charge Usage Station Removal: The motivation of this algorithm is to make use of the battery as much as possible before a recharging is needed and increase the efficiency in using the stations. We promote the removal of the stations which an EV visits with relatively higher charge level. The stations are sorted in the non-increasing order of the battery level of the EVs that visit them for recharging and σ stations are removed starting from the first station in the order.

Full Charge Station Removal: The algorithm identifies the stations where EVs are fully charged and removes γ of them randomly.

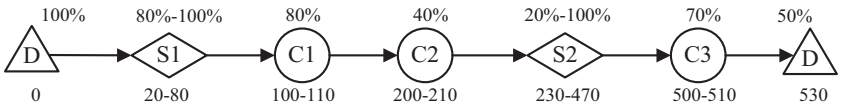
After the removal algorithms, the destroyed solution may become infeasible with respect to the state of charge. Consider the route shown in Fig. 3(a) as an example. The% numbers above the route indicate the state of charge at the arrival to and departure from a node whereas the numbers under the route show the arrival and departure times. When S1 is removed from the route, the EV can still visit C1 and C2 in the given sequence. However, since its battery is empty, the recharging takes longer at S2, which delays its arrival to C3. Since the EV departs from C3 at a later time, it cannot return to D before the latest arrival time of 550 as shown in Fig. 3(b).

Fig. 4 illustrates how the battery infeasibility may occur after a SR. Consider the feasible route in Fig. 4(a). The EV is charged to full at S1. However, when S1 is removed, the battery level is not sufficient to return to the depot after visiting C3 as shown in Fig. 4(b).

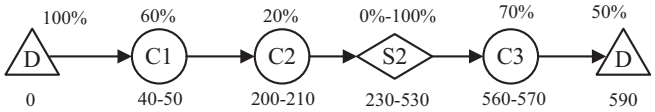
4.3. Insertion algorithms

4.3.1. Customer insertion

We use the *Greedy* and *Regret Insertion* algorithms from the literature. Greedy Insertion algorithm determines the best insertion position for customer *i* by calculating the cost of inserting it between all feasible pairs of nodes *j* and *k* and selecting the position with the minimum cost. The procedure is repeated for all customers and the customer who has the minimum insertion cost is inserted to its designated position. Regret-*k* Insertion prevents the myopic nature of Greedy Insertion by

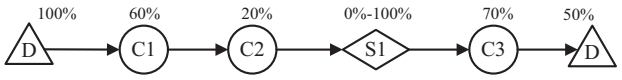


(a) Feasible route before SR

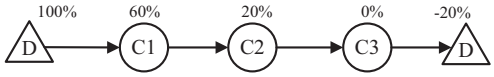


(b) Time-window infeasible route after SR

Fig. 3. Example of time-window infeasibility after SR.



(a) Feasible route before SR



(b) Battery infeasible route after SR

Fig. 4. Example of battery infeasibility after SR.

avoiding the customers which may yield higher costs in the subsequent iterations. It calculates the difference between the cost of the first and k^{th} best insertions of the customers and insert the one with the highest difference to its best position. In our ALNS we utilize *Regret*-2 and *Regret*-3 methods. In addition, we propose the *Time Based Insertion* and adapt the *Zone Insertion* of Demir et al. (2012) as follows:

Time Based Insertion: In this algorithm, the insertion cost is calculated as the difference between the total route times before and after the insertion of a customer. For each customer, the algorithm determines the best insertion position among all routes based on this insertion cost. The customer that increases the route time the least is selected and inserted. The procedure is repeated for the remaining customers until all customers are inserted. The aim of this algorithm is to increase the number of customers visited by an EV by combining compatible customers with respect to their time windows or distances.

Zone Insertion: The algorithm uses the Time Based Insertion criterion above when selecting a customer. However, instead of investigating all routes in the solution, it considers the routes within a randomly selected zone only. The zones are determined in the same way with *Zone Removal*.

Note that a customer insertion may be feasible with respect to service time-window but infeasible with respect to the battery state (referred to as battery infeasibility). In that case, the Greedy Station Insertion (described in Section 4.2.2) is applied to make the destroyed solution charge feasible.

To determine the battery state of charge and the recharge amount at a station visited in the implementation of CI algorithms we use the assumptions stated at the end of Section 3: an EV departs from the depot with a full battery and returns to the depot by completely consuming its battery if it has been recharged at least once along its route. So, in the case the EV is recharged only once during its route then: (i) if the customer is inserted between the depot and the station the insertion only affects the arrival state of charge at the station; (ii) if the customer is inserted between the station and the depot the recharge amount is increased such that the EV returns to the depot with empty battery.

If multiple recharges exist along the route and the customer is inserted between the depot and the first station visited, we follow the procedure (i) described above. If the customer is inserted between two consecutive stations or between the last station visited and the depot the amount recharged at the last station visited is increased by the additional energy needed to visit that customer. If the recharge duration makes the insertion infeasible with respect to service time window of an existing customer, then we attempt to reduce the recharge duration by increasing the battery charge level at the arrival to that station. This is achieved by recharging the EV longer at the previous station making sure that the time-window feasibility of the customers visited between these two consecutive stations is maintained. In any case, if the insertion is feasible with respect to service time window but battery infeasible the Greedy Station Insertion (see Section 4.2.2) is applied to make the destroyed solution charge feasible.

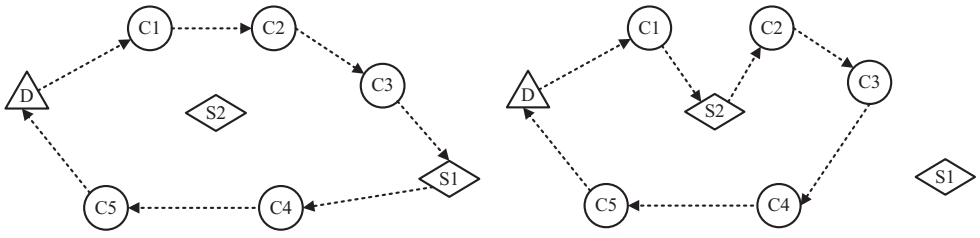
4.3.2. Station insertion

After removing some stations, the current feasible solution may become battery infeasible. In order to repair the solution, stations must be inserted to the infeasible routes. We make an infeasible route feasible by identifying the first customer at which the vehicle arrives with a negative battery level and inserting a station into the partial route prior to that customer. The difference from CI algorithms is that SI algorithms do not necessarily insert the stations which have been removed in SR. Since the stations are always available and it is assumed that as many stations as needed are available, any station can be inserted throughout the algorithm. The SR and SI procedures are illustrated on an example in Fig. 5. A feasible route is depicted in Fig. 5(a). Suppose, S1 is removed using an SR algorithm. Next, S2 is inserted between C1 and C2 by maintaining both time-window and battery feasibility. The resulting route in Fig. 5(b) is shorter than the initial.

We use the following three SI algorithms:

Greedy Station Insertion (GSI): This algorithm determines the first customer in the route at which the vehicle arrives with negative battery level and inserts the "best" (which increases the distance least) station on the arc between that customer and the previous customer. If this insertion is not feasible, then the previous arcs are attempted in the same manner.

Greedy Station Insertion with Comparison: The algorithm determines the best station on the arc leading to the customer where the battery level is negative as in GSI and compares the outcome with the case of inserting the corresponding best



(a) Feasible solution (b) Improved solution after SR and SI

Fig. 5. An improved route after SR and SI procedure.

station on the immediate predecessor arc. The insertion which increases the route distance the least is performed. If both insertions are infeasible, the GSI procedure is applied by considering the previous arcs.

Best Station Insertion: We determine the best station insertions between the customer that the EV arrives at with negative battery level and the depot or a previously visited station by considering all the arcs backwards in the route. We select the best feasible insertion and perform it.

Algorithm 2: ALNS algorithm

```
Generate an initial solution
j ← 1
repeat
    if j ≡ 0(mod N_SR) then
        Select SR algorithm and remove stations
        Select SI algorithm and repair solution
    else if j ≡ 0(mod N_RR) then
        for n_RR iterations do
            Select RRR or GRR algorithm and remove customers
            Select CI algorithm and repair solution
        end for
    else
        Select CR algorithm and remove customers
        if destroyed solution infeasible then
            Perform Greedy Station Insertion
            Select CI algorithm and repair solution
        end if
    end if
    Using SA criterion to accept/reject solution
    j ← j + 1
    if j ≡ 0(mod N_C) then
        Update adaptive weights of CR and CI algorithms
    else if j ≡ 0(mod N_S) then
        Update adaptive weights of SR and SI algorithms
    end if
until stop-criterion met
```

The procedure is repeated for all customers where the EV arrives with negative battery level. If a station insertion cannot be performed feasibly, we return to the previous feasible solution. The battery state of the EV and/or the recharge quantities in the implementation of the SI algorithms are determined in a similar fashion as in CI algorithms.

The pseudocode of the ALNS approach is provided in Algorithm 2.

5. Computational study

To validate the performance of the proposed ALNS approach we perform computational experiments using the EVRPTW data. This data set consists of 36 small and 56 large instances generated by Schneider et al. (2014) based on the well-known VRPTW instances of Solomon. The large instances include three main problem classes where 100 customers and 21 recharging stations are clustered (C), randomly distributed (R), and both clustered and randomly distributed (RC) over a 100×100 grid. Each set has also two subsets, type 1 and type 2, which differ by the length of the time windows and the vehicle load and battery capacities. The small instances include three subsets of 12 problems, each involving 5, 10 and 15 customers drawn randomly from the large instances.

We first tuned the parameters values using a subset of EVRPTW instances. Then, we solved the large EVRPTW problems and compare the results with the benchmarks reported in the literature. Finally, we report the solutions we achieved using the EVRPTW-PR setting and discuss the results. The algorithm was coded in the Java programming language.

5.1. Parameter tuning

Our tuning methodology is in line with those adopted in the literature (Ropke and Pisinger, 2006a, 2006b; Pisinger and Ropke, 2007; Demir et al., 2012; Emeç et al., 2016). We selected six problems and performed ten runs by considering up to 10 values for each parameter. We omitted C1 and C2 problem classes since they usually converged to the same solutions for different parameter values and did not provide much information about the contribution of the parameter value on the solution quality. Consequently, we selected the instances R107, RC101, RC104, RC105, R205 and RC205 for parameter tuning.

For each value we calculated the average percent deviation from the average of the best achieved solutions, determined the one that yielded the least average percent deviation, and fixed the parameter value. We repeated this procedure until all parameter values had been tuned. In EVRPTW, we set the initial values according to the best values reported in Emeç et al. (2016). In EVRPTW-PR, we initialized the values using the best values obtained for EVRPTW and performed another parameter tuning using the same procedure. The parameters, their considered values and the corresponding deviations, and their final values selected are given in Appendix A.

We observed that if the score of the worse solution (σ_3) is greater than the score of the better solution (σ_2) it allows diversification by rewarding non-improved solutions as noted in Ropke and Pisinger (2006a) and Demir et al. (2012). So, our setting of the parameters σ_1, σ_2, and σ_3 also rewards a worse solution more than a better solution as follows: $\sigma_1 \geqslant \sigma_3 \geqslant \sigma_2$.

Ropke and Pisinger (2006a) set the number of iterations to 25,000 and noted that additional runtime had a minor contribution to the solution quality. Our convergence analysis showed similar results. So, we also set the number of iterations to 25,000. The lower and upper limits for the number of customers n_c to be removed are set as $\underline{n_c} = \min\{0.1|N|, 30\}$ and $\overline{n_c} = \min\{0.4|N|, 60\}$, respectively. The removal algorithms are selected in an adaptive manner from the set of algorithms CR.

5.2. Numerical results for EVRPTW instances

In this section, we investigate the effectiveness of the proposed algorithm in the case where partial recharges are not allowed and aim at validating its performance by benchmarking it against state-of-the-art methodologies designed for solving that particular case. Table 1 compares our average results and the average of the results reported in Goeke and Schneider (2015) and Hiermann et al. (2015) to the original results given in Schneider et al. (2014). Our results were obtained using the parameters tuned to the values given in Appendix A. The benchmark results from the literature were also obtained in the same way by fixing the parameters of the corresponding algorithm to the values tuned. The first column in Table 1 denotes the instances. '*#Veh*' and '*TD*' refer to the number of vehicles and the total distance, respectively. '$\Delta\%$' is the percentage deviation from the distances reported in Schneider et al. (2014), if the number of vehicles is same, and is calculated as $(TD^{SSG} - TD^{M})/TD^{M}$, where *SSG* stands for the VNS/TS of Schneider et al. (2014) and *M* refers to the corresponding method, i.e. GS for Goeke and Schneider (2015), HPH for Hiermann et al. (2015), and KÇ for this study. A negative $\Delta\%$ value means improvement.

Our results show that our ALNS approach performs better in type-2 problems but it converges to solutions with one additional vehicle in several instances as compared to other methods. Even though the performance of Hiermann et al. (2015) is better in type-1 problems, the overall performance of our approach is similar. We also observe that the recent work of Goeke and Schneider (2015) has a superior performance, improving many of the best solutions to date. We note that Goeke and Schneider (2015) used the numbers of vehicles achieved in Schneider et al. (2014) as *a priori* information to construct their initial routes, which might have a positive effect both in run time and solution quality. Nevertheless, with fixed parameters our approach improved the best solutions of 11 instances.

In Table 2, we report the best solutions we observed throughout our entire experimental study and compare them against the best known solutions (BKS) from the literature achieved in a similar way. Note that Schneider et al. (2014) and Goeke and Schneider (2015) reported the best results they obtained throughout all experiments they performed as their BKS whereas Hiermann et al. (2015) performed their tests with fixed parameters only. The results in Table 2 also show that our ALNS approach performs better in type-2 problems. Although it finds solutions with one additional vehicle in some instances, it improved the best known solutions of four instances, of which three are type-1 problems. The new best solutions are indicated in bold. Furthermore, it achieved the same best known solution in 16 instances. In comparison, Schneider et al. (2014) found the best known solutions in 18 instances whereas Hiermann et al. (2015) and Goeke and Schneider (2015) improved the solutions of 4 and 30 instances, respectively.

With respect to computational effort, it is not possible to make a fair comparison among the algorithms since they utilize different processors and data structures. We only note that Schneider et al. (2014), Hiermann et al. (2015), and Goeke and Schneider (2015) report average run times of 15.34 min,[1] 15.92 min,[2] and 2.78 min,[3] respectively, whereas our average run time is comparable to the first two with 12.26 min.[4] Goeke and Schneider (2015) has an outstanding run time performance; yet it not possible to assess the contribution of feeding the *a priori* number of vehicles information to this performance.

5.3. Experiments on EVRPTW instances using PR scheme

We first analyze the performance of our ALNS on small EVRPTW instances. This allows us to make a comparison against the (near-)optimal solutions found by using CPLEX 12.6.1 both for the FR and PR cases. Next, we solve the large

[1] Intel Core i5 processor with 2.67 GHz speed and 4 GB RAM, operating on Windows 7 Professional.
[2] Intel Core2 Quad CPU Q6600 processor with 2.40 GHz speed and 4 GB RAM, operating on 64-bit Linux.
[3] Intel Core i7 processor with 2.8 GHz speed and 8 GB RAM, operating on Windows 7 Enterprise.
[4] Intel Xeon E5 processor with 3.30 GHz speed and 32 GB RAM, operating on 64-bit Windows 7.

Table 1
Average results for EVRPTW obtained with fixed parameters.

Instance type	SSG		GS		HPH		KÇ	
	#Veh	*TD*	*#Veh*	*Δ%*	*#Veh*	*Δ%*	*#Veh*	*Δ%*
c1	10.67	1050.04	10.67	−0.31	10.67	0.16	10.89	0.78
c2	4.00	640.92	4.00	0.00	4.00	0.06	4.00	0.00
r1	12.83	1268.60	12.83	−0.80	13.00	0.11	13.25	0.69
r2	2.64	919.04	2.64	−0.47	2.64	0.53	2.82	−0.07
rc1	13.13	1415.84	13.13	−0.50	13.00	−0.48	13.38	0.13
rc2	3.13	1146.76	3.13	−0.15	3.13	0.95	3.25	0.08
Average				−0.41		0.25		0.25

SSG: Schneider et al. (2014), GS: Goeke and Schneider (2015), HPH: Hiermann et al. (2015), and KÇ: Our ALNS.

Table 2
Comparison with the best known solutions of EVRPTW instances.

	BKS			KÇ				BKS			KÇ		
Inst.	*#Veh*	*TD*	Ref.	*#Veh*	*TD*	*Δ%*	Inst.	*#Veh*	*TD*	Ref.	*#Veh*	*TD*	*Δ%*
c101	12	1053.83	SSG	12	1053.83	0.00	c201	4	645.16	SSG	4	645.16	0.00
c102	11	1051.38	GS	11	1056.12	0.45	c202	4	645.16	SSG	4	645.16	0.00
c103	10	1034.86	GS	11	1001.81	–	c203	4	644.98	SSG	4	644.98	0.00
c104	10	961.88	GS	10	**951.57**	**−1.08**	c204	4	636.43	SSG	4	636.43	0.00
c105	11	1075.37	SSG	11	1075.37	0.00	c205	4	641.13	SSG	4	641.13	0.00
c106	11	1057.65	HPH	11	1057.65	0.00	c206	4	638.17	SSG	4	638.17	0.00
c107	11	1031.56	SSG	11	1031.56	0.00	c207	4	638.17	SSG	4	638.17	0.00
c108	10	1095.66	GS	11	1015.68	–	c208	4	638.17	SSG	4	638.17	0.00
c109	10	1033.67	GS	10	1069.16	3.32							
r101	18	1663.04	HPH	18	1679.06	0.95	r201	3	1264.82	SSG	3	1265.67	0.07
r102	16	1487.41	GS	16	1519.80	2.13	r202	3	1052.32	SSG	3	1052.32	0.00
r103	13	1271.35	GS	13	1312.50	3.14	r203	3	895.54	GS	3	895.54	0.00
r104	11	1088.43	SSG	12	1071.89	–	r204	2	779.49	GS	2	780.98	0.19
r105	14	1442.35	GS	15	1383.29	–	r205	3	987.36	GS	3	987.36	0.00
r106	13	1324.10	GS	14	1276.15	–	r206	3	922.19	GS	3	922.70	0.06
r107	12	1150.95	GS	12	**1148.43**	**−0.22**	r207	2	845.26	GS	2	847.14	0.22
r108	11	1050.04	SSG	11	1051.59	0.15	r208	2	736.12	GS	2	736.12	0.00
r109	12	1261.31	GS	13	1214.72	–	r209	3	867.05	GS	3	871.22	0.48
r110	11	1119.50	GS	12	1097.89	–	r210	3	846.20	GS	3	**843.65**	**−0.30**
r111	12	1106.19	SSG	12	1109.14	0.27	r211	2	827.89	GS	3	761.56	–
r112	11	1016.63	GS	11	1038.74	2.13							
rc101	16	1726.91	HPH	16	1731.07	0.24	rc201	4	1444.94	SSG	4	1446.84	0.13
rc102	14	1552.08	HPH	15	1551.69	–	rc202	3	1410.74	GS	3	1450.34	2.73
rc103	13	1350.09	GS	13	1351.73	0.12	rc203	3	1055.19	GS	3	1069.27	1.32
rc104	11	1227.25	GS	11	1232.45	0.42	rc204	3	884.80	GS	3	887.45	0.30
rc105	14	1475.31	HPH	14	**1473.24**	**−0.14**	rc205	3	1273.55	GS	3	1277.60	0.32
rc106	13	1427.21	GS	14	1414.99	–	rc206	3	1188.63	GS	3	1207.64	1.57
rc107	12	1274.89	SSG	12	1283.05	0.64	rc207	3	985.03	GS	3	994.48	0.95
rc108	11	1197.83	GS	11	1209.11	0.93	rc208	3	836.29	GS	3	841.34	0.60
Avg	12.21			12.52		0.60		3.19			3.22		0.33

SSG: Schneider et al. (2014), GS: Goeke and Schneider (2015), HPH: Hiermann et al. (2015), and KÇ: Our method.

instances using two different ALNS implementations and discuss potential gains that can be achieved through different PR strategies.

5.3.1. Numerical results for small-size instances

Schneider et al. (2014) provided the optimal solution for 25 small-size problems and upper bound for the remaining 11 for the FR case using CPLEX 12.2 with a time limit of 7200 s. We solved all these problems with CPLEX 12.6.1 and confirmed the optimality of these upper bounds. Note that ALNS was also able to find the optimal solution for all these instances.

For the PR case, we report the solutions obtained by CPLEX and ALNS in Table 3 in comparison with the optimal solutions achieved with FR. 'FR Optimal' refers to the optimal solution of the FR scheme and '$t(sec)$' is the run time in seconds. The time limit for CPLEX is set to 7200 s. 'Δ^{CPLEX}%' denotes the percentage deviation of the total distance of the ALNS solution from the total distance of the solution found by CPLEX. 'Δ^{FR}%' gives the percentage deviation from the optimal distance of the solution with the FR scheme and shows the benefit of adopting the PR strategy.

Table 3
Comparison of results obtained with CPLEX and ALNS on the small-size instances.

	FR optimal		PR CPLEX			PR ALNS				
Instance	*#Veh*	*TD*	*#Veh*	*TD*	*t(sec)*	*#Veh*	*TD*	Δ^{CPLEX} %	Δ^{FR} %	*t(sec)*
C101-5	2	257.75	2	257.75	0.31	2	257.75	0.00	0.00	0.03
C103-5	1	176.05	1	175.37	2.73	1	175.37	0.00	−0.39	0.05
C206-5	1	242.55	1	242.56	5.38	1	242.56	0.00	0.00	0.07
C208-5	1	158.48	1	158.48	1.37	1	158.48	0.00	0.00	0.06
R104-5	2	136.69	2	136.69	0.47	2	136.69	0.00	0.00	0.04
R105-5	2	156.08	2	156.08	3.39	2	156.08	0.00	0.00	0.04
R202-5	1	128.78	1	128.78	0.95	1	128.78	0.00	0.00	0.08
R203-5	1	179.06	1	179.06	1.12	1	179.06	0.00	0.00	0.10
RC105-5	2	241.30	2	233.77	3.06	2	233.77	0.00	−3.22	0.03
RC108-5	2	253.93	2	253.93	3.76	2	253.93	0.00	0.00	0.04
RC204-5	1	176.39	1	176.39	2.17	1	176.39	0.00	0.00	0.08
RC208-5	1	167.98	1	167.98	1.05	1	167.98	0.00	0.00	0.07
C101-10	3	393.76	3	388.25	50.26	3	388.25	0.00	−1.42	0.10
C104-10	2	273.93	2	273.93	5.15	2	273.93	0.00	0.00	0.17
C202-10	1	304.06	1	304.06	7.52	1	304.06	0.00	0.00	0.20
C205-10	2	228.28	2	228.28	2.01	2	228.28	0.00	0.00	0.16
R102-10	3	249.19	3	249.19	1.83	3	249.19	0.00	0.00	0.11
R103-10	2	207.05	2	206.12	6.76	2	206.12	0.00	−0.45	0.17
R201-10	1	241.51	1	241.51	11.40	1	241.51	0.00	0.00	0.21
R203-10	1	218.21	1	218.21	1.62	1	218.21	0.00	0.00	0.62
RC102-10	4	423.51	4	423.51	3.07	4	423.51	0.00	0.00	0.09
RC108-10	3	345.93	3	345.93	2.90	3	345.93	0.00	0.00	0.09
RC201-10	1	412.86	1	412.86	7200.00	1	412.86	0.00	0.00	0.17
RC205-10	2	325.98	2	325.98	3.26	2	325.98	0.00	0.00	0.19
C103-15	3	384.29	3	348.46	1008.00	3	348.46	0.00	−10.28	0.23
C106-15	3	275.13	3	275.13	0.47	3	275.13	0.00	0.00	0.15
C202-15	2	383.62	2	383.62	24.07	2	383.62	0.00	0.00	0.29
C208-15	2	300.55	2	300.55	0.92	2	300.55	0.00	0.00	0.26
R102-15	5	413.93	5	412.78	7200.00	5	412.78	0.00	−0.28	0.12
R105-15	4	336.15	4	336.15	1.39	4	336.15	0.00	0.00	0.09
R202-15	2	358.00	2	358.00	462.89	2	358.00	0.00	0.00	0.51
R209-15	1	313.24	1	313.24	610.64	1	313.24	0.00	0.00	0.92
RC103-15	4	397.67	4	397.67	20.27	4	397.67	0.00	0.00	0.12
RC108-15	3	370.25	3	370.25	101.45	3	370.25	0.00	0.00	0.15
RC202-15	2	394.39	2	394.39	113.43	2	394.39	0.00	0.00	0.31
RC204-15	1	407.45	1	403.38	7200.00	1	382.22	−5.54	−1.01	1.35
Average					668.47			−0.15	−0.47	0.21

The results show that our ALNS is able to solve all small instances very efficiently. CPLEX found the optimal solution for 33 instances within the time limit and ALNS was able to obtain the optimal solution in all these instances. In RC201-10 and R102-15, we found the same upper bound as CPLEX and in RC204-15 our solution is 5.54% better than the upper bound provided by CPLEX. The average run time of ALNS is only 0.21 s whereas CPLEX spent 668 s on the average. It should be noted that the run time of CPLEX is not the time it takes until it finds the optimal solution or the upper bound but the time it stops after the optimality conditions have been satisfied or the time limit has been reached.

When we compare the solutions that we found with the PR scheme to the optimal solutions of the FR scheme, we see that the number of EVs utilized remained same and the routes were improved only in seven instances. We believe that the potential advantage of the PR strategy is not evident in these instances because their sizes are too restrictive to allow alternate routes. Nevertheless, the savings in total distance may exceed 10% as we observed in C103-15.

5.3.2. Numerical results for large-size instances

In our experimental study with the large instances we also implement a strategy where the partial recharge is allowed at a pre-determined constant amount and test different amounts for comparison purposes. In that case, the implementation of SI algorithms is same but the PR is performed at a constant level. If the battery is not sufficient to complete the route we fully recharge the battery at the last station visited before the infeasibility occurs, if the time windows of the customers permit.

The results are illustrated in Table 4. The results obtained by of the FR scheme are given in the 'FR' column for comparison. The next three main columns report the best solutions found by ALNS using different PR strategies. '*q free*' refers to the case where the PR is performed at any level, i.e. the recharge amount is a decision variable, whereas in '$q = 0.3$', '$q = 0.4$', and '$q = 0.5$' the PR is allowed at the constant levels of 30%, 40%, and 50% of the battery capacity, respectively. Note that in some

Table 4
EVRPTW-PR results for different recharge strategies.

	FR		PR (q free)			PR (q = 0.3)			PR (q = 0.4)			PR (q = 0.5)		
Inst.	*#Veh*	*TD*	*#Veh*	*TD*	*Δ%*	*#Veh*	*TD*	*Δ%*	*#Veh*	*TD*	*Δ%*	*#Veh*	*TD*	*Δ%*
c101	12	1053.83	12	1051.23	−0.25	12	1043.38	−1.00	12	1045.98	−0.75	12	1053.82	0.00
c102	11	1056.12	11	1034.24	−2.12	11	1034.48	−2.09	11	1032.49	−2.29	11	1040.34	−1.52
c103	11	1001.81	**10**	973.39	–	**10**	981.16	–	**10**	1028.32	–	**10**	1030.42	–
c104	10	951.57	10	886.72	−7.31	10	891.05	−6.79	10	892.84	−6.58	10	894.03	−6.44
c105	11	1075.37	11	1037.78	−3.62	11	1051.98	−2.22	11	1051.98	−2.22	11	1054.36	−1.99
c106	11	1057.65	11	1024.18	−3.27	11	1030.41	−2.64	11	1036.88	−2.00	11	1046.57	−1.06
c107	11	1031.56	**10**	1058.11	–	11	1014.45	−1.69	11	1013.63	−1.77	11	1013.63	−1.77
c108	11	1015.68	**10**	1033.50	–	11	999.55	−1.61	11	1002.54	−1.31	11	1000.62	−1.51
c109	10	1069.16	10	960.03	−11.37	10	990.53	−7.94	10	947.20	−12.88	10	946.84	−12.92
c201	4	645.16	4	629.95	−2.41	4	642.84	−0.36	4	642.84	−0.36	4	641.13	−0.63
c202	4	645.16	4	629.95	−2.41	4	641.07	−0.64	4	641.07	−0.64	4	641.07	−0.64
c203	4	644.98	4	629.95	−2.39	4	641.07	−0.61	4	641.07	−0.61	4	641.07	−0.61
c204	4	636.43	4	629.95	−1.03	4	635.80	−0.10	4	636.43	0.00	4	636.43	0.00
c205	4	641.13	4	629.95	−1.77	4	638.17	−0.46	4	631.26	−1.56	4	638.17	−0.46
c206	4	638.17	4	629.95	−1.30	4	641.13	0.46	4	631.26	−1.09	4	638.17	0.00
c207	4	638.17	4	629.95	−1.30	4	641.13	0.46	4	638.17	0.00	4	638.17	0.00
c208	4	638.17	4	629.95	−1.30	4	634.19	−0.63	4	631.26	−1.09	4	638.17	0.00
r101	18	1679.06	18	1661.33	−1.07	18	1636.69	−2.59	18	1651.80	−1.65	18	1679.27	0.01
r102	16	1519.80	16	1461.48	−3.99	16	1466.58	−3.63	16	1461.38	−4.00	16	1467.92	−3.53
r103	13	1312.50	13	1262.75	−3.94	14	1254.22	–	13	1265.90	−3.68	13	1276.80	−2.80
r104	12	1071.89	**11**	1078.99	–	**11**	1131.39	–	12	1065.14	−0.63	12	1065.17	−0.63
r105	15	1383.29	15	1373.94	−0.68	15	1380.62	−0.19	15	1380.44	−0.21	15	1396.79	0.97
r106	14	1276.15	**13**	1310.46	–	14	1288.90	0.99	14	1295.60	1.50	14	1284.19	0.63
r107	12	1148.43	12	1118.91	−2.64	12	1132.35	−1.42	12	1131.01	−1.54	12	1126.42	−1.95
r108	11	1051.59	11	1031.14	−1.98	11	1045.97	−0.54	11	1044.82	−0.65	11	1048.49	−0.30
r109	13	1214.72	13	1201.04	−1.14	13	1193.76	−1.76	13	1209.30	−0.45	13	1194.80	−1.67
r110	12	1097.89	**11**	1112.80	–	**11**	1090.92	–	12	1095.98	−0.17	12	1093.73	−0.38
r111	12	1109.14	12	1084.13	−2.31	12	1102.07	−0.64	12	1096.22	−1.18	12	1101.27	−0.71
r112	11	1038.74	11	1017.31	−2.11	11	1035.16	−0.35	11	1017.52	−2.09	11	1037.90	−0.08
r201	3	1265.67	3	1266.06[a]	0.03	3	1266.54	0.07	3	1274.00	0.65	3	1262.10	−0.28
r202	3	1052.32	3	1052.32	0.00	3	1054.70	0.23	3	1053.57	0.12	3	1055.48	0.30
r203	3	895.54	3	895.54	0.00	3	896.71	0.13	3	895.83	0.03	3	895.70	0.02
r204	2	780.98	2	780.14	−0.11	3	720.15	–	3	723.08	–	2	801.29	2.53
r205	3	987.36	3	987.36	0.00	3	989.03	0.17	3	995.03	0.77	3	991.52	0.42
r206	3	922.70	3	922.70	0.00	3	925.40	0.29	3	934.88	1.30	3	925.67	0.32
r207	2	847.14	2	846.59	−0.06	2	853.12	0.70	3	811.63	–	2	849.49	0.28
r208	2	736.12	2	736.12	0.00	2	736.75	0.09	2	736.75	0.09	2	737.40	0.17
r209	3	871.22	3	868.95	−0.26	3	871.31	0.01	3	872.03	0.09	3	871.55	0.04
r210	3	843.65	3	843.36	−0.03	3	850.84	0.84	3	849.15	0.65	3	850.16	0.77
r211	3	761.56	**2**	862.56	–	3	761.56	0.00	3	766.56	0.65	3	766.56	0.65
rc101	16	1731.07	16	1684.84	−2.74	**15**	1743.90	–	16	1689.40	−2.47	16	1704.19	−1.58
rc102	15	1551.69	**14**	1155.50	–	**14**	1566.40	–	**14**	1555.90	–	**14**	1558.51	–
rc103	13	1351.73	13	1329.58	−1.67	13	1351.51	−0.02	13	1342.49	−0.69	13	1345.04	−0.50
rc104	11	1232.45	11	1202.93	−2.45	11	1226.33	−0.50	11	1229.38	−0.25	11	1223.79	−0.71
rc105	14	1473.24	14	1458.49	−1.01	14	1458.28	−1.03	14	1467.44	−0.40	14	1470.40	−0.19
rc106	14	1414.99	**13**	1422.96	–	**13**	1440.45	–	**13**	1455.21	–	**13**	1417.40	–
rc107	12	1283.05	12	1261.03	−1.75	12	1272.09	−0.86	12	1265.16	−1.41	12	1261.44	−1.71
rc108	11	1209.11	11	1185.68	−1.98	11	1184.06	−2.12	11	1235.85	2.16	11	1209.64	0.04
rc201	4	1446.84	4	1446.84	0.00	4	1454.11	0.50	4	1463.70	1.15	4	1456.02	0.63
rc202	3	1450.34	3	1416.96	−2.36	4	1244.76	–	4	1245.10	–	4	1242.59	–
rc203	3	1069.27	3	1069.27	0.00	3	1089.74	1.88	3	1092.22	2.10	3	1084.06	1.36
rc204	3	887.45	3	887.76	0.04	3	887.29	−0.02	3	892.25	0.54	3	886.23	−0.14
rc205	3	1277.60	3	1262.22	−1.22	4	1162.16	–	4	1145.34	–	4	1151.93	–
rc206	3	1207.64	3	1213.89	0.51	3	1206.09	−0.13	3	1208.36	0.06	3	1209.42	0.15
rc207	3	994.48	3	993.49	−0.10	3	996.35	0.19	3	992.14	−0.24	3	993.26	−0.12
rc208	3	841.34	3	839.71	−0.19	3	847.82	0.76	3	843.98	0.31	3	844.76	0.40
Avg					−1.64			−0.80			−0.91			−0.73

[a] 1258.39 was also observed during our tests.

instances, the results obtained using the PR scheme are worse than the best results found with the FR scheme. For consistency, we preferred to give them as reported by ALNS even though the solutions to EVRPTW are also feasible to EVRPTW-PR.

These results clearly show the advantage of using the PR strategy over the FR restriction: when PR quantity is not fixed (q free) the average total distance reduces by 1.64% in the instances for which the numbers of vehicles used with FR and PR schemes are same. On the other hand, the reductions are 0.80%, 0.91%, and 0.73% if PR is performed at the level of $q = 0.3$, $q = 0.4$, and $q = 0.5$, respectively. In addition, we observe that solutions with one less vehicle are found in nine instances when q is free while the same situation occurs in six instances for $q = 0.3$ and in three instances for $q = 0.4$ and $q = 0.5$. The improved numbers are shown in bold. Moreover, we see PR has more potential for saving from the number of vehicles in type-1 problems, which is an expected outcome since those problems are more restrictive due to narrow time-windows and shorter route durations. Due to the same fact, we also observe that the average distances improve significantly in type-1 instances, c1 problem set in particular. When q is free the average improvement in total distance is 2.89% in type-1 problems whereas the average improvement in type-2 problems is only 0.66%.

Not surprisingly ALNS using the variable PR amount has a better performance compared to ALNS with constant PR. On the other hand, its average computation time is 9.7% longer compared to the case when $q = 0.3$ (16.77 min vs. 15.29 min). The difference in run time is particularly significant in r2 and rc2 problems where ALNS with variable PR takes 25.1% and 10.3% more time, respectively. Since the time-windows are wider and routes are longer in these problem types, ALNS needs more time to evaluate the insertions and determine the recharge quantity. Nevertheless, the additional time brings substantial improvements both in the number of vehicles and total distance traveled. Overall, these results suggest the PR scheme is effective, particularly in cases where the time windows are more restrictive.

6. Conclusions and future research

In this paper, we investigated the partial recharge strategies for the EVRPTW, namely EVRPTW-PR, and proposed an ALNS algorithm to solve it. Some of the existing ALNS mechanisms were adopted from the literature whereas new removal and insertion mechanisms specific to EVRPTW were developed to handle the visits to recharging stations and to incorporate the PR decisions.

We used the instances generated by Schneider et al. (2014) to validate the performance of the proposed ALNS. We first solved the EVRPTW instances and benchmarked our results with those of reported in Schneider et al. (2014), Goeke and Schneider (2015), and Hiermann et al. (2015). We also reported 4 new best-known solutions. For the proposed EVRPTW-PR we solved the same instances. The results revealed that the routes can be significantly improved when PR is allowed, even if at a pre-determined constant level.

In this study, we only allowed PR using the same power level. The problem can be extended to a multiple recharge power options at different speeds and costs as discussed in Felipe et al. (2014). Further research on this topic may also address the heterogeneous fleet case. The heterogeneity within this context does not only arise from the vehicle capacities but from their batteries as well since the cruising range of EVs and discharge/recharge durations differ depending on their battery condition and age. Furthermore, the travel times may vary due to traffic conditions, accidents, construction, etc., which may have a significant impact on the routing decisions due to limited driving range of the EVs. In addition, we assume that recharging stations are always available, which may not be true in real life and there may be queues in the stations. So, variability in both travel and recharging times arises as an interesting and challenging topic to be investigated within the stochastic context.

Acknowledgments

The authors would like to thank the guest editors and the two anonymous reviewers for their constructive and valuable comments. This research was partially supported by The Scientific and Technical Research Council of Turkey through the BIDEB 2210 Program graduate scholarship to the first author and Grant #113M522 to the second author.

Appendix A. Parameter tuning

All parameters are summarized in Table A.1 and the results of the parameter tuning procedure for solving the EVRPTW-PR are given in Table A.2. The tuning sequence is from top to bottom. All parameters were initially set to the values given in the "Initial Value" column. We first considered ten different values for σ_2 shown in the first row and performed ten runs on the six instances selected for tuning. Next, we calculated the average percentage deviation 'Dev%' of the average solution achieved with that value from the best solution found in all runs. Then, we determined 0.22% as the least deviation and fixed the value of the parameter σ_2 to 20, which had achieved this best average performance. We repeated this procedure for the remaining parameters in the order given until all parameter values had been tuned. The best parameter values (the ones yielding the minimum deviations) are indicated in bold. Note that the initial values in Table A.2 are the values we determined when we applied the parameter tuning for solving EVRPTW. We do not give the detailed setting for that case since the tuning approach and the values considered are same.

Table A.1
Notation and description of the parameters.

σ_2	Score of the better solution
N_C	# of iterations between which adaptive weights of CR and CI algorithms are updated
ρ	Roulette wheel parameter
σ_1	Score of the best solution
σ_3	Score of the worse solution
ϕ_1	First Shaw parameter
ϕ_2	Second Shaw parameter
ϕ_3	Third Shaw parameter
ϕ_4	Fourth Shaw parameter
ε	Cooling rate of SA
μ	Initial temperature control parameter of SA
κ	Worst removal determinism factor
η	Shaw removal determinism factor
n_Z	Number of zones in zone removal
N_{SR}	# of iterations between which SR is performed
N_S	# of iterations between which adaptive weights of SR and SI algorithms are updated
m_r	Route removal upper bound
N_{RR}	# of iterations between which route removal algorithms are performed
n_{RR}	# of consecutive iterations during which route removal algorithms are performed

Table A.2
Parameter tuning.

Parameter		Initial value	Values tested								
σ_2	Value	6	0	2	4	9	12	14	16	18	**20**
	Dev%	0.30	0.27	0.25	0.28	0.27	0.28	0.25	0.25	0.28	**0.22**
N_C	Value	500	50	100	150	**200**	250	300	350	400	450
	Dev%	0.28	0.33	0.27	0.25	**0.23**	0.32	0.30	0.27	0.25	0.32
ρ	Value	0.2	0.05	0.1	0.15	**0.25**	0.3	0.35	0.4	0.45	0.5
	Dev%	0.30	0.33	0.25	0.30	**0.25**	0.27	0.32	0.27	0.25	0.30
σ_1	Value	33	**25**	30	35	40	45	50			
	Dev%	0.32	**0.25**	0.32	0.32	0.37	0.30	0.28			
σ_3	Value	21	3	6	9	12	13	15	24	27	30
	Dev%	0.37	0.42	0.40	0.45	0.47	0.42	0.45	0.43	0.47	0.48
ϕ_1	Value	5	**0.5**	1	3	7	9	11	13	15	
	Dev%	0.55	**0.55**	0.60	0.55	0.58	0.63	0.57	0.65	0.63	
ϕ_2	Value	1	0.25	3	5	7	9	11	**13**	15	
	Dev%	0.57	0.62	0.57	0.62	0.63	0.60	0.62	**0.55**	0.67	
ϕ_3	Value	13	**0.15**	1	3	5	7	9	11	15	
	Dev%	0.60	**0.53**	0.58	0.53	0.57	0.55	0.53	0.62	0.53	
ϕ_4	Value	0.25	1	2	3	4	5	6	7	8	9
	Dev%	0.57	0.62	0.58	0.62	0.68	0.62	0.63	0.58	0.67	0.62
ε	Value	0.9995	0.999	0.9991	0.9992	0.9993	**0.9994**	0.9996	0.9997	0.9998	0.9999
	Dev%	0.48	0.57	0.48	0.43	0.53	**0.38**	0.42	0.48	0.42	0.42
μ	Value	0.05	0.1	0.15	0.2	0.25	0.3	0.35	**0.4**	0.45	0.5
	Dev%	0.55	0.60	0.58	0.58	0.65	0.58	0.58	**0.55**	0.58	0.62
κ	Value	1	2	3	**4**	5	6				
	Dev%	0.57	0.63	0.58	**0.55**	0.60	0.57				
η	Value	10	2	4	6	8	**12**				
	Dev%	0.58	0.55	0.60	0.62	0.63	**0.55**				
n_Z	Value	15	5	7	9	11	13	19	21	**25**	30
	Dev%	0.60	0.60	0.65	0.55	0.58	0.63	0.60	0.57	**0.53**	0.58
N_{SR}	Value	10	20	30	40	50	**60**	70	80	90	100
	Dev%	0.75	0.78	0.82	0.80	0.80	**0.72**	0.82	0.73	0.77	0.77
N_S	Value	1000	1500	2000	2500	3000	3500	4000	4500	5000	**5500**
	Dev%	0.70	0.78	0.75	0.77	0.77	0.83	0.73	0.80	0.77	**0.70**
m_r	Value	0.4	**0.3**	0.5	0.6						
	Dev%	0.58	**0.57**	0.67	0.65						
N_{RR}	Value	6000	**2000**	2500	3000	3500	4000	4500	5000	5500	6500
	Dev%	0.85	**0.77**	0.87	0.78	0.77	0.85	0.80	0.78	0.80	0.88
n_{RR}	Value	1000	750	**1250**	1500	1750	2000	2250	2500	2750	3000
	Dev%	0.43	0.43	**0.40**	0.45	0.53	0.47	0.48	0.48	0.43	0.48

References

Adulyasak, Y., Cordeau, J.-F., Jans, R., 2014. Optimization-based adaptive large neighborhood search for the production routing problem. Transport. Sci. 48 (1), 20–45.
Aksen, D., Kaya, O., Salman, S., Tüncel, Ö., 2014. An adaptive large neighborhood search algorithm for a selective and periodic inventory routing problem. Eur. J. Oper. Res. 239 (2), 413–426.

Artmeier, A., Haselmayr, J., Leucker, M., Sachenbacher, M., 2010. The optimal routing problem in the context of battery-powered electric vehicles. In: 2nd International Workshop on Constraint Reasoning and Optimization for Computational Sustainability, Bologna, Italy.
Azi, N., Gendreau, M., Potvin, J.-Y., 2014. An adaptive large neighborhood search for a vehicle routing problem with multiple routes. Comput. Oper. Res. 41, 167–173.
Conrad, R.G., Figliozzi, M.A., 2011. The recharging vehicle routing problem. In: Doolen, T., Van Aken, E. (Eds.), Proceedings of the 2011 Industrial Engineering Research Conference.
Demir, E., Bektaş, T., Laporte, G., 2012. An adaptive large neighborhood search heuristic for the pollution-routing problem. Eur. J. Oper. Res. 223 (2), 346–359.
Desaulniers, G., Errico, F., Irnich, S., Schneider, M., 2014. Exact Algorithms for Electric Vehicle Routing Problems with Time Windows. Working Paper.
Emeç, U., Çatay, B., Bozkaya, B., 2016. An adaptive large neighborhood search for an e-grocery delivery routing problem. Comput. Oper. Res. 69, 109–125.
Erdogan, S., Miller-Hooks, E., 2012. A green vehicle routing problem. Transp. Res. Part E 48, 100–114.
Felipe, Á., Ortuño, M.T., Righini, G., Tirado, G., 2014. A heuristic approach for the green vehicle routing problem with multiple technologies and partial recharges. Transp. Res. Part E 71, 111–128.
Goeke, D., Schneider, M., 2015. Routing a mixed fleet of electric and conventional vehicles. Eur. J. Oper. Res. 245 (1), 81–99.
Hemmelmayr, V.C., Cordeau, J.-F., Crainic, T.G., 2012. An adaptive large neighborhood search heuristic for two-echelon vehicle routing problems arising in city logistics. Comput. Oper. Res. 39 (12), 3215–3228.
Hiermann, G., Puchinger, J., Hartl, R.F., 2015. The Electric Fleet Size and Mix Vehicle Routing Problem with Time Windows and Recharging Stations. Working Paper.
Masson, R., Lehuédé, F., Péton, O., 2013. An adaptive large neighborhood search for the pickup and delivery problem with transfers. Transport. Sci. 47 (3), 344–355.
Omidvar, A., Tavakkoli-Moghaddam, R., 2012. Sustainable vehicle routing: Strategies for congestion management and refueling scheduling. In: Proceedings of the IEEE International Energy Conference and Exhibition, Florence, Italy, pp. 1089–1094.
Pisinger, D., Ropke, S., 2007. A general heuristic for vehicle routing problems. Comput. Oper. Res. 34 (8), 2403–2435.
Qu, Y., Bard, J.F., 2012. A GRASP with adaptive large neighborhood search for pickup and delivery problems with transshipment. Comput. Oper. Res. 39 (10), 2439–2456.
Ribeiro, G.M., Laporte, G., 2012. An adaptive large neighborhood search heuristic for the cumulative capacitated vehicle routing problem. Comput. Oper. Res. 39 (3), 728–735.
Ropke, S., Pisinger, D., 2006a. An adaptive large neighborhood search heuristic for the pickup and delivery problem with time windows. Transport. Sci. 40 (4), 455–472.
Ropke, S., Pisinger, D., 2006b. A unified heuristic for a large class of vehicle routing problems with backhauls. Eur. J. Oper. Res. 171 (3), 750–775.
Schneider, M., Stenger, A., Goeke, D., 2014. The electric vehicle routing problem with time windows and recharging stations. Transport. Sci. 48 (4), 500–520.
Shaw, P., 1998. Using constraint programming and local search methods to solve vehicle routing problems. In: Proceedings of the 4th International Conference on Principles and Practice of Constraint Programming, Springer, New York, pp. 417–431.
Solomon, M.M., 1987. Algorithms for the vehicle routing and scheduling problems with time window constraints. Oper. Res. 35 (2), 254–265.
Touati-Moungla, N., Jost, V., 2012. Combinatorial optimization for electric vehicles management. J. Energy Power Eng. 6 (5), 738–743.
Wang, H., Shen, J., 2007. Heuristic approaches for solving transit vehicle scheduling problem with route and fueling time constraints. Appl. Math. Comput. 190, 1237–1249.
Wang, H., Cheu, R.L., 2012. Operations of a taxi fleet for advance reservations using electric vehicles and charging stations. J. Transport. Res. Board 2352, 1–10.
Worley, O., Klabjan, D., 2012. Simultaneous vehicle routing and charging station siting for commercial electric vehicles. In: IEEE International Electric Vehicle Conference, Greenville, SC, pp. 1–3.