An Adaptive Large Neighborhood Search for Relocating Vehicles in Electric Carsharing Services

Maurizio Bruglieri*

Dipartimento di Design, Politecnico di Milano

Ferdinando Pezzella

Dipartimento di Ingegneria dell'Informazione, Universitá Politecnica delle Marche

Ornella Pisacane

Dipartimento di Ingegneria dell'Informazione, Universitá Politecnica delle Marche

Abstract

We propose an Adaptive Large Neighborhood Search metaheuristic to solve a vehicle relocation problem arising in the management of electric carsharing systems. The solution approach, aimed to optimize the total profit, is tested on three real-like benchmark sets of instances. It is compared with a Tabu Search, ad hoc designed for this work, with a previous Ruin and Recreate metaheuristic and with the optimal results obtained via Mixed Integer Linear Programming. We also develop a bounding procedure to evaluate the solution quality when the optimal solution is not available.

Keywords: one-way carsharing, operator-based relocation, profit optimization, pick-up and delivery problem, Ruin and Recreate metaheuristic, Tabu Search

1. Introduction

The carsharing systems allow users sharing vehicles, available on-demand, by paying a charge that usually depends on the real time of use (also for a fraction of hour). In this way, the customers can use a vehicle without owning it, according

*Corresponding author
Email address: maurizio.bruglieri@polimi.it (Maurizio Bruglieri)

to the principle of *pay-as-you-drive*, and the fixed costs (e.g., maintenance, taxes, assurance), due to the ownership of a vehicle, are zeroed.

Moreover, such systems contribute also to reduce the traffic congestion and incentive the occasional use of vehicles. This favors also a reduction of the harmful emissions mainly due to the use of private cars, consistent with the recent rules imposed by the European Committee aiming to reduce the emissions, from 130 g/km of the 2015 to 95 g/km for the 2020.

However, the efficient management of carsharing systems poses new strategic, tactical and operational decisions [8]. Tactical decisions mainly concern the fleet dimensioning, i.e., to determine the appropriate number of vehicles to share. Among the strategic decisions, both locating and sizing the stations (i.e., determining the number of their parking lots) play a key role in the context of the station-based systems. Finally, among the operational decisions, the pricing and the vehicle relocation are both of interest.

This paper focuses on *one-way* carsharing systems, in which it is allowed users to pick-up a vehicle from a station and to delivery it to a different one. The flexibility of this type of carsharing systems is paid in terms of possible unbalance between the demand and the supply of vehicles at stations. For this reason, implementing efficient relocation policies of vehicles among the stations becomes crucial in order to guarantee that they are available at *the right station, at the right time*.

In this work, the Electric Vehicle Relocation Problem (E-VReP) is addressed. The vehicles shared among the users are electric. The workers relocate the EVs moving from a station of delivery to one of pick-up by folding bikes and vice versa, from a station of pick-up to one of delivery by directly driving the EVs. In the latter case, they put their folding bicycles in the trunk of the EV. A solution to the E-VReP therefore provides the *routing* and the *scheduling* of each worker used and it has to be feasible with regard both the battery levels and the time windows constraints required by the users. The aim is to maximize the total profit of the system, given by the difference between the total revenue, due to the relocation requests satisfied, and the total cost, due to the workers used. This

problem is shown to be APX-hard in [14]. First, the authors demonstrate that it is strongly NP-hard by reduction from the Orienteering Problem (OP). Indeed, they show that each instance of the OP can be transformed in polynomial time into an instance of the E-VReP such that from its optimal solution it is possible to obtain the optimal solution of the OP instance. In this way, the optimal solution of the E-VReP yields in polynomial time the optimal solution of the OP. Moreover, since the reduction from the OP is gap-preserving (because the E-VReP optimal value for the transformed instance coincides with the OP optimal value for the original instance), the E-VReP cannot be approximated to within a factor of 1481/1480 , unless P = NP, since this property holds for the OP (see Theorem 7.2 of [7]). For this reason, we propose an *Adaptive Large Neighborhood Search* (ALNS) metaheuristic.

The main contributions of this paper are: the development of a metaheuristic solution approach (ALNS) able to solve in few seconds also instances of large size (i.e., order of one hundred relocation requests); the definition of several insertion and removal heuristics (necessary to the ALNS) tailored for the E-VReP; a careful parameter tuning of the ALNS to increase its effectiveness; an experimental analysis of the contribution of each insertion and removal heuristic (used in the ALNS) in determining its effectiveness; the design and testing of a Tabu Search (TS) that uses the same moves of the ALNS, in order to compare two metaheuristics; the development of bounding procedures to evaluate the metaheuristic solution quality when the optimal solution cannot be detected.

The paper is organized as in the following. In Section 2, we give an overview of the scientific literature, focusing on the contributions related to the vehicle relocation or to the ALNS applied to similar Vehicle Routing Problems. In Section 3, we state the problem and its assumptions while in Section 4 we describe all the components of the ALNS. In section 5, we describe a TS metaheuristic properly defined for the E-VReP. In Section 6, we present two bounding methods based on two different relaxations of a Mixed Integer Linear Programming (MILP) formulation of the E-VReP and a third tighter bound based on a combination of them. In Section 7, we discuss the computational experiments carried

out on three sets of benchmark instances. In particular, the results obtained by the ALNS are compared with those of both the TS and a previous Ruin and Recreate (R&R) metaheuristic. Moreover, they are also compared with the optimal results obtained through both the MILP formulation (when available) and the bounding procedures. Finally, Section 8 draws some conclusions.

2. Literature Review

The carsharing systems were introduced in the years 70'-80' ([39]) but it is only in the last years that they have seen a significant diffusion, especially in urban areas. Several factors can affect the performance of a carsharing system, from the strategic, the tactical and the operational perspective [24, 10]. Among them, particular attention is focused on the way of relocating the vehicles in order to assure a balance between the supply and the demand of vehicles at the stations [32]. According to the operating model implemented, the carsharing system can be [24]:*one-way*, in which the station where a vehicle is delivered has not to be necessarily the same of the one where it has been picked up; *round-trip*, where the users have to deliver the vehicles at the same stations where they have been picked up; *free-floating*, where the users can deliver the vehicles at any parking space of the city.

This work focuses on the one-way carsharing systems where the vehicle relocation among the stations becomes more crucial. The vehicle relocation can be usually performed [5]: based on the immediate needs (*static relocation*); forecasting the future demand of vehicles at the stations (*historical relocation*); using the real knowledge of the demand, on the basis of the users' reservations (*exact predictive relocation*). Moreover, the workers of the carsharing system can relocate the vehicles (*operator-based relocation*), by directly driving them from a point of pick-up to one of delivery. Alternatively, the vehicle relocation can be done through the users of the system (*user-based relocation*) who are incentivized to leave the vehicle at a specific station that has an immediate need [6]. In the recent work of [16], a decision support system is proposed for solv-

ing the user-based vehicle relocation problem by applying economic incentives ruled by a threshold policy and taking into account the stochastic reactions of the customers. While, in [18] , the discrete event systems are used for easily representing the dynamics of the carsharing system. The user-based relocation is addressed through an optimal relocation policy in a rolling horizon framework.

The operator-based relocation is addressed from the system profit maximization point of view, in [25]; jointly to the staff-rebalance, in [33]; for minimizing the total waiting time, in [21]; for minimizing the total cost due to the workers used, the rejection of the requests and the maintenance of the requests not fulfilled in [37]. A multi-objective version has been studied in [8] with the aim of taking into account the optimization of different point of views, e.g., maximizing the net revenue of the system providers and the users benefit. Recently, in [9], a simulation-optimization based framework has been designed to determine a trade-off among the vehicle cost, the personnel cost and the level of service (offered to the users) considering both the vehicle and personnel relocation.

Very recently, a Tabu Search has been developed to solve a multiobjective optimization problem arising in one-way electric car-sharing systems [1]. In particular, the number of vehicles used, the number of workers employed, the total distances traveled by them during the relocation operations and the users unsatisfied demand are minimized, at the same time. Numerical results on simulated instances (based on Geographical Information System QGIS) demonstrate that their Tabu Search approach is able to find near-optimal solutions in a reasonable computational time.

In the operator-based relocation, the workers usually use trucks to move the vehicles. However, this relocation policy not only may cause an increment of the traffic congestion but also poses some critical issues, e.g., stations may not be easily reachable by trucks. For this reason, in [12], the idea that the workers can directly move the vehicles from pick-up stations to one of delivery driving them and they can go from a delivery station to one of pick-up thanks to a folding bicycle, has been introduced. The authors model, through MILP, the problem of routing and scheduling the workers to maximize the total number of

requests served. Moreover, the additional assumption that the shared resources are Electric Vehicles (EVs) complicates the relocation problem since their limited driving range has to be properly taken into account. For example, the EVs have to be recharged at stations beforehand to be used; their battery level has to be the one required by the user at a station and so on. A complete and very recent review on this topic can be found in [10]. Under the same assumptions, [13] designs a simulator, able to generate real-like instances of the problem, using some data from the Milan transport agency AMAT and the main energy supplier company in Milan (A2A). The relocation in electric carsharing systems has been studied from the methodological perspective also in [28], [34] and [15]) and it shares some features with the Pick-up and Delivery Problem (PDP) [23]. More specifically, due to the alternation of the pickup and the delivery requests in a route, the E-VReP shares features with both the 1-skip collection problem ([2]) and the Roll-on Roll-off problem ([3], [4]).

Recently, [14] investigates the economic sustainability of the E-VReP. In fact, a revenue is associated with each relocation request satisfied while a cost is due to the use of the workers. Therefore, the E-VReP is addressed with the aim of maximizing the total profit. Since such a new variant is shown to be APX-hard, the authors propose a R&R metaheuristic that starts from a solution found by a Nearest Neighborhood Search and exploits four different types of moves.

The aim of this work is to propose an Adaptive Large Neighborhood Search (ALNS) to solve the E-VReP since it proved to be very effective for other challenging problems, e.g., the PDP [36], the very large scale Vehicle Routing Problem (VRP) [26], the capacitated arc-routing problem with stochastic demands [27], the cumulative capacitated VRP [35], the Pollution Routing Problem [17]. An ALNS has been recently designed and developed for the share-a-ride problem in [29], succeeding to perform very well from both the solution quality point of view and the computational time effort.

3. Problem statement and assumptions

In this section, we introduce the statement, the notation and the assumptions of the E-VReP and we illustrate the problem also with a detailed example.

A one-way carsharing system made up of a homogeneous fleet of EVs is considered. Each fully charged EV can cover a maximum distance L that depends on the kind of EV considered. When the EV is not fully charged, L is assumed to be linearly proportional to the residual charge of the battery. This is an exemplification usually done in the literature ([12]) in order not to take into account also the slope of the path. For instance, an EV with a residual charge of 50% can cover a distance equal to $L/2$ km. Moreover, the time, Γ, for fully recharging an exhausted battery is known. Although the behavior of the charging time function depends on the battery technology used [31], for the sake of simplicity it is assumed to be linear, as done also in several papers, e.g., in [22], [38] and [41]. We assume that each EV is always picked up and returned to a parking station equipped with a charging dock so it can be recharged when it is not in use. Since in a one-way carsharing service, the cars can be returned to parking stations which are different from the pick-up point, some vehicles need to be moved in order to prevent a station from running out of either EVs or parking lots.

Let D be the set of delivery requests (i.e., the requests for delivering EVs in order to prevent a station from running out of EVs) and let P be the set of pick-up requests (i.e., requests for EVs that need to be moved to vacant parking lots). Each relocation request $r \in R = P \cup D$ is characterized by a revenue rev_r, a parking location v_r, i.e., a node of the road network, by the residual charge of the battery ρ_r (measured as a fraction of the total battery capacity) and by a time window $[\tau_r^{min}, \tau_r^{max}]$ where τ_r^{min} and τ_r^{max} represent the earliest time and the latest time allowed to carry out the request r, respectively. If $r \in D$, ρ_r represents the minimum charge level that the EV battery must have at time τ_r^{max}. Consequently, an EV delivered before τ_r^{max} may also have a battery charge level less than ρ_r. Indeed, in order to satisfy the battery constraint, it is

only necessary that the charge level ρ_r is reached within τ_r^{max}, considering that the EV is immediately recharged after delivery. Whereas, if $r \in P$, ρ_r indicates the available battery charge level of the EV at the time τ_r^{min}. Finally, the length of the shortest path from the parking location of the request r and the one of the request r' is denoted by $l_{rr'}$.

Since the fleet of EVs is homogeneous each delivery request can be satisfied by picking up each EV of a pick-up request provided that it is compatible with the time windows and the battery charge level. Given a team of K workers, each one with cost C, who leave a single depot (denoted by 0), even at different times, using folding bicycles, we aim to determine their routes and schedules so as to maximize the total profit given by the difference between the total revenue, i.e., the sum of the revenues of all the relocation requests satisfied, and the total cost, obtained by multiplying by C the number of the workers used. Each route starts and ends from/to the depot and consists of an alternating sequence of pick-up and delivery requests, such that their time windows are respected, the battery charge level of each pick-up request is sufficient to reach the parking station of the next delivery request, the battery charge levels required at the delivery requests are satisfied and the route duration does not exceed a given threshold T (i.e. the shifts of the workers). It is worth remarking that the problem models the trade-off between the point of view of the manager and of the users of the carsharing system, without including the worker perspective. In fact, the workers are subject to contracts that usually limit the maximum duration of a shift and not the number of relocation requests they have to manage.

Moreover, it is assumed that each worker directly drives an EV from a station to another in order to satisfy a delivery request. While, a worker moves by a folding bicycle from a station to another for satisfying a pick-up request. The average speeds of an EV and of a bicycle are denoted by s' and s'', respectively. In addition, the average time for parking an EV and taking the bike out of the EV trunk is denoted by q' while, the average time for unloading the bike from the EV trunk and leaving the parking lot with the EV is indicated by q''.

Figure 1 shows a feasible solution for an instance of the problem. In this instance, each parking station is assumed to be distant 5 km from the adjacent ones; also the depot is supposed to be 5 km far from every parking station. Moreover we assume $K = 2$, $C = 60$ €, $rev_r = 35$ € $\forall r \in R$, $L = 150$ km, $\Gamma = 4$ hours, $s' = 25$ km/h, $s'' = 15$ km/h and $q' = q'' = 1$ minute. In this way worker W_1 can leave from the depot at 8.10 arriving by bike in P_1 at 8.30; leaves from P_1 at 8.31 driving the EV and arrives at D_5 at 8.43 where the EV is delivered at 9 (left hand side TW of D_5); then arrives by bike at P_4 at 9.21 and driving the EV of P_4 arrives at D_3 at 9.33, where the EV is delivered at 10 (earliest time window of D_3) and then arrives at the depot by bike at 10.21. In this way, the total time spent by worker W_1 is 2 hours and 11 minutes and therefore it respects the work shift $T = 5$ hours. While worker W_2 can leave from the depot at 14.10 arriving by bike at P_2 at 14.30, leaves from P_2 at 14.31 driving the EV and arrives at D_1 at 14.43 where the EV is immediately delivered. In this way W_2 arrives by bike to the depot at 15.04, spending in total 54 minutes. We notice that sometimes the battery charge level of the picked-up EV is lower than the value requested for the delivered EV. For instance, in the route performed by W_2, the available battery charge level of the EV at P_2 at time 14.30 is 0.4 while, its minimum charge level at D_1 at time 17.00 must be 0.9. Nevertheless, the solution is feasible since the EV is delivered in D_1 at 14.43 with a battery level of 0.37 (since the battery consumption to go from P_2 to D_1 is 0.03) but at the latest time window of D_1, i.e. 17, the EV reaches a battery level of $0.94 > 0.9$ since in the 137 minutes, from 14.43 to 17, the battery level increases of $137/240 = 0.32$, since the EV is always connected to the docking station. This solution has a total profit of 90 €.

4. Adaptive Large Neighborhood Search for the E-VReP

ALNS is an extension of Large Neighborhood Search (LNS) originally proposed by [40]. The basic idea of LNS is to search in large neighborhoods, which may contain more and potentially better solutions compared to small neigh-

Figure 1: An instance of the E-VReP with four pick-up requests (P_i for i = 1,...,4) and six delivery requests (D_i for i = 1,...,6). For each request, the battery level is indicated on the left of the node, while the time window is shown on the right. Two workers, W_1, W_2, leaving a common depot (the square node) are available, with work shift $T = 5$ hours. The dashed arcs denote that a worker is moving by bicycle (the travel time of these arcs is 20 minutes) while the solid arcs indicate that a worker is driving an EV (their travel time is 12 minutes).

borhoods. The neighborhood of a solution is defined implicitly by a destroy method, i.e., a *removal heuristic*, and a repair method, i.e., an *insertion heuristic*. The former disrupts part of the current solution while the latter rebuilds the destroyed solution. In ALNS, several removal heuristics and insertion heuristics are employed (see Algorithm 1). In each iteration, the removal/insertion heuristics to be applied on the current solution are selected according to their *scores*, which are adaptively adjusted based on the performance of these heuristics in the previous iterations. The removal/insertion heuristics that have performed well in the previous iterations are more likely to be chosen in the current iteration (lines 8, 9).

For accepting a solution, we use the same criterion defined in the Simulated Annealing algorithm. In particular, let be $X_{current}$ and X_{new} the current and the new solution obtained, respectively and let be $\pi(X_{current})$ and $\pi(X_{new})$ their related total profit. If the new solution has a total profit higher than the one of the current solution, then it is accepted (lines 11–12). On the contrary (lines 13–17), the new solution is accepted with a probability equal to $e^{(-(\pi(X_{new})-\pi(X_{current}))/\tau)}$, where τ denotes the value of the temperature that is initially set to P_{start} times the total profit of the initial solution (with P_{start} an input parameter). The temperature τ decreases at each iteration through a cooling rate $h \in \{0, 1\}$ (line 25). The heuristic ends when the stop criterion is met.

In our implementation, ALNS is run for multiple restarts. The initial solution is obtained through the Nearest Neighborhood Heuristic (NNH, line 2). In each restart ALNS stops when either a predefined maximum number of it-

erations, N_{max}, or a maximum number of non-improving iterations, Z_{max}, is reached (line 7).

4.1. Adaptive Weight Adjustment Procedure (AWAP)

The AWAP, used at step 26 of ALNS, allows to adjust automatically the probabilities of using the removal and insertion heuristics on the basis of statistics from earlier iterations. To this purpose a *score* s_h is associated with each heuristic, which measures how well the heuristic has performed recently. The whole search is divided into a number of segments. A segment is a number NS of iterations of the ALNS. Three different score increments, σ_1,σ_2,σ_3, are used to update the scores of the removal/insertion heuristic depending on their result in each iteration of a segment.

The pseudocode of the AWAP is shown in Algorithm 2. With $mod(N, NS)$ we indicate the remainder of the division of N by NS (it is used to determine the iteration N corresponding to the end of the segment). The case of line 4 indicates that if a heuristic is able to find a new overall best solution, then it is rewarded. In similar way, in line 7, if a heuristic has been able to find a solution that has not been visited before and it is accepted by the accepting criteria of the ALNS, then the heuristic has been successful, since it has brought the search forward. AWAP distinguishes between the two situations corresponding to parameters line 7 and line 10 in order to prefer heuristics that can improve the solution, but that can also diversify the search, and the latter are rewarded by the case of line 10. To this purpose, we memorize all the new solutions accepted, together with their total profit values, using the latter as first check to quickly verify if a solution was already accepted before.

Finally, a *reaction factor*, r, is introduced in the score updating formula (line 18) to control how quickly the weight-adjustment algorithm reacts to changes in the effectiveness of the heuristics (if $r = 0$ the scores are not used and thus the probabilities do not change, if $r = 1$ the scores obtained in the last segment determine the probabilities).

Algorithm 1 Pseudocode for ALNS

Initialize probability $P^0_{RH}$for each removal heuristic $RH \in \mathcal{RH}$ and probability P^0_{IH} for each insertion heuristic $IH \in \mathcal{IH}$ using a uniform probability distribution;
Generate an initial solution X_{NNH} by NNH algorithm;
Set $X_{best} := X_{current} := X_{NNH}$;
Let UR be the set of unsatisfied requests;
Set $\tau := \pi(X_{NNH})P_{start}$;
Set $N := 0, t = 0$;
while $N < N_{max}$ and $Z < Z_{max}$ **do**
    Select a removal heuristic $RH \in \mathcal{RH}$ with probability P^t_{RH};
    Select an insertion heuristic $IH \in \mathcal{IH}$ with probability P^t_{IH};
    Let X_{new} be the solution obtained applying removal heuristic RH to $X_{current}$ and then insertion heuristic IH;
    if $\pi(X_{new}) > \pi(X_{current})$ **then**
        set $X_{current} := X_{new}$;
    else
        set $\nu := e^{(-(\pi(X_{new})-\pi(X_{current}))/\tau)}$;
        generate a random number $\epsilon \in [0, 1]$;
        if $\epsilon < \nu$ **then**
            set $X_{current} := X_{new}$;
        end if
    end if
    if $\pi(X_{current}) > \pi(X_{best})$ **then**
        set $X_{best} := X_{new}, Z := 0$;
    else
        set $Z := Z + 1$;
    end if
    Set $\tau := h\tau$;
    Update probabilities using the AWAP procedure;
    Let $N := N + 1$;
end while

Algorithm 2 Pseudocode for AWAP

if $mod(N, NS) = 0$ **then**
    Set $s_H := 0$, $\theta_H := 0$ for each $H \in \mathcal{RH} \cup \mathcal{IH}$;
end if
if the last removal-insertion heuristics applied, (RH, IH), resulted in a new global best solution **then**
    set $s_{RH} := s_{RH} + \sigma_1$;
    set $s_{IH} := s_{IH} + \sigma_1$;
else if the last remove-insertion heuristics applied, (RH, IH), resulted in a solution that has not been accepted before and the cost of the new solution is better than the cost of current solution **then**
    set $s_{RH} := s_{RH} + \sigma_2$;
    set $s_{IH} := s_{IH} + \sigma_2$;
else if the last remove-insertion heuristics applied, (RH, IH), resulted in a solution that has not been accepted before and the cost of the new solution is worse than the cost of current solution, but the solution was accepted **then**
    set $s_{RH} := s_{RH} + \sigma_3$;
    set $s_{IH} := s_{IH} + \sigma_3$;
end if
set $\theta_{RH} := \theta_{RH} + 1$;
set $\theta_{IH} := \theta_{IH} + 1$;
if $mod(N, NS) = NS - 1$ **then** ▷ Update probabilities after the last iteration of the segment
    for each $H \in \mathcal{RH} \cup \mathcal{IH}$ **do**
        set $P_H^{t+1} := P_H^t(1 - r) + r\frac{s_H}{\theta_H}$;
    end for
    Set $t := t + 1$;
end if

4.2. Removal heuristics

In our implementation of the ALNS for the EVReP, we consider three kinds of removal heuristics:

1. *Random removal*: this heuristic randomly eliminates λ pairs of pick-up and delivery requests among those served in the current solution and puts them in the set of Unserved Requests (UR). We obtain four heuristics of this kind, RH1, RH2, RH3 and RH4, considering the following four values of $\lambda = \{1, \lceil 0.1|SR| \rceil, \lceil 0.2|SR| \rceil, \lceil 0.3|SR| \rceil\}$, where SR is the set of Served Requests in the current solution.

2. *Worst route removal*: it removes as a whole the route with the least total profit. We call this heuristic RH5.

3. *Shaw removal*: this heuristic, introduced by [40], removes requests that are somewhat similar, on the basis of a specific relatedness indicator, since we expect it to be easier to mix similar requests around and thus create new and maybe better solutions. Given two requests i and j, either both of pickup or of delivery, we define their *relatedness* $R(i, j)$ in the following way
$$R(i,j) = w_1 \frac{d_{ij}}{d^{max}} + w_2 \frac{|t_i - t_j|}{(t^{max} - t^{min})} + w_3 |\rho_i - \rho_j|$$
where w_1, w_2, w_3 are three weights with sum 1, d_{ij} is the distance between the locations of requests i and j; d^{max} is the maximum distance between two request locations; t_i and t_j are the times in which, in the current solution, requests i and j are satisfied, respectively; t^{max} and t^{min} are the maximum latest time and the minimum earliest time of all the requests served, respectively. We report the detailed pseudocode of the Shaw removal heuristic, based on that of [36], in Algorithm 3. In the following we call this heuristic RH6.

In Algorithm 3, q is the total number of pickup-delivery pairs removed from X and p is a real number not lower than 1 that introduces some randomness in the selection of the removed requests. Indeed, $p = 1$ corresponds to a completely

Algorithm 3 Pseudocode for Shaw removal

1: Let (r_1, r_2) be a randomly selected pairs of pickup and delivery requests consecutively served in current feasible solution X;
2: Set $S := \{r1, r2\}$;
3: **while** $|S|/2 < q$ **do**
4: Let r be a randomly selected request from S;
5: Let α be an array with size n_α containing all requests of X not in S;
6: Sort α in such a way that $i < j \Rightarrow R(r, \alpha[i]) < R(r, \alpha[j])$;
7: Choose a random number y from the interval [0,1)
8: Set $S := S \cup \{\alpha[\lfloor y^p n_\alpha \rfloor]\}$;
9: Add to S the request associated with $\alpha[\lfloor y^p n_\alpha \rfloor]$ in the solution X;
10: **end while**
11: Remove the requests in S from X;
12: Set $UR := UR \cup S$;

random removal algorithm, while for a very high value of p, y^p tends to 0 and therefore at step 8 we obtain the lowest index element of the sorted array α, i.e., the most related request to r.

4.3. Insertion heuristics

In the ALNS we consider four insertion heuristics, IH1, IH2, IH3 and IH4. They exploit different Local Searches based on four classic moves of the Vehicle Routing Problem: the two intra-moves $Insertion$ and $3-opt$, and the two inter-route moves $Cross$ and $Relocation$, also used in the R&R developed in [14].

The move $Insertion(\omega, p, d, h)$ inserts the pair of requests (p, d), with $p \in P$ and $d \in D$, after the hth pair of requests, in route ω.

The move $3-opt(\omega, (d_1, p_1), (d_2, p_2), (d_3, p_3))$ is similar to the $3-opt$ move for the TSP: the three bike arcs $(d_1, p_1), (d_2, p_2), (d_3, p_3)$ are removed and then the route is rebuilt in the only possible feasible way (with no inversion of the other arcs).

The move $Cross(\omega_1, \omega_2, (d_1, p_1), (d_2, p_2), (d_3, p_3), (d_4, p_4))$ involves a pairs of routes, ω_1 and ω_2, and consists in crossing the two routes by eliminating the four bike arcs $(d_1, p_1), (d_2, p_2), (d_3, p_3), (d_4, p_4)$ where the first two belong

to ω_1 and the last two to ω_2 (e.g., given $\omega_1 = \{0, 1, 2, 3, 4, 5, 6, 0\}$ and $\omega_2 = \{0, 7, 8, 9, 10, 11, 12, 13, 14, 0\}$, the move $Cross(\omega_1, \omega_2, (2, 3), (4, 5), (8, 9), (12, 13))$, generates the two new routes $\omega_1 = \{0, 1, 2, 9, 10, 11, 12, 5, 6, 0\}$ and $\omega_2 = \{0, 7, 8, 3, 4, 13, 14, 0\}$).

The move $Relocation(\omega_1, \omega_2, (p_1, d_1), h)$ also involves a pairs of routes, ω_1 and ω_2, and relocates the pair of requests (p_1, d_1) of ω_1 after the hth pair of requests belonging to route ω_2.

We indicate by $LS-insertion$, $LS-3-opt$, $LS-Cross$ and $LS-Relocation$ a local search based on each neighborhood above described, respectively. The insertion heuristics IH1–IH4 exploit these local searches as described in the pseudocodes of Algorithms 4–7.

In particular, in heuristic $IH2$, the effect of applying an $LS - Cross$, fixing $d_1 = p_2 = 0$ in route ω_1, consists in moving all the requests of ω_1 in ω_2, and in moving in ω_1 only some of the requests of ω_2, i.e., those between d_3 and p_4 (excluding d_3 and p_4). Thus, if all the requests of a route ω_1 can be easily served by route ω_2 (e.g., since their locations are near those of ω_2), they are given as a whole to ω_2. In symmetric way, the same reasoning holds if $d_3 = p_4 = 0$ in ω_2.

Algorithm 4 Pseudocode for IH1

1: **repeat**
2: Minimize the total duration of the current solution applying to each route an $LS - 3 - opt$ and then an $LS - Cross$;
3: Maximize the total profit of the current solution applying to each route an $LS - Insertion$ considering all pairs of requests (p, d) of UR and all possible values of h; among all feasible insertions with the same total profit, apply that at minimum increasing of total duration;
4: **until** the solution changes

Algorithm 5 Pseudocode for IH2

1: Maximize the total profit of the current solution applying an $LS - Cross$ with only $d_1 = p_2 = 0$ in route ω_1 or $d_3 = p_4 = 0$ in route ω_2;
2: Eliminate the possible route with negative profit;
3: Add the requests eliminated in UR

Algorithm 6 Pseudocode for IH3

1: Maximize the total profit of the current solution applying an $LS - Relocation$;
2: Eliminate the possible route with negative profit;
3: Add the requests eliminated in UR.

Algorithm 7 Pseudocode for IH4

1: Create a new route $\omega = \{0, 0\}$ (i.e. a degenerate route from depot to depot);
2: Maximize the total profit of the current solution applying an $LS - insertion$ to ω considering all pairs of requests (p, d) of UR and all possible values of h; among all feasible insertion with the same total profit, apply that at minimum increasing of total duration.

5. A Tabu Search metaheuristic for the E-VReP

Tabu Search (TS) is a metaheuristic introduced by [20] and successfully applied to solve challenging vehicle routing problems (e.g., [19], [11],[30] and [1]). It consists in exploring the search space by moving from a feasible solution to its best neighbor even if the latter worsens the objective function. This allows escaping from local optima. To avoid cycling, solutions already examined are forbidden, that is, declared *tabu*. For this reason, a *Tabu List* (TL), with length at most equal to the number of iterations, is introduced to memorize the already considered solutions.

We properly design a multistart TS algorithm for solving the E-VReP. Its pseudocode is reported in Algorithm 8. A starting feasible solution is generated applying the NNH algorithm. It represents the current solution, $X_{current}$, and also the best solution found until now, X_{best}. TL is initialized by $X_{current}$.

At each iteration, the TS applies a local search to $X_{current}$, exploring its neighborhood $\mathcal{N}(X_{current})$ excluding the elements in TL. In the tabu list all the local optima obtained in the TS together with their total profit values are stored, using the latter as first check to quickly verify if a solution was already accepted before (as also made for AWAP Algorithm, in line 10).

In order to compare on equal terms the TS with the ALNS, the neighborhood consists in all the 24 possible combinations of the six Removal heuristics with the four Insertion heuristics already introduced for the ALNS. $X_{current}$ is updated

with the local optimum obtained and it is added to TL. If $X_{current}$ improves X_{best}, the latter is replaced by $X_{current}$.

After Z_{max} consecutive iterations without improving the best value of the objective function, the current round of TS is stopped since the search is considered to be trapped in a deep local optimum (*while* loop on Z). Then the search is restarted from a new solution. While each round of TS explores in detail a region of the search space, each restart moves the search to a new region. The restart consists in completely destroying the current solution by moving all the requests in the set of the unsatisfied requests ($UR := R$), in applying the insertion heuristic $IH4$ and then the random removal heuristic $RH1$.

Finally, the TS stops when a maximum number of iterations N_{max} is reached (*while* loop on N).

Algorithm 8 Pseudocode for multistart TS

Generate an initial solution X_{NNH} by NNH algorithm;
Set $X_{best} := X_{current} := X_{NNH}$;
Let TL be the TL list: $TL = \{X_{NNH}\}$;
Set $N := 0$; ▷ Initialization of the iteration counter
while $N < N_{max}$ **do**
    Set $Z := 0$; ▷ Initialization of consecutive iteration counter without improvement
    while $Z < Z_{max}$ **do**
        Let be $X_{current} = argmax\{\pi(x) : x \in \mathcal{N}(X_{current}) \setminus TL\}$;
        if $\pi(X_{current}) \geq \pi(X_{best})$ **then**
            $X_{best} := X_{current}$;
            $Z := 0$;
        else $Z := Z + 1$;
        end if
        $TL := TL \cup \{X_{current}\}$
        $N := N + 1$;
    end while
    Set $UR := R$ and apply $IH4$ and $RH1$; ▷ Creation of a new solution
end while

6. Bounding procedures

In order to evaluate the quality of the solutions determined by the meta-heuristics, also when the optimal value of the E-VReP is not available, we propose three methods to compute upper bounds. They are based on the MILP formulation of the E-VReP proposed in [14, 12] and that we recall here.

$$\max \sum_{k=1}^{K} \sum_{(i,j)\in A: i\neq 0} rev_i x_{ijk} - C \sum_{k=1}^{K} \sum_{(0,j)\in A} x_{0jk} \tag{1}$$

$$\sum_{j\in\delta^+(0)} x_{0jk} \leq 1 \quad \forall k = 1,\ldots,K \tag{2}$$

$$\sum_{k=1}^{K} \sum_{j\in\delta^+(i)} x_{ijk} \leq 1 \quad \forall i \in P \cup D \tag{3}$$

$$\sum_{j\in\delta^+(i)} x_{ijk} - \sum_{j\in\delta^-(i)} x_{jik} = 0 \quad \forall i \in P \cup D \cup \{0\},\ \forall k = 1,\ldots,K \tag{4}$$

$$t_{ik} + c_{ij} x_{ijk} \leq t_{jk} + T(1 - x_{ijk}) \quad \forall (i,j) \in A : j \neq 0,\ \forall k = 1,\ldots,K \tag{5}$$

$$t_{ik} + c_{i0} x_{i0k} \leq t_{0k} + T + T(1 - x_{i0k}) \quad \forall i \in \delta^-(0),\ \forall k = 1,\ldots,K \tag{6}$$

$$\tau_i^{min} \leq t_{ik} \leq \tau_i^{max} \quad \forall i \in P \cup D,\ \forall k = 1,\ldots,K \tag{7}$$

$$d_{ij} x_{ijk} \leq L(\rho_i + \frac{t_{ik} - \tau_i^{min}}{\Gamma}) \quad \forall (i,j) \in A : i \in P, j \in D,\ \forall k = 1,\ldots,K \tag{8}$$

$$\rho_i + \frac{t_{ik} - \tau_i^{min}}{\Gamma} - \frac{d_{ij}}{L} x_{ijk} \geq \rho_j - \frac{\tau_j^{max} - t_{jk}}{\Gamma} - (\rho_j + 1)(1 - x_{ijk}) \tag{9}$$

$$\forall (i,j) \in A : i \in P, j \in D,\ \forall k = 1,\ldots,K$$

$$1 - \frac{d_{ij}}{L} x_{ijk} \geq \rho_j - \frac{\tau_j^{max} - t_{jk}}{\Gamma} - (\rho_j + 1)(1 - x_{ijk}) \tag{10}$$

$$\forall (i,j) \in A : i \in P, j \in D,\ \forall k = 1,\ldots,K$$

$$x_{ijk} \in \{0,1\} \quad \forall (i,j) \in A,\ \forall k = 1,\ldots,K \tag{11}$$

$$t_{ik} \geq 0 \quad \forall i \in P \cup \in D \cup \{0\},\ \forall k = 1,\ldots,K \tag{12}$$

In such formulation, x_{ijk} represents the routing variables of the k-th worker, A is the set of the arcs of the modeling graph, linking each pair of compatible relocation requests. Since each arc connects a pair of requests, the objective function (1) represents the total profit given by the difference of the total revenue associated with all the requests served and the total cost proportional by C to the number of workers employed. Constraints (2) take into account that at most K workers are available. Therefore, at most K routes can be generated imposing that at most one arc for each worker can leave the depot node 0. Constraints (3) impose that each request is satisfied at most once. Flow conservation constraints (4) guarantee that the solution is a collection of cycles. Constraints (5) rule the time variables ensuring that the visit time of a node is given by the sum of the visit time of its predecessor and the operational time to go from the predecessor to the current node. The input parameter c_{ij} denotes the time to reach the location of the request j from the location of the request i. It is worth noting that such constraints are not imposed for the depot node in order to ensure that the route can pass through it. At the same time, they prevent the solution from containing isolated cycles that do not pass through the depot. In this way, the formulation does not require additional sub-tour elimination constraints. Constraints (6) ensure that the duration of each route does not exceed the threshold T. Constraints (7) enforce the time windows for the pickup and delivery requests. Constraints (8) model the fact that the maximum distance traveled by an EV is linearly proportional to the residual charge. We observe that if $\rho_i + \frac{t_{ik} - \tau_i^{min}}{\Gamma} > 1$ such constraints become redundant since the graph topology prevents already the existence of arcs (i, j) with $d_{ij} > L$. Finally constraints (9) and (10) ensure that an EV is delivered with a battery level such that at the time τ_j^{max} a charge level not lower than ρ_j will be achieved.

Concerning the bound computation, we suppose that the number of available workers K is greater than 1. In fact, when $K = 1$, the optimal value of the E-VReP can be obtained in easier way through the MILP formulation, since the number of decision variables x_{ijk} depends on K (thus in this case the bound computation is not necessary). Moreover, hereafter we suppose that there exists

a feasible route such that the sum of the profit of the requests served is greater than C. This ensures that the optimal solution of the E-VReP is positive for any value of K (also for $K = 1$).

A first method to compute an upper bound, UB_1, to the optimal value of the E-VReP, consists straightforwardly in multiplying by K the optimal profit obtained solving the MILP formulation of the E-VReP (1)-(12) with only one worker (i.e., $K = 1$). Indeed let z_1^* the latter profit and let z^{max} the maximum profit of a worker in the optimal solution of the original problem. By definition it results $z^{max} \leq z_1^*$, therefore for the optimal value z^* of the E-VReP it happens $z^* \leq Kz^{max} \leq Kz_1^* = UB_1$.

A second method to compute an upper bound, UB_2, consists in solving with one worker the relaxation of the MILP formulation (1)-(12) where the time window constraints (7) are removed and the worker's working time is extended to KT. UB_2 is an upper bound for the original problem because any feasible solution of the latter consists in at most K routes, each one with duration less than T, and therefore they can be also traveled by one worker within the working time KT satisfying all constraints except the time window constraints (7).

Compared to the bounding procedure UB_1, UB_2 exploits more in depth the information of all the available requests and avoids that the profit of the same request can be considered more than once in the bound computation (as it happens in UB_1). To exploit the advantages of both the bounding procedures, we combine them in only one bound, UB_3 defined as in the following:

$$UB_3 = \min\left\{\max\left\{\tilde{z}_2^* - C\left\lceil\frac{\tilde{z}_2^*}{\tilde{z}_1^*}\right\rceil, \left\lfloor\frac{\tilde{z}_2^*}{\tilde{z}_1^*}\right\rfloor \tilde{z}_1^* - C\left\lfloor\frac{\tilde{z}_2^*}{\tilde{z}_1^*}\right\rfloor\right\}, Kz_1^*\right\} \tag{13}$$

where $\tilde{z}_1^*$ is the revenue associated with the solution detected in the bounding procedure UB_1, i.e., $\tilde{z}_1^* = z_1^* + C$, and $\tilde{z}_2^* = UB_2 + C$, in similar way.

Through the following two propositions we prove respectively that UB_3 is an upper bound for the E-VReP and it dominates both the previous upper bounds UB_1 and UB_2.

Proposition 1. *UB_3, defined in (13), provides an upper bound to the E-VReP.*

Proof. Since $\tilde{z}_1^*$ is the revenue associated with the solution detected in the bounding procedure UB_1, it is the maximum revenue that can be collected by one worker. Since $\tilde{z}_2^*$ is the revenue associated with the solution detected in the bounding procedure UB_2, it is the maximum revenue that can be collected by all the available workers. Therefore the minimum number of workers necessary to collect the revenue $\tilde{z}_2^*$ is $\left\lceil \frac{\tilde{z}_2^*}{\tilde{z}_1^*} \right\rceil$ and then the corresponding profit is $\tilde{z}_2^* - C \left\lceil \frac{\tilde{z}_2^*}{\tilde{z}_1^*} \right\rceil$. With one less worker the collected revenue decreases of $\tilde{z}_2^* - \left\lfloor \frac{\tilde{z}_2^*}{\tilde{z}_1^*} \right\rfloor \tilde{z}_1^*$ and therefore if this quantity is less than C, it is more profitable to use one less worker (i.e., $\left\lfloor \frac{\tilde{z}_2^*}{\tilde{z}_1^*} \right\rfloor$ workers) and the profit becomes as indicated in the second argument of the maximum in formula (13). This bound is strengthened by taking the minimum with Kz_1^* since the latter coincides with the upper bound UB_1. ■

Proposition 2. *UB_3 dominates both the upper bounds UB_1 and UB_2, i.e., $UB_3 \leq UB_1$ and $UB_3 \leq UB_2$.*

Proof. UB_3 is by definition lower or equal to UB_1, since it is defined as the minimum with $Kz_1^* = UB_1$. UB_3 is also lower or equal to UB_2 since if UB_3 is equal to the first argument of the maximum in formula (13) then it becomes $\tilde{z}_2^* - Ch$ with $h \geq 2$ and therefore it is strictly lower than $\tilde{z}_2^* - C = UB_2$. Otherwise, if UB_3 is equal to the second argument of the maximum then it becomes $\left\lfloor \frac{\tilde{z}_2^*}{\tilde{z}_1^*} \right\rfloor \tilde{z}_1^* - Ch \leq \tilde{z}_2^* - C = UB_2$ being in this case $h \geq 1$. Finally, if UB_3 is equal to the second argument of the minimum in formula (13) then it is lower than both the previous values already discussed and thus also than UB_2. ■

7. Computational experiments

This section describes the computational experiments carried out to assess the performances of the ALNS proposed in Section 4. To this purpose, we compare the results found by the ALNS with those obtained by the MILP and the R&R, described in [14] and by the TS proposed in Section 5. Both the ALNS and TS were implemented in Java and were run on an Intel Core I7-2630QM CPU with 2 GHz and 8 GB RAM. Table 1 shows the setting of the parameters

used in the ALNS. This setting has been obtained after a careful tuning also based on some reasonings. For instance, among them, how we set parameter NS representing the number of iterations over which, according to the AWAP procedure, the distribution of the most successful combinations of the removal-insertion heuristics is evaluated. NS has to be big enough to allow testing the performances of different combinations of removal-insertion heuristics, but not too big otherwise the AWAP procedure becomes too time consuming. At each iteration, a removal heuristic is combined with an insertion one in random way, initially with the uniform distribution and then with the probability values set by AWAP. Since there are 6 removal heuristics and 4 insertion ones, in order to test all their combinations, NS has to be at least 6*4=24. Moreover, in order to test also the performances of combinations with small probabilities of being selected, for instance 1/4 of the probability of the uniform distribution, we need $NS \geq 96$. For this reason we set $NS = 100$.

To solve the MILP, we use the state of the art solver $CPLEX12.5$, with a CPU time limit of $43,200$ seconds.

Parameter	$Nmax$	$Zmax$	$Pstart$	h	NS	σ_1	σ_2	σ_3	r	w_1	w_2	w_3
Setting	1000	50	100	0.999	100	33	9	13	0.1	$\frac{1}{3}$	$\frac{1}{3}$	$\frac{1}{3}$

Table 1: Setting of the parameters used in the ALNS

Regarding the TS algorithm, we found that a value of $N_{max} = 100$ and of $Z_{max} = 20$ is the right compromise between the solution quality and the total computational effort.

7.1. Test instances

The numerical experiments are carried out on three sets of realistic instances that share the input parameters shown in Table 2.

Parameter	T	s'	s''	q'	q''	L	Γ	C
Setting	300 minutes	25km/h	15km/h	1 minute	1 minute	150km	240 minutes	60 €

Table 2: Setting of the main input parameters of the E-VReP

The first set, AMAT, taken from the literature and described in [13], is

introduced in order to experiment with relocation requests having the same revenue (20 €). In this set, $15 \leq |R| \leq 31$, with an average value of 22.

The second set, V-AMAT [14], is used in order to analyze instances having different revenues, with $15 \leq |R| \leq 49$ and an average of 27 requests. The revenue of each request is made up by both a variable and a fixed component. The former is proportional to the user's rent-time while the latter is introduced to prevent customers, that require the service for a short rent-time, from being never served. Finally, a new set of instances, L-AMAT, has been specifically created for testing the ALNS on large size instances. It is made up of 20 instances, with $83 \leq |R| \leq 100$ and an average of 89 requests having the same revenue (20 €).

Concerning the number of available workers K, for the instance sets AMAT and V-AMAT we consider $K = 4$, while for the set L-AMAT, $K = 8$.

7.2. Computational results on AMAT set

Table 3 shows the numerical results obtained on the AMAT set, in which the first three columns report the instance name, the number of pick-up requests ($|P|$) and of delivery ($|D|$), respectively. The column $ALNS$ reports the solution found by the ALNS, with reference to both the total profit (π) and the number of workers used (K). The column $GapMILP$ shows the gaps between the solutions found by the ALNS and by the MILP, with reference to both the total profit ($\Delta\pi(\%)$) and the number of workers used (ΔK). Similarly, the column $GapR\&R$ shows the gaps between the solutions found by ALNS and by the R&R, the column $GapTS$ reports the gaps between the solutions detected by ALNS and the ones of TS, and the column $GapUB$ reports the gaps with the upper bound UB_3.

The percentage gap between the total profit of the solution found by the ALNS (π) and the one of the solution found by the MILP ($\pi^{'}$) is computed as in the following:

$$\Delta\pi = \frac{\pi - \pi^{'}}{\max\{\pi, \pi^{'}\}}.100 \tag{14}$$

The gap between the number of workers used in the solution found by the ALNS (K) and that of the solution found by the MILP ($K^{'}$) is computed as:

$$\Delta K = K - K^{'} \tag{15}$$

The gaps between the solutions found by the ALNS and those of the R&R as well as those of the TS and the gaps with UB_3 are computed in similar way.

The last four columns show the CPU times of the MILP, that of R&R, that of TS and of the ALNS, respectively. The symbol † indicates that the MILP solver reached the CPU time limit. The cases in which the ALNS outperforms in terms of total profit the R&R or the TS or the MILP (since the latter reached the CPU time limit) are highlighted in boldface. These conventions also hold in the next tables.

Instance	$\|P\|$	$\|D\|$	ALNS		GapMILP		GapR&R		GapTS		GapUB	CPU(seconds)			
			π	K	$\Delta\pi(\%)$	ΔK	$\Delta\pi(\%)$	ΔK	$\Delta\pi(\%)$	ΔK	$\Delta\pi(\%)$	$MILP$	$R\&R$	TS	$ALNS$
AMAT1	15	12	360	2	0.00	0	0.00	0	0.00	0	0.00	$43,200^{\dagger}$	0.51	8.56	0.36
AMAT2	15	7	220	1	0.00	0	0.00	0	0.00	0	0.00	779.02	0.68	2.54	0.14
AMAT3	16	6	180	1	0.00	0	0.00	0	0.00	0	0.00	1,864.60	0.42	2.43	0.14
AMAT4	15	14	400	2	0.00	0	0.00	1	0.00	0	9.09	3,640.22	1.61	10.75	0.56
AMAT5	13	11	320	2	0.00	0	0.00	0	0.00	0	5.88	4,200.40	0.48	5.94	0.33
AMAT6	7	8	180	1	0.00	0	0.00	0	0.00	0	0.00	3,852.14	0.36	1.52	0.10
AMAT7	9	8	220	1	0.00	0	0.00	0	0.00	0	0.00	1,025.10	0.13	1.28	0.11
AMAT8	11	10	240	2	0.00	0	**8.33**	1	0.00	0	0.00	3,765.38	0.19	3.37	0.21
AMAT9	15	7	220	1	0.00	0	0.00	0	0.00	0	0.00	3,678.24	0.52	3.00	0.15
AMAT10	11	11	320	2	0.00	0	0.00	0	0.00	0	0.00	$43,200^{\dagger}$	0.47	4.31	0.23
AMAT11	10	8	260	1	0.00	0	0.00	0	0.00	0	0.00	17,049.23	0.15	2.08	0.17
AMAT12	12	6	180	1	0.00	0	0.00	0	0.00	0	0.00	76.26	0.13	1.57	0.10
AMAT13	13	12	340	1	0.00	0	**5.88**	-1	0.00	0	0.00	3,658.22	0.27	2.55	0.31
AMAT14	6	12	180	1	0.00	0	0.00	0	0.00	0	0.00	13,780.46	0.13	0.88	0.13
AMAT15	11	5	140	1	0.00	0	0.00	0	0.00	0	0.00	13.02	0.14	2.88	0.08
AMAT16	16	11	340	1	0.00	0	0.00	0	0.00	0	0.00	$43,200^{\dagger}$	0.27	5.04	0.46
AMAT17	15	16	480	2	**7.57**	0	0.00	0	0.00	0	0.00	$43,200^{\dagger}$	15.13	13.22	1.33
AMAT18	10	12	280	2	0.00	0	0.00	0	0.00	0	0.00	$43,200^{\dagger}$	8.95	4.68	0.21
AMAT19	15	7	220	1	0.00	0	0.00	0	0.00	0	0.00	6,249.01	0.14	2.98	0.14
AMAT20	17	8	260	1	0.00	0	0.00	0	0.00	0	0.00	$43,200^{\dagger}$	0.17	2.82	0.20
AMAT21	12	16	360	2	0.00	0	0.00	0	0.00	0	0.00	$43,200^{\dagger}$	3.47	9.29	0.40
AMAT22	13	8	220	1	0.00	0	0.00	0	0.00	0	0.00	527.25	0.14	2.99	0.18
AMAT23	9	9	260	1	0.00	0	0.00	0	0.00	0	0.00	2,529.19	0.14	1.67	0.15
AMAT24	12	7	220	1	0.00	0	0.00	0	0.00	0	0.00	16,782.88	0.14	2.70	0.13
AMAT25	7	9	180	1	0.00	0	0.00	0	0.00	0	0.00	4,443.23	2.30	2.83	0.39
AMAT26	11	11	340	1	0.00	0	0.00	0	0.00	0	0.00	$43,200^{\dagger}$	0.32	5.58	0.28
AMAT27	14	12	360	2	0.00	0	0.00	0	0.00	0	0.00	3,890.55	10.33	7.41	0.29
AMAT28	13	11	280	2	0.00	0	**7.14**	1	0.00	0	12.50	$43,200^{\dagger}$	0.22	5.17	0.29
AMAT29	13	10	280	2	0.00	0	0.00	0	0.00	0	0.00	3,658.79	6.45	5.16	0.27
AMAT30	9	15	260	1	0.00	0	0.00	0	0.00	0	0.00	3,700.04	0.19	4.78	0.23
AVG			**270**	**1**	**0.25**	**0**	**0.71**	**0**	**0.00**	**0**	**0.92**	**16,265.44**	**1.82**	**4.33**	**0.27**

Table 3: Computational Results on the AMAT set

Analyzing the results reported in Table 3,it is possible to see that both the ALNS and TS find the same results. Moreover, in all instances except one (AMAT29) they detect the optimal solution since either their gap with the upper bound is 0 or their gap with the MILP is 0 and the latter does not reach the maximum CPU time limit. Concerning the CPU time, the ALNS outperforms the TS requiring on average 0.27 seconds against 4.33 seconds of the TS. It also outperforms, by far, CPLEX that requires on average $16,265.44$ seconds to solve the MILP. Finally, there is one case in which the solution found by the ALNS is better than the one detected by the MILP (since the latter reached the CPU time limit) while there are three cases in which it is better than the R&R.

7.2.1. Analysis on the impact of the heuristics

We investigated the impact of the 10 heuristics (six of Removal, RH1-RH6, and four of Insertion, IH1-IH4), computing the average number of times they are used on the 30 instances of the AMAT set.This analysis is reported in Table 4. We observe that for the Removal heuristics, there is not a strong dominance of one of them respect to the others, except for the fact that the RH3 is less used. While, concerning the Insertion heuristics, the use of the IH4 dominates that of the others.

	RH1	RH2	RH3	RH4	RH5	RH6	IH1	IH2	IH3	IH4
Number of times	12	12	6	10	9	8	15	13	8	21

Table 4: Average number of times of usage of the removal/insertion heuristics

We analyze more in detail the impact of each heuristic on the solution quality selecting, as an example, the instance AMAT 18 (Table 5). When either only one Removal heuristic or only one Insertion heuristic is used, the results are always worse than when all of them are used, with the average profit worsening of about 35.71%. While, when either only one Removal heuristic or only one Insertion heuristic is excluded, the results are worse in the 40% of the cases, with the average profit worsening of about 2.86%.

When only IH1 is used, the worsening of the total profit is about 14.28% (passing from 280 to 240). On the other hand, when either only one of the 6

Removal heuristics or IH2 is used, the profit worsening is about 75% (from 280 to 160). Finally, when either IH3 or IH4 is used, the profit worsening is about 21.43% (from 280 to 220).

If only RH1 is excluded, the profit worsening is about 7.14% (from 280 to 260). The same happens when either only RH2 or RH6 or IH4 is excluded. Therefore, these heuristics show to give a contribution to the solution quality that is higher than the others.

	RH1	RH2	RH3	RH4	RH5	RH6	IH1	IH2	IH3	IH4
Total profit (in €) using only 1 heuristic	160	160	160	160	160	160	240	160	220	220
Total profit (in €) excluding only 1 heuristic	260	260	**280**	**280**	**280**	260	**280**	**280**	**280**	260

Table 5: Impact of the heuristics on the solution quality of the AMAT 18

To assess the behavior of the removal/insertion heuristics, we also adopt an entropy-based diversity measure as in [29]. In particular, the diversity δ is computed on a set SOL of solutions obtained in the last IT iterations (i.e., IT is like an interval of observation for the diversity indicator). According to [29], we introduce a coincidence matrix CM where $CM_{ii} = 0$ and

$$CM_{ij} = \sum_{sol \in SOL} C^{ij}_{sol}, \forall i, j = 1, \ldots, |R| \tag{16}$$

with $C^{ij}_{sol} = 1$ if and only if request i and request j are consecutively served in the solution sol. Therefore, δ is computed as in the following:

$$\delta = -\sum_{i=1}^{|R|} \sum_{j=1}^{|R|} q_{ij} ln(q_{ij}) \tag{17}$$

where $q_{ij} = 1$ if $CM_{ij} = 0$ while $q_{ij} = CM_{ij}/IT$, otherwise. When the solutions generated are little diversified, δ tends to 0. Indeed if in the last IT iterations the solutions are the same then $q_{ij} = 1 \ \forall i, j \in R$ and therefore δ=0.

By computing δ, we can observe two types of behavior:if δ is small then the behavior is *converging*, i.e., with a high percentage of pair of requests served in common among the solutions generated in the last IT iterations, and it is *chaotic*, otherwise.

The entropy-based diversity measure is performed on the instance AMAT 10, fixing IT equal to 5 and running the ALNS for 100 iterations. We chose

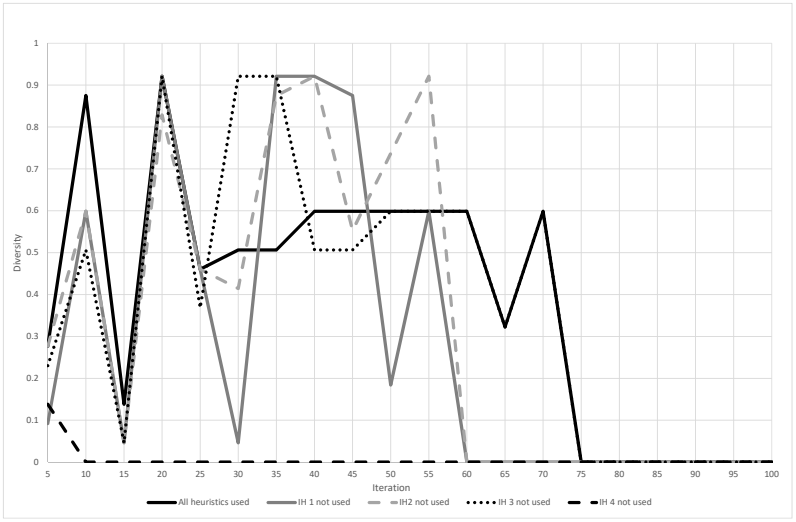

Figure 2: Entropy-based diversity analysis of Insertion heuristics on AMAT 10

this number of iterations because for this instance we obtained the convergence of the ALNS within this value.

Figure 2 shows the values of δ when one of the Insertion heuristics is not used at a time. In particular, we can observe a very converging behavior when IH4 is not used since in this case δ slumps to 0 immediately after the 10-th iteration, while if all the insertion heuristics are used this happens only after the 75-th iteration. This behavior is coherent with the fact that the IH4 heuristic is the most disruptive one, since creates a new route $\omega = \{0, 0\}$, (i.e., a degenerate route where no request is served), and thus, it gives the possibility to include new relocation requests.

In similar way, Figure 3 shows the trend of δ (always on instance AMAT 10) when one of the Removal heuristics is not used at a time. In particular, in this case, we observe that there is not a more converging behavior of one heuristic than the others and thus, all of them tend to diversify.

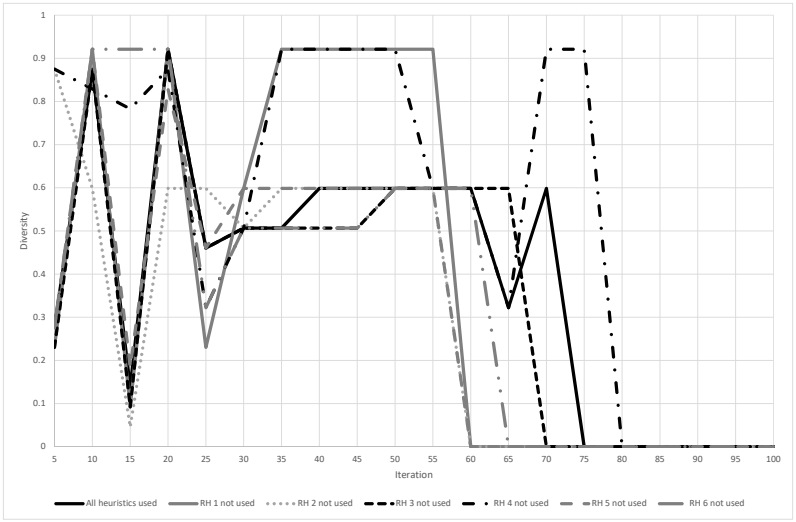

Figure 3: Entropy-based diversity analysis of Removal heuristics on AMAT 10

7.3. Computational results on V-AMAT set

In this section, we compare the solutions found by the ALNS with those found by the MILP, the R&R, the TS and with the upper bound UB_3, on the set V-AMAT.

The meaning of the columns in Table 6 is the same of the one shown in Table 3. The gaps are computed as explained in Section 7.2. The cases in which the ALNS finds the best total profit are remarked in boldface.

From the results shown in Table 6 we can certify that the ALNS finds the optimum on 22 instances (over 30) since either its gap with the upper bound is 0 or it is so its gap with the MILP and the latter does not reach the maximum CPU time limit. Moreover, since its average gap with the upper bound is only of 0.49%, we deduce that the solutions found by the ALNS on the other instances are very near to the optimal ones. The ALNS is able to improve on average of the 1.52% the best profit found by CPLEX in the CPU time limit.

Concerning the comparisons with R&R, the ALNS provides on 19 instances

Instance	$\|P\|$	$\|D\|$	ALNS		GapMILP		GapR&R		GapTS		GapUB	CPU(seconds)			
			π	K	$\Delta\pi(\%)$	ΔK	$\Delta\pi(\%)$	ΔK	$\Delta\pi(\%)$	ΔK	$\Delta\pi(\%)$	$MILP$	$R\&R$	TS	$ALNS$
V-AMAT1	32	17	462.99	2	**12.46**	-1	**14.23**	0	**7.12**	0	0.50	$43,200^{\dagger}$	13.65	12.85	8.84
V-AMAT2	19	4	44.76	1	0.00	0	0.00	0	0.00	0	0.00	27.29	0.40	1.04	0.10
V-AMAT3	17	14	354.74	2	0.00	0	**0.04**	0	0.00	0	0.00	$43,200^{\dagger}$	18.60	16.38	10.82
V-AMAT4	21	8	219.63	1	0.00	0	**0.07**	0	0.00	0	0.00	$43,200^{\dagger}$	0.92	5.00	0.50
V-AMAT5	18	2	9.61	1	-0.06	0	0.00	0	0.00	0	0.03	0.14	0.19	0.40	0.09
V-AMAT6	35	13	331.00	2	0.00	0	**0.65**	0	**0.30**	0	0.00	$43,200^{\dagger}$	10.33	32.05	1.87
V-AMAT7	15	6	143.52	1	0.00	0	**0.41**	0	0.00	0	0.00	430.42	1.30	1.39	0.20
V-AMAT8	13	9	213.74	1	0.00	0	**0.37**	0	**0.36**	0	0.00	$43,200^{\dagger}$	1.28	5.59	0.28
V-AMAT9	15	8	179.34	1	0.00	0	**0.03**	0	0.00	0	0.00	$43,200^{\dagger}$	0.83	4.11	0.20
V-AMAT10	12	6	146.24	1	0.00	0	0.00	0	**23.50**	0	0.00	22.22	1.92	0.81	0.15
V-AMAT11	14	11	281.61	1	**8.64**	-1	**8.65**	-1	**0.74**	0	0.54	$43,200^{\dagger}$	2.33	5.88	0.80
V-AMAT12	19	5	112.14	1	0.00	0	**0.35**	0	**0.35**	0	0.00	2550.65	1.22	2.18	0.12
V-AMAT13	20	12	318.14	1	**7.70**	-1	**7.75**	-1	0.00	0	0.13	$43,200^{\dagger}$	11.62	11.26	0.67
V-AMAT14	16	7	147.62	1	0.00	0	**0.35**	0	0.00	0	0.00	$43,200^{\dagger}$	1.12	3.26	0.17
V-AMAT15	24	17	458.55	2	**12.21**	-1	**12.21**	-1	-0.20	0	0.86	$43,200^{\dagger}$	10.62	24.20	2.10
V-AMAT16	17	9	214.95	1	0.00	0	**0.10**	0	0.00	0	0.00	$43,200^{\dagger}$	5.06	5.66	0.27
V-AMAT17	29	13	327.07	2	0.00	0	**0.67**	0	**0.13**	0	0.07	$43,200^{\dagger}$	8.42	23.32	3.95
V-AMAT18	19	2	8.40	1	0.00	0	0.00	0	0.00	0	0.02	0.86	0.39	1.22	0.07
V-AMAT19	18	7	145.58	1	0.00	0	**0.44**	0	0.00	0	0.00	$43,200^{\dagger}$	2.54	1.98	0.26
V-AMAT20	16	6	146.56	1	0.00	0	0.00	0	0.00	0	0.00	1598.44	2.01	2.17	0.13
V-AMAT21	13	2	8.34	1	0.00	0	0.00	0	0.00	0	0.00	0.07	0.01	0.06	0.06
V-AMAT22	18	5	113.10	1	0.00	0	0.00	0	0.00	0	0.00	614.58	1.22	1.62	0.13
V-AMAT23	1	20	494.79	3	**5.44**	0	**5.44**	0	**5.44**	0	11.80	$43,200^{\dagger}$	2.92	29.87	1.08
V-AMAT24	22	4	79.29	1	0.00	0	**0.38**	0	**1.50**	0	0.00	61.59	0.98	0.69	0.11
V-AMAT25	18	9	248.65	1	0.00	0	0.00	0	0.00	0	0.00	$43,200^{\dagger}$	4.79	3.23	2.20
V-AMAT26	33	7	181.41	1	-0.65	0	0.00	0	0.00	0	0.65	681.72	3.59	3.49	0.26
V-AMAT27	16	5	112.88	1	0.00	0	0.00	0	0.00	0	0.00	820.10	1.09	1.01	0.12
V-AMAT28	12	3	9.13	1	0.00	0	0.00	0	0.00	0	0.00	0.06	0.61	0.13	0.08
V-AMAT29	14	4	43.01	1	0.00	0	**1.44**	0	**3.81**	0	0.00	3.53	1.11	0.15	0.07
V-AMAT30	18	6	145.20	1	0.00	0	**0.11**	0	**0.11**	0	0.00	$43,200^{\dagger}$	1.44	3.26	0.21
AVG			**190.07**	**1**	**1.52**	**0**	**1.79**	**0**	**1.44**	**0**	**0.49**	**23,267.06**	**3.75**	**6.81**	**1.20**

Table 6: Computational Results on the V-AMAT set

better profits while there are no cases in which it performs worse than the R&R. In this way the average relative profit improvement is of 1.79%. Compared to the TS, the ALNS provides on 11 instances better profits while only for one instance ($V-AMAT15$) a worse profit, with an average relative profit improvement of 1.44%. Concerning the CPU time, the ALNS outperforms all the other solution methods requiring on average 1.20 seconds against 6.81 seconds of the TS, 3.75 seconds of the R&R and $23,267.06$ seconds of the MILP.

7.4. Computational results on L-AMAT set

In this section, we describe the numerical results found by running the proposed ALNS on the large size instance set L-AMAT. Unlike the previous two sections, we can compare the solution quality only with the R&R and the TS

metaheuristic due to the prohibitive computational times required by the MILP solver.

The third and fourth column of Table 7 show the solutions found by the ALNS, in terms of both the total profit (π) and the number of workers used (K), respectively. The fifth and sixth column report the gaps (computed as explained is Section 7.2) between the ALNS and the R&R in terms of both the total profit ($\Delta\pi(\%)$) and the number of workers used (ΔK), respectively. The seventh column reports the relative percentage gap between the ALNS and the upper bound UB_3 described in Section 6. Finally, the last three columns show the CPU times required by the R&R, the TS and the ALNS, respectively.

INSTANCE	$\|P\|$	$\|D\|$	ALNS		GapR&R		GapTS		GapUB	CPU(seconds)		
			π	K	$\Delta\pi(\%)$	K	$\Delta\pi(\%)$	K	$\Delta\pi(\%)$	$R\&R$	TS	$ALNS$
L-AMAT1	37	50	1240	4	**32.26**	2	0.00	0	4.62	804.77	475.25	83.20
L-AMAT2	49	47	1520	6	**50.00**	4	0.00	0	7.32	971.53	653.11	53.20
L-AMAT3	51	38	1200	4	**36.67**	2	**1.67**	-1	3.23	822.41	488.98	80.58
L-AMAT4	46	44	1420	5	**49.30**	3	**1.41**	-1	6.58	521.42	496.49	86.51
L-AMAT5	45	40	1300	5	**44.62**	3	0.00	0	8.45	725.41	519.98	82.04
L-AMAT6	42	44	1320	6	**48.48**	4	**1.52**	1	9.59	517.55	496.74	89.98
L-AMAT7	47	41	1360	4	**38.24**	2	**1.47**	-1	6.85	637.87	576.48	80.47
L-AMAT8	45	49	1340	5	**40.30**	3	**2.99**	-2	9.46	813.30	962.24	98.78
L-AMAT9	45	38	1240	4	**35.48**	2	**1.61**	-1	7.46	1027.73	498.19	94.86
L-AMAT10	55	28	940	3	**19.15**	1	**12.77**	0	2.08	815.34	494.17	126.89
L-AMAT11	44	43	1420	5	**46.48**	3	**2.82**	0	5.33	1234.40	973.74	98.32
L-AMAT12	54	41	1300	5	**35.38**	3	**4.62**	-1	8.45	1354.91	1227.10	136.32
L-AMAT13	50	39	1260	5	**42.86**	3	**4.76**	-1	8.70	1336.03	447.69	82.95
L-AMAT14	42	44	1400	4	**42.86**	2	**1.43**	-1	0.00	1135.49	421.18	99.14
L-AMAT15	44	47	1460	5	**39.73**	3	**2.74**	0	3.95	1565.84	523.41	102.50
L-AMAT16	60	34	1080	4	**33.33**	2	0.00	0	8.47	1122.22	404.71	112.80
L-AMAT17	46	39	1280	4	**28.13**	2	**1.56**	-1	3.03	1310.57	413.93	107.68
L-AMAT18	49	43	1440	4	**36.11**	2	**1.39**	-1	4.00	1387.37	470.73	114.44
L-AMAT19	45	55	1500	5	**41.33**	3	**4.00**	-1	3.85	1789.55	692.63	154.64
L-AMAT20	46	39	1260	5	**39.68**	3	0.00	0	8.70	1014.27	374.36	105.45
AVG			**1,314**	**5**	**39.02**	**3**	**2.34**	**-1**	**6.01**	**1045.40**	**580.55**	**99.54**

Table 7: Computational Results on the L-AMAT set

From Table 7 we can see that the ALNS outperforms the R&R on all the 20 instances, with an average relative profit improvement of the 39.02% and an about ten times lower average computational effort. We observe that R&R always uses 3 workers while the ALNS on average employs 5 workers. This is possible thanks to the introduction of the Insertion heuristic IH4 that allows to increase the number of workers used and thus to serve more relocation requests.

The ALNS outperforms the TS on 15 instances with an average relative profit

improvement of the 2.34% and an average computational effort more than five times lower. Finally, since the average gap with the upper bound is of about 6%, we can deduce that the solutions found by the ALNS are near enough to the optimal ones also for this challenging set of instances.

8. Conclusions

In this work, we developed an Adaptive Large Neighborhood Search (ALNS) and a Tabu Search (TS) metaheuristic for solving the Electric Vehicle Relocation problem (E-VReP) in one-way carsharing systems. First of all, the solution approaches were tested on two real-like benchmark sets of instances for the E-VReP (AMAT and V-AMAT) already introduced in the literature. Moreover, to verify their performances also on challenging large size instances, we created a new set (L-AMAT) made up of 20 instances with on average 89 relocation requests. In order to evaluate the metaheuristic solution quality on this challenging set, for which the optimal solution cannot be detected by CPLEX, we also developed a bounding procedure based on a combination of two different relaxations of the MILP formulation of the E-VReP. The ALNS and TS were also compared with a previous Ruin and Recreate (R&R) metaheuristic representing the state-of-the-art solution approach for the E-VReP.

The numerical results show that on the AMAT set, both the ALNS and TS find the same results, but while the TS requires on average 4.33 seconds, the ALNS only requires 0.27 seconds. Moreover, on all the instances, except one, they detect exactly the optimal solution and outperform the R&R in three instances (with an average relative improvement of 0.71%).

On the V-AMAT set, the ALNS finds the optimum for 22 instances (over 30) and near-optimal solutions for the other instances since its average relative gap with the upper bound is only of the 0.49%. Moreover, the ALNS is able to improve on average of the 1.52% the best profit found by CPLEX within the CPU time limit. Compared to the R&R, the ALNS provides for 19 instances better profits and it never performs worse than the R&R, yielding an average

relative profit improvement of 1.79%. Compared to the TS, the ALNS provides in 11 instances better profits while only for one instance a worse profit, with an average relative profit improvement of 1.44%. Concerning the computational time, the ALNS outperforms all the other solution methods requiring on average 1.20 seconds against 6.81 seconds of the TS, 3.75 seconds of the R&R and more than $23,267$ seconds of CPLEX.

Finally, on the L-AMAT set, the ALNS outperforms the R&R for all the instances, with an average relative profit improvement of the 39.02% and an about ten times lower average computational effort. Moreover, it outperforms the TS on 15 instances with an average relative profit improvement of the 2.34% and an average computational effort more than five times lower. Finally, since the average gap with the upper bound is of about the 6%, we can conclude that the solutions found by the ALNS are near enough to the optimal ones also for this challenging set of instances.

Hence, from the experimental campaign, the proposed ALNS revealed to be by far the most effective and efficient metaheuristic to solve the E-VReP. Through a deep analysis of the removal/insertion heuristics used in the ALNS, also based on an entropy diversity measure, we obtained some empirical justifications of its effectiveness. Indeed, we detected the significant importance of the insertion heuristic $IH4$ in diversifying the solution search and thus, in helping the ALNS to escape from the local optima.

References

[1] A. Ait-Ouahmed, D. Josselin, and F. Zhou. Relocation optimization of electric cars in one-way car-sharing systems: modeling, exact solving and heuristics algorithms. *International Journal of Geographical Information Science*, 32(2):367–398, 2018.

[2] C. Archetti and M. G. Speranza. Vehicle routing in the 1-skip collection problem. *Journal of the Operational Research Society*, 55(7):717–727, 2004.

[3] R. Aringhieri, M. Bruglieri, F. Malucelli, and M. Nonato. An asymmetric vehicle routing problem arising in the collection and disposal of special waste. *Electronic notes in discrete mathematics*, 17:41–47, 2004.

[4] R. Aringhieri, M. Bruglieri, F. Malucelli, and M. Nonato. A special vehicle routing problem arising in the optimization of waste disposal: A real case. *Transportation Science*, 2017.

[5] M. Barth and M. Todd. Simulation model performance analysis of a multiple station shared vehicle system. *Transportation Research Part C: Emerging Technologies*, 7(4):237–259, 1999.

[6] M. Barth, M. Todd, and L. Xue. User-based vehicle relocation techniques for multiple-station shared-use vehicle systems. 2004.

[7] A. Blum, S. Chawla, D.R. Karger, T. Lane, A. Meyerson, and M. Minkoff. Approximation algorithms for orienteering and discounted-reward tsp. *SIAM Journal on Computing*, 37(2):653–670, 2007.

[8] B. Boyacı, K. G Zografos, and N. Geroliminis. An optimization framework for the development of efficient one-way car-sharing systems. *European Journal of Operational Research*, 240(3):718–733, 2015.

[9] B. Boyacı, K. G Zografos, and N. Geroliminis. An integrated optimization-simulation framework for vehicle and personnel relocations of electric carsharing systems with reservations. *Transportation Research Part B: Methodological*, 95:214–237, 2017.

[10] G. Brandstätter, C. Gambella, M. Leitner, E. Malaguti, F. Masini, J. Puchinger, M. Ruthmair, and D. Vigo. Overview of optimization problems in electric car-sharing system design and management. In *Dynamic Perspectives on Managerial Decision Making*, pages 441–471. Springer, 2016.

[11] O. Bräysy and M. Gendreau. Vehicle routing problem with time windows, part ii: Metaheuristics. *Transportation science*, 39(1):119–139, 2005.

[12] M. Bruglieri, A. Colorni, and A. Luè. The relocation problem for the one-way electric vehicle sharing. *Networks*, 64(4):292–305, 2014.

[13] M. Bruglieri, A. Colorni, and A. Luè. The vehicle relocation problem for the one-way electric vehicle sharing: an application to the milan case. *Procedia-Social and Behavioral Sciences*, 111:18–27, 2014.

[14] M. Bruglieri, F. Pezzella, and O. Pisacane. Heuristic algorithms for the operator-based relocation problem in one-way electric carsharing systems. *Discrete Optimization*, 23:56–80, 2017.

[15] G. Cao, L. Wang, Y. Jin, J. Yu, W. Ma, Q. Liu, A. He, and T. Fu. Determination of the vehicle relocation triggering threshold in electric car-sharing system. In *Proceedings of 2016 Chinese Intelligent Systems Conference*, pages 11–22. Springer, 2016.

[16] M. Clemente, M. P. Fanti, G. Iacobellis, M. Nolich, and W. Ukovich. A decision support system for user-based vehicle relocation in car sharing systems. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, 2017.

[17] E. Demir, T. Bektaş, and G. Laporte. An adaptive large neighborhood search heuristic for the pollution-routing problem. *European Journal of Operational Research*, 223(2):346–359, 2012.

[18] A. Febbraro, N. Sacco, and M. Saeednia. One-way carsharing: solving the relocation problem. *Transportation Research Record: Journal of the Transportation Research Board*, (2319):113–120, 2012.

[19] M. Gendreau, A. Hertz, and G. Laporte. A tabu search heuristic for the vehicle routing problem. *Management science*, 40(10):1276–1290, 1994.

[20] F. Glover. Heuristics for integer programming using surrogate constraints. *Decision sciences*, 8(1):156–166, 1977.

[21] P. Halffmann, S. O Krumke, A. Quilliot, A. K Wagler, and J-T. Wegener. On the online min-wait relocation problem. *Electronic Notes in Discrete Mathematics*, 50:281–286, 2015.

[22] F. He, Y. Yin, and J. Zhou. Deploying public charging stations for electric vehicles on urban road networks. *Transportation Research Part C: Emerging Technologies*, 60:227–240, 2015.

[23] H. Hernández-Pérez and J.-J. Salazar-González. A branch-and-cut algorithm for a traveling salesman problem with pickup and delivery. *Discrete Applied Mathematics*, 145(1):126–139, 2004.

[24] D. Jorge and G. Correia. Carsharing systems demand estimation and defined operations: a literature review. *European Journal of Transport and Infrastructure Research*, 13(3):201–220, 2013.

[25] D. Jorge, G. Correia, and C. Barnhart. Comparing optimal relocation operations with simulated relocation policies in one-way carsharing systems. *IEEE Transactions on Intelligent Transportation Systems*, 15(4):1667–1675, 2014.

[26] J. Kytöjoki, T. Nuortio, O. Bräysy, and M. Gendreau. An efficient variable neighborhood search heuristic for very large scale vehicle routing problems. *Computers & operations research*, 34(9):2743–2757, 2007.

[27] G. Laporte, R. Musmanno, and F. Vocaturo. An adaptive large neighbourhood search heuristic for the capacitated arc-routing problem with stochastic demands. *Transportation Science*, 44(1):125–135, 2010.

[28] J. Lee and G.-L. Park. Planning of relocation staff operations in electric vehicle sharing systems. In *Asian Conference on Intelligent Information and Database Systems*, pages 256–265. Springer, 2013.

[29] B. Li, D. Krushinsky, T. Van Woensel, and H. A Reijers. An adaptive large neighborhood search heuristic for the share-a-ride problem. *Computers & Operations Research*, 66:170–180, 2016.

[30] Xiangyong Li, Stephen CH Leung, and Peng Tian. A multistart adaptive memory-based tabu search algorithm for the heterogeneous fixed fleet open vehicle routing problem. *Expert Systems with Applications*, 39(1):365–374, 2012.

[31] F. Marra, G. Y. Yang, C. Traeholt, E. Larsen, C.N. Rasmussen, and S. You. Demand profile study of battery electric vehicle under different charging options. *IEEE Power and Energy Society General Meeting*, pages 1–7, 2012.

[32] L. M. Martínez, G. Correia, F. Moura, and M. Mendes Lopes. Insights into carsharing demand dynamics: Outputs of an agent-based model application to lisbon, portugal. *International Journal of Sustainable Transportation*, 11(2):148–159, 2017.

[33] M. Nourinejad and M. J Roorda. Carsharing operations policies: a comparison between one-way and two-way systems. *Transportation*, 42(3):497–518, 2015.

[34] M. Repoux, B. Boyaci, and N. Geroliminis. An event-based simulation for optimising one-way carsharing systems. In *Presentation at the 14th Swiss Transport Research Conference, Monte Verità/Ascona, Switzerland*, 2014.

[35] G. M. Ribeiro and G. Laporte. An adaptive large neighborhood search heuristic for the cumulative capacitated vehicle routing problem. *Computers & operations research*, 39(3):728–735, 2012.

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

[37] G. Santos and G. Correia. A mip model to optimize real time maintenance and relocation operations in one-way carsharing systems. *Transportation Research Procedia*, 10:384–392, 2015.

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

[39] S. Shaheen, D. Sperling, and C. Wagner. Carsharing in europe and north american: past, present, and future. *University of California Transportation Center*, 2001.

[40] P. Shaw. Using constraint programming and local search methods to solve vehicle routing problems. In *International Conference on Principles and Practice of Constraint Programming*, pages 417–431. Springer, 1998.

[41] Y. W. Wang and C. C. Lin. Locating multiple types of recharging stations for battery-powered electric vehicle transport. *Transportation Research Part E: Logistics and Transportation Review*, 58:76–87, 2013.