The Dynamic Electric Carsharing Relocation Problem

Hellem, Simen; Julsvoll, Carl Andreas; Moan, Magnus; Andersson, Henrik; Fagerholt, Kjetil; Pantuso, Giovanni

Published in:
EURO Journal on Transportation and Logistics

DOI:
10.1016/j.ejtl.2021.100055

Publication date:
2021

Document version
Publisher's PDF, also known as Version of record

Document license:
CC BY-NC-ND

Citation for published version (APA):
The Dynamic Electric Carsharing Relocation Problem

Simen Hellem\textsuperscript{a}, Carl Andreas Julsvoll\textsuperscript{a}, Magnus Moan\textsuperscript{a}, Henrik Andersson\textsuperscript{a}, Kjetil Fagerholt\textsuperscript{a}, Giovanni Pantuso\textsuperscript{b,}\textsuperscript{c,}\textsuperscript{*}

\textsuperscript{a}Department of Industrial Economics and Technology Management, Norwegian University of Science and Technology, Trondheim, Norway
\textsuperscript{b}Department of Mathematical Sciences, University of Copenhagen, Copenhagen, Denmark

\textbf{A B S T R A C T}

This article addresses a relocation and recharging problem faced by modern carsharing operators who manage a fleet of electric vehicles. As customers utilize the fleet, batteries are depleted and vehicles are possibly left in low-demand locations. Consequently, carsharing operators need to arrange the charging of depleted batteries and the relocation of poorly positioned vehicles in order to better meet the demand of the customers. Most of these activities require the intervention of dedicated staff. This article provides a framework for planning recharging and relocation activities based on periodically routing and scheduling a number of dedicated staff as a result of updated system information. The periodic planning problem is formulated as a Mixed Integer Linear Program and solved in a rolling-horizon fashion. For the solution of the problem a fast Adaptive Large Neighborhood Search heuristic is proposed. Tests based on data for the city of Oslo show that the heuristic can deliver, in reasonable computational time, high quality solutions for instances compatible with real-life planning problems.

\textbf{1. Introduction}

Carsharing systems, which have existed in various forms for several decades, have recently gained traction due to the enabling power of internet technology and the increased awareness regarding environmental issues. A carsharing system is owned and maintained by a Carsharing Organization (CSO). First-time users typically sign up through a website or a mobile application to get access to the system, possibly paying a subscription fee. Users already in the system can then locate, possibly reserve, and unlock the available cars typically via a mobile application, and pay based on the time of usage (e.g., a per-minute fee), sometimes in addition to a drop-off fee based on the zone of the city where the car is returned (Shaheen et al., 2015; Hansen and Pantuso, 2018; Pantuso, 2020). Modern carsharing services mainly exist in two forms. Station-based systems restrict users to pick up and return cars at available stations. These can be further distinguished into one- and two-way systems, which, respectively allow and forbid the user to return the car to a station different from the pick up station. Free-floating systems do not necessarily include stations and cars can be picked up and returned at any common parking spot within the specified business area.

Modern carsharing systems give rise to new and unexplored planning problems which are attracting the interest of the operations research community. At different strategic levels, CSOs need to decide, for example, fleet size (George and Xia, 2011; Cepolina and Farina, 2012), station locations (Weikl and Bogenberger, 2012), trip-booking scheme (Correia et al., 2014; Kaspi et al., 2014), and pricing scheme (Hansen and Pantuso, 2018; Pantuso, 2020). At the operational level, CSOs deal with recharging/refueling, maintenance and, particularly in one-way systems, with relocating vehicles in order to better meet transportation demand. In fact, asymmetric patterns in transportation demand cause cars to remain parked in low-demand zones with consequent under-supply in high-demand zones. In some cases, this phenomenon is contrasted with pricing-based initiatives. As an example, in the city of Milan, the CSO Share-Now adopts a pricing scheme which charges users for parking in unfavorable zones of the city, Share-Now (2021a). However, staff-based relocation of cars is often unavoidable. This is particularly true for free-floating systems operating a fleet of (at least some) electric vehicles (example of these are Share-Now in Copenhagen, Share-Now (2021b) and Vy Bybil in Oslo, Vy (2021)). In fact, since users are not required to return cars at charging stations, the CSO’s staff often needs to ensure recharging.

In this paper we introduce the Dynamic Electric Carsharing Relocation Problem (DE-CRP) for one-way, either free-floating or station-based, systems. A solution to the problem consists of: (i) an assignment of cars with low battery level to charging stations, (ii) an assignment of cars in need for relocation to under-supplied zones/stations, (iii) an assignment of car-moves to employees, and (iv) routes and schedules...
for the service employees. The problem is dynamic in the sense that new information about the distribution of cars and their state of charge is received continuously. In addition, transportation demand changes in a stochastic way during the business hours or planning horizon. The overall goal of the problem is to maximize profits by providing a suitable distribution of vehicles. This is consistent with the business objective of private CSOs.

The contributions of this paper can be stated as follows.

- First, we formalize the DE-CRP and offer a rolling-horizon solution procedure, based on periodic on-line re-optimization of charging and relocation activities as the landscape of the problem changes during the planning horizon.
- Second, we provide a Mixed Integer Programming (MIP) formulation for the periodic re-optimization problem. Compared with the available literature, the models take a different approach with respect to modeling relocation tasks. Particularly, it is based on a-priori generated possible car-moves, which are then assigned to available employees. Preliminary results from Hellem et al. (2018) show that this formulation is superior to a traditional arc-flow formulation. The re-optimization problem can be seen as a subproblem of the DE-CRP and is referred to as the Electric Carsharing Relocation Problem (E-CRP).
- Third, we provide an efficient solution method based on an Adaptive Large Neighborhood Search (ALNS) for the re-optimization problem which is able to solve large-scale instances of the problem within reasonable computational time.
- Finally, we test the proposed solution method in a simulation framework based on real-time traffic data from Oslo, Norway.

The methodology can adapt to both station-based and free-floating systems. In the latter case a discretization of the business area into zones is necessary, as exemplified in our case study. The remainder of this paper is organized as follows. In Section 2, the DE-CRP is discussed in conjunction with the related literature. In Section 3, the DE-CRP is formally introduced. A solution method for the DE-CRP, based on periodic re-optimization of relocation activities is presented in Section 4. Section 5 introduces the Electric Carsharing Relocation Problem (E-CRP) and the ALNS heuristic for solving it. The simulation used to test the solution method as well as the test instances are described in Section 6. A computational study in presented in Section 7 and, finally, conclusions are drawn in Section 8. The Appendix contains a Mixed Integer Linear Programming formulation of the E-CRP.

2. Literature overview

A variety of strategies have been proposed to address the relocation of vehicles. The great majority of the studies focus on station-based systems. Barth and Todd (1999) examine a station-based electric carsharing system through a discrete-event simulation model which includes a number of heuristic algorithms to determine when and how a relocation must happen (i.e., how many vehicles to move from a station to another). Kek et al. (2009) also consider staff-based relocations in a simulation model. In addition, the authors propose an optimization model which allocates staff to relocation activities. The scope of the optimization model is to minimize the total relocation cost. Jorge et al. (2014) propose an optimization model to determine the number of cars to relocate between pairs of stations. The model is also tested in a simulation framework. Boyaci et al. (2015) take into account staff-based relocations in an optimization model that determines the optimal fleet size, number of stations, and their locations in one-way station- and reservation-based car-sharing systems. Boyaci et al. (2017) propose a multi-objective MIP to determine the optimal temporal and spatial distribution of vehicles in stations and also the personnel responsible for the relocation. Nair and Miller-Hooks (2011) propose a stochastic MIP involving joint chance constraints which generates least-cost relocation plans such that a proportion of all short-term demand is met. Brandstätter et al. (2016) provide a broader overview of the methods available for station-based systems, such as the problem of finding optimal locations and sizes for charging stations as studied in Brandstätter et al. (2020).

A number of studies add a further level of planning detail, and consider also the routing of the relocation staff. Bruglieri et al. (2014) consider the relocation problem for a fleet of electric vehicles in one-way station-based systems. The authors propose the use of staff travel by means of folding bicycles that can be loaded into the trunk of the electric vehicle to relocate. The authors refer to this problem as the Electric Vehicle Relocation Problem (E-VReP). A solution to the E-VReP provides the routing and scheduling of each worker employed. Bruglieri et al. (2017) expand the E-VReP by introducing the costs related to repositioning staff and the revenue associated with each relocation request satisfied, and thus seek to maximize the total profit. Recently, Bruglieri et al. (2019) propose an Adaptive Large Neighborhood Search heuristic to solve large instances of the problem. Gambella et al. (2018) present two models for the relocation problem, including staff routing, one during operating hours maximizing the profit of relocating cars, and one for non-operating hours maximizing the level of the most depleted battery. Ait-Ouahmed et al. (2018) also consider the joint routing of staff and vehicles, and propose a Tabu Search heuristic that first considers only relocations of cars to meet the demand, and then assigns service employees to the relocations found in the first phase. Finally, Wang et al. (2019) propose a method to determine the number of vehicles needed at each station, the relocations to perform accordingly, and how relocations are allocated to the available staff. The determination of the necessary vehicle balance at each charging station is based on historical data and on the computation of a threshold which ensures that the probability of the station running out of cars in a given time horizon is sufficiently low. Following, an optimization method determines which relocations to perform and how these are allocated to the available staff.

The literature concerning free-floating systems is more sparse. Kortum and Machemehl (2012) propose a procedure for the relocation of cars. After an initial allocation, vehicles are moved from one zone to another according to relative levels of demand. The procedure stops when there is no unmet demand in the entire system or when the vehicles end in a zone with no demand to carry it into another zone. Weikl and Bogenberger (2015) introduce a relocation model for systems with both conventional and electric vehicles. In case of imbalances, the model is able to recommend profit-maximizing car relocations. Relocations are combined with the unplugging and recharging of electric vehicles and the refueling of conventional vehicles. Both Weikl and Bogenberger (2015) and Kortum and Machemehl (2012) partition the operating area into zones, which basically transforms the system into a station-based system.

Our work shares similarities with available studies. For example, similarly to Bruglieri et al. (2014, 2017, 2019), we assume that service employees travel by folding bikes or public transport in-between relocating cars and, similarly to Weikl and Bogenberger (2015), we combine recharging and maintenance activities with relocation. In addition, in this paper we advance the state-of-the-art by means of the following additions. First, we simultaneously address both the routing of the employees (as in, e.g., Bruglieri et al. (2014), Gambella et al. (2018), Ait-Ouahmed et al. (2018) and Wang et al. (2019)) and joint relocation and recharging decisions (as in Weikl and Bogenberger (2015)). The state-of-charge of the vehicles is also taken into account in Wang et al. (2019). The authors ensure that the relocation moves are feasible with respect to the state-of-charge. However, the authors do not address recharging decisions. In addition, the method we propose is tailored for free-floating systems, and as such does not require the detailed information at the station-level used in the method proposed by Wang et al. (2019), but rather information at the level of geographical zones. Second, we propose an alternative formulation based on a-priori defined car-moves. In contrast, in Bruglieri et al. (2019) the
problem is based on pick-up and delivery requests, and in Ai{t-Ouahmed et al. (2018) solutions are represented using customer demands and the corresponding relocations needed to fulfill them. The formulation based on car-moves can potentially reduce the search space and offer better scalability. Finally, we propose a framework for explicitly addressing a dynamic problem where new information is received continuously through the planning horizon, providing a closer representation of the actual decision process of real-world carsharing operators.

The DE-CRP shares characteristics with Dynamic Vehicle Routing Problems (DVRP). In the definition of Psaraftis et al. (2016) a VRP is the actual decision process of real-world carsharing operators. Finally, we propose a framework for explicitly addressing scalability. Particularly, Ulmer et al. (2017) present a Markov Decision Process (MDP) framework for DVRPs which generalizes the previous work of Thomas (2007) and Seconmandi and Margot (2009). They show how a route-based MDP can be used to model a DVRP while at the same time being closely coupled with solution methods that optimize iteratively. However, MDP models suffer excessively the curse of dimensionality, as the state-space tends to become too large for real-life-size instances. In fact, most solution methods solve the DVRP using periodic replanning in a rolling-horizon framework. In this case, a static vehicle routing problem (VRP) is used for periodically replanning over portions of the planning horizon as input data is updated. As an example, Chen and Xu (2006) solve a DVRP with hard time windows using fixed intervals between consecutive replanning. In Kilby et al. (1998) replanning is triggered when new demand arrives. Yang et al. (2002) show how the framework can be used in combination with a variety of solution methods for the underlying static VRP, including both heuristics and exact optimization methods. In this paper we also propose a solution method based on periodic replanning for portions of the planning horizon.

3. Problem description

In the Dynamic Electric Carsharing Relocation Problem (DE-CRP), we consider a CSO managing a fleet of electric cars in a car-sharing service over the entire day (24 h). The positions and charging states of the cars change throughout the day as a result of the users’ driving activities. The demand of shared cars in the different areas of the city also varies over the day. An area of the city may correspond with a well-defined geographical zone in free-floating systems (as assumed in the rest of this article) or with a specific station in station-based systems. To ensure continuity of the service and a profitable distribution of cars, the CSO needs to charge cars with too low battery levels, perform necessary maintenance, and relocate cars in order to better meet demand. For these activities, the CSO uses dedicated service employees.

The service employees use public transport or folding bikes that can fit in the trunk of a car to reach cars subject to relocation. Cars subject to relocation are those in areas with an excess of available cars, or those in need of charging. Upon reaching a car, the employee performs necessary small maintenance tasks and then moves the car, corresponding to driving it to a charging station if its battery level is below a given threshold, or relocating it to a deficit area that has fewer cars than needed at the given time of the day. Once the car has been moved, either to a charging station and/or to a deficit area, the employee travels to another car in need of intervention.

The decision process concerns the service employees. When an employee arrives at a car, a decision is made about where it should be relocated, and when the car arrives at its destination, the decision is to which car the employee should go to next.

Therefore, we define the DE-CRP as the problem of determining:

(i) the assignment of cars in need of charging to available charging stations,
(ii) the assignment of cars in areas with an excess of available cars to deficit areas,
(iii) the assignment of employees to car-moves, that is the relocation of cars from their current position to their assigned charging stations or to deficit areas, and
(iv) routes and schedules for the activities of the service employees. A route consists of relocations of cars and travels between relocation activities. A relocation plan, consisting of the above mentioned decisions, is required for the entire day, and activities are performed as the system is perturbed by user activities. The scope of the CSO is that of performing these activities such that profits are maximized, where revenue consists of the remuneration for rentals, and costs include the cost of the movements with rental cars, tolls and wear. It should be emphasized that in contrast to most other vehicle routing and pickup and delivery problems, the DE-CRP also includes determining where to relocate the different cars.

4. Solution method for the DE-CRP

In practice, CSOs face the problem by periodically planning relocation and recharging activities throughout the day with updated information and demand outlook as the system is perturbed by user activities. Therefore, we adopt this organization of work and set to solve the DE-CRP by periodically re-optimizing relocation and recharging activities at a finite number of time points referred to as decision stages.

Let the system state at a given decision stage describe the current position and battery level of each car not currently in use, the position of each service employee, and the travel times for rental cars, public transport and folding bikes. At each decision stage, decisions regarding a portion of the whole planning horizon, referred to as planning period, are made. Such decisions are based on the current system state and on demand forecast for the planning period as well as a period of time after the planning period referred to as look-ahead period. Particularly, at each decision stage, a static and open subproblem is solved. We refer to this problem as the Electric Carsharing Relocation Problem (E-CRP).

The problem is static since we assume that all information is known at the time of planning and it is open since there is no defined depot where the service employees must start and end their routes.

Given a solution to the E-CRP, relocations and charging activities for the first part of the planning period are implemented accordingly, while activities further ahead in the future are planned at a future decision stage (they are only included to avoid myopic solutions for the here-and-now decisions in the first part of the planning period). It should be emphasized that the look-ahead period is only used for forecasting the ideal state at the end of the planning period. The resulting rolling-horizon framework is illustrated in Fig. 1.

In the remainder of the article we assume a free-floating system. As in Weikl and Bogenberger (2015) and Folkestad et al. (2020), the business area of the CSO is divided into zones. A parking zone is a geographical area where rental cars can be parked and picked up by the customers. Multiple cars can be located within a parking zone. The position of each parked car is tracked and used when calculating the driving time to the charging stations and the center of the deficit zones.

A parking zone is characterized by an ideal state which indicates how many sufficiently charged cars should be located in the zone at the end of the planning period to satisfy future demand. Since the replanning of relocation and recharging activities is done frequently, the uncertainty in demand is not explicitly handled in the model but instead reflected in the ideal state.

Each charging station has a finite capacity. Only cars currently in need of charging can be parked at a charging station. A charging station is located inside a parking zone. However, they are considered separated entities, as shown in Fig. 2.

We make the following assumptions. There are always available parking spaces in parking zones. When cars are fully charged, they are automatically made available to customers in the surrounding parking zone, unassisted by service employees. This corresponds to mark the car as available in the booking system when it is fully charged, and it allows customers to pick up the car directly from charging station (with the
Fig. 1. Solution of the DE-CRP by period re-optimization.

Fig. 2. Location of a charging station. The charging station is separated from its associated parking zone.

5. An adaptive large neighborhood search heuristic for the E-CRP

Preliminary testing of the model (summarized in Appendix) using Xpress 29.01.10 run on a 3.4 GHz Intel E5 processor showed that instances with more than ten zones and ten cars could not be solved in a reasonable time, see Fig. 3. We can therefore conclude that solving real-life instances of problem (4) in reasonable time using only a commercial solver is in practice impossible. Instead, we propose an Adaptive Large Neighborhood Search (ALNS) heuristic, as introduced by Ropke and Pisinger (2006), which has proven to be efficient for solving large-scale vehicle routing problems. Similarly to Ropke and Pisinger (2006), Shaw removal and k-regret are utilized for the Large Neighborhood Search (LNS), while Tabu Search (TS) is adopted as the local search.

The heuristic is divided into two recurring processes, TS and LNS. The TS performs a local search until \( I_{des} \) iterations without improvements have been performed. Then, the LNS destroys and repairs the solution provided by TS, guiding the search into a new neighborhood of the search space in which TS is reactivated. The algorithm terminates after \( I_{max} \) LNS iterations without improvement or \( T_{max} \) seconds (when the first of the two conditions is met). The pseudo-code of the ALNS heuristic is provided in Algorithm 1. After the construction of an initial solution, TS performs a local search in a neighborhood \( M \) provided by the function \( FindNeighborhood \). The neighborhood consists of all the solutions which can be obtained by altering the current solution using one or several of the available local search operators (LSOs). The heuristic chooses the best solution in the given neighborhood. If the solution improves the current best solution, it is updated. Here \( f(s) \) denotes the objective function value of solution \( s \). Otherwise, after
Algorithm 1: Adaptive Large Neighborhood Search Heuristic

Input: R Set of candidate car-moves
Output: Ordered list of car-moves for each service employee
1 Solution s = ConstructionHeuristic(R)
2 Best solution \( s_{best} = s \)
3 while stopping criteria not met do
4 \( B = \text{Find Neighborhood}(s) \)
5 \( s \in \arg \max_{s \in B} f(s) \)
6 if \( f(s) > f(s_{best}) \) then
7 \( s_{best} = s \)
8 else if non-improving TS iterations \( \geq I^{ts} \) then
9 \( s = \text{Large Neighborhood Search}(s) \)
10 end
11 UpdateWeights()
12 end

5.1. Solution representation

The key entity of the solution representation is a car-move, which define a feasible relocation of a car from its origin to a given destination. This means to relocate a sufficiently charged car from an excess zone to a deficit zone, a parking-move, or a car in need of charging to a charging station, a charging-move. Let R be the set of all feasible car-moves. The set of car-moves is created from the current state of the carsharing system.

A solution \( s \) is represented by two lists, \( \gamma \) and \( \beta \). The first list, \( \gamma \), contains the used car-moves and is divided into one list for each service employee. Let \( K \) denote the set of service employees and \( \gamma_k \) the ordered list of car-moves performed by service employee \( k \). The second list, \( \beta \), contains the unused car-moves, not present in \( \gamma \). At most one car-move for each car may be present in \( \gamma \). The route of service employee \( k \) is derived by iteratively visiting the origin and destination of each car-move \( r \in \gamma_k \). Fig. 4 shows an example of this. Two service employees relocate five cars. Employee 1 relocates cars 1, 2, and 3 with destinations 2, 3, and 5, respectively, while employee 2 relocates cars 1 and 3 with destinations 4 and 5 with destinations 2 and 5.

5.2. Feasibility and objective function

The ALNS allows infeasible solutions during the search to widen the search space. Two types of violations are allowed. First, \( \gamma_k \) may contain more car-moves than service employee \( k \) can handle within the planning period \( T \). In this case, a solution is punished at a cost \( C^L \) for each car-move that is handled outside the planning period. The ordered list of car-moves performed by employee \( k \), \( \gamma_k \), can easily be made feasible by moving the car-moves outside the planning period from \( \gamma_k \) to \( \beta \). Second, the capacity of charging stations may be violated. Such violation is punished at a cost \( C^I \) for each car in excess of capacity.

The objective value is calculated by means of Eq. (1). The number of parking-moves that are rewarded is denoted \( r^p \), and the reward per move is \( C^P \). The variable \( \phi \) is the number of charging-moves performed within the planning period, and \( C^{C,R} \) is the reward per move. The total time used by service employee \( k \) is denoted \( t_k \), and idle time and overtime is punished by \( C^E \) and \( C^{ET} \) in the third and fourth terms, respectively. The idle time cost is introduced to address that we deal with only a portion of the entire planning horizon in the E-CRP. Idle time costs encourage employees to complete their tasks as soon as possible, and thus leave the company in a better position to address the next re-optimization. The variable \( \mu_k \) is 1 if car-move \( r \) is performed, and 0 otherwise, and \( C^{CH} \) is the cost of wear, tolls and electricity for car-move \( r \) where \( C^R \) is the cost per time unit and \( T^H \) the time needed to perform the car-move. Note that \( (F)r^+ \) is short for \( max(F,0) \).

The terms on the second line of (1) are penalties for infeasibility and rewards for early charging. The variable \( r^C \) denotes the total capacity violation at all charging stations, while the variable \( \mu_k^C \) is 1 if car-move \( r \) is performed after the planning period, and 0 otherwise. To test an early charging strategy, we also include a revenue for charging early, where \( \mu_k \) is the time charging-move \( r \) is performed and \( \mu_k^E = \mu_k \) if it is not performed. \( C^{CHE} \) is the reward per unit of time and \( R^C \) is the set of charging moves.

Thus the heuristic prioritizes early charging moves as these may be beneficial in a dynamic setting to reduce the number of cars that become unavailable later. Notice that the last term of (1) is not included in the objective function of model (4).

\[
\begin{align*}
    f(s) = & C^L \sum r^p + C^C \sum \phi - \sum_{r \in \gamma_k} C^R (T - t_k)^+ - \sum_{r \in \gamma_k} C^{ET} (T - t_k)^+ \\
    & - \sum_{r \in \gamma_k} C^{CH} \mu_k - C^E \sum_{r \in \gamma_k} C^{CHE} \mu_k^E + \sum_{r \in R^C} C^{CHE} (T - t_k)^+ \\
\end{align*}
\]

5.3. Construction of the initial solution

An initial solution is created in a greedy fashion. Initially, \( \beta \) contains all car-moves and \( \gamma \) is empty. The heuristic iterates through the service employees, thus the employees are handled one at a time. For each employee, \( k \), the best insertion (in terms of objective value) of a car-move at the end of \( T_k \) is performed, given that the corresponding car \( c \) does not yet have a car-move in \( \gamma \). The remaining car-moves for car \( c \) remain in \( \beta \). Tasks are added to one employee until no more task improving the objective function are found. The heuristic then continues in the same way with the next employee, each time adding a move at the end of a given \( T_k \).

5.4. Local neighborhood search

In each iteration, the Tabu Search generates a local neighborhood \( B \), using a chosen LSO. The LSOs that are available are called Intra, Inter, Inter-2, and Swap. The Intra LSO moves a car-move within the list of car-moves for one given service employee, while the Inter LSO moves a car-move from one service employee to another. The Inter-2 LSO moves two consecutive car-moves from one service employee to another, while the Swap LSO swaps two car-moves between two service employees. We also use similar LSOs where car-moves to/from \( \beta \) (i.e. the list of unused car-moves) are moved from/to \( \gamma_k \) (i.e. a route for a given service employee), and a car-move from \( \gamma_k \) is replaced with a car-move from \( \beta \) for the same/different car.

Different ways of generating the neighborhood \( B \) were tested and preliminary testing showed that a Random Weighted Enumeration method gave the best results. Here, the TS first selects one LSO in
a roulette wheel fashion, based on adaptive weights (described in Section 5.6). Second, a neighborhood with $M_{\text{max}}$ solutions is generated randomly with the selected LSO. We use a best improvement strategy when searching in $M$ for an improving solution, meaning that a most improving neighbor is selected (note that there might be multiple ones). If there does not exist any improving neighbor solutions, a neighbor that worsens the solution the least is chosen. The selected LSO is added to the tabu list. The tabu list is adaptive, limited by an upper and lower threshold. If the last $I^b$ iterations have been unsuccessful in finding a local improvement, the length of the tabu list is doubled. Likewise, if at least one of the previous $I^s$ iterations has been successful, the length of the tabu list is halved.

5.5. Large neighborhood search

The large neighborhood search consists of combinations of destroy and repair heuristics. The destroy heuristics remove car-moves from $\gamma$. Subsequently, repair heuristics insert car-moves into $\gamma$. The degree in which a current solution is destroyed and repaired is denoted $T$, $T = 0.1$ means that 10% of the car-moves in $\gamma$ are removed. The destroy and repair heuristics are chosen in a roulette wheel fashion, individually, based on adaptive weights.

The destroy heuristics are Random Removal, Worst Removal and Shaw Removal. Random Removal sequentially removes car-moves randomly and uniformly from $\gamma$, to diversify the search. Worst Removal greedily removes the car-moves causing the largest decrease in the objective function value from the current solution $\gamma$. The intention is that more beneficial car-moves can replace these car-moves. Shaw removal was first introduced by Shaw (1997). The technique increases $\gamma$ is that more beneficial car-moves can replace these car-moves. Shaw removal was first introduced by Shaw (1997). The technique increases

$$
\Delta R = |c(1) - c(2)|
$$

The first and second terms consider the relatedness between car-moves’ origin and destination, respectively. The third term checks if both car-moves are charging-moves or parking-moves, while the two final terms compare handling time and start time, respectively. The parameters $\omega_1, \ldots, \omega_5$ weight the importance of each of the five measures. The lower the values of $R(r_1, r_2)$, the more related the two car-moves are. Initially, a random car-move is chosen from $\gamma$ and inserted into a list of removed car-moves. While keeping track of the car-moves that are already removed, a random car-move from $r_1$ is then chosen. The car-move in $\gamma$ most similar to $r_1$ according to Eq. (2) is then removed and placed in the list of removed car-moves. This process repeats until a proportion $\Gamma$ of the car-moves in $\gamma$ is removed.

The repair heuristics are Greedy Insertion and Regret Insertion. Greedy Insertion greedily inserts car-moves yielding the greatest improvement to the objective function value. The Regret Insertion is similar to the $k$-Regret used in Ropke and Pisinger (2006). The heuristic considers the alternative costs of inserting a car-move into $y$ by comparing the objective function value of the best insertion with the $k$ best insertions, favoring the car-move with the largest difference. We have used both the 2-Regret and the 3-Regret Insertions heuristics.

5.6. Adaptive weights adjustments

Adaptive weights guide both the TS and LNS. Each LSO $q$ in the TS, as well as the destroy and repair heuristic in the LNS, has a weight $w_q$ associated with it, which is updated based on its performance once in every segment of iterations, similar to Ropke and Pisinger (2006). A segment for the LSO used in the TS consists of $I^w$ consecutive iterations. Similarly, a segment for the destroy and repair heuristics used in the LNS consists of minimum $I^{leq}$ iterations.

Eq. (3) shows how the weights $w_q$ for all LSOs are updated. $\theta_q$ is the number of times LSO $q$ have been used in the last segment, while $a$ is a parameter that controls the degree for which weights are updated. $\mu_q$ is the accumulated score in the current segment based on its performance, similar to Ropke and Pisinger (2006). In the special case where $\mu_q$ and $\theta_q$ are both zero, the last term in Eq. (3) is set to zero. The weights for the LNS heuristics are updated in a similar way.

$$
w_q = w_q(1 - a) + a \frac{\mu_q}{\theta_q}
$$

(3)

6. Simulation, implementation and test instances

We test the performance of the periodic re-optimization framework for the DE-CRP through simulation within a Rolling Horizon framework, which is described in Section 6.1. Section 6.2 presents the hardware and software used as well as the test instances which are based on real traffic data from the city of Oslo.

6.1. Simulation environment

The simulation environment consists of three components, as illustrated in Fig. 5. The E-CRP Solver finds solutions to the E-CRPs using the ALNS heuristic. The E-CRP is solved periodically given the current system state and demand forecast for the planning period and look-ahead period. The Customer Demand component provides both predicted and realized customer demand. Finally, the Simulation Model, which is the core of the simulation environment, simulates the real-life system by keeping track of the evolution of the system state as demand materializes and service employees move around the city to relocate and recharge the cars. Fig. 5 illustrates the connection between the three components. The Simulation Model feeds the current system state to the E-CRP Solver. After the E-CRP Solver is done, it returns the routes for each service employee and the relocations to the Simulation Model. The Simulation Model simulates both the travels of the employees and the realized customer demand.

Let parameters $T_{\text{start}}$ and $T_{\text{end}}$ represent the start and end time of the planning horizon of the DE-CRP, i.e the total time which we simulate over. Let $T_{\text{decision}}$ represent the frequency of the decision stage, that is how often replanning is performed by calling the E-CRP solver (i.e., the ALNS heuristic). Let $T$ represent the length of the planning period when solving the E-CRP, and let the look-ahead period have the same
length as the planning period. Let $T_{\text{charge}}$ specify the time (in minutes) it takes to fully charge a rental car with an empty battery. Let $T_{\text{range}}$ be the time a fully charged rental car can drive. Let cars with battery level below the threshold $\zeta_{\text{upper}}$ be considered for recharging. Customers can still rent cars with battery levels between $\zeta_{\text{upper}}$ and $\zeta_{\text{lower}}$ as the battery level is sufficient for shorter trips. Rental cars with battery levels below $\zeta_{\text{lower}}$ are not available to customers. The pseudo-code for the Simulation Model is shown in Algorithm 2. The simulation is run after every decision stage in the Rolling Horizon framework. Parameters such as $T_{\text{start}}$ and $T_{\text{increment}}$ are fed to the Simulation Model specifying the start and the duration of the period to simulate within each pair of decision stages. The Simulation Model divides events into departures and arrivals. Departures consist of potential relocations and customer requests, while arrivals include relocations and customer rentals that are performed within the simulation time.

**Algorithm 2: Simulation Model**

```
Input: $T_{\text{start}}$, $T_{\text{increment}}$, CustomerArrivals, EmployeeArrivals, EmployeeRoutes
Output: System state
1 CustomerRequests = CustomerDemand.getActualDemand($T_{\text{start}}$, $T_{\text{start}} + T_{\text{increment}}$)
2 NextEvent = findNextEvent()
3 $t ←$ NextEvent.getTime()
4 while $t < T_{\text{start}} + T_{\text{increment}}$ do
5   System state, CustomerArrivals, EmployeeArrivals = doEvent(NextEvent)
6   NextEvent ← findNextEventAfter($t$)
7   updateBatteryLevels($t$, min(NextEvent.getTime(), $T_{\text{start}} + T_{\text{increment}}$))
8   $t ←$ NextEvent.getTime()
9 end
```

Lines 4–9 in Algorithm 2 show the simulation of tasks from $T_{\text{start}}$ until the end of the simulation period. The variable $t$ is used to track the start time of the previous event. Repeatedly, the Simulation Model finds the next event to happen after the time $t$. This is a simple process of finding the earliest arrival or departure of service employees and customers. Another event which may occur is that a rental car finishes charging. In this case, the fully charged car is moved from its charging node to the associated parking node. Every time a new task is approved, the battery levels of the cars are updated in line 7.

6.2. Test instances and implementation

We have generated test instances based on the geographical layout and on historical traffic flow from the city of Oslo, including the city center and surrounding suburban areas. The nodes are created using a grid structure, defining each node as a square of size 500 $\times$ 300 meters. Each node represents a parking node and charging stations are uniformly distributed in the operating area to the available parking nodes. A subset of the 225 nodes in Fig. 6 defines each test instance. Travel time data for car, bike, and public transport is collected from Google Maps. The travel time between two nodes is defined by the fastest travel option available, car for charging and parking moves and bike or public transport for the transportation between car-moves.

There are usually many parking slots along the streets and many parking garages within the studied area (and in particular for electric cars). We therefore believe that the assumption that there is always available parking spaces is valid.

Three instance classes are created for the DE-ERP as shown in Table 1. Common for all instance classes is that all charging stations have a capacity of six charging cars, and that there are approximately three times as many cars as nodes. The simulation is done over a 12-hour period starting at 6 AM in the morning. This entails solving several different instances for each instance class, where each instance represents a snapshot of the carsharing system at a given re-optimization time, as explained below. On average, 22.5 cars are requested in each node during the 12-hour period. This implies that there are 6–7 times more customer requests than cars in the system.

For each instance in each instance class, ideal states are generated based on historical traffic flow patterns in the city of Oslo, obtained from Google Maps and assuming that customers renting cars are most likely to follow the traffic flow pattern. In the morning, traffic flows from the suburban areas into the city center. These flows decrease towards noon. From noon until 3–4 PM, the traffic from the city center to the suburban areas gradually increases with a rush hour peak around 4 PM. These findings led to a simple three-folded categorization of nodes: nodes with morning rush and lower demand in the afternoon, nodes with a steady and moderate level of demand during the entire planning horizon, and nodes with low morning demand but high afternoon demand. Consequently, the demand in each node, corresponding to the ideal state, is assumed to follow a Poisson process with arrival rate changing during the day. The arrival rate is indicated by parameter $\lambda_t$, $s \in \{H, M, L\}$ as reported in Table 2. For instance, this means that nodes with morning rush have a rate of $\lambda_H$ in the morning which linearly decreases towards $\lambda_L$ in the afternoon. For simplicity, we have assumed that customers always travel at least for ten minutes. To this, the travel time between the departure and destination node is added. Since customers may have errands to run, each travel time of customers is adjusted by a factor drawn from a uniform distribution $U \sim \text{unif}(1, 1.4)$.

The parameters common to all test instances are shown in Table 3. At $T_{\text{start}} = 6$ AM, it is assumed that the distribution of available rental cars is close to the ideal state. The initial battery levels of the cars are uniformly distributed. The cost parameters from Section 5 are inherently dependent on the specific CSO, e.g., the expected profit for an individual available shared car and the cost of wear, tolls and employees. In absence of a focal real-case, in our tests the cost parameters have the following values: $C^C = 10, C^CE = 30, C^T = 0.01, C^{FT} = 0.5, C^R = 0.2, C^{1} = 100, C^{L} = 10, C^{CBE} = 0.1$. These values have been chosen based on two principles. First, the relative size of each cost component should reflect the importance of each cost. Secondly, each cost should incorporate its value in a dynamic long-term environment, e.g., the benefit of charging a car is not observable directly, but is beneficial when simulating an entire day.

The final ALNS parameters are reported in Table 4 and based on comprehensive testing performed by Hellem et al. (2018). In addition, Hellem et al. (2018) show that the ALNS heuristic produces high quality solutions to the E-ERP for instances where the commercial software Xpress fails. Furthermore, Hellem et al. (2018) show that, compared with a greedy construction heuristic, the fully calibrated ALNS heuristic finds solutions on average 45.1% closer to the best-known.
Table 3
Parameters used in the Rolling Horizon framework and the Simulation model.

<table>
<thead>
<tr>
<th>Notation</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Start time business hours</td>
<td>(T_{\text{start}})</td>
</tr>
<tr>
<td>End time business hours</td>
<td>(T_{\text{end}})</td>
</tr>
<tr>
<td>Time increments</td>
<td>(T_{\text{increment}})</td>
</tr>
<tr>
<td>Planning period</td>
<td>(T)</td>
</tr>
<tr>
<td>Overtime</td>
<td>(T_{\text{overtime}})</td>
</tr>
<tr>
<td>Charging time</td>
<td>(T_{\text{charge}})</td>
</tr>
<tr>
<td>Car range</td>
<td>(T_{\text{range}})</td>
</tr>
<tr>
<td>Upper battery threshold</td>
<td>(\zeta_{\text{upper}})</td>
</tr>
<tr>
<td>Lower battery threshold</td>
<td>(\zeta_{\text{lower}})</td>
</tr>
</tbody>
</table>

Table 4
ALNS: Final parameter values.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>(T_{\text{MAX}})</td>
<td>180 s</td>
<td>Max running time (seconds)</td>
</tr>
<tr>
<td>(B_{\text{INIT}})</td>
<td>2</td>
<td>Initial tabu list size</td>
</tr>
<tr>
<td>(B_{\text{MIN}})</td>
<td>2</td>
<td>Minimal tabu list size</td>
</tr>
<tr>
<td>(B_{\text{MAX}})</td>
<td>1024</td>
<td>Maximal tabu list size</td>
</tr>
<tr>
<td>(I_{\text{K}})</td>
<td>125000</td>
<td>Max number of iteration without improvement</td>
</tr>
<tr>
<td>(I_{\text{W}})</td>
<td>5 ln</td>
<td>The number of iterations before the LSO weights are updated</td>
</tr>
<tr>
<td>(I_{\text{DES}})</td>
<td>120 ln</td>
<td>Iterations without global improvement before destroy and repair</td>
</tr>
<tr>
<td>(I_{\text{F}})</td>
<td>6</td>
<td>Iterations without local improvement before increasing the tabu list size</td>
</tr>
<tr>
<td>(I_{\text{S}})</td>
<td>3</td>
<td>Iterations with local improvements before decreasing the tabu list size</td>
</tr>
<tr>
<td>(M_{\text{MAX}})</td>
<td>25 ln (</td>
<td>C</td>
</tr>
<tr>
<td>(\Gamma)</td>
<td>0.4</td>
<td>The destroy/repair factor</td>
</tr>
<tr>
<td>(R_{0})</td>
<td>1</td>
<td>LSO score for finding a new local solution</td>
</tr>
<tr>
<td>(R_{0})</td>
<td>23</td>
<td>LSO score for finding a new global best solution</td>
</tr>
<tr>
<td>(R_{1})</td>
<td>13</td>
<td>LSO score for finding a new and better local solution</td>
</tr>
<tr>
<td>(R_{1})</td>
<td>23</td>
<td>Destroy and repair score for finding a better local solution</td>
</tr>
<tr>
<td>(R_{2})</td>
<td>13</td>
<td>Destroy and repair score for finding a new and better global solution</td>
</tr>
<tr>
<td>(\alpha)</td>
<td>0.1</td>
<td>Update factor for both LSO and repair and destroy weights</td>
</tr>
<tr>
<td>(\omega_1 \ldots \omega_5)</td>
<td>0.315, 0.315, 0.315, 0.005, 0.05</td>
<td>Weights for Shaw Removal</td>
</tr>
</tbody>
</table>

The hardware and software used to implement and test the solution method for the DE-CRP are presented in Table 5. The ALNS heuristic from Section 5 and the simulation model from Section 6 have been implemented in Java 9.0.4. The maximum computation time to solve each E-CRP in the Rolling Horizon simulation framework is set to three minutes.

7. Computational study

We tested the performance of the periodic re-optimization framework for the DE-CRP on the test instances described in Section 6.2. In the following, we first show the results from testing the proposed solution method, before we discuss managerial insights.

7.1. Results

The evaluation of the solution method is based on the objectives presented in Section 3. The degree of demand served, referred to as DS, is the most important key performance indicator. The number of rental cars charged by the service employees during the business hours is also presented. To calibrate the solution method, two tests are considered; Section 7.1.1 tests the length of the planning period when solving each subproblem E-CRP, \(\overline{T}\), while Section 7.1.2 tests the replanning frequency, \(T_{\text{increment}}\). Each test is run over ten days with different realizations of customer requests. The average scores over all days are used as a basis for comparison. To reduce the variance of the results, all models are run on the same set of realized customer requests.

7.1.1. Planning period

This test explores the effects of changing the length of the planning period \(\overline{T}\). The length of the planning period restricts the number of
relocations that the ALNS outputs for the service employees. Ideally, the E-CRP model would consider the whole planning horizon. However, there are three main arguments against using long planning periods in the proposed solution method. First, longer planning periods increase the search space due to the increased number of possible routes for the service employees. The larger search space may, in turn, increase the computational time needed for the ALNS to find good solutions. Second, the future states of the system are stochastic due to varying customer demand and travel times. A solution looking optimal at the moment may, therefore, not even be feasible after the next couple of minutes due to unforeseen events. Finally, since the solution method for the DE-CRP re-plans sequentially, the actions performed by the service employees are usually only the first couple of actions provided by the ALNS. Hence, the use of longer planning periods involves more calculations of needless actions that are not likely to be performed.

Table 6 substantiates these arguments where a planning period of 60 min slightly outperforms the alternatives. When using shorter planning periods, the solutions provided by the ALNS become more greedy, explaining the reduction in demand served when using a planning period of 40 min. In addition, with a planning period of 60 min, the method is able to charge a higher number of cars. This is in turn beneficial as it puts the CSO in a better position with regards to being able to satisfy future demand (beyond the planning horizon of the subproblem). The cars that are being charged are in most cases not available for rental due to low battery levels.

7.1.2. Frequency of replanning

This test explores the effect of changing the replanning frequency $T_{increment}$. Given the result from Section 7.1.1, all tests use a planning period of 60 min. The results of three different replanning frequencies are presented in Table 7. A replanning frequency of 15 min performs slightly better than replanning frequencies of 10 and 20 min when it comes to demand served. Replanning more often should, intuitively, do no worse than replanning more seldom. However, it is noteworthy that optimizing too often may have negative effects. One possible explanation is that high replanning frequencies imply small changes to the system’s state between decision stages. Replanning too often may, therefore, diminish the possible long-term benefits of the routes. However, it may increase the number of cars charged. One explanation is that cars are classified earlier as in need of charging by the simulation model. In addition, frequent replanning increases the probability of charging rental cars. The probability increases because the ALNS has the option to charge cars early more frequently.

7.1.3. Comparison with a greedy construction heuristic

For each individual decision stage, i.e. for each E-CRP subproblem, the ALNS has an average performance increase of 45.1% based on the objective function from Appendix compared with the construction heuristic. Table 8 shows the results from using the construction heuristic in the Rolling Horizon simulation framework. Interestingly, when solving the DE-CRP, the difference in DS is only 7.86%. This implies that the uncertainty faced when solving the DE-CRP reduces the performance gap between the two methods. However, it is noteworthy that the difference of 7.86% in DS corresponds to an additional 175 performance gap between the two methods. However, it is noteworthy that the uncertainty faced when solving the DE-CRP reduces the performance gap between the two methods. However, it is noteworthy that the uncertainty faced when solving the DE-CRP reduces the performance gap between the two methods.

7.2. Managerial insights

In this section we discuss some managerial insights for CSOs that can be gained from various tests. Sections 7.2.1 and 7.2.2 discuss insights of operational character, while tactical and strategic concerns are addressed in Sections 7.2.3 and 7.2.4.

7.2.1. Benefits of charging cars early

One objective of the proposed solution method is to charge cars in need of charging. However, there are no guarantees that these relocations are done by the service employees if they are not among the first relocations in the solutions to the E-CRP subproblems. Hence, prioritizing early charging of cars seems beneficial and has been rewarded 0.1 per time unit in this study.

In the following, we test the effect of including this early charging reward by comparing it with the results without (denoted Regular). The results show that rewarding early charging improves the demand served by approximately 4%. As shown in Fig. 7, rewarding early charging of cars, keeps the number of cars in need of charging at a low and steady level compared to the Regular setting. When charging cars early, the short-term demand served is slightly decreased. However, it is evident that the long-term costs of not meeting future demand are higher than the short-term losses.

The benefit of charging early boils down to the preferences and opening hours of the CSO. For instance, if a CSO only allows car rentals during the daytime, it seems beneficial to prioritize serving demand short-term and do most of the recharging of cars during the night. However, charging all cars during the night requires a sufficient number of service employees to work the night shift. For instance, charging all cars in test instance D-50-170-12-6 would require 12 h of work with the given workforce. To prevent too much work at night, it can therefore be advantageous to use strategies like early charging.

7.2.2. Destinations to consider for relocation

Appendix introduced the set of car-moves which defined the possible destinations each car can be relocated to. Similar to the method in Kirchler and Calvo (2013) for the Dial-a-Ride problem, it is possible to reduce the search space by removing car-moves not likely to be part of good solutions. Fig. 8 shows the distributions of all car-moves present in the best solutions to instance D-50-170-12-6 found by the ALNS heuristic. The distribution is calculated by comparing the car-moves to the longest available travel time present in the test instance. The figure indicates that it may be possible to significantly reduce the search space without degrading the quality of the solutions found by the

<table>
<thead>
<tr>
<th>Hardware and software used in testing</th>
</tr>
</thead>
<tbody>
<tr>
<td>Processor: 3.4 GHz Intel E5</td>
</tr>
<tr>
<td>Memory: 512 GB RAM</td>
</tr>
<tr>
<td>Operating system: CentOS 7.4</td>
</tr>
<tr>
<td>Java version: 9.0.4</td>
</tr>
</tbody>
</table>

Fig. 7. Development of cars in need of charging for D-50-170-12-6.
Table 6
Demand served and cars charged for different planning periods.

<table>
<thead>
<tr>
<th>Instance</th>
<th>$T = 40$ min</th>
<th>$T = 60$ min</th>
<th>$T = 80$ min</th>
<th>$T = 100$ min</th>
<th>$T = 120$ min</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>DS % Cars charged</td>
<td>DS % Cars charged</td>
<td>DS % Cars charged</td>
<td>DS % Cars charged</td>
<td>DS % Cars charged</td>
</tr>
<tr>
<td>D-20-65-5-3</td>
<td>58.35 53</td>
<td>64.10 58</td>
<td>60.48 53</td>
<td>59.48 60</td>
<td>58.98 53</td>
</tr>
<tr>
<td>D-50-170-12-6</td>
<td>60.36 139</td>
<td>63.74 136</td>
<td>62.81 131</td>
<td>61.31 129</td>
<td>61.77 124</td>
</tr>
<tr>
<td>D-120-380-24-12</td>
<td>57.51 279</td>
<td>58.31 285</td>
<td>57.37 266</td>
<td>58.41 255</td>
<td>56.19 251</td>
</tr>
<tr>
<td>Average</td>
<td>58.74 157</td>
<td>62.05 160</td>
<td>60.22 150</td>
<td>59.30 146</td>
<td>58.98 144</td>
</tr>
</tbody>
</table>

Green cells indicate best values for each test instance.

Table 7
Demand served and cars charged for different re-planning frequencies.

<table>
<thead>
<tr>
<th>Instance</th>
<th>$T_{inc} = 10$ min</th>
<th>$T_{inc} = 15$ min</th>
<th>$T_{inc} = 20$ min</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>DS % Cars charged</td>
<td>DS % Cars charged</td>
<td>DS % Cars charged</td>
</tr>
<tr>
<td>D-20-65-5-3</td>
<td>59.96 59</td>
<td>64.10 58</td>
<td>63.06 58</td>
</tr>
<tr>
<td>D-50-170-12-6</td>
<td>63.47 144</td>
<td>63.74 136</td>
<td>63.52 140</td>
</tr>
<tr>
<td>D-120-380-24-12</td>
<td>56.61 279</td>
<td>58.31 285</td>
<td>58.40 279</td>
</tr>
<tr>
<td>Average</td>
<td>60.01 161</td>
<td>62.05 160</td>
<td>61.66 159</td>
</tr>
</tbody>
</table>

Green cells indicate best values for each test instance.

Table 8
Comparing the calibrated solution method to the Construction heuristic.

<table>
<thead>
<tr>
<th>Instance</th>
<th>Construction heuristic</th>
<th>ALNS</th>
<th>DS % Cars charged</th>
<th>DS % Cars charged</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Δ</td>
<td>DS % Cars charged</td>
<td>DS % Cars charged</td>
</tr>
<tr>
<td>D-20-65-5-3</td>
<td>54.73 47</td>
<td>−7.86 pp</td>
<td>54.19 129</td>
<td>−31</td>
</tr>
<tr>
<td>D-50-170-12-6</td>
<td>55.5 112</td>
<td></td>
<td>54.19 129</td>
<td></td>
</tr>
<tr>
<td>D-120-380-24-12</td>
<td>52.33 229</td>
<td></td>
<td>54.19 129</td>
<td></td>
</tr>
<tr>
<td>Average</td>
<td>54.19 129</td>
<td>−7.86 pp</td>
<td>54.19 129</td>
<td>−31</td>
</tr>
</tbody>
</table>

Fig. 8. Distributions of car-moves for test instance D-50-170-12-6. Parking-moves and charging-moves are the distributions for car-moves present in best-found solutions. All available car-moves are the distribution for all car-moves identified.

7.2.3. Number of service employees

ALNS. This implies that cars should in most cases be relocated locally, a finding which considerably simplifies the operational problem. This can be utilized to reduce the computational time for solving the E-CRP subproblems at each decision stage. Testing indicates that these findings also hold for instances D-20-65-5-3 and D-120-380-24-12.

Fig. 9. Difference in demand served when varying the number of service employees. The difference is compared to using 12 employees from the original test instance D-50-170-12-6.

Fig. 9 that the marginal value of additional service employees is diminishing. Having too few service employees is punished by low levels of demand served, while too many service employees yield no significant improvement in demand served. Based on the specific cost and revenue values of the CSOs, the optimal number of employees should be chosen where the marginal revenue of an additional employee is close to the marginal cost of an employee.

7.2.4. Number of charging stations

For the rental cars to be available for customers during the operating hours, charging of cars is crucial. However, the number of cars that can be charged is restricted by the number of available charging stations. Due to capital costs associated with charging stations, this test explores the importance of a sufficient number of charging stations in the carsharing system. Based on test instance D-50-170-12-6, two additional test instances are generated; one with six charging stations and one with 24 charging stations.

Fig. 10 shows that halving the number of charging stations results in fewer cars available for customers. Hence, DS is reduced by 3.76%. When doubling the number of charging stations, DS is increased by 1.87%. However, the increase in demand served diminishes when doubling the number of charging stations. Similar to the case of service employees, the number of charging stations should be chosen such that the marginal revenue from adding a charging station equals the marginal cost.
In this article, the level of granularity of the discretization of the contribution to the creation of efficient, and lasting carsharing systems is considered. In total, we consider the proposed solution method a significant improvement. Novel search methods have been presented. The proposed solution method adopts a Rolling Horizon framework, solving subproblems (E-CRPs) of the DE-CRP at different decision stages.

The subproblems are solved using an Adaptive Large Neighborhood Search (ALNS) heuristic based on Ropke and Pisinger (2006). The objective is to maximize profits by providing a suitable number of cars charged following an expected ideal distribution of vehicles. A solution consists of routes for each service employee, which cars to relocate, and where to relocate them. The solution method allows solutions where service employees originate and end at all locations in the operating area. Thus, the method is capable of solving the E-CRP that arises in free-floating carsharing systems with electric vehicles, at any point in time.

The E-CRP is a variation of classical vehicle routing and pickup delivery problems. The problem structure of the E-CRP allows identification of the minimal set of possible relocation destinations for each car. Each element in this set is denoted a car-move. The proposed solution method derives solutions by searching in and combining elements from the set of car-moves. The use of car-moves is a novelty which significantly simplifies the pickup and delivery aspect of the E-CRP, and thus reducing the complexity of the problem.

A simulation model is developed to test the proposed solution method on problem instances for the DE-CRP. The simulation model mimics the work day of an artificial carsharing organization. The solution method is able to provide efficient solutions for test instances of at least 120 nodes and 380 rental cars. When stress-testing the solution method, it serves 62% of customers on average during a period of 12 h. This equals 1,674 customer rentals served. Compared to a greedy heuristic, an additional 200 customers are served using the proposed solution method.

In conclusion, solving the DE-CRP with the proposed solution method provides high-quality solutions in reasonable computation time for the problem instance tested. Novel search methods have been introduced that effectively deal with the large search space of the problem. In total, we consider the proposed solution method a significant contribution to the creation of efficient, and lasting carsharing systems.

Still, a number of questions remain to be addressed in future research. In this article, the level of granularity of the discretization of the business area was arbitrarily decided before the analysis. Furthermore, every zone of the city was subject to the same discretization. It could be argued that central zones of the city might benefit from a finer discretization, or that a different level of granularity might partially affect the results. The effect of the discretization strategy on the relocation actions is to be clarified. We also assumed that cars must be fully charged once plugged in. This might reduce the ability of the model to satisfy demand. An extension of the model is envisaged where the decision of unplugging a partially charged car in order to fulfill demand is endogenous to the model. Finally, the current method takes the ideal state (i.e., the ideal number of cars) in each zone as an input coming from an exogenous analysis of historical demand. However, demand is influenced by supply (and thus by the deployment of the fleet) as well as by competition. This interplay also requires a dedicated analysis.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgment

This work was partly supported by the Research Council of Norway through the AXIOM project. This support is gratefully acknowledged.

Appendix. The E-CRP subproblem

We propose a Mixed Integer Linear Programming formulation for the E-CRP subproblem. The appendix is self-explanatory meaning that all notion used in the formulation is introduced here. In the cases when the same sets and parameters are used in the heuristic presented in Section 5 and here, the same notation is used.

The operating area is modeled as a complete graph, where nodes represent parking zones and charging stations and edges represent movements between zones and stations. The weight of an edge represents the travel time, and might change between decision stages due to different traffic conditions. Each parking node may be a surplus or a deficit node, if the number of available cars in the corresponding parking zone is higher or lower, respectively, than the ideal state. Let \( N \) be the set of nodes, \( N^C \) the set of charging nodes and \( N^p \) the set of parking nodes, with \( N^C \cap N^p = \emptyset \) and \( N = N^C \cup N^p \). Furthermore, let \( N^p + \subseteq N^p \) be the set of surplus nodes and \( N^p - \subseteq N^p \) the set of deficit nodes, with \( N^p + \cap N^p - = \emptyset \). However, since some parking nodes are at their ideal state we do not necessarily have \( N^p - = N^p \cup N^p + \). Furthermore, let \( N^{PC} \subseteq N^p \) be the set of parking nodes with cars in need of charging. Note that \( N^{PC} \) may be disjoint from both \( N^p + \) and \( N^p - \) as some nodes in \( N^{PC} \) might be at their ideal state.

Let \( C \) be the set of cars potentially subject to relocation, i.e. those either in a surplus node or those in need of charging. Let \( R \) be the set of car-moves. A car-move \( r \) is defined as a triplet \((c, d(r), d'(r))\) with \( c \in C \) being a car, \( d(r) \in N^p + \cup N^{PC} \) its origin node, and \( d'(r) \in N^p - \cup N^C \) its destination node. Particularly, for sufficiently charged cars in surplus nodes, car-moves always go to deficit nodes and are referred to as parking-moves. Similarly, for cars in need of charging, car-moves always go to charging nodes and are referred to as charging-moves. A car in need of charging can only be subject to charging moves. This means that even if the charging station is within a deficit zone, the car in need of charging is not counted towards the deviation from the ideal state.

Parking-moves and charging-moves are illustrated in Fig. 11. Let \( R_{EC} \) be the set of possible car-moves for car \( c \). Let \( R_{EC}^{FD} \) be the set of parking-moves with destination deficit node \( i \). Similarly, let \( R_{EC}^{PC} \) be the set of charging-moves that originate in node \( i \in N^{PC} \). Finally, let \( R_{EC}^{FD} \) be the set of charging-moves that end in charging node \( i \). Note that we consider only car-moves that contribute to increasing demand satisfied,
indicates that the fifth task \( m_5 \) denoted by \( T_5 \) employee can be derived from the sequence of the tasks assigned. The same applies to service employees. In fact, at the beginning of the planning period, the cost per unit of time of the relocation activities, which includes charging station, for charging-moves, or the center of the deficit zones, is car-move. This time includes driving time, parking, possibly plugging in, which coincides with \( d(1) \) is shown as node \( d \), which has been determined through the optimization. From \( d(1) \) the employee travels (by means of a folding bike or by public transport) to the origin of the second car-move \( d(2) \) and completes the second car-move by relocating the car to \( d(2) \), which coincides with \( d(1) \) and is therefore shown as node \( d \). The service employee finally travels to the origin of the third car-move \( d(3) \) and relocates the car to \( d(3) \) (again shown as node \( d \) in the figure since it coincides with \( d(1) \) and \( d(2) \)). This example also serves to illustrate that the problem is open-ended, that is, there is no depot and the routes of the service employees can terminate in any node, and that each node can be visited several times. Furthermore, let variable \( t_{ik} \) indicate the time when task \( m \) is started by employee \( k \) and let variables \( t_{ik}^s \) and \( t_{ik}^r \) represent the time used in excess and in short of \( t_{ik} \), respectively, by employee \( k \). A complete list of the notation can be found in the appendix.

Hence, the E-CRP can be stated as follows:

\[
\begin{align*}
\text{max } z &= \sum_{i \in N^P} \sum_{k \in K} \sum_{r \in R^P} \sum_{m \in M} C^D x_{km} + \sum_{i \in N^P} \sum_{k \in K} \sum_{r \in R^P} \sum_{m \in M} C^B x_{km} \\
&\quad - \sum_{k \in K} C^T t_{ik}^s - \sum_{k \in K} C^T t_{ik}^r - \sum_{k \in K} \sum_{r \in R} \sum_{m \in M} C^E t_{ik} x_{km} \\
\text{s.t. } &\sum_{k \in K} \sum_{m \in M} x_{km} \leq 1, \quad \forall c \in C, \\
&\sum_{k \in K} x_{km} \leq 1, \quad \forall k \in K, m \in M, \\
&\sum_{k \in K} x_{km} \leq \sum_{r \in R} x_{km}, \quad \forall k \in K, m \in M \setminus \{|M|\}, \\
&\sum_{k \in K} \sum_{m \in M} x_{km} \leq \sum_{i \in N^P} x_{i,km}, \\
&\sum_{i \in N^P} \sum_{k \in K} \sum_{r \in R} x_{km} \leq \sum_{i \in N^P} \sum_{k \in K} \sum_{r \in R} x_{i,km}, \quad i \in N^P.
\end{align*}
\]
\[
\sum_{k \in K, r \in R^\text{C}, m \in M} x_{km} \leq S^C_i \\
\sum_{k \in K, r \in R^\text{D}, m \in M} x_{km} \leq N^C_i
\]
\[i \in N^{PC} \quad (4f)\]
\[j \in N^C \quad (4g)\]

\[
I_{km} + T^{\text{H}}_r \leq \sum_{i \in R} T_{d(i) \in P_i} x_{k(i+1)} - M_i (1 - x_{km}) \leq I_{km+1}
\]
\[k \in K, r \in R, m \in M \setminus \{1, |M|\} \quad (4h)\]

Objective function \((4a)\) consists of the sum of the benefit for reaching the ideal state at deficit nodes and the benefit for recharging cars with depleted battery, minus the cost of the employees’ idle time, the cost for exceeding the planning horizon, and the cost of all relocation activities as a consequence of wear, tolls and electricity. The cost for the idle time of employees is introduced to take into account that, in the E-CRP, we deal with only a portion of the entire planning horizon. Idle time costs encourage employees to complete their tasks as soon as possible, and thus leave the company in a better position to address the next re-optimization. The objective function includes the main drivers of a CSOs decisions: on the one hand the need of ensuring a “well deployed” and ready to use (i.e., charged) fleet, on the other hand, the need to contain the costs deriving from ensuring such level of service. Constraints \((4b)\) state that each car can be relocated at most once (i.e., only one of the car-moves associated with the car can be performed). Constraints \((4c)\) make sure that each task of employee can consist of at most one car-move. This also means that some task may not be associated with a car-move (i.e., not performed). Constraints \((4d)\) state the precedence between consecutive car-moves. Constraints \((4e)\) and \((4f)\) limit the number of cars that can be moved to deficit nodes and the number of cars in need of charging, respectively.

Constraints \((4g)\) enforce the capacity of the charging stations. Constraints \((4h)\) ensure consistent tracking of the time used for each employee, car-move and task. Basically, it means that the time when starting a car-move must be greater than or equal to the time the employee started on its preceding car-move plus the time he/she spent to perform that preceding car-move and the time needed to travel from the destination of that preceding car-move to the beginning of the current one. Here, \(M_i\) is a constant large enough to make the constraint redundant when \(x_{km} = 0\). Constraints \((4i)\) state that the starting time of two consecutive tasks must be non-decreasing. Constraints \((4j)\) ensure that no car-move can start before the earliest availability of its car. Constraints \((4k)\) state that the first task assigned to an employee cannot start before its earliest availability plus the time to reach the origin of the first car-move. Constraints \((4l)\) ensure the upper bound on over time. Finally, constraints \((4n)-(4q)\) define the domain for the decision variables.

References


