The Dynamic Electric Carsharing Relocation Problem

Simen Hellem[1], Carl Andreas Julsvoll[1], Magnus Moan[1], Henrik Andersson[1], Kjetil Fagerholt[1], and Giovanni Pantuso[2,*]

[1]*Department of Industrial Economics and Technology Management,, Norwegian University of Science and Technology, Trondheim, Norway*
[2]*Department of Mathematical Sciences,, University of Copenhagen, Copenhagen, Denmark*
[*]*Corresponding author, gp@math.ku.dk*

November 26, 2021

Abstract

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.

Keywords: Carsharing, Urban mobility, Adaptive Large Neighborhood Search, Dynamic routing.

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]). 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/refuelling, 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 takes 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. Boyacı 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. Boyacı 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 traveling 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 using 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 give 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], 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 Ait-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 *dynamic* if *the input of the problem is received and updated concurrently with the determination of the route*, thus showing an evident parallel with the DE-CRP. The interest for DVRPs has increased in the recent years, with 51 out of the total 117 published after 2011 (Psaraftis et al. [2016]). Particularly, Ulmer et al. [2017] present a Markov Decision Process (MDP) framework for DVRPs which generalizes the previous work of Thomas [2007] and Secomandi 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 hours). 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 users' 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 Figure 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 centre 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 Figure 2.

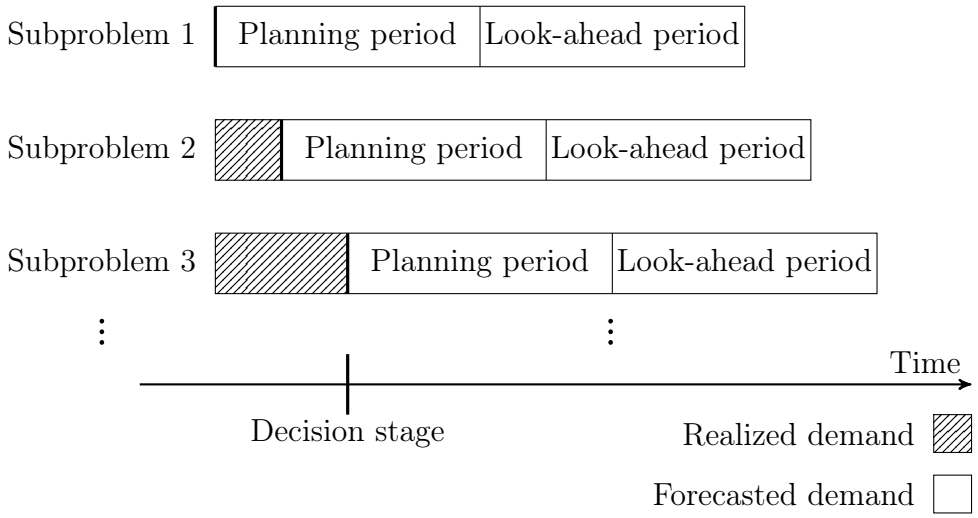

Figure 1: Solution of the DE-CRP by period re-optimization

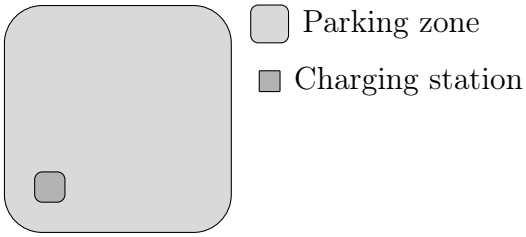

Figure 2: Location of a charging station. The charging station is separated from its associated parking zone

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 necessary information on how to unplug it). Nevertheless, the charging spot will remain occupied in future reoptimizations, until the car has been picked up. We also assume that rental cars with remaining battery level below a set threshold are unavailable for customers until they are fully charged again. Similarly, cars which are subject to repositioning are not available to customers. It is only possible to book a rental car for use at the current time, i.e. booking future usage is not possible. A service employee is assumed to use the fastest means of transportation available, either folding bike or public transportation, when traveling between car relocations. We assume that the time used to relocate cars to parking nodes includes the additional time required to find an available parking spot. Similarly, the time to relocate cars to charging stations includes additional time required to start the charging process. Finally, cars currently charging are assumed to be unavailable also for service employees.

We define the E-CRP as the problem of determining, for a portion of the planning horizon referred to as the planning period (see Figure 1): i) the assignment of cars in need of charging to available charging stations, ii) the assignment of cars in excess zones to

deficit zones, iii) the assignment of employees to car relocation tasks, and vi) routes and schedules for the relocation activities of the service employees. When an E-CRP is solved, the system state is known and a demand forecast for the planning period and look-ahead period (see Figure 1) is provided. A Mixed Integer Linear Formulation of the E-CRP is presented in Appendix A.

5 An Adaptive Large Neighborhood Search Heuristic for the E-CRP

Preliminary testing of the model (summarized in Appendix A) using Xpress 29.01.10 run on a 3.4GHz Intel E5 processor showed that instances with more than ten zones and ten cars could not be solved in a reasonable time, see Figure 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.

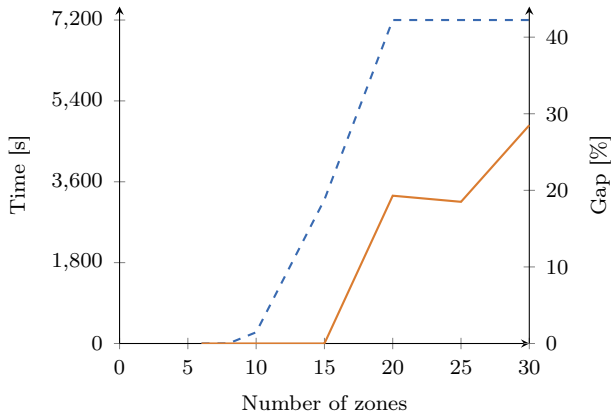

Figure 3: Solution time, blue dashed line, and gap, orange solid line, for the preliminary testing of solving problem (4) using Xpress 29.01.10 run on a 3.4GHz Intel E5 processor. The number of cars in the system is approximately the same as the number of zones. A time limit of 7,200 s has been used.

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^R 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 $\mathbb{M}$ 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 I^{des} iterations without improvements, the search restarts in another neighborhood by applying LNS, and the weights of the heuristic (the parameters that calibrate its behavior, and which are discussed later) are updated.

Algorithm 1: Adaptive Large Neighborhood Search Heuristic

Input: $\mathcal{R}$ Set of candidate car-moves
Output: Ordered list of car-moves for each service employee $k \in \mathcal{K}$

```
Solution s = ConstructionHeuristic(R)
Best solution s^best = s
while stopping criteria not met do
    M = FindNeighborhood(s)
    s ∈ argmax_{s∈M} f(s)
    if f(s) > f(s_best) then
        s^best = s
    else if non-improving TS iterations ≥ I^des then
        s = LargeNeighborhoodSearch(s)
    end
    UpdateWeights()
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 $\mathcal{R}$ 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, γ and β. The first list, γ, contains the used car-moves and is divided into one list for each service employee. Let $\mathcal{K}$ denote the set of service employees and γ_k the ordered list of car-moves performed by service employee k. The second list, β, contains the unused car-moves, not present in γ. At most one car-move for each car may be present in γ. The route of service employee k is derived by iteratively visiting the origin and destination of each car-move $r \in \gamma_k$. Figure 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 4 and 5 with destinations 2 and 5.

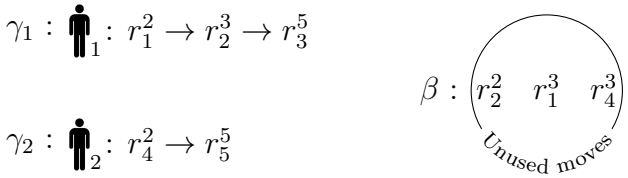

Figure 4: Solution representation. γ_k is the set of car-moves, in order, for service employee k. β is the set of unused car-moves. r_c^i represent moving car c to destination i.

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, γ_k may contain more car-moves than service employee k can handle within the planning period $\overline{T}$. 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, γ_k, can easily be made feasible by moving the car-moves outside the planning period from γ_k to β. 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 Equation (1). The number of parking-moves that are rewarded is denoted τ^P, and the reward per move is C^D. The variable ϕ is the number of charging-moves performed within the planning period, and C^{Ch} 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^T 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 μ_r is 1 if car-move r is performed, and 0 otherwise, and $C^R T_r^H$ is the cost of wear, tolls and electricity for car-move r where C^R is the cost per time unit and T_r^H the time needed to perform the car-move. Note that $(F)^+$ 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 τ^C denotes the total capacity violation at all charging stations, while the variable μ_r^E 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 t_r is the time charging-move r is performed and $t_r = \overline{T}$ if it is not performed. C^{ChE} is the reward per unit of time and $\mathcal{R}^{Ch}$ 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).

$$
\begin{aligned}
f(s) = C^D\tau^P + C^{Ch}\phi - \sum_{k\in\mathcal{K}} C^T(\overline{T} - t_k)^+ - \sum_{k\in\mathcal{K}} C^{ET}(t_k - \overline{T})^+ - \sum_{r\in\mathcal{R}} C^R T_r^H \mu_r \\
- C^I\tau^C - \sum_{r\in\mathcal{R}} C^L \mu_r^E + \sum_{r\in\mathcal{R}^{Ch}} C^{ChE}(\overline{T} - t_r)^+
\end{aligned} \tag{1}
$$

5.3 Construction of the Initial Solution

An initial solution is created in a greedy fashion. Initially, β contains all car-moves and γ is empty. The heuristic iterates through the service employees, thus the employees are handled one at a time. For each employee, $k \in \mathcal{K}$, the best insertion (in terms of objective value) of a car-move at the end of γ_k is performed, given that the corresponding car c does not yet have a car-move in γ. The remaining car-moves for car c remain in β. 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 γ_k.

5.4 Local Neighborhood Search

In each interation, the Tabu Search generates a local neighborhood $\mathbb{M}$, 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 β (i.e. the list of unused car-moves) are moved from/to γ_k (i.e. a route for a given service employee), and a car-move from γ_k is replaced with a car-move from β for the same/different car.

Different ways of generating the neighborhood $\mathbb{M}$ 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^{max} solutions is generated randomly with the selected LSO. We use a *best improvement* strategy when searching in $\mathbb{M}$ 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 γ. Subsequently, repair heuristics insert car-moves into γ. The degree in which a current solution is destroyed and repaired is denoted Γ, $\Gamma = 0.1$ means that 10 % of the car-moves in γ 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 γ, to diversify the search. Worst Removal greedily removes the car-moves causing the largest decrease in the objective function value from the current solution γ. The intention is that more beneficial car-moves can replace these car-moves. Shaw removal was first introduced by Shaw [1997]. The technique increases the number of unique objects in the solution, defining a relatedness measure $R(r_1, r_2)$ between car-moves r_1 and r_2 to identify which objects to remove. Equation (2) shows our definition of $R(r_1, r_2)$. Here, function $\Delta(n, m)$ gives the geographical distance between nodes n and m, while function $c(r)$ returns one if the destination of car-move r is a charging node. The functions $o(r)$ and $d(r)$ give the origin and destination of car-move r, respectively.

$$\begin{aligned} R(r_1, r_2) =& \omega_1 \Delta(o(r_1), o(r_2)) + \omega_2 \Delta(d(r_1), d(r_2)) + \omega_3 |c(r_1) - c(r_2)| \\ &+ \omega_4 |T^H_{r_1} - T^H_{r_2}| + \omega_5 |T^S_{r_1} - T^S_{r_2}| \end{aligned} \tag{2}$$

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 γ and inserted into a list of removed car-moves. While keeping track of the car-moves that are already removed, a random car-move r_1 from this list is chosen. The car-move in γ most similar to r_1 according to Equation (2) is then removed and placed in the list of removed car-moves. This process repeats until a proportion Γ of the car-moves in γ 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 γ by comparing the objective function value of the best insertion with the k best insertions, favouring 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^{des} iterations.

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

$$w_q = w_q(1 - \alpha) + \alpha \frac{\mu_q}{\theta_q} \tag{3}$$

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 Figure 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. Figure 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.

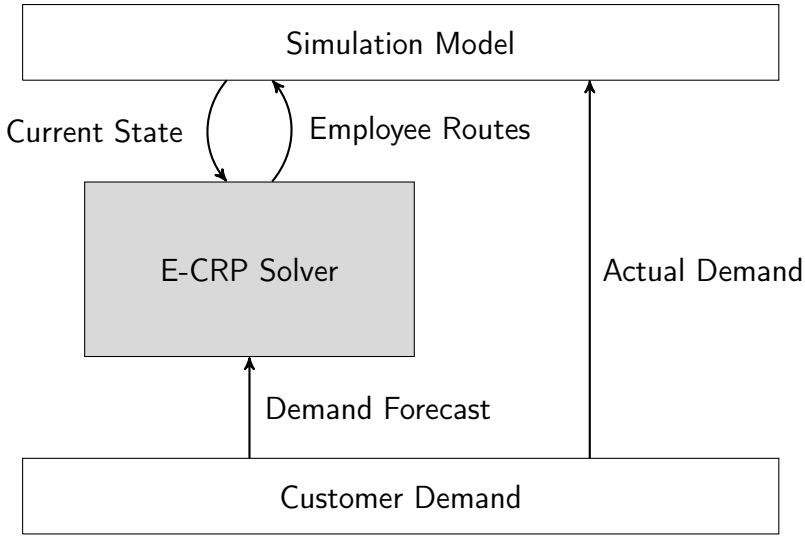

Figure 5: The Rolling Horizon framework components. The Simulation Model can be exchanged by a component tracking actual events in a real world scenario.

Let parameters T_{start} and T_{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 $T_{increment}$ 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 $\overline{T}$ 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_{charge} specify the time (in minutes) it takes to fully charge a rental car with an empty battery. Let T_{range} be the time a fully charged rental car can drive. Let cars with battery level below the threshold ξ_{upper} be considered for recharging. Customers can still rent cars with battery levels between ξ_{upper} and ξ_{lower} as the battery level is sufficient for shorter trips. Rental cars with battery levels below ξ_{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_{start} and $T_{increment}$ 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.

Lines 4-9 in Algorithm 2 show the simulation of tasks from T_{start} until the end of the simulation period, $T_{start} + T_{increment}$. 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

Algorithm 2: Simulation Model

Input: T_{start}, $T_{increment}$, CustomerArrivals, EmployeeArrivals, EmployeeRoutes
Output: System state

```
1 CustomerRequests = CustomerDemand.getActualDemand(T_start, T_start + T_increment)
2 NextEvent = findNextEvent()
3 t ← NextEvent.getTime()
4 while t < T_start + T_increment do
5 |   System state, CustomerArrivals, EmployeeArrivals = doEvent(NextEvent)
6 |   NextEvent ← findNextEventAfter(t)
7 |   updateBatteryLevels(t, min(NextEvent.getTime(), T_start + T_increment))
8 |   t ← NextEvent.getTime()
9 end
```

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 × 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 Figure 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-CRP 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.

Table 1: Instance classes and respective size and constant parameters used for generating instances in the Rolling Horizon framework.

Test Instance	Nodes	Cars	Service Employees	Charging stations
D-20-65-5-3	20	65	5	3
D-50-170-12-6	50	170	12	6
D-120-380-24-12	120	380	24	12

Figure 6: Nodes in the city of Oslo. Used as a basis for all test instances created

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 λ_s, $s \in \{H, M, L\}$ as reported in Table 2. For instance, this means that nodes with morning rush have a rate of λ_H in the morning which linearly decreases towards λ_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 $U \sim \text{unif}(1, 1.4)$.

Table 2: Expected number of cars requested for the three scenarios used in the Poisson process.

	Notation	**Number of cars demanded/hour**
High demand	λ_H	4
Medium demand	λ_M	1
Low demand	λ_L	0.3

The parameters common to all test instances are shown in Table 3. At $T_{start} = 6$ 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: $C^D = 10, C^{Ch} = 30, C^T = 0.01, C^{ET} = 0.5, C^R = 0.2, C^I = 100, C^L = 10, C^{ChE} = 0.1$. 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.

Table 3: Parameters used in the Rolling Horizon framework and the Simulation model.

	Notation	**Value**
Start time business hours	T_{start}	6 AM
End time business hours	T_{end}	6 PM
Time increments	$T_{increment}$	15 min
Planning period	$\overline{T}$	60 min
Overtime	$\overline{T}^L$	10 min
Charging time	T_{charge}	210 min
Car range	T_{range}	120 min
Upper battery threshold	ξ_{upper}	40%
Lower battery threshold	ξ_{lower}	20%

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-CRP 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.

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.

Table 4: ALNS: Final Parameter Values

Parameter	Value	Description
T^{MAX}	180	Max running time (seconds)
B^{INIT}	2	Initial tabu list size
B^{MIN}	2	Minimal tabu list size
B^{MAX}	1024	Maximal tabu list size
I^{R}	125000	Max number of iteration without improvement
I^{W}	$5 \ln \lvert C \rvert$	The number of iterations before the LSO weights are updated
I^{DES}	$120 \ln \lvert C \rvert$	Iterations without global improvement before destroy and repair
I^{B}	6	Iterations without local improvement before increasing the tabu list size
I^{S}	3	Iterations with local improvements before decreasing the tabu list size
M^{MAX}	$25 \ln \lvert C \rvert$	Neighborhood size
Γ	0.4	The destroy/repair factor
R_Q^N	1	LSO score for finding a new local solution
R_Q^G	23	LSO score for finding a new global best solution
R_Q^L	13	LSO score for finding a new better local solution
R_U^G	23	Destroy and repair score for finding a better global solution
R_U^L	13	Destroy and repair score for finding a new and better local solution
α	0.1	Update factor for both LSO and repair and destroy weights
$\omega_1 \dots \omega_5$	0.315, 0.315, 0.315, 0.005, 0.05	Weights for Shaw Removal

Table 5: Hardware and software used in testing

Processor	3,4GHz Intel E5
Memory	512GB RAM
Operating System	CentOS 7.4
Java version	9.0.4

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, $T_{increment}$. 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 minutes 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 minutes. In addition, with a planning period of 60 minutes, 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.

Table 6: Demand served and cars charged for different planning periods

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

Green cells indicate best values for each test instance

7.1.2 Frequency of replanning

This test explores the effects of changing the replanning frequency $T_{increment}$. Given the result from Section 7.1.1, all tests use a planning period of 60 minutes. The results of three different replanning frequencies are presented in Table 7. A replanning frequency of 15 minutes performs slightly better than replanning frequencies of 10 and 20 minutes 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.

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

Instance	$T_{increment} = 10\,min$		$T_{increment} = 15\,min$		$T_{increment} = 20\,min$	
	DS %	Cars charged	DS %	Cars charged	DS %	Cars charged
D-20-65-5-3	59.96	59	64.10	58	63.06	58
D-50-170-12-6	63.47	144	63.74	136	63.52	140
D-120-380-24-12	56.61	279	58.31	285	58.40	279
Average	60.01	161	62.05	160	61.66	159

Green cells indicate best values for each test instance

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 Section A 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 customers served throughout the 12-hour period for the largest test instance. Furthermore, the construction heuristic charges fewer cars, most likely due to inefficient relocations.

Table 8: Comparing the calibrated solution method to the Construction Heuristic

Instance	**Construction Heuristic**	
	DS %	Cars charged
D-20-65-5-3	54.73	47
D-50-170-12-6	55.5	112
D-120-380-24-12	52.33	229
Average	54.19	129
Δ to ALNS	-7.86 pp	-31

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 Figure 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

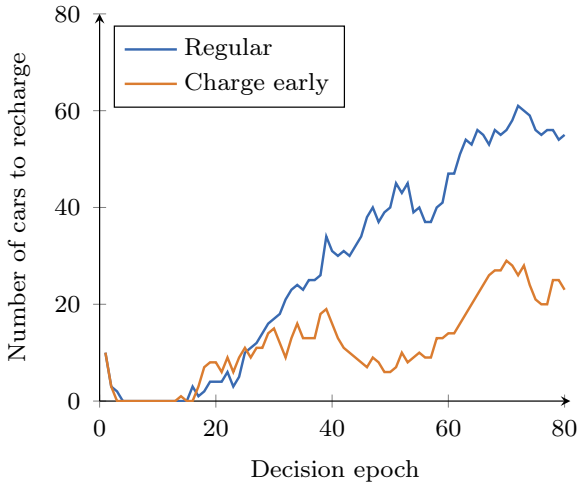

Figure 7: Development of cars in need of charging for D-50-170-12-6

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 hours 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

Section A 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. Figure 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 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.

7.2.3 Number of Service Employees

The optimal number of service employees used in a carsharing system is dependent on the problem instance as well as the CSO's preference regarding the trade-off between costs and customer satisfaction. Intuitively, increasing the number of service employees strictly improves the performance of the system. However, it is evident from Figure 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 yields 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.

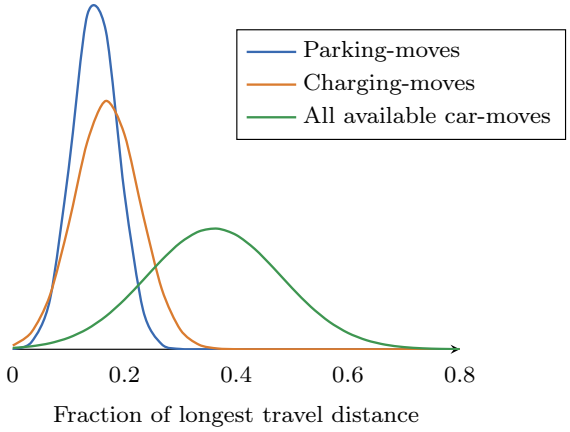

Figure 8: Distributions of car-moves for test instance D-50-170-12-6. Parking-moves and charging-moves are the distributions for car-moves present in best-found solutions. All available car-moves are the distribution for all car-moves identified.

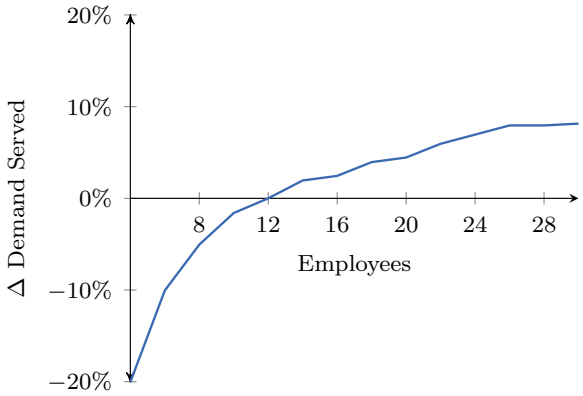

Figure 9: Difference in demand served when varying the number of service employees. The difference is compared to using 12 employees from the original test instance D-50-170-12-6.

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.

Figure 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.

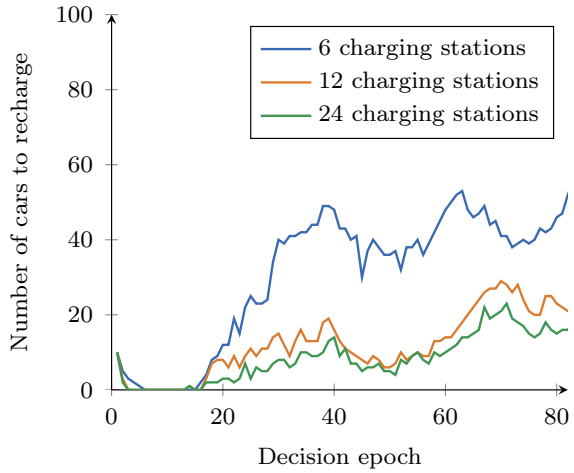

Figure 10: Development of cars in need of charging over the planning horizon for instances with a different number of charging stations.

8 Conclusions

This paper presents a solution method for the Dynamic Electric Carsharing Relocation Problem (DE-CRP). The DE-CRP considers routing of both service employees and rental cars in a free-floating carsharing system. Folding bikes have been assumed as a means of transportation for the service employees, as it offers high flexibility in urban areas. The presented 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 rental 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 and 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 hours. 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.

Acknowledgments

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

References

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.

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

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*, 240:718–733, 2015.

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(Supplement C):214–237, 2017.

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.

G. Brandstätter, M. Leitner, and I. Ljubić. Location of charging stations in electric car sharing systems. *Transportation Science*, 54(5):1408–1438, 2020.

M. Bruglieri, A. Colorni, and A. Luè. The vehicle relocation problem for the one-way electric vehicle sharing: an application to the Milan case. *Networks*, 64:292–305, 2014.

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.

M. Bruglieri, F. Pezzella, and O. Pisacane. An Adaptive Large Neighborhood Search for relocating vehicles in electric carsharing services. *Discrete Applied Mathematics*, 253: 185–200, 2019.

E. M. Cepolina and A. Farina. A new shared vehicle system for urban areas. *Transportation Research Part C: Emerging Technologies*, 21:230–243, 2012.

Z.-L. Chen and H. Xu. Dynamic Column Generation for Dynamic Vehicle Routing with Time Windows. *Transportation Science*, 40:74–88, 2006.

G. Correia, D. Jorge, and D. Atunes. The added value of accounting for users' flexibility and information on the potential of a station-based one-way car-sharing system. *Journal of Intelligent Transportation Systems*, 18(3):299–308, 2014.

C. A. Folkestad, N. Å. Hansen, K. Fagerholt, H. Andersson, and G. Pantuso. Optimal charging and repositioning of electric vehicles in a free-floating carsharing system. *Computers & Operations Research*, 113, 2020.

C. Gambella, E. Malaguti, F. Masini, and D. Vigo. Optimizing relocation operations in electric car-sharing. *Omega*, 81:234 – 245, 2018.

D. K. George and C. H. Xia. Fleet-sizing and service availability for a vehicle rental system via closed queueing networks. *European Journal of Operational Research*, 211: 198–207, 2011.

R. G. Hansen and G. Pantuso. Pricing car-sharing services in multi-modal transportation systems: An analysis of the cases of Copenhagen and Milan. In R. Cerulli, A. Raiconi, and S. Voß, editors, *Computational Logistics*, pages 344–359, Cham, 2018. Springer International Publishing.

S. Hellem, C. A. Julsvoll, and M. Moan. The Dynamic Electric Vehicle Relocation Problem - An Adaptive Large Neighborhood Search. Master thesis, Norwegian University of Science and Technology, 2018.

D. Jorge, G. H. A. 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.

M. Kaspi, T. Raviv, and M. Tzur. Parking reservation policies in one-way vehicle sharing systems. *Transportation Research Part B: Methodological*, 62:35–50, 2014.

A. G. Kek, R. L. Cheu, Q. Meng, and C. H. Fung. A decision support system for vehicle relocation operations in carsharing systems. *Transportation Research Part E: Logistics and Transportation Review*, 45(1):149–158, 2009.

P. Kilby, P. Prosser, and P. Shaw. Dynamic VRPs: A Study of Scenarios. Technical report, University of Strathclyde: Glasgow, UK, 1998.

D. Kirchler and R. W. Calvo. A Granular Tabu Search algorithm for the Dial-a-Ride Problem. *Transportation Research Part B: Methodological*, 56:120–135, 2013.

K. Kortum and R. B. Machemehl. Free-floating carsharing systems: innovations in membership prediction, mode share, and vehicle allocation optimization methodologies. Technical report, University of Texas at Austin, 2012.

R. Nair and E. Miller-Hooks. Fleet management for vehicle sharing operations. *Transportation Science*, 45(4):524–540, 2011.

G. Pantuso. Formulations of a carsharing pricing and relocation problem. In E. Lalla-Ruiz, M. Mes, and S. Voß, editors, *Computational Logistics*, pages 295–310, Cham, 2020. Springer International Publishing. ISBN 978-3-030-59747-4.

H. N. Psaraftis, M. Wen, and C. A. Kontovas. Dynamic vehicle routing problems: Three decades and counting. *Networks*, 67(1):3–31, 2016.

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.

N. Secomandi and F. Margot. Reoptimization Approaches for the Vehicle-Routing Problem with Stochastic Demands. *Operations Research*, 57:214–230, 2009.

S. Shaheen, N. D. Chan, and H. Micheaux. One-way carsharing's evolution and operator perspectives from the Americas. *Transportation*, 42(519):519–536, 2015.

Share-Now. `https://share-now.assetbank-server.com/assetbank-share-now/assetfile/8565.pdf`, 2021a. Accessed: 2021-10-14.

Share-Now. `https://www.share-now.com/dk/en/copenhagen/`, 2021b. Accessed: 2021-10-14.

P. Shaw. A new local search algorithm providing high quality solutions to vehicle routing problems. Technical report, Department of Computer Science, University of Strathclyde, Glasgow, United Kingdom, 1997.

B. W. Thomas. Waiting Strategies for Anticipating Service Requests from Known Customer Locations. *Transportation Science*, 41(3):319–331, 2007.

M. Ulmer, J. Goodson, D. Mattfeld, and B. Thomas. Route-Based Markov Decision Processes for Dynamic Vehicle Routing Problems. Working paper, 2017.

Vy. `https://www.vy.no/alt-om-reisen/andre-transportmidler/vybil`, 2021. Accessed: 2021-10-14.

L. Wang, Q. Liu, and W. Ma. Optimization of dynamic relocation operations for one-way electric carsharing systems. *Transportation Research Part C: Emerging Technologies*, 101:55–69, 2019.

S. Weikl and K. Bogenberger. Relocation Strategies and Algorithms for free-floating Car Sharing Systems. *IEEE Intelligent Transportation Systems Magazine*, 5:100–111, 2012.

S. Weikl and K. Bogenberger. A practice-ready relocation model for free-floating carsharing systems with electric vehicles – Mesoscopic approach and field trial results. *Transportation Research Part C: Emerging Technologies*, 57:206–223, 2015.

J. Yang, P. Jaillet, and H. Mahmassani. Real-Time Multi-Vehicle Truckload Pick-Up and Delivery Problems. *Transportation Science*, 38:135–148, 2002.

A 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 $\mathcal{N}$ be the set of nodes, $\mathcal{N}^C$ the set of charging nodes and $\mathcal{N}^P$ the set of parking nodes, with $\mathcal{N}^C \bigcap \mathcal{N}^P = \emptyset$ and $\mathcal{N} = \mathcal{N}^C \cup \mathcal{N}^P$. Furthermore, let $\mathcal{N}^{P+} \subseteq \mathcal{N}^P$ be the set of surplus nodes and $\mathcal{N}^{P-} \subseteq \mathcal{N}^P$ the set of the deficit nodes, with $\mathcal{N}^{P+} \bigcap \mathcal{N}^{P-} = \emptyset$. However, since some parking nodes are at their ideal state we do not necessarily have $\mathcal{N}^P = \mathcal{N}^{P-} \cup \mathcal{N}^{P+}$. Furthermore, let $\mathcal{N}^{PC} \subseteq \mathcal{N}^P$ be the set of parking nodes with cars in need of charging. Note that $\mathcal{N}^{PC}$ may be disjoint from both $\mathcal{N}^{P+}$ and $\mathcal{N}^{P-}$ as some nodes in $\mathcal{N}^{PC}$ might be at their ideal state.

Let $\mathcal{C}$ be the set of cars potentially subject to relocation, i.e. those either in a surplus node or those in need of charging. Let $\mathcal{R}$ be the set of *car-moves*. A car-move r is defined as a triplet $(c, o(r), d(r))$ with $c \in \mathcal{C}$ being a car, $o(r) \in \mathcal{N}^{P+} \cup \mathcal{N}^{PC}$ its origin node, and $d(r) \in \mathcal{N}^{P-} \cup \mathcal{N}^C$ 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 Figure 11. Let $\mathcal{R}_c$ be the set of possible car-moves for car c. Let $\mathcal{R}_i^{PD}$ be the set of parking-moves with destination deficit node i. Similarly, let $\mathcal{R}_i^{CO}$ be the set of charging-moves that originate in node $i \in \mathcal{N}^{PC}$. Finally, let $\mathcal{R}_i^{CD}$ 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, i.e., only car-moves from surplus nodes or from nodes with cars in need of charging.

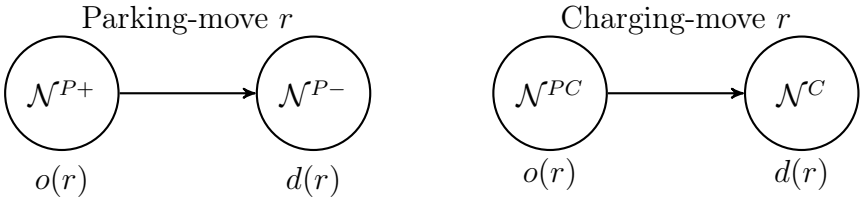

Figure 11: Car-moves divided into parking-moves and charging-moves

The E-CRP assigns car-moves to service employees and decides the order in which they are carried out. This is done in the following way. Let $\mathcal{K}$ the set of service employees. Let $\mathcal{M}$ be an ordered set of possible abstract tasks to perform, where $|\mathcal{M}|$ is the total number of tasks an employee might perform during the planning horizon. The set of tasks is identical for all employees, that is each employee is assigned an identical abstract set of tasks $\mathcal{M}$ to perform. The set of tasks is ordered, in the sense that task $m \in \mathcal{M}$ must be performed before task $m + 1 \in \mathcal{M}$. An abstract task $m \in \mathcal{M}$ becomes concrete when

it is assigned to a car-move from $\mathcal{R}$. As an example, assume employee k is assigned to perform two car-moves, r_1 and r_2, both in $\mathcal{R}$, and to be performed in this order. Then, tasks m_1 and $m_2 \in \mathcal{M}$ become concrete: task $m_1 \in \mathcal{M}$ corresponds to car-move r_1 and task $m_2 \in \mathcal{M}$ corresponds to car-move r_2. This will be further clarified when introducing the decision variables.

For each car-move $r \in \mathcal{R}$ let T_r^H be the time needed to perform the car-move. This time includes driving time, parking, possibly plugging to a power source, and basic maintenance. We track the position of each parked car and calculate the driving time from this location to the charging station, for charging-moves, or the centre of the deficit zones, for parking-moves. Cars may be unavailable at the start of the planning period. Hence, let T_r^{SC} indicate the earliest start time of car-move r. The same applies to service employees. In fact, at the beginning of the planning period, they might still be completing some tasks assigned to them during the previous planning period. Therefore, let T_k^{SO} be the earliest start time for service employee k. Furthermore, let node $o(k) \in \mathcal{N}$ be the position of employee k at time T_k^{SO}. Travel times between nodes i and j, using folding bikes or public transport, are denoted by T_{ij}. The total planning period is denoted $\overline{T}$. However, some overtime $\overline{T}^L$ is allowed.

The initial deficit of cars from the ideal state in parking node i, is denoted S_i^{0-}. For parking nodes $i \in \mathcal{N}^{PC}$, S_i^C denotes the initial number of cars that require charging. Every charging node i has an available capacity of N_i^{CS}. Since N_i^{CS} represents the available, and not the total, capacity, it should be noted that it can vary from one decision stage to the next. We count the number of cars currently being charged at charging station i in the beginning of the planning period and subtract this number from the actual capacity to get N_i^{CS}.

Let C^{Ch} be the remuneration for each car recharged, and C^D the remuneration for each car relocated to decrease the deficit in a parking node. These parameters may correspond to, for example, the expected revenue generated by means of a fully charged car. Let C^{ET} be the cost per time unit used beyond the allocated planning period. This cost is possibly artificially set in order to prioritize timely completion of tasks, especially during the day when the system is used the most. Let C^R be the cost per unit of time of the relocation activities, which includes wear, tolls and electricity. Finally, let C^T be the cost per time unit of the idle time of the service employees, used to increase the activity within the planning period.

Let decision variables x_{krm} indicate that service employee k performs car-move r as task number m. As an example, $x_{k_1,r_2,m_5} = 1$ indicates that the fifth task (m_5) on the agenda of employee k_1 is car-move r_2. Thus, abstract task m_5 becomes concrete when associated with a car-move (r_2 in this case). Hence, the route of the service employee can be derived from the sequence of the tasks assigned. Figure 12 illustrates an example in which a service employee performs three car-moves. Initially the service employee is in its origin $o(k)$. The employee then travels to the origin of the first car-move, $o(1)$, and relocates the corresponding car in need of charging to its destination $d(1)$ (shown as node d in the figure), 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 $o(2)$ 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 $o(3)$ 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

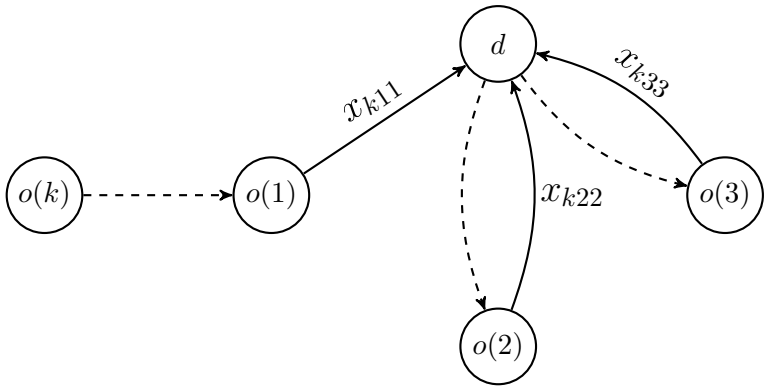

Figure 12: Example illustrating a service employee k performing three car-moves. Dashed lines indicate traveling by folding bike or public transport, between car-moves. Solid lines indicate service employee movements with cars to relocate. In this example, all car-moves go to the same destination shown by node d, which corresponds to $d(1)$, $d(2)$ and $d(3)$.

of the service employees can terminate in any node, and that each node can be visited several times. Furthermore, let variable t_{km} indicate the time when task m is started by employee k and let variables t_k^+ and t_k^- represent the time used in excess and in short of $\overline{T}$, 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:

$$\max z = \sum_{i\in\mathcal{N}^{P-}}\sum_{k\in\mathcal{K}}\sum_{r\in\mathcal{R}_i^{PD}}\sum_{m\in\mathcal{M}} C^D x_{krm} + \sum_{i\in\mathcal{N}^{PC}}\sum_{k\in\mathcal{K}}\sum_{r\in\mathcal{R}_i^{CO}}\sum_{m\in\mathcal{M}} C^{Ch} x_{krm} - \sum_{k\in\mathcal{K}} C^T t_k^- - \sum_{k\in\mathcal{K}} C^{ET} t_k^+ - \sum_{k\in\mathcal{K}}\sum_{r\in\mathcal{R}}\sum_{m\in\mathcal{M}} C^R T_r^H x_{krm} \tag{4a}$$

$$\sum_{k\in\mathcal{K}}\sum_{r\in\mathcal{R}_c}\sum_{m\in\mathcal{M}} x_{krm} \le 1 \qquad c\in\mathcal{C} \tag{4b}$$

$$\sum_{r\in\mathcal{R}} x_{krm} \le 1 \qquad k\in\mathcal{K}, m\in\mathcal{M} \tag{4c}$$

$$\sum_{r\in\mathcal{R}} x_{kr(m+1)} \le \sum_{r\in\mathcal{R}} x_{krm} \qquad k\in\mathcal{K}, m\in\mathcal{M}\setminus\{|\mathcal{M}|\} \tag{4d}$$

$$\sum_{k\in\mathcal{K}}\sum_{r\in\mathcal{R}_i^{PD}}\sum_{m\in\mathcal{M}} x_{krm} \le S_i^{0-} \qquad i\in\mathcal{N}^{P-} \tag{4e}$$

$$\sum_{k\in\mathcal{K}}\sum_{r\in\mathcal{R}_i^{CO}}\sum_{m\in\mathcal{M}} x_{krm} \le S_i^{C} \qquad i\in\mathcal{N}^{PC} \tag{4f}$$

$$\sum_{k\in\mathcal{K}}\sum_{r\in\mathcal{R}_i^{CD}}\sum_{m\in\mathcal{M}} x_{krm} \le N_i^{CS} \qquad i\in\mathcal{N}^{C} \tag{4g}$$

$$t_{km} + T_r^H x_{krm} + \sum_{v\in\mathcal{R}} T_{d(r)o(v)} x_{kv(m+1)} - M_r(1-x_{krm}) \le t_{k(m+1)}$$

$$k \in \mathcal{K}, r \in \mathcal{R}, m \in \mathcal{M} \setminus \{|\mathcal{M}|\} \tag{4h}$$

$$t_{km} \leq t_{k(m+1)} \qquad k \in \mathcal{K}, m \in \mathcal{M} \setminus \{|\mathcal{M}|\} \tag{4i}$$

$$T_r^{SC} x_{krm} \leq t_{km} \qquad k \in \mathcal{K}, r \in \mathcal{R}, m \in \mathcal{M} \tag{4j}$$

$$(T_k^{SO} + T_{o(k)o(r)}) x_{kr1} \leq t_{k1} \qquad k \in \mathcal{K}, r \in \mathcal{R} \tag{4k}$$

$$t_{k|\mathcal{M}|} + \sum_{r \in \mathcal{R}} T_r^H x_{kr|\mathcal{M}|} + t_k^- - t_k^+ = \overline{T} \qquad k \in \mathcal{K} \tag{4l}$$

$$t_{k|\mathcal{M}|} + \sum_{r \in \mathcal{R}} T_r^H x_{kr|\mathcal{M}|} \leq \overline{T} + \overline{T}^L \qquad k \in \mathcal{K} \tag{4m}$$

$$x_{krm} \in \{0, 1\} \qquad k \in \mathcal{K}, r \in \mathcal{R}, m \in \mathcal{M} \tag{4n}$$

$$t_{km} \geq 0 \qquad k \in \mathcal{K}, m \in \mathcal{M} \tag{4o}$$

$$t_k^+ \geq 0 \qquad k \in \mathcal{K} \tag{4p}$$

$$t_k^- \geq 0 \qquad k \in \mathcal{K} \tag{4q}$$

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 each 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_r is a constant large enough to make the constraint redundant when $x_{krm} = 0$. 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) keep track of the deviations from the planning period. Constraints (4m) enforce the upper bound on over time. Finally, constraints (4n)-(4q) define the domain for the decision variables.