European Journal of Operational Research xxx (xxxx) xxx

Contents lists available at ScienceDirect

European Journal of Operational Research

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

Production, Manufacturing, Transportation and Logistics

# Joint relocation and pricing in electric car-sharing systems

Ulrik Eilertsen [a], Olav M. Falck-Pedersen [a], Jone V. Henriksen [a], Kjetil Fagerholt [a,*], Giovanni Pantuso [b]

[a] Department of Industrial Economics and Technology Management, Norwegian University of Science and Technology (NTNU), Alfred Getz veg 3, NO-7491 Trondheim, Norway

[b] Department of Mathematical Sciences, University of Copenhagen, Copenhagen, DK-2100, Denmark

ARTICLE INFO

*Keywords:*
Transportation
Car-sharing
Pricing
Stochastic programming
Adaptive Large Neighborhood Search

ABSTRACT

In this paper we study the integrated planning problem of determining car-sharing prices between zones of the operating area and routing employees (operators) to relocate cars in preparation for future uncertain demand. We present a novel two-stage integer stochastic programming model for this problem together with a heuristic algorithm, based on Adaptive Large Neighborhood Search (ALNS), to obtain solutions to realistically sized instances. We test the ALNS heuristic on a set of instances generated based on data from a real car-sharing organization and show that it outperforms a commercial solver.

## 1. Introduction

Car-sharing has experienced significant growth in recent years and is expected to further increase in popularity (Schiller et al., 2017). Car-sharing systems are typically divided into *station-based* and *free-floating*. In station-based systems, cars must be picked up from and returned to one of the available service stations. In free-floating systems, cars can be returned to any common parking space within the operating area. Both systems can be configured for either *one-way* rentals, where cars can returned to a station/location different from the pick-up location, or *two-way* rentals, where cars must be returned where they were picked up.

One-way free-floating systems have gained popularity due to the higher level of flexibility for the users. However, such configuration makes the system vulnerable to geographical and temporal mismatches between supply and demand. As a prime form of response, the car-sharing organization (CSO) normally relocates cars to areas where demand is high and away from areas where demand is low (Boldrini et al., 2017). These rebalancing activities are performed by dedicated staff, hence the name *operator-based rebalancing*. In the situation when the fleet consists of electric cars – which has become very common in recent years (Cheng et al., 2019) – an even more important task for the employees is to ensure that cars with depleted batteries are driven to charging stations.

Most CSOs today use pricing strategies that consist of a fixed per-minute fee independent of time of the day, origin and destination of the trip, or other factors. However, in many industries revenue management practices help better utilize customers' differences in their willingness to pay for a given service, and through that increase profits, e.g., Klein et al. (2020). In car-sharing systems, this could translate e.g., into time- or zone-dependent fees. In addition, such pricing strategies may become instrumental in reducing the need for operator-based relocations. For example, a CSO could set a low, or even negative, price for trips originating in a zone where there is already an abundance of cars and terminating to a zone where the demand exceeds supply, or to a charging station if the battery level of the car is low. Synergies could arise from jointly optimizing pricing and relocation decisions.

Therefore, in this paper we consider the operational planning problem faced by a CSO of jointly setting car-sharing prices and deciding operator-based relocations to face future (uncertain) demand and maximize (expected) profits. Particularly, the pricing strategy we consider consists of setting origin- and destination-specific prices in order to favor (prevent) (un)favorable movements of cars. To account for uncertainty in customer preferences with respect to the mode of transport, we model this problem as a stochastic program and denote it as the *Stochastic Electric Vehicle Relocation and Pricing Problem* (SE-VRePP). The main contributions of this paper can be summarized as follows: (1) we propose a novel two-stage stochastic integer programming model for the SE-VRePP which addresses pricing decisions as well as the relocation activities, including detailed routing and scheduling of the operators' (employees') tasks; (2) we develop an Adaptive Large Neighborhood Search (ALNS) heuristic for solving realistically-sized instances; (3) we test the ALNS heuristic on a set of instances generated based on data from Vybil, which is a real CSO operating in Oslo, Norway; and (4) we analyze the effect of jointly planning pricing and relocation decisions

* Corresponding author.
*E-mail address:* kjetil.fagerholt@ntnu.no (K. Fagerholt).

https://doi.org/10.1016/j.ejor.2023.12.001
Received 11 January 2023; Accepted 1 December 2023
Available online 6 December 2023

through simulations offering managerial insights. As further explained in Section 2, the literature on joint pricing and relocation decisions is sparse and this study extends it in a number of ways.

The outline of the remainder of this paper is as follows. Section 2 presents a review of the relevant literature, while Section 3 gives a detailed description of the SE-VRePP and formulates it as a two-stage stochastic programming model. Section 4 outlines the heuristic solution algorithm. Finally, the computational study is presented in Section 5, before we draw some concluding remarks in Section 6.

## 2. Summary of the available literature

Operator-based rebalancing of cars is one of the main challenges for CSOs. To address this challenge, the scientific literature has grown substantially since the onset of car-sharing services, providing methods for various configurations of the system. Examples are methods that focus on addressing rental demand uncertainty using a variety of methods including prediction, (Hellem et al., 2021; Weikl & Bogenberger, 2013), machine learning, and data driven optimization or stochastic programming (Brandstätter et al., 2017; Fan, 2013; Huo et al., 2020; Li et al., 2019; Santoso et al., 2005). Particular attention has also been paid to the growing adoption of electric fleets, see e.g., Boyacı et al. (2015, 2017), Bruglieri et al. (2014), Folkestad et al. (2019), Hellem et al. (2021), Xu and Meng (2019). The surveys in Golalikhani et al. (2021), Illgen and Höck (2019), Wu and Xu (2022) provide thorough analyses of the available models and methods.

Operator-based rebalancing is, however, inherently inefficient and expensive in the large scale, especially if performed while the system in being used most (e.g., during day hours). For this reason, pricing-based mechanisms to balance demand and supply at the geographical and temporal level have been proposed. The central concept of such method is to make more attractive certain movements of cars that are, for various reasons, considered beneficial by the CSO.

The research literature on pricing method is also growing considerably. A classification of the literature is proposed by Pantuso (2022), who divide the proposed pricing strategies in *individual* and *collective*. Individual pricing strategies require an interaction between the CSO and the individual user by means of which the trip details (including price) are agreed upon. As an example, the CSO may offer a specific user a reward in exchange for returning the car to a more favorable position. Further examples of individual pricing strategies can be found, e.g., in Wagner et al. (2015), Waserhole and Jost (2016), Di Febbraro et al. (2018),Stokkink and Geroliminis (2021), Liu et al. (2021), Wu et al. (2021), Wang et al. (2021) and Wang and Ma (2019). Collective pricing strategies are instead targeted to the entire user base and aim to influence the rental demand by means of prices. As an example, the CSO may decrease the price of rentals to/from selected zones. Further examples of collective pricing strategies can be found, e.g., in Jorge et al. (2015), Hansen and Pantuso (2018), Kamatani et al. (2019), Xu et al. (2018), Xie et al. (2019), Ren et al. (2019), Lu et al. (2021), Kikuchi and Miwa (2021), Pantuso (2020), Li et al. (2022) and Pantuso (2022). It should be noted however, that the classification is not meant to be precise as some methods could be classified equally well in both categories.

From the analysis of the research literature, it emerges that both pricing and relocation decisions have received considerable attention in the research literature, and method are available for various configurations of the car-sharing service. Nevertheless, the gap that emerges from the literature, is that pricing and relocation decisions have mainly been considered as separate entities. We argue instead that since they may be used to ease the balance between demand and supply, they should be considered in the same decision process, and that the combination of the two can reinforce the ability of the CSO to prevent imbalances. Contributions in this direction, can be found in Xu et al. (2018), Li et al. (2022) and Pantuso (2022). Xu et al. (2018) formulate the joint pricing and relocation problem as a non-linear mixed-integer programming problem in order to account for demand elasticity. We show that the problem can be formulated using a linear mixed-integer two-stage stochastic programming problem by accounting directly for uncertainty in individual customers preferences. Li et al. (2022) present a simulation–optimization framework to determine both the prices between car-sharing stations and the relocations to perform. However, the optimization model does not explicitly determine the activities of the operations, which are instead taken care of by the simulation framework. In this paper, we extend the recipe of Pantuso (2022), which model the joint problem of deciding car-sharing prices and relocations by a two-stage stochastic program to account for the uncertainty in customer's preferences in terms of transport mode. However, in contrast to our work, rebalancing decisions are modeled at a rather high level of abstraction. Particularly, they model the number of relocations to perform between zones without addressing how such movements should be performed in practice by operators (and even if these are feasible given the available staff). We fill this gap by adding to pricing decisions a more detailed model of operators activities. In addition, we ensure that the expected number of cars with low battery levels relocated to charging stations (either by employees or by customers) exceeds a given threshold. This allows us to adjust prices also in such a way to incentivize recharging activities made by customers.

## 3. Problem description and mathematical model

We formulate the SE-VRePP as a two-stage stochastic integer program where prices and relocations are decided in the first decision stage and rentals are decided in the second decision stage. We start by describing the problem and introducing the mathematical notation in Section 3.1 before presenting the full mathematical model in Section 3.2.

### 3.1. Problem description and notation

We assume the CSO operates a homogeneous set $\mathcal{V}$ of electric vehicles in a business area which is suitably partitioned in a set $\mathcal{I}$ of zones. A subset $\mathcal{I}^{CS} \subseteq \mathcal{I}$ of the zones includes the zones containing charging stations. The number of available charging spots in zone $i \in \mathcal{I}^{CS}$ is given by $N_i^{CS}$. Observe that the zones containing charging stations can be used both for normal parking as well as for charging cars.

In order to maximize expected profits for a given target period (e.g., two hours in the morning or afternoon), some time before the beginning of the target period, the CSO makes decisions regarding the prices applicable during the target period and how to relocate cars within the business area in preparation for the target period.

At the time decisions are made, each car $v$ initially located in zone $o(v) \in \mathcal{I}$. A subset of cars $\mathcal{V}^B \subset \mathcal{V}$ contains the cars in need of charging. We assume that, in order to ensure continuity of the service, at least a fraction $N^B$ of the cars in need of charging must be brought to a charging station, either by operators or as a result of rentals.

In order to perform relocations and recharging activities, we assume that the CSO employs a homogeneous set $\mathcal{E}$ of operators with operator $e$ located in zone $o(e)$ at the time decisions are made. We assume operator are possibly initially occupied in relocation tasks assigned to them during previous planning phases. The earliest start time employee $e$ is available for performing a task is given by $T_e^{SO}$.

Each operator has a set $\mathcal{M}$ of abstract tasks used to keep track of the sequence of car moves to perform. The set of possible car-moves that can be performed by operators for all cars is given by $\mathcal{R}$. We identify the origin and the destination of car-move $r$ by $o(r)$ and $d(r)$. An abstract task $m \in \mathcal{M}$ becomes concrete once it is assigned to a specific car-move $r \in \mathcal{R}$. We also define the subset $\mathcal{R}_v^V$ of car-moves available for each car $v$ (observe, e.g., that for cars in need of charging $\mathcal{R}_v^V$ only includes moves to charging stations), and the subset $\mathcal{R}_i^N$ for all car-moves with destination zone $i$. $\mathcal{R}_i^{CD} \subseteq \mathcal{R}_i^N$ contains the set of car-moves

with destination at charging zone $i$ (also referred to as charging moves). It should be noted that cars in need of charging can only be moved to a charging zone (a zone with at least one charging station). This move can be done by either an operator or a customer. Cars not in need of charging can be moved to all zones, including charging zones. However, cars not in need of charging will not occupy a charging station slot if moved to a charging zone. Furthermore, we assume that a car is labeled 'in need of charging' if its battery level is below a certain threshold. At this point, if the car is to be moved, it must be moved to a charging zone (a zone with at least one charging station), and further be put to charging (either by an employee or a customer). This means that a car in need of charging is still available for a customer to rent, but the customer must drive it to a charging station within a charging zone (and the customer will be informed about it). For this to work, we assume a car in need of charging always has enough battery capacity to reach any available charging station in the operating area. Since cars may be busy at the time of planning (e.g., due to ongoing relocations) we let the first time point when a car is available for a car-move $r$ be $T_r^{SC}$.

Similar to Hellem et al. (2021) and others, we assume that the operators can travel between car moves by using folding bicycles that can fit into the trunk of the cars or by using public transport (whichever is fastest). Therefore, we let $T_r^H$ be the time necessary to complete car-move $r$. This parameter includes the time it takes to get from the origin to the destination zone of the car-move, in addition to the approximated time it takes to find a parking spot or initiate charging. The time it takes to travel between zones $i$ and $j$ is given by $T_{ij}$. Let also $C^R$ be the cost per time unity of driving a car. The salary for the given set of employees is assumed fixed and therefore not a part of the model. We let time $\overline{T}^1$ represent the time when the target period starts. This entails that all relocation activities must be completed before $\overline{T}^1$.

We let binary variable $z_{iv}$ take the value 1 if car $v$ is made available to customers in zone $i$ at the beginning of the target period, 0 otherwise, as a result of the operator-based relocations. Furthermore, binary variable $x_{erm}$ takes the value 1 if operator $e$ performs car-move $r$ as their task number $m$, 0 otherwise. We also let non-negative variable $t_{em}$ represent the time when operator $e$ starts performing task $m$.

In addition to relocation activities, by adjusting rental prices between the different zones of the operating area, the CSO can make certain cars and trips more (or less) attractive to customers. Particularly, similarly to Pantuso (2020, 2022), we assume that the pricing mechanism is made of a per-minute fee independent of the origin and destination of the trip, and a pick-up/drop-off fee which instead depends on the origin and destination zones of the trip. We assume the CSO can adopt a set $\mathcal{L}$ of pick-up/drop-off fee levels, with $L_l$ being the dollar value of fee $l$. Consequently, we define binary variable $\lambda_{ijl}$ to take value 1 if level $l$ is chosen between zone $i$ and $j$, 0 otherwise.

Once pricing and relocation decisions have been made, during the target period customers interact with the car-sharing service. We let $\mathcal{K}$ be the set of all potential car-sharing customers. Particularly, this set includes all the customers requiring transportation within the operating area, independently of the transportation mode. We define subsets $\mathcal{K}_i$ and $\mathcal{K}_{ij}$ containing the customers traveling from zone $i$, and from zone $i$ to zone $j$, respectively.

Rental demand is, nevertheless, uncertain at the time of planning. In fact, customers react to pricing decisions and choose their preferred mode of transport in a way that is partially unknown to the decision maker. Particularly, we assume each customer is characterized by unique preferences, captured by a well-specified choice model (Bierlaire & Sharif Azadeh, 2016). This entails that the random term of the choice model is a fully specified random variable. In line with Paneque et al. (2021), Pantuso (2020, 2022), we assume that each customer chooses the transport mode that maximizes their utility. Given a set $\mathcal{T} = \{CS, \dots\}$ of transport services (which includes car-sharing), the utility received by customer $k$ when using transport service $t$ to move from its origin $o(k)$ to its destination $d(k)$ is given by

$$U_{kt} = F_k\left(p_{o(k),d(k),t}, \pi_1, \dots, \pi_N\right) + \tilde{\xi}_{kt}$$

where $F_k(\cdot)$ is a function that describes the preferences of the customer with respect to the price $p_{o(k),d(k),t}$, and a number of additional characteristics of the service $\pi_1, \dots, \pi_N$ which we assume are exogenous to the decision problem at hand. For car-sharing services, $p_{o(k),d(k),t} = P^M T^D_{o(k),d(k)} + \sum_{l\in\mathcal{L}} L_l \lambda_{o(k),d(k),l}$, where $P^M$ is the per-minute fee and $T^D_{o(k),d(k)}$ is the driving time between $o(k)$ and $d(k)$. For other transport services $t$, $p_{o(k),d(k),t}$ is a known parameter. The random variable $\tilde{\xi}_{kt}$, with $\tilde{\xi} := (\tilde{\xi}_{kt})_{k\in\mathcal{K},t\in\mathcal{T}}$, captures the portion of the customer's preferences unknown to the CSO. Different specifications of a probability distribution for $\tilde{\xi}$ lead to different choice models, see e.g., Train (2009). We let $S$ be a set of realizations (scenarios) of $\tilde{\xi}$, with $P_s$ being the probability of scenario $s$, and $\xi_s$ the realization of $\tilde{\xi}$ under scenario $s$, that is $\xi_s := (\xi_{kts})_{k\in\mathcal{K},t\in\mathcal{T}}$.

Given a scenario $s$, the random element of the choice model $\xi_s$ materializes, and the preferences of each customer with respect to the transport mode are fully known for that scenario. For each scenario we initialize a set of customer requests, $\mathcal{D}(\xi_s)$. The set $\mathcal{D}(\xi_s)$ contains a request for each customer $k$ for which there exists at least one fee level $l$ in $\mathcal{L}$ for which the customer would prefer car-sharing over other transport alternative. That is, there exists at least one fee level for which the customer finds their highest utility in using car-sharing. (Observe, thus, that the customers which would never prefer car-sharing regardless of the fee, are not in the set). More formally, there exists one request for each customer $k$ for which

$$\mathcal{L}_k := \left\{ l \in \mathcal{L} \,|\, F_k(p_{o(k),d(k),CS}, \pi_1, \dots, \pi_N) + \xi_{kts} > U_{kt} \forall t \in \mathcal{T}, t \neq CS \right\} \neq \emptyset$$

The origin, destination and customer of request $d \in \mathcal{D}(\xi_s)$ are denoted by $i(d)$, $j(d)$ and $k(d)$, respectively, and the highest fee level at which customer $k(d)$ would prefer car-sharing to other transportation modes is represented by $l(d)$, that is $l(d) := \arg\max_{l\in\mathcal{L}_{k(d)}}\{L_l\}$. We further create subsets $\mathcal{D}_{ij}(\xi_s) \subseteq \mathcal{D}(\xi_s)$, containing the requests going from zone $i$ to zone $j$, $\mathcal{D}^{CD}(\xi_s) \subseteq \mathcal{D}(\xi_s)$ containing the requests with destination in a charging zone, and $\mathcal{D}^{CD}_i(\xi_s) \subseteq \mathcal{D}^{CD}(\xi_s)$ containing the requests going to charging zone $i \in \mathcal{I}^{CS}$.

Most car-sharing services operate with a First-Come-First-Served mechanism (Wang & Liao, 2021). For this reason we let set $\mathcal{D}_d(\xi_s) = \{p \in \mathcal{D}(\xi_s) : i(p) = i(d), k(p) < k(d)\} \subseteq \mathcal{D}(\xi_s)$ contain the requests that have precedence over request $d$, i.e., we simply assume that the index of the customer indicates the arrival time at the vehicle. Finally, we let $\mathcal{L}_d \subseteq \mathcal{L}$ be the subset of fees that are less than the highest fee level acceptable for request $d$, $l(d)$ and $R_{dl}$ be the profit of satisfying request $d$ with fee level $l$ (i.e., the sum of the per-minute fee of the trip and the pick-up and drop-off fees).

During the target period (second decision stage) cars are rented by customers. We let binary variable $y_{vdls}$ take the value 1 if car $v$ satisfies request $d$ with the fee level $l$ in scenario $s$, 0 otherwise. Observe that $y_{vdls}$ are second-stage decision variables and are defined for each scenario $s$.

All the notation is summarized in Appendix A.

### 3.2. Model

In this section, we present the mathematical model for the SE-VRePP.

$$\max z = \max -\sum_{e\in\mathcal{E}}\sum_{r\in\mathcal{R}}\sum_{m\in\mathcal{M}} C^R T_r^H x_{erm} + \sum_{s\in S} P_s \left( \sum_{d\in\mathcal{D}(\xi_s)}\sum_{v\in\mathcal{V}}\sum_{l\in\mathcal{L}_d} R_{dl} y_{vdls} \right) \tag{1a}$$

The objective function (1a) represents the total expected profit for the CSO. The first term represents the (deterministic) costs born to perform operator-based relocations, while the second term represents the expected revenue for the rentals occurred during the target period.

$$\sum_{l\in\mathcal{L}} \lambda_{ijl} = 1, \qquad i, j \in \mathcal{I} \tag{1b}$$

$$1 - \sum_{e\in\mathcal{E}}\sum_{r\in\mathcal{R}_v^V}\sum_{m\in\mathcal{M}} x_{erm} = \sum_{i\in\mathcal{I}} z_{iv}, \qquad v \in \mathcal{V}^B \tag{1c}$$

$$\sum_{e\in\mathcal{E}}\sum_{r\in\mathcal{R}_i^N\cap\mathcal{R}_v^V}\sum_{m\in\mathcal{M}} x_{erm} = z_{iv}, \qquad i \in \mathcal{I}\backslash\{o(v)\}, v \in \mathcal{V}\backslash\{\mathcal{V}^B\} \tag{1d}$$

$$1 - \sum_{e\in\mathcal{E}}\sum_{r\in\mathcal{R}_v^V}\sum_{m\in\mathcal{M}} x_{erm} = z_{o(v),v}, \qquad v \in \mathcal{V} \tag{1e}$$

$$\sum_{i\in\mathcal{I}} z_{iv} \le 1, \qquad v \in \mathcal{V} \tag{1f}$$

The constraints that handle the logic of the first decision stage are as follows. Constraints (1b) ensure that exactly one fee is assigned between each pair of zones. Constraints (1c)–(1f) handle the logic concerning the availability of cars in each zone. Constraints (1c) state that a car in need of charging is not available for rentals if any employee has moved it to a charging station. Constraints (1d) ensure that if a car is relocated to a given zone, it becomes available for rentals in that zone during the target period. Constraints (1e) force a car to be available in its original zone if it is not relocated and similarly unavailable in its original zone if it is moved. Finally, Constraints (1f) make sure that a car is available for rental in at most one zone.

$$\sum_{r\in\mathcal{R}} x_{e,r,(m+1)} \le \sum_{r\in\mathcal{R}} x_{erm}, \qquad e \in E, m \in \mathcal{M}\backslash\{|\mathcal{M}|\} \tag{1g}$$

$$\sum_{r\in\mathcal{R}} x_{erm} \le 1, \qquad e \in \mathcal{E}, m \in \mathcal{M} \tag{1h}$$

$$\sum_{e\in\mathcal{E}}\sum_{r\in\mathcal{R}_v^V}\sum_{m\in\mathcal{M}} x_{erm} \le 1, \qquad v \in \mathcal{V} \tag{1i}$$

$$\sum_{e\in\mathcal{E}}\sum_{i\in\mathcal{I}^{CS}}\sum_{r\in\mathcal{R}_i^{CD}}\sum_{m\in\mathcal{M}} x_{erm} + \sum_{s\in S} P_s\left(\sum_{v\in\mathcal{V}^B}\sum_{d\in\mathcal{D}^{CD}(\xi_s)}\sum_{l\in\mathcal{L}} y_{vdls}\right) \ge N^B\cdot|\mathcal{V}^B| \tag{1j}$$

Constraints (1g)–(1j) handle the logic concerning the relocation of cars. Constraints (1g) ensure that the tasks assigned to the employees are performed in the given order, that is task $m$ is performed before task $m+1$. Constraints (1h) state that any employee task can consist of at most one car-move. Constraints (1i) ensure that each car is relocated at most once. Finally, Constraint (1j) ensures that the expected number of cars in need of charging that are moved to charging stations, either by employees or customers, is higher than the defined threshold. Observe that the number of cars moved by customers to charging stations is influenced by the prices set to and from the charging stations, as it will be more evident in the second-stage constraints. Furthermore, note that this constraint ensures that the threshold is exceeded only on expectation. It is therefore possible that, in some scenarios, there result fewer cars plugged in than desired. However, in the long term, over many repetitions, the Law of Large Numbers ensures that the number of cars plugged in coincides with the expectation. It should be noted that the parameter $N^B$ can be changed over the course of a day, i.e., from one model run to another. This means for example that if the CSO knows that they will need many cars in the near future, they can increase $N^B$ and vice versa. In this way, our approach can easily be adapted to the case for when we aim to have a given number of cars available. As an example, suppose there are 30 cars in the fleet out of which six cars are in need of charging, and we aim to have at least 27 cars (i.e., minimum 90% of the fleet should be sufficiently charged). In this case we can set $N^B = 0.5$, which will make sure that at least three out of the six cars in need of charging will be moved to a charging station, and we end up with the desired number of available cars of 27 (on average).

$$t_{em} + T_r^H\cdot x_{erm} + \sum_{r_1\in\mathcal{R}\backslash\{r\}} T_{d(r),o(r_1)} x_{e,r_1,(m+1)} - M_r(1 - x_{erm}) \le t_{e,(m+1)},$$
$$e \in \mathcal{E}, r \in \mathcal{R}, m \in \mathcal{M}\backslash\{|\mathcal{M}|\} \tag{1k}$$

$$(T_e^{SO} + T_{o(e),o(r)})\cdot x_{er1} \le t_{e1}, \qquad e \in \mathcal{E}, r \in \mathcal{R} \tag{1l}$$

$$T_r^{SC} x_{erm} \le t_{em}, \qquad e \in \mathcal{E}, r \in \mathcal{R}, m \in \mathcal{M} \tag{1m}$$

$$t_{e|\mathcal{M}|} + \sum_{r\in\mathcal{R}} T_r^H x_{er|\mathcal{M}|} \le \overline{T}^1, \qquad e \in \mathcal{E} \tag{1n}$$

Constraints (1k)–(1n) handle the temporal aspects of the relocation tasks. Constraints (1k) state that a new task cannot start until the previous task has been completed. Here $M_r$ is a sufficiently large constant. Note here that the parameter $T_{d(r),o(r_1)}$ is the travel time (either by folding bikes or public transport, whichever is fastest) between the destination zone of car-move $r$ to the origin zone of car-move $r_1$. Hence, the constraints ensure that if an employee performs car-move $r$ as it task number $m$, it cannot perform any car-move $r_1$ as it next task $m+1$ before the employee has finished car-move $r$ (second term on the lhs) plus the time to travel between the two car-moves (third term on the lhs). Constraints (1l) make sure that the first task of each employee starts after the earliest start time for the employee adjusted with the travel time from the employee's origin to the origin of the car-move. Constraints (1m) ensure that the start time of each task for each employee is after the earliest start time for the specific car-move. Finally, Constraints (1n) make sure that the last task for each employee is completed before the beginning of the target period.

Constraints (1o)–(1y) handle the logic of the second decision stage concerning the assignment of cars to customer requests. Observe that these constraints have to hold almost surely and thus for each scenario.

$$\sum_{e\in\mathcal{E}}\sum_{v\in\mathcal{V}^B}\sum_{r\in\mathcal{R}_i^{CD}\cap\mathcal{R}_v^V}\sum_{m\in\mathcal{M}} x_{erm} + \sum_{v\in\mathcal{V}^B}\sum_{d\in\mathcal{D}_i^{CD}(\xi_s)}\sum_{l\in\mathcal{L}} y_{vdls} \le N_i^{CS},$$
$$i \in \mathcal{I}^{CS}, s \in S \tag{1o}$$

Constraints (1o) ensure that in none of the scenarios considered the capacities of the charging zones are exceeded.

$$\sum_{v\in\mathcal{V}}\sum_{l\in\mathcal{L}_d} y_{vdls} \le 1, \qquad d \in \mathcal{D}(\xi_s), s \in S \tag{1p}$$

$$\sum_{d\in\mathcal{D}(\xi_s)}\sum_{l\in\mathcal{L}_d} y_{vdls} \le 1, \qquad v \in \mathcal{V}, s \in S \tag{1q}$$

Constraints (1p) make sure that, for each scenario, each request is satisfied at most by one car at some fee level. Constraints (1q) ensure that each car satisfies at most one request at some fee level.

$$\sum_{l\in\mathcal{L}_{d_1}} y_{vd_1ls} + \sum_{d_2\in\mathcal{D}_{d_1}(\xi_s)}\sum_{l\in\mathcal{L}_{d_2}} y_{vd_2ls} \le z_{i(d_1),v}, \qquad d_1 \in \mathcal{D}(\xi_s), v \in \mathcal{V}, s \in S \tag{1r}$$

$$y_{vd_1l_1s} + \sum_{d_2\in\mathcal{D}_{d_1}(\xi_s)}\sum_{l_2\in\mathcal{L}_{d_2}} y_{vd_2l_2s} + \sum_{v_1\in\mathcal{V}:v_1\neq v} y_{v_1d_1l_1s} \ge \lambda_{i(d_1),j(d_1),l_1} + z_{i(d_1),v} - 1,$$
$$d_1 \in \mathcal{D}(\xi_s), v \in \mathcal{V}, l_1 \in \mathcal{L}_{d_1}, s \in S \tag{1s}$$

Constraints (1r) state that a given car can satisfy a request $d_1$ only if it is available in the zone of the request and it is not used by a customer arriving earlier, i.e., a customer with a lower index. Constraints (1s) state that a request $d_1$ at a given fee level $l_1$ must be satisfied by car $v$ if fee level $l_1$ is chosen for that request and the car is available in the origin zone, unless the car has been used to satisfy a request $d_2$ with higher priority or the request $d_1$ has been satisfied by another car $v_1$.

$$\sum_{v\in\mathcal{V}} y_{vdls} \le \lambda_{i(d),j(d),l}, \qquad d \in \mathcal{D}(\xi_s), l \in \mathcal{L}_d, s \in S \tag{1t}$$

Constraints (1t) state that request can be satisfied at a given fee level $l$ only if the fee level between its origin and destination has been chosen in the first stage.

$$\lambda_{ijl} \in \{0,1\}, \qquad i,j \in \mathcal{I}, l \in \mathcal{L} \tag{1u}$$

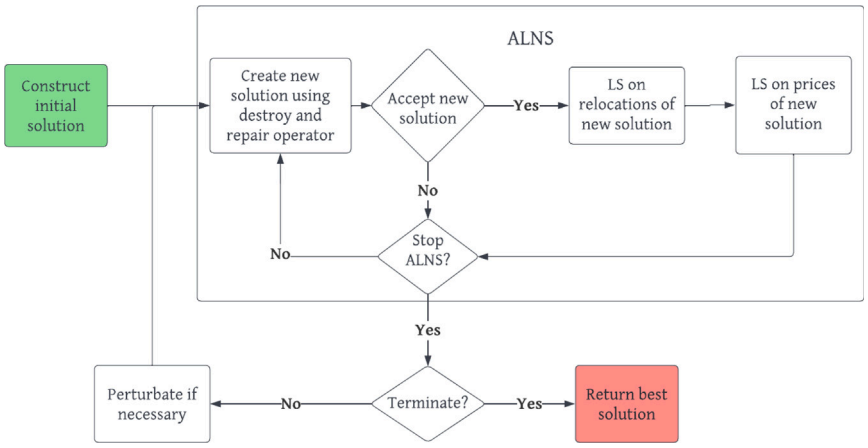


**Fig. 1.** Flowchart presenting the different elements of the heuristic.

$$z_{iv} \in \{0,1\}, \quad i \in \mathcal{I}, v \in \mathcal{V} \tag{1v}$$

$$x_{erm} \in \{0,1\}, \quad e \in \mathcal{E}, r \in \mathcal{R}, m \in \mathcal{M} \tag{1w}$$

$$t_{em} \geq 0, \quad e \in \mathcal{E}, m \in \mathcal{M} \tag{1x}$$

$$y_{vdls} \in \{0,1\}, \quad v \in \mathcal{V}, d \in \mathcal{D}(\xi_s), l \in \mathcal{L}, s \in \mathcal{S} \tag{1y}$$

Finally, Constraints (1u)–(1y) define the domain of the decision variables.

Observe that the model assumes that, in the target period, each car satisfies at most one request. In practice, it is possible that the same car is used more than once during the target period, e.g., for several short trips. Therefore, the model proposed provides a pessimistic estimate of actual profits during the target period.

## 4. Solution algorithm

The SE-VRePP is an extension of the car-sharing relocation studied by Hellem et al. (2021), where it was shown that a heuristic was needed to solve the problem. Furthermore, the problem is to be solved in a setting where relatively short solution times (i.e., around 10 min) are required to work in a real life setting. For this reason, we propose a heuristic solution algorithm. Particularly, we develop an Adaptive Large Neighborhood Search (ALNS) that addresses pricing decisions, integrated with two different local search algorithms that address pricing and relocation decisions, respectively. An overview of the algorithm is given in Fig. 1.

The algorithm starts by providing an initial solution using a construction heuristic. The initial solution is continuously improved based on different destroy and repair operators that act on the pricing decisions only, following the principles of ALNS (Ropke & Pisinger, 2006). Following, two local search heuristics apply smaller changes to the pricing and relocation decisions, respectively. The motivation for this choice is that we saw in the preliminary testing with the commercial MIP solver (Gurobi) that changes in the pricing decisions had a much larger impact on the objective value than changes in the relocation decisions. Hence, we decided to use a "heavy machinery" (i.e., the ALNS) for the pricing, while only using a "light machinery" (i.e., local search) for the relocation part. This process continues until a termination condition is met. The final element of the algorithm is the *perturbation* process following a principle based on Lourenço et al. (2003). This aims for diversifying the search even further in a somewhat similar manner as in Li et al. (2016) where they (i) both run their ALNS algorithm with multiple initial solutions, and (ii) use two destroy operators combined if the best solution has not improved in a number of iterations. In our algorithm, the perturbation is performed only when the solution has not improved for a number of iterations, and it works in a similar manner as a random destroy operator, although destroying a larger part of the pricing solution.

The different elements of the algorithm are described in detail in the following sections.

### 4.1. Solution representation and heuristic objective function

A solution $\gamma$ consists of both a relocation solution, $\gamma^E$, and a pricing solution, $\gamma^P$ (the $x$ and $\lambda$ variables of the model presented in Section 3.2, respectively). The relocation solution, $\gamma^E$, consists of an ordered set of tasks (car-moves) to be performed by each employee. The pricing solution, $\gamma^P$, is an $|\mathcal{I}| \times |\mathcal{I}|$ matrix where entry $(i, j)$ represents the fees between zones $i$ and $j$.

The calculation of the value of a heuristic solution is based on objective function (1a). As the representation of the model is different in the heuristic model than in the mathematical formulation, we use a different notation to formulate the *heuristic objective function*. Let $\gamma^E_e$ denote the ordered list of car-moves to be performed by employee $e$ in the relocation solution, and $r$ denote a car-move in this list. Further, recall that the per-minute cost of performing a car-move is $C^R$, and that the time to complete car-move $r$ is denoted as $T^H_r$. The heuristic evaluation of the relocation solution can then be written as

$$f_E = -\sum_{e \in \mathcal{E}} \sum_{r \in \gamma^E_e} C^R T^H_r. \tag{2}$$

Let $\gamma^P_{ij}$ denote the assigned fee $l$ between zones $i$ and $j$ in the pricing solution. Further recall that $\mathcal{D}(\xi_s)$ is the set of requests $d$ for a given realization of $\xi_s$ in scenario $s$, and that $\mathcal{L}_d$ is the set of fees $l$ sufficiently low to satisfy request $d \in \mathcal{D}(\xi_s)$ for a given realization of $\xi_s$ in scenario $s$. Each request $d$ has an origin zone $i(d)$ and a destination zone $j(d)$. For a given pricing solution $\gamma^P_{ij}$, we create a subset of $\mathcal{D}(\xi_s)$ with all potential requests $d \in \mathcal{D}(\xi_s)$ with origins $i(d)$ and destinations $j(d)$ which could be satisfied by the fee $l = \gamma^P_{i(d),j(d)}$. Mathematically, this subset, denoted as $\mathcal{D}^H(\xi_s)$, is

$$\mathcal{D}^H(\xi_s) = \{d \mid d \in \mathcal{D}(\xi_s) \wedge \gamma^P_{i(d),j(d)} \in \mathcal{L}_d\}.$$

The set $\mathcal{D}^H(\xi_s)$ is ordered by customer priority, so a request in the set has precedence over all subsequent requests. To determine whether a request $d \in \mathcal{D}^H(\xi_s)$ is allocated a vehicle in scenario $s$, we must consider the following three requirements: There is a vehicle in the request origin zone, $i(d)$; If the vehicle is in need of charging, the destination zone, $j(d)$, must have a charging station with available capacity; The vehicle has not previously been allocated to serve another request with precedence over the current request. The heuristic handles this by

iterating over the ordered set of requests $\mathcal{D}^H(\xi_s)$ for each scenario $s$ while checking the three requirements, resulting in the set of *satisfied requests* for each realization of $\xi_s$ for all scenarios $s$, $\mathcal{D}^{SR}(\xi_s)$. We can therefore express the heuristic evaluation of the pricing solution as

$$f_P = \frac{1}{|S|} \sum_{s \in S} \left( \sum_{d \in \mathcal{D}^{SR}(\xi_s)} R_{d,\gamma^P_{i(d),j(d)}} \right). \tag{3}$$

Here $S$ is the set of scenarios and $R_{d,\gamma^P_{i(d),j(d)}}$ is the profit of satisfying request $d$ with fee level $l = \gamma^P_{i(d),j(d)}$, as described in Section 3.1. Combining and maximizing the expressions in Eqs. (2) and (3), we get the *heuristic objective function* defined as $f_H = f_E + f_P$.

### 4.2. Constructing the initial solution

The construction heuristic starts with an empty solution, $\gamma_\emptyset$, and modifies it until a feasible solution is obtained. In the relocation part of the empty solution, $\gamma^E_\emptyset$, none of the employees have any tasks to perform. In the pricing part of the empty solution, $\gamma^P_\emptyset$, the fee assigned for all pairs of zones is the lowest fee.

The only constraint that might make an empty solution infeasible is Constraint (1j), which requires a certain share of the cars with low battery to be plugged in within the target period. To satisfy this constraint, either employees need to relocate the cars to charging stations or customers need to drop off cars to charging stations. To find a feasible solution, charging-moves are added to the task-list of the employees $\gamma^E$, provided there is space left. This procedure of adding charging-moves is repeated until the solution satisfies Constraint (1j).

To illustrate this, consider a case with six cars in need of charging. Assume Constraints (1j) demand that at least 50% of the cars in need of charging are to be put to charging. Furthermore, assume that in every scenario, only one customer wants to drop off the car in need of charging at a charging zone. In this case, two cars still need to be moved to charging stations. To meet this requirement, these cars must be relocated by employees. The construction heuristic then iterates over all car-moves moving a car in need of charging to a charging zone, unless the car is the one moved by a customer. Car moves are then randomly assigned to employees provided that their set of tasks respects time limit constraints.

### 4.3. Adaptive large neighborhood search

An outline of the Adaptive Large Neighborhood Search (ALNS) is shown in Algorithm 1. A set of operators are used in the ALNS to explore the neighborhood of a pricing solution. These are divided into two categories, namely destroy and repair operators. The destroy operator breaks down a pricing solution, bringing the solution closer to an empty solution. The repair operator takes the destroyed pricing solution and rebuilds it to a complete solution.

**Algorithm 1** ALNS of the current solution

```
function ALNS(γ)
    currentSolution = γ
    bestSolution = γ
    visitedSolutions = [currentSolution] ▷ list of all solutions visited to this point
    temp = calculate initial temperature
    coolingRate = calculate cooling rate
    while Stopping criterion not met do
        newSolution = choose destroy operator and destroy currentSolution
        newSolution = choose repair operator and repair newSolution
        if (newSolution not in visitedSolutions) AND (newSolution accepted by simulated annealing) then
            add newSolution to visitedSolutions
            if newSolution better than currentSolution then
                perform best improving local search on prices in newSolution
            else
                perform first improving local search in newSolution
            end if
            perform local search on relocations in newSolution
            if newSolution better than bestSolution then
                bestSolution = newSolution
            end if
            add newSolution to visitedSolutions    ▷ newSolution has been updated
            currentSolution = newSolution
        end if
        update temperature based on cooling rate
        if Time for new segment then
            Start new segment
        end if
    end while
    return bestSolution
end function
```

#### Destroy operators

The destroy operators break down the pricing solution by removing the assigned fees for a predefined number of zone pairs, $\eta$. The ALNS algorithm has three different destroy operators to choose from.

The *random removal* operator randomly chooses $\eta$ pairs of zones and removes their assigned fees. Random removal helps diversify the search space and therefore decreases the chance of getting stuck in a local optimum.

The *worst removal* operator removes the $\eta$ zone pairs that contribute the least to the objective value. In combination with a repair operator, this operator helps replace the worst fees with potentially better ones. To help diversify the search, a determinism parameter, $p_{worst} \geq 1$, is included. The list, $L$, of zone pairs is sorted in non-decreasing order of contribution to the objective value (the zone pair at the top of the ranking is the one with the fee that contributes the least to the objective value). The zone pair to be removed from the solution and $L$ is found in $L$ at index $i = y^{(p_{worst})} * |L|$, where $y \in [0, 1)$ is a uniformly random number. Consequently, a low value of $p_{worst}$ increases the degree of randomness, and vice versa. This process is repeated $\eta$ times.

The *related removal* operator removes $\eta$ zone pair fees that are similar based on certain criteria. Shaw (1998) first introduced related removal with the intention of removing similar parts of a solution so that the reconstruction and maintenance of feasibility would be easier. This was also used in the original ALNS by Ropke and Pisinger (2006). We adapt the related removal operator to fit the pricing solution of the SE-VRePP. In order to compare different tuples of zone pairs and fees, $t = ((i, j), l)$, a relatedness measure $R(t_1, t_2)$ is used. This measure examines the travel distance between both the origin, $d^o(t_1, t_2)$ and destination $d^d(t_1, t_2)$ of the tuples $t_1$ and $t_2$. In addition, the relatedness considers the difference in fees for $t_1$ and $t_2$. Let $l_1$ and $l_2$ be the fees of the zone pairs of $t_1$ and $t_2$, respectively. The complete relatedness measure between $t_1$ and $t_2$ is then given by

$$R(t_1, t_2) = q_o d^o(t_1, t_2) + q_d d^d(t_1, t_2) + q_f |l_1 - l_2|, \tag{4}$$

where $q_o, q_d$ and $q_f$ are weights for the origin zone distance, destination zone distance and difference in fees, respectively.

First, a tuple $t_1$ with its zone pair is randomly selected and added to a list of removed zone pairs, $L_{removed}$. Next, a random zone pair from $L_{removed}$ is used to compare to the other zone pairs that are not yet included in $L_{removed}$. The relatedness measure is calculated and sorted in ascending order in a new list $L_{relatedness}$. The lower the relatedness $R(t_1, t_2)$ is, where $t_2$ is a tuple with a zone pair not in $L_{removed}$, the more

**Table 1**
Rewards for destroy and repair operators.

| Parameter | Reward criterion | Description |
|---|---|---|
| $\rho_G$ | $f_H(\gamma') > f_H(\gamma_{best})$ | The new solution is a new global best. |
| $\rho_L$ | $f_H(\gamma') > f_H(\gamma_{current})$ | The new solution is a new local best, but not a new global best. |
| $\rho_N$ | $f_H(\gamma') < f_H(\gamma_{current})$, Accepted | The new solution is non-improving, but is accepted. |

similar the two pairs are. To allow for some randomness when selecting zone pair from $L_{relatedness}$ to remove and add to $L_{removed}$, a determinism parameter $p_{related} \geq 1$ is included. This works in the same manner as for the worst removal operator. The process of choosing zone pairs from $L_{removed}$, calculating relatedness with other zone pairs, sorting, and finally selecting the zone pair to remove is repeated $\eta$ times.

*Repair operators*

The repair operators take an incomplete pricing solution as input and returns a complete one where all zone pairs have an assigned fee. The repair operators go through the $\eta$ zone pairs with missing fees and assign these a new fee. The assignment of fees is done according to one out of three different repair operators.

The *random insertion* operator goes through the set of $\eta$ zone pairs with missing fees, and randomly assigns it a new fee $l$ from the set of fees $\mathcal{L}$. This operator helps diversify the search and can help mitigate the risk of getting stuck in a local optimum.

The *greedy insertion* operator greedily chooses the fee that increases the objective value the most for each of the $\eta$ destroyed zone pairs. The order of reassigning new fees to these zone pairs is random. This operator is an efficient way of finding potentially improving fees for the destroyed zone pairs.

The *random greedy insertion* operator includes some degree of randomness to the greedy insertion operator described above. Similar to worst and related removal, there is a determinism parameter $p_{greedy} \geq 1$. For each destroyed zone pair, the random greedy operator evaluates the possible fees, and inserts the fee and the heuristic objective value for using that fee into a list $L_{obj.val}$. The list is sorted in descending order, so the fee with the highest objective value is at the first index. The fee to be assigned to the zone pair is then chosen in the same manner as for the worst and related removal, where a value of $p_{greedy}$ close to one increases the randomness, and a high value of $p_{greedy}$ increases the chance of choosing the fee with the highest objective value. This process is repeated for each of the $\eta$ destroyed zone pairs. This operator combines the increased diversification from random insertion with the increased possibilities of choosing fees that positively affect the objective value from the greedy insertion operator.

*Choosing destroy and repair operators*

Let $\Omega^D$ and $\Omega^R$ be the set of all destroy operators and repair operators, respectively, and let $\Omega^O = \Omega^D \cup \Omega^R$. A pair containing one destroy operator, $d \in \Omega^D$, and one repair operator, $r \in \Omega^R$, is chosen in each iteration of the ALNS. The selection is based on the *roulette wheel selection principle* where the destroy and repair operators are drawn from a weighted probability distribution of all possible destroy and repair operators. Each pair of destroy and repair operators is associated with a weight, $w_o$, where $o \in \Omega^O$, and operators with a good performance in the past get a higher chance of being selected.

Running the ALNS is an iterative process divided into segments, i.e., a batch of iterations. During a segment, data regarding performance for each operator is collected. When a segment is over, we update the operator weights based on these data. The purpose of a segment is to update the weights at regular intervals, after the performance of the destroy and repair operators have been evaluated. This process defines the adaptive part of the ALNS.

At initialization, all weights are assigned a value of 1, making the probability of choosing an operator equal for all operators. During a segment, the chosen operators are assigned a reward, $\rho$, based on their performance. The rewards are accumulated over the iterations in a segment, resulting in an overall segment evaluation denoted as $\pi$. An operator can receive different rewards based on whether an operator contributes to a new global best solution, a new local best solution or accepts a non-improving local solution, as shown in Table 1.

At the end of a segment, the operators' weights are updated according to the following equation: $w_o = w_o(1-\alpha) + \alpha\frac{\pi_o}{\sigma_o}$, where $\pi_o$ is the accumulated reward for operator $o$, and $\sigma_o$ is the number of times operator $o$ is used in the most recent segment. $\alpha$ is a parameter between zero and one, used to control how large an impact the evaluation of an operator in the most recent segment has on the operator's weight. If $\alpha$ is closer to one, the weights are highly determined by the associated operators' success in the most recent segment. However, if $\alpha$ is closer to zero, the weights remain rather stable, only adapting slightly based on the evaluation from the most recent segment.

*Acceptance and stopping criterion*

When a new solution is constructed by the destroy and repair operators, it is evaluated to see whether it is accepted as the current new solution. We use the simulated annealing acceptance criterion, which is the most commonly used acceptance criterion within ALNS (Santini et al., 2018). We extend this acceptance criterion with the addition of a *tabu list* to keep track of already explored solutions, in order not to accept any previously evaluated solutions.

The ALNS algorithm stops after a predefined maximum number of iterations.

### 4.4. Local search algorithms

We extend the ALNS with local search heuristics with the aim of finding better solutions in the close neighborhood of the current solution. We include two different local search components, i.e., one to improve the pricing solution, $\gamma^P$, and one to improve the relocation solution, $\gamma^E$. The inputs to each of the local searches are the solution to be explored and a strategy, either being *best improving* or *first improving*. With a best improving search, the entire neighborhood defined by the given Local Search Operators (LSOs) is evaluated, while the first improving search explores the given neighborhood until a better neighbor is found, after which the solution is updated. Independently of the LSO applied, this process is repeated until a stopping criterion is met.

*Local search for the pricing solution*

We use a single LSO for the pricing solution denoted the *price move operator*. Recall that the pricing solution consists of a fee between all zone pairs. With the price move operator a solution is said to be a neighboring solution if all fees except one remain the same. Thereby, for a certain solution, one finds the neighbors by altering exactly one of the fees. The price move operator represents the neighborhood as a list of zone pairs. The list is constructed in a random order to broaden the search. The operator iterates over the list, testing different fees for each zone pair. With a *first improving* search strategy, the iteration stops when an improving solution is found. With a *best improving* search, the search would span $|\mathcal{I}|^2 \cdot |\mathcal{L}|$ zone pair fees, returning the best found solution.

**Table 2**
The test instance types used in the computational study. For each type of test instance, three different versions are generated.

| Test instance type | Zones | Cars | Employees | Customers per zone | Scenarios | Size |
|---|---|---|---|---|---|---|
| 5-4-1 | 5 | 4 | 1 | 10 | 25 | |
| 6-6-1 | 6 | 6 | 1 | 10 | 25 | Small |
| 8-8-1 | 8 | 8 | 1 | 10 | 25 | |
| 10-9-2 | 10 | 9 | 2 | 10 | 25 | |
| 15-12-2 | 15 | 12 | 2 | 10 | 25 | Medium |
| 20-15-2 | 20 | 15 | 2 | 10 | 25 | |
| 30-20-2 | 30 | 20 | 2 | 10 | 25 | |
| 40-25-2 | 40 | 25 | 2 | 10 | 25 | Large |
| 50-30-2 | 50 | 30 | 2 | 10 | 25 | |

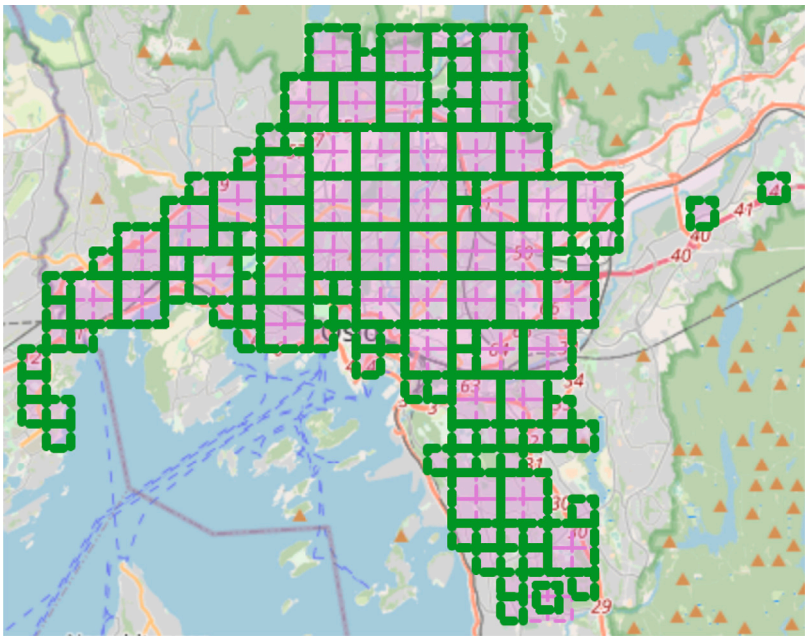

**Fig. 2.** The operating area set to Oslo, divided into a grid of 113 zones of sizes 800 × 800 m or 400 × 400 m, defined by the green lines.

*Local search for the operator-based relocation*

We propose three LSOs in this search. The *add move* and the *remove move* operators investigate whether it is profitable to add or remove a car-move from one of the employees' task lists, respectively. The evaluation is performed by calculating the objective values of the new solutions obtained. The *inter swap* operator investigates the effect of swapping car-moves present in the solution between employees. This will not have effects on the expected revenues but may reduce the relocation cost and time for each employee, and hence impact the objective value.

### 4.5. Complexity reduction

In order to increase the efficiency of the heuristic search, size reduction techniques and certain computational simplifications have been utilized.

*Customers*

As discussed in 3.2, the utility a transportation mode yields for a potential customer is composed of a deterministic and a stochastic term. This entails that a potential customer might have different preferences in different scenarios. This may result in a customer preferring car-sharing with given price levels in some scenarios, while preferring other means of transportation in others. To reduce the search space, we attempt to separate the customers into two categories: those considered *relevant customers* and those considered *irrelevant customers*. A potential customer is considered relevant if he/she prefers the car-sharing option in at least $\phi_{customer} \cdot 100\%$ of the scenarios, and irrelevant if not. Only the relevant customers are considered in the search. As an example, suppose $\phi_{customer} = 0.7$ and there are 25 scenarios (which are the values we use in our tests). In this case, only the customers that have car-sharing as their preferred mode of transportation in at least $0.7 \cdot 25 = 17.5$ (i.e., 18) of the scenarios will be included.

*ALNS search space*

To improve the performance of the ALNS, the destroy and repair operators are only changing the prices of zone pairs that are found relevant. A zone pair is considered relevant if it fulfills three conditions: (1) There is at least one car present in the zone pair's origin zone. This might be due to the car being there in the first place or that an employee has relocated a car to this zone. (2) There is at least one relevant customer in the zone pair's origin zone. This has the effect of finding a zone where there is demand for cars. (3) Among the relevant customers, at least one has the zone pair's destination zone as its desired destination. By only evaluating the relevant zone pairs, we avoid evaluating changes in zone pairs that do not have an impact on the objective value. It should be noted that this concept of relevant zone pairs is considered also when the initial solution is constructed.

*Local search space*

The local search for pricing solutions uses the same search space as defined for the ALNS by only investigating what we define as relevant

**Table 3**
Comparison of the ALNS and Gurobi. Three versions of each test instance type are generated. Each test instance is solved ten times with the ALNS for a maximum of 600 s each time. Gurobi is likewise used to solve the full stochastic program with a 600 s time limit. Average objective values, runtimes and gaps are reported. The gap is based on the distance to the best upper bound found by Gurobi.

| Test instance type | Gurobi (600 s) | | | ALNS Heuristic (600 s) | | |
|---|---|---|---|---|---|---|
| | Objective value | Gap (%) | Time (s) | Objective value | Gap (%) | Time (s) |
| 5-4-1 | 252.76 | 0.00 | 13.11 | 252.76 | 0.00 | 3.45 |
| 6-6-1 | 411.32 | 0.00 | 32.43 | 411.32 | 0.00 | 8.31 |
| 8-8-1 | 570.81 | 0.00 | 132.77 | 570.81 | 0.00 | 67.82 |
| 10-9-2 | 579.98 | 3.28 | 600.00 | 590.50 | 1.40 | 133.83 |
| 15-12-2 | 819.51 | 13.06 | 600.00 | 817.68 | 13.30 | 274.80 |
| 20-15-2 | 833.49 | 44.56 | 600.00 | 968.09 | 17.62 | 318.13 |
| 30-20-2 | 1454.11 | 27.65 | 600.00 | 1567.24 | 17.06 | 376.45 |
| 40-25-2 | 1264.69 | 79.10 | 600.00 | 1786.20 | 22.65 | 426.45 |
| 50-30-2 | (–) | (–) | 600.00 | 2037.64 | (–) | 370.77 |
| Average Total | 687.42 | 18.63 | 419.81 | 1001.47 | 11.91 | 393.71 |

(–) in *Objective Value* indicates no feasible solution found.
(–) in *Gap (%)* indicates no feasible upper bound found.
Average values are found replacing (–) with 0.

**Table 4**
Comparison of the ALNS heuristic and the commercial solver Gurobi. Three versions of each test instance type are generated. Each test instance is run ten times with the ALNS heuristic for a maximum of 3600 s. The results from Gurobi after a maximum of 3600 s are registered as a benchmark. Average objective values, runtimes and gaps are reported. The gap is based on the distance to the best upper bound found by Gurobi.

| Test instance type | Gurobi (3600 s) | | | ALNS Heuristic (3600 s) | | |
|---|---|---|---|---|---|---|
| | Objective value | Time (s) | Gap (%) | Objective value | Time (s) | Gap (%) |
| 10-9-2 | 597.65 | 2844.57 | 0.00 | 588.71 | 402.91 | 1.53 |
| 15-12-2 | 861.34 | 3600.96 | 4.82 | 821.18 | 494.93 | 9.93 |
| 20-15-2 | 1099.88 | 3600.29 | 3.24 | 993.66 | 584.22 | 14.28 |
| 30-20-2 | 1746.79 | 3601.52 | 6.61 | 1607.98 | 1801.81 | 15.78 |
| 40-20-2 | 2155.20 | 3600.31 | 10.02 | 1993.30 | 2973.28 | 18.89 |
| 50-30-2 | (–) | 3600.57 | (–) | 2222.51 | 1875.70 | (–) |
| Average Total | 1076.81 | 3334.17 | 4.11 | 1200.97 | 1355.48 | 10.07 |

(–) in *Objective Value* indicates no feasible solution found.
(–) in *Gap (%)* indicates no feasible upper bound found.
Averages are found replacing (–) with 0.

**Table 5**
Number of operator-based relocations performed with and without adaptable prices. The results are average values from running the ALNS heuristic ten times on three versions of each test instance type, with a time limit of 600 s per run.

| Instance type | Uniform pricing | Adaptable pricing |
|---|---|---|
| 5-4-1 | 1.00 | 2.10 |
| 6-6-1 | 1.00 | 2.77 |
| 8-8-1 | 1.00 | 1.83 |
| 10-9-2 | 1.00 | 2.33 |
| 15-12-2 | 1.00 | 4.63 |
| 20-15-2 | 1.93 | 7.27 |
| 30-20-2 | 2.87 | 7.13 |
| 40-25-2 | 2.63 | 5.70 |
| 50-30-2 | 4.00 | 6.13 |
| Average | 1.83 | 4.43 |

zone pairs. The local search for relocations has extended this definition to include car-moves that are considered relevant when using the *add move* LSO. For each zone, we calculate a cars-to-customers ratio. This ratio is calculated by including only relevant customers in the zone, and all cars originating or currently being in the zone as a consequence of relocation. If a given zone has a cars-to-customers ratio exceeding $\phi_{cars-to-customers}$, all car-moves with the given zone as destination zone are excluded from the relevant car-moves. This reduction in search space makes the algorithm only consider moving cars to areas which do not already have a high density of cars compared to customers.

To restrict the search space even further, the car-moves are filtered based on the time it takes to perform them. If the relocation time of a car-move exceeds $\phi_{relocation-time} \cdot \overline{T}^1$ the car-move is considered irrelevant. The only exception is if the car to be relocated is in need of charging. This is done to eliminate very time-consuming car-moves which are unlikely to be part of the optimal solution.

A final effort to reduce runtime is defined through the LSOs themselves. Each LSO has a time limit of $T_{LSO}$ seconds, avoiding a *best improving* search from being a bottleneck in larger instances of the problem. Furthermore, the local search has an overall time limit of $T_{LS}$ seconds. To avoid repeatedly investigating the same part of the search space for $T_{LSO}$ seconds, the search space is randomly ordered at the beginning of each local search.

## 5. Computational study

This section presents the computational study. The solution algorithm was implemented in *Python 3.9.6* while we used *Gurobi 9.5.0* as the solver of the two-stage stochastic program. In the following, Section 5.1 describes the generation of the test instances, before we test the performance of the proposed heuristic in Section 5.2. Finally, we test the effect of including pricing decisions in Section 5.3.

### 5.1. Data and test instances

We generate a number of test instances based on data from Vybil, a CSO operating in Oslo, Norway. Particularly, we divide the operating area into 113 zones of size 800 × 800 or 400 × 400 meters as shown in Fig. 2. Each test instance is denoted by a code *X-Y-Z*, where *X* is the number of zones, *Y* is the number of cars, and *Z* is the number of employees. Table 2 summarizes the instances we generate and the respective number of zones, cars, employees, customers, and scenarios. The test instance types are divided into categories based on their sizes.

**Table 6**
The number of customers served (Requests Satisfied) and the average profit they generate (Profit per Request) using uniform and adaptable pricing strategies. The results are average values from running the ALNS heuristic ten times on three versions of each test instance type, with a time limit of 600 s per run.

| Instance type | Uniform pricing | | Adaptable pricing | |
|---|---|---|---|---|
| | Requests satisfied | Profit per request | Request satisfied | Profit per request |
| 5-4-1 | 2.53 | 57.04 | 3.36 | 79.23 |
| 6-6-1 | 4.45 | 54.18 | 4.75 | 79.11 |
| 8-8-1 | 6.16 | 64.13 | 6.77 | 79.25 |
| 10-9-2 | 7.40 | 52.98 | 7.71 | 75.41 |
| 15-12-2 | 10.37 | 57.41 | 10.45 | 76.20 |
| 20-15-2 | 11.54 | 56.40 | 12.18 | 76.92 |
| 30-20-2 | 16.26 | 66.93 | 17.11 | 75.96 |
| 40-25-2 | 20.59 | 69.99 | 20.48 | 76.51 |
| 50-30-2 | 25.01 | 74.46 | 24.51 | 77.56 |
| Average | 11.59 | 61.50 | 11.93 | 77.35 |

**Table 7**
Evaluation of how pricing decisions affect the overall profit during a work day. The accumulated objective values over the ten hours in the work day are displayed in the columns Objective Value. The column "Increase" shows the average increase in profit from having adaptable prices. The average runtimes the ALNS heuristic spent finding the best known solution for each hour are displayed in the columns Time (s).

| Instance type | Uniform prices | | Adaptable prices | | |
|---|---|---|---|---|---|
| | Obj. Value | Runtime | Obj. Value | Increase (%) | Runtime |
| 5-4-1 | 1731.09 | 3.17 | 2246.12 | 29.8 | 40.12 |
| 6-6-1 | 2842.57 | 6.85 | 3514.55 | 23.6 | 101.71 |
| 8-8-1 | 3838.62 | 18.05 | 4757.58 | 23.9 | 261.65 |
| 10-9-2 | 3674.10 | 52.82 | 5071.25 | 38.0 | 297.26 |
| 15-12-2 | 5190.65 | 101.00 | 6817.08 | 31.3 | 520.91 |
| 20-15-2 | 6951.42 | 185.79 | 8686.37 | 25.0 | 521.26 |
| 30-20-2 | 10 101.74 | 276.44 | 11 069.95 | 9.6 | 441.68 |
| 40-25-2 | 13 746.34 | 429.30 | 15 238.85 | 10.9 | 456.56 |
| 50-30-2 | 17 335.61 | 442.62 | 18 860.54 | 8.8 | 480.31 |
| Average | 7268.02 | 168.45 | 8473.59 | 16.6 | 346.83 |

For each type of test instance, three different random instances are generated.

The number of charging zones is dependent on the number of zones in the test instance. As Vybil offers 35 parking lots, out of which nine are equipped with charging stations,[1] we apply a ratio of charging zones per zone of 25%. We assume that each charging station has a capacity of three cars. Furthermore, the number of cars used by Vybil in need of charging is typically between 10% and 15% of their fleet. Thus, in what follows we assume that 15% of the cars require charging by the end of the target period. The size of the set of customers, $|\mathcal{K}|$, is arbitrarily set to ten times the number of zones in the instance. We use Google Maps API *Distance Matrix*,[2] to generate travel times for the various transport means ($T^T_{ijt}$) and relocation times ($T^H_r$).

For a given instance, the initial position of cars and employees and the origin and destination of the customers are randomly generated using the technique of Pantuso (2020, 2022), which assigns to each zone a probability depending on the distance from the center. The instances are generated in such a way that the zones closer to the center are more probable.

Employees are defined by their origin location and the earliest time at which they can start performing relocations. The start time is drawn from a weighted distribution ranging from 0 to 15 (minutes), where values closer to 0 are more likely to be drawn. A car is defined by its origin zone and whether or not it is in need of charging. Cars in need of charging are uniformly drawn among the set of all cars. The set of possible car-moves that are generated for a car $v$, $\mathcal{R}_v$, includes car-moves to all zones to which car $v$ can be relocated. If a car $v$ requires charging, $v \in \mathcal{V}^B$, only car-moves to charging zones are allowed. We assume we plan one hour in advance of the target period, thus we set $\overline{T}^1 = 60$ minutes, and we assume that each employee can perform at most $|\mathcal{M}| = 5$ car moves during the 60 minutes. The cost of relocation activities is set to NOK (Norwegian Kroner) 0.43 per minute, based on the typical energy consumption of the electric cars used and Norwegian electricity prices.

We assume that the city offers public transportation (PT) and bicycle (B) as alternatives to car-sharing, that is $\mathcal{T} = \{CS, PT, B\}$. We use choice model (5), introduced by Modesti and Sciomachen (1998) and used, e.g., by Hansen and Pantuso (2018), Pantuso (2020, 2022) as a proxy for the behavioral model of the customers. It must be noted, however, that since the model is not validated by data, it does not allow us to draw sensible conclusions related to how people react to prices and incentives. Its purpose is solely that of testing the model and the method introduced.

$$F_{kt}(p_{ijt}, T^T_{ijt}, T^{Walk}_{ijt}, T^{Wait}_{ijt}) = \beta^P_k p_{ijt} + \beta^{CS}_k T^T_{ijt}\delta_{CS}(t) + \beta^{PT}_k T^T_{ijt}\delta_{PT}(t) + \beta^B_k T^T_{ijt}\delta_B(t) + \beta^{Walk}_k T^{Walk}_{ijt} + \beta^{Wait}_k T^{Wait}_{ijt} + \xi_{kts}. \quad (5)$$

The model expresses the (dis)utility of customer $k$ traveling between their origin and destination, say $i$ and $j$, with a given transport mean $t$, as a function of price $p_{ijt}$, walking time $T^{Walk}_{ijt}$ required e.g., to commute or to reach transit stations, waiting time $T^{Wait}_{ijt}$, and travel time $T^T_{ijt}$. Function $\delta_A(t) : \mathcal{T} \to \{0, 1\}$ is the indicator function $\delta_A(t) = 1$ if $t \in \{A\}$, 0 otherwise. The $\beta$ parameters reflect the sensitivity of the customer with respect to the different attributes and their values are provided Table 11 in Appendix A. Particularly, we assume all customers have the same sensitivity to all parameters except price. To model different price sensitivities we set two different values of $\beta^P_k$, say high and low (also provided in Table 11), and assign each customer a price sensitivity level with equal probability. We assume the $\tilde{\xi}_{kt}$ random variables are independent and follow a Gumbel distribution, thus obtaining a Logit choice model. Particularly, for the Gumbel distribution we use the standard deviation of the deterministic part of the customers utility ($F_{kt}$) across all $t$ and $k$. We use 25 scenarios given by iid samples from the underlying Gumbel distribution.

Finally, we set a NOK 6 per-minute fee for car-sharing and we define five different pick-up/drop-off fees, namely $\mathcal{L} = \{-40, -20, 0, 20, 40\}$ NOK. Observe that negative fees represent a bonus for the user. Bicycles are assumed to be free of charge and we assume a public transport ticket costs NOK 38 between all pairs of relevant zones, as is the case in Oslo.

### 5.2. Computational results

We performed a parameter tuning of the ALNS heuristic on a separate set of test instances, following the procedure by Ropke and Pisinger (2006). The parameters to tune, their initial values, their

[1] https://www.vy.no/alt-om-reisen/andre-transportmidler/vybil/vybil-parkeringsplass accessed 2022-05-26.

[2] https://developers.google.com/maps/documentation/distance-matrix/overview accessed 2021-11-10.

**Table 8**
Sets used in the model.

| Notation | Explanation |
|---|---|
| $\mathcal{I}$ | Set of zones |
| $\mathcal{I}^{CS}$ | Set of charging zones |
| $\mathcal{E}$ | Set of employees |
| $\mathcal{M}$ | Set of abstract employee tasks |
| $\mathcal{V}$ | Set of car-sharing vehicles |
| $\mathcal{V}^B$ | Set of vehicles in need of charging |
| $\mathcal{R}$ | Set of car-moves |
| $\mathcal{R}^V_v$ | Set of car-moves for vehicle $v$ |
| $\mathcal{R}^N_i$ | Set of car-moves with destination zone $i$ |
| $\mathcal{R}^{CD}_i$ | Set of car-moves with destination zone $i \in \mathcal{I}^{CS}$ |
| $\mathcal{K}$ | Set of customers |
| $\mathcal{K}_i$ | Set of customers traveling from zone $i$ |
| $\mathcal{K}_{ij}$ | Set of customers traveling from zone $i$ to $j$ |
| $\mathcal{S}$ | Set of scenarios |
| $\mathcal{L}$ | Set of possible combined pick-up and drop-off fee levels |
| $\mathcal{D}(\xi_s)$ | Set of requests for a given realization of $\xi_s$ in scenario $s$ |
| $\mathcal{D}^{CD}(\xi_s)$ | Set of requests with destination in a charging zone for a given realization of $\xi_s$ in scenario $s$ |
| $\mathcal{D}^{CD}_i(\xi_s)$ | Set of requests with destination in charging zone $i$ for a given realization of $\xi_s$ in scenario $s$ |
| $\mathcal{D}_d(\xi_s)$ | Set of requests that has precedence over request $d$ in scenario $s$ |
| $\mathcal{D}_{ij}(\xi_s)$ | Set of requests going from zone $i$ to $j$ in scenario $s$ |
| $\mathcal{L}_d$ | Set of combined pick-up and drop-off fee levels that are less than or equal to the highest fee accepted for request $d$ |

**Table 9**
Parameters used in the model.

| Notation | Explanation |
|---|---|
| $C^R$ | Accumulated toll, maintenance and energy (electricity) cost per minute of driving a car |
| $T^H_r$ | The time it takes to perform the car-move $r$ |
| $P_s$ | Probability of scenario $s$ occurring |
| $R_{dl}$ | Profit of satisfying request $d$ with pick-up fee level $l$ |
| $o(v)$ | Origin of car $v$ at start of planning horizon |
| $o(r)$, $d(r)$ | Origin and destination of car-move $r$, respectively |
| $T^{SO}_e$ | Earliest start time employee $e$ is available to perform a task |
| $o(e)$ | Origin of employee $e$ at start of planning horizon |
| $\overline{T}^1$ | Time length of the first stage |
| $T^{SC}_r$ | First available time a car is available for car-move $r$ |
| $T_{ij}$ | The time it takes to travel between zone $i$ and $j$ by bicycle |
| $L_l$ | Value of a fee at fee level $l$ |
| $N^{CS}_i$ | Number of available charging slots in zone $i \in \mathcal{I}^{CS}$ |
| $N^B$ | Share of cars in need of charging which needs to be charged in the planning horizon |
| $i(d)$, $j(d)$ | Origin and destination of customer request $d$, respectively |
| $k(d)$ | Customer of customer request $d$ |
| $l(d)$ | The highest fee level at which customer $k(d)$ would prefer car-sharing to other transportation modes |
| $M_r$ | Big-M notation for each car-move $r$ |

**Table 10**
Decision variables used in the model.

| Notation | Explanation |
|---|---|
| $z_{iv}$ | 1 if vehicle $v$ is available to customers in zone $i$ at the beginning of the second stage, at time $\overline{T}^1$, 0 otherwise. |
| $\lambda_{ijl}$ | 1 if fee level $l$ is used between zones $i$ and $j$, 0 otherwise. |
| $x_{erm}$ | 1 if employee $e$ performs car-move $r$ as her task number $m$, 0 otherwise. |
| $y_{vdls}$ | 1 if vehicle $v$ satisfies request $d$ with the fee level $l$ in scenario $s$, 0 otherwise. |

tuning intervals and their final values are summarized in Appendix C. It should be noted that the performance of the ALNS was quite robust with respect to its parameter values. We also assessed the value of adaptivity in the ALNS, and it emerged that it yields an average objective value improvement of 2.4%, which is higher that what was found in the meta-analysis by Turkeš et al. (2021). We also tested the value of including the two local search algorithms on the pricing part of the problem (i.e., on top of the ALNS). This showed that including the local search gave an average improvement of 6.4%.

In the following we compare the performance of our heuristic with the commercial solver. We use a time limit of 600 seconds to test the algorithm on real-life-like settings. The average results over the three versions of each instance type are presented in Table 3. To account for the randomness of the algorithm, the objective values reported for the heuristic are averages over ten runs of each of the three versions of the instance types. The gaps are measured as the distance in the objective values in percentage to the best upper bounds found by the commercial solver, Gurobi. Note that these gaps are pessimistic as they represent maximum distances to the optimal solutions.

We observe from Table 3 that the two solution methods have achieved the same optimal solutions for the small instances, with the ALNS being faster. For the remaining instances, the ALNS is superior to Gurobi in all but the instances of type $15-12-2$, where there is a minor difference in the gaps of 0.24%. The runtimes indicate that the ALNS heuristic finds the best-known solutions more efficiently than Gurobi. This supports the applicability of the ALNS heuristic as opposed to Gurobi in a real-life situation where decisions must be made frequently. The gaps to the upper bounds are somewhat large for the larger test instances, most likely due to poor upper bounds, but still smaller than those of the solutions found by Gurobi.

In Table 4 we compare the performance of Gurobi and the ALNS with a time limit of one hour (3600 s). It should be emphasized that a running time of one hour is significantly longer than what can be accepted in a real-life setting (as discussed above), which is considered

**Table 11**
Predefined parameters and set sizes common to all test instances.

| | Notation | Value |
|---|---|---|
| Number of scenarios | $\|S\|$ | 25 |
| Probability of scenarios | $P_s, s \in S$ | $1/\|S\|$ |
| Number of fees | $\|\mathcal{L}\|$ | 5 |
| Fees [NOK] | $L_l,\ l \in \mathcal{L}$ | {-40, 20, 0, 20, 40} |
| Beginning of the target period [minutes] | $\overline{T}^1$ | 60 |
| Initial capacity at charging station | $N_i^{CS}, i \in \mathcal{I}^{CS}$ | 3 |
| Cost of relocation per minute [NOK] | $C^R$ | 0.43 |
| Customer sensitivity to: | | |
| Price (P) | $\beta_k^P$ | {-7, -19} |
| Car-sharing travel time (CS) | $\beta_k^{CS}$ | −1 |
| Pubic transportation travel time (PT) | $\beta_k^{PT}$ | −2 |
| Bicycle travel time | $\beta_k^B$ | −2.5 |
| Walking distance | $\beta_k^{Walk}$ | −3 |
| Waiting time | $\beta_k^{Wait}$ | −6 |
| Prices per trip [NOK] using: | | |
| Car-sharing (CS) | $p_{ijCS}$ | $6 \cdot T_{ijCS}$ + fee |
| Public transportation (PT) | $p_{ijPT}$ | 38 |
| Bicycle | $p_{ijB}$ | 0 |

to be around ten minutes. We can observe that in the long run Gurobi delivers better solutions than the ALNS on most instances. Nevertheless, Gurobi fails to deliver a solution on the largest instances, which shows that a heuristic is needed (especially since we in practice need solutions within at around ten minutes).

### *5.3. Effects from pricing decisions and simulation over a longer planning period*

In this section, we report on the effect of including pricing decisions.

We start by assessing the effect of adjustable prices. Particularly, we compare the results obtained when solving the problem with and without the possibility to set zone-specific pick-up and drop-off fees. In the latter case we simply set the fee to 0 on all origin–destination pair, so that the only price paid by the customers is the per minute fee, as is common in most car-sharing services, including our case company. Our tests indicate that, on the instances presented in Table 2, the profit from including pricing decisions increases by 19.5% on average.

Pricing decisions have an impact also on the relocation decisions. The results in Table 5 illustrate that, when prices can adapt to the origin and destination zone, the number of relocations increases on average. Particularly, for instances larger than $10 - 9 - 2$, the average number of relocations performed nearly triples when introducing pricing decisions, going from 2.49 to 6.17. These results can be explained with the fact that pricing decisions allow exploiting the potential customers' willingness to pay. In fact, in Table 6, showing the number of customers served and their average profits, it can be observed that the average profit per request satisfied is significantly higher when adapting prices. As a consequence, a higher number of zones become attractive destinations for operator-based relocations. On the contrary, keeping prices fixed on the entire business area exposes to the risk of underselling the service in certain zones.

In a real-life setting, the pricing and relocation problem would be solved periodically, e.g., every hour in preparation for the following hour. The long-term effects of adjusting prices during the day can therefore be properly assessed only in a simulation framework. Hence, we developed a rolling horizon simulation framework, which attempts to replicate a real-life situation where the CSO makes frequent decisions concerning prices and operator-based relocations.

We periodically solve the SE-VRePP with our heuristic as follows. First, pricing and relocation decisions are made one hour ahead of the target period. Following, all operator-based relocations are performed during the first hour, thus before the beginning of the target period. After the relocations are performed, we enter the second stage (the target period) of the problem. A random scenario among all 25 possible scenarios is selected to be rolled out, and cars are allocated to customers. After this, we update the state of the system (i.e., for cars and employees) and move one hour ahead in time and do the same over again, until we have reached the end of the simulation period. Following this procedure, we simulate each test instance ten times for ten hours representing a work day from 7 a.m. to 5 p.m. The same procedure is adopted assuming no pricing decision are made, i.e., static pricing (only relocations). Table 7 summarizes the results of these simulations. Note that each line in the table represents the average values over three instances of the given instance type.

As shown in Table 7, we observe a significant improvement in objective values for all test instances when optimizing pricing decisions (adaptable prices) compared to having uniform prices. The average improvement over all test instances is 16.6%. An interesting aspect to consider is how the joint optimization of prices and relocations affects the need for employees to perform charging-moves. We notice that the numbers of customers satisfied do not significantly differ for the two pricing strategies, i.e., on average 120.7 customers satisfied with a uniform pricing strategy versus 118.2 with optimized prices. On the other hand, the average profit per request is significantly higher with optimized zone prices, i.e., 71.6 vs. 58.3 NOK/request. However, we have to stress that the behavioral model utilized is only a proxy for a validated model. Therefore, we are not able to provide a more thorough assessment of customer's choices.

Finally, from Table 7 it also emerges that the added value of adaptable prices decreases with the size of the fleet. This signals that a sufficiently large fleet is able to cover enough demand to partially compensate a uniform pricing strategy.

## 6. Summary and concluding remarks

We have in this paper studied the Stochastic Electric Vehicle Relocation and Pricing Problem (SE-VRePP), which integrates operator-based relocation decisions and pricing decisions in car-sharing systems when faced with uncertain future demand/customer behavior. We modeled the SE-VRePP as a two-stage stochastic programming model. Since this model could only be solved by a commercial solver for very small problem instances, we proposed a new heuristic solution algorithm, which uses an Adaptive Large Neighborhood Search (ALNS) heuristic combined with local search for the pricing part of the problem and another local search for the operator-based relocations.

We performed a computational study on test instances generated using realistic data from our case company, Vybil, which is a car-sharing organization (CSO) in Oslo, Norway. The tests showed that our heuristic found the same optimal solutions as the commercial solver for small instances and significantly better solutions for the larger and more realistic ones. Furthermore, simulation results linked optimized prices to substantial profit increases. Nevertheless, these results were obtained with a non-validated model of customer behavior, therefore no general conclusions can be drawn in this sense.

The research presented in this paper leaves room for further improvements. Particularly, in practice, customers choose not only based on pricing decisions but also based on the distance to the nearest shared car. This distance, in turn, depends on or can be influenced by relocation decisions. Extensions of the model presented could be developed to model more precisely the interaction between decisions and customers preferences. Furthermore, pricing and relocation decisions are in practice made periodically during the day as a result of the updated status of the system. A multistage extension of the model presented would serve better to capture this dynamics. Finally, our ALNS algorithm is relatively complex as it includes many components. It probably also has room for other improvements in order to provide even better solutions. Hence, there is a value in simplifying and improving our heuristic, something we leave for future research.

## Appendix A. Summary of notation

See Tables 8–10.

**Table 12**
The parameters used in the ALNS heuristic. The initial values, the tuning intervals and the determined values are displayed. Untuned parameters are displayed only with their determined values.

| Parameter | Initial value | Tuning interval | Determined value | Description |
|---|---|---|---|---|
| $\phi_{customer}$ | 1.0 | $[0.3, 0.7]$ | 0.7 | Relevant request threshold. |
| $\phi_{cars-to-customer}$ | 1.0 | $[0.3, 0.7]$ | 0.7 | Car-to-customer ratio threshold. |
| $\phi_{relocation-time}$ | 1.0 | $[0.3, 0.7]$ | 0.7 | Relocation time threshold. |
| $\rho_G$ | 10 | $[10, 40]$ | 20 | Weight reward parameter for a new globally best solution. |
| $\rho_L$ | 5 | $[10, 20]$ | 15 | Weight reward parameter for a new locally best solution. |
| $\rho_N$ | 1 | $[5, 10]$ | 10 | Weight reward parameter for a new non-improving solution. |
| $\alpha$ | 0.1 | $[0.4, 0.7]$ | 0.7 | Reward decay parameter. |
| $\eta$ | $[0.05, 1.00]$ | $[0.05, 0.70]$ | $[0.15, 0.70]$ | Neighborhood size distribution interval. |
| $q_o$ | 0.05 | $[0.05, 0.2]$ | 0.05 | Related Removal weight for origin zone distance. |
| $q_d$ | 0.05 | $[0.05, 0.2]$ | 0.05 | Related Removal weight for destination zone distance. |
| $q_f$ | 0.1 | $[0.1, 0.4]$ | 0.1 | Related Removal weight for fee difference. |
| $p_{worst}$ | 1 | $[1, 6]$ | 2 | Worst Removal determinism parameter. |
| $p_{related}$ | 1 | $[1, 6]$ | 2 | Related Removal determinism parameter. |
| $p_{greedy}$ | 1 | $[1, 6]$ | 4 | Random Greedy Insertion determinism parameter. |
| $\tau_{strategy}$ | *reheat* | $reheat, no-reheat$ | *reheat* | Reheat strategy for Simulated Annealing. |
| $\tau_{init}$ | | | $\frac{f(\gamma_{init}) \cdot 0.1}{\ln 0.5}$ | Initial temperature for Simulated Annealing. |
| $\tau_{final}$ | | | $\frac{f(\gamma_{best}) \cdot 0.1}{\ln (0.01)}$ | Final temperature for Simulated Annealing. |
| $\zeta$ | | | $1 - (\frac{\tau_{final}}{\tau_{init}})^{\frac{1}{I^{ALNS}}}$ | Cooling rate for Simulated Annealing. |
| $I^{ALNS}$ | | | 100 | Number of iterations within the ALNS. |
| $I^{S}$ | | | 10 | Number of iterations within a segment. |
| $I^{main}$ | | | 50 | Number of iterations in the main framework. |
| $T^{limit}$ | | | 600 s | Runtime limit for the heuristic. |
| $T^{LSO}$ | | | 0.5 s | Runtime limit for a local search operator. |
| $T^{LS}$ | | | 2.0 s | Runtime limit for local search. |
| $\kappa$ | | | 50% | Share of pricing solution set to initial values in perturbation process |

## Appendix B. Predefined parameters common to all test instances

See Table 11.

## Appendix C. Summary of the tuning of the parameters of the ALNS heuristic

See Table 12.

## References

Bierlaire, M., & Sharif Azadeh, S. (2016). *Demand-based discrete optimization: Technical report*.

Boldrini, C., Incaini, R., & Bruno, R. (2017). Relocation in car sharing systems with shared stackable vehicles: Modelling challenges and outlook. In *2017 IEEE 20th international conference on intelligent transportation systems* (pp. 1–8). IEEE.

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

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

Brandstätter, G., Kahr, M., & Leitner, M. (2017). Determining optimal locations for charging stations of electric car-sharing systems under stochastic demand. *Transportation Research, Part B (Methodological)*, *104*, 17–35.

Bruglieri, M., Colorni, A., & Luè, A. (2014). The vehicle relocation problem for the one-way electric vehicle sharing: An application to the Milan case. *Procedia - Social and Behavioral Sciences*, *111*, 18–27.

Cheng, Y., Deng, X., & Zhang, M. (2019). A novel business model for electric car sharing. In *International workshop on frontiers in algorithmics* (pp. 76–87). Springer.

Di Febbraro, A., Sacco, N., & Saeednia, M. (2018). One-way car-sharing profit maximization by means of user-based vehicle relocation. *IEEE Transactions on Intelligent Transportation Systems*, *20*(2), 628–641.

Fan, W. D. (2013). Management of dynamic vehicle allocation for carsharing systems: Stochastic programming approach. *Transportation Research Record*, *2359*(1), 51–58.

Folkestad, C., Hansen, N., Fagerholt, K., Andersson, H., & Pantuso, G. (2019). Optimal charging and repositioning of electric vehicles in a free-floating carsharing system. *Computers & Operations Research*, *113*, Article 104771.

Golalikhani, M., Oliveira, B. B., Carravilla, M. A., Oliveira, J. F., & Antunes, A. P. (2021). Carsharing: A review of academic literature and business practices toward an integrated decision-support framework. *Transportation Research Part E: Logistics and Transportation Review*, *149*, Article 102280.

Hansen, R. G., & Pantuso, G. (2018). Pricing car-sharing services in multi-modal transportation systems: An analysis of the cases of copenhagen and milan. In R. Cerulli, A. Raiconi, & S. Voß (Eds.), *Computational logistics* (pp. 344–359). Cham: Springer International Publishing.

Hellem, S., Julsvoll, C. A., Moan, M., Andersson, H., Fagerholt, K., & Pantuso, G. (2021). The dynamic electric carsharing relocation problem. *EURO Journal on Transportation and Logistics*, *10*, 100055.

Huo, X., Wu, X., Li, M., Zheng, N., & Yu, G. (2020). The allocation problem of electric car-sharing system: A data-driven approach. *Transportation Research Part D: Transport and Environment*, *78*, Article 102192.

Illgen, S., & Höck, M. (2019). Literature review of the vehicle relocation problem in oneway car sharing networks. *Transportation Research, Part B (Methodological)*, *120*, 193–204.

Jorge, D., Molnar, G., & de Almeida Correia, G. H. (2015). Trip pricing of one-way station-based carsharing networks with zone and time of day price variations. *Transportation Research, Part B (Methodological)*, *81*, 461–482.

Kamatani, T., Nakata, Y., & Arai, S. (2019). Dynamic pricing method to maximize utilization of one-way car sharing service. In *2019 IEEE international conference on agents* (pp. 65–68). IEEE.

Kikuchi, R., & Miwa, H. (2021). Dynamic pricing method for one-way car sharing service to meet demand and to maximize profit under given utility function. *Lecture Notes on Data Engineering and Communications Technologies*, *65*, 324–332.

Klein, R., Koch, S., Steinhardt, C., & Strauss, A. K. (2020). A review of revenue management: Recent generalizations and advances in industry applications. *European Journal of Operational Research*, *284*(2), 397–412.

Li, Y., Chen, H., & Prins, C. (2016). Adaptive large neighborhood search for the pickup and delivery problem with time windows, profits, and reserved requests. *European Journal of Operational Research*, *252*(1), 27–38. http://dx.doi.org/10.1016/j.ejor.2015.12.032, URL https://www.sciencedirect.com/science/article/pii/S0377221715011716.

Li, H., Hu, L., & Jiang, Y. (2022). Dynamic pricing, vehicle relocation and staff rebalancing for station-based one-way electric carsharing systems considering nonlinear charging profile. *Transportation Letters*, 1–26.

Li, X., Wang, C., & Huang, X. (2019). A two-stage stochastic programming model for car-sharing problem using kernel density estimation. Concordia University: arXiv.

Liu, Y., Xie, J., & Chen, N. (2021). Offline-online approximate dynamic programming for stochastic carsharing systems with relocation incentives. Available at SSRN 3882532.

Lourenço, H. R., Martin, O. C., & Stützle, T. (2003). Iterated local search. In *Handbook of metaheuristics* (pp. 320–353). Springer.

Lu, R., Correia, G. H. d. A., Zhao, X., Liang, X., & Lv, Y. (2021). Performance of one-way carsharing systems under combined strategy of pricing and relocations. *Transportmetrica B: Transport Dynamics*, *9*(1), 134–152.

Modesti, P., & Sciomachen, A. (1998). A utility measure for finding multiobjective shortest paths in urban multimodal transportation networks. *European Journal of Operational Research*, *111*(3), 495–508.

Paneque, M. P., Bierlaire, M., Gendron, B., & Azadeh, S. S. (2021). Integrating advanced discrete choice models in mixed integer linear optimization. *Transportation Research, Part B (Methodological)*, *146*, 26–49.

Pantuso, G. (2020). Formulations of a carsharing pricing and relocation problem. In *International conference on computational logistics* (pp. 295–310). Springer.

Pantuso, G. (2022). Exact solutions to a carsharing pricing and relocation problem under uncertainty. *Computers & Operations Research*, *144*, Article 105802.

Ren, S., Luo, F., Lin, L., Hsu, S.-C., & Li, X. I. (2019). A novel dynamic pricing scheme for a large-scale electric vehicle sharing network considering vehicle relocation and vehicle-grid-integration. *International Journal of Production Economics*, *218*, 339–351.

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

Santini, A., Ropke, S., & Hvattum, L. M. (2018). A comparison of acceptance criteria for the adaptive large neighbourhood search metaheuristic. *Journal of Heuristics*, *24*(5), 783–815.

Santoso, T., Ahmed, S., Goetschalckx, M., & Shapiro, A. (2005). A stochastic programming approach for supply chain network design under uncertainty. *European Journal of Operational Research*, *167*(1), 96–115.

Schiller, T., Scheidl, J., & Pottebaum, T. (2017). Car sharing in Europe - business models, national variations and upcoming disruptions. https://www2.deloitte.com/content/dam/Deloitte/de/Documents/consumer-industrial-products/CIP-Automotive-Car-Sharing-in-Europe.pdf. (Accessed 28 November 2021).

Shaw, P. (1998). Using constraint programming and local search methods to solve vehicle routing problems. In *1998 CP International conference on principles and practice of constraint programming* (pp. 417–431).

Stokkink, P., & Geroliminis, N. (2021). Predictive user-based relocation through incentives in one-way car-sharing systems. *Transportation Research, Part B (Methodological)*, *149*, 230–249.

Train, K. E. (2009). *Discrete choice methods with simulation*. Cambridge University Press.

Turkeš, R., Sörensen, K., & Hvattum, L. M. (2021). Meta-analysis of metaheuristics: Quantifying the effect of adaptiveness in adaptive large neighborhood search. *European Journal of Operational Research*, *292*(2), 423–442.

Wagner, S., Willing, C., Brandt, T., & Neumann, D. (2015). Data analytics for location-based services: Enabling user-based relocation of carsharing vehicles. In *International conference on information systems*.

Wang, N., Jia, S., & Liu, Q. (2021). A user-based relocation model for one-way electric carsharing system based on micro demand prediction and multi-objective optimization. *Journal of Cleaner Production*, *296*, Article 126485.

Wang, D., & Liao, F. (2021). Analysis of first-come-first-served mechanisms in one-way car-sharing services. *Transportation Research, Part B (Methodological)*, *147*, 22–41.

Wang, L., & Ma, W. (2019). Pricing approach to balance demands for one-way car-sharing systems. In *2019 IEEE intelligent transportation systems conference* (pp. 1697–1702). IEEE.

Waserhole, A., & Jost, V. (2016). Pricing in vehicle sharing systems: Optimization in queuing networks with product forms. *EURO Journal on Transportation and Logistics*, *5*(3), 293–320.

Weikl, S., & Bogenberger, K. (2013). Relocation strategies and algorithms for free-floating car sharing systems. *IEEE Intelligent Transportation Systems Magazine*, *5*(4), 100–111.

Wu, C., Le Vine, S., Sivakumar, A., & Polak, J. (2021). Dynamic pricing of free-floating carsharing networks with sensitivity to travellers' attitudes towards risk. *Transportation*, 1–24.

Wu, T., & Xu, M. (2022). Modeling and optimization for carsharing services: A literature review. *Multimodal Transportation*, *1*(3), Article 100028.

Xie, R., Wei, W., Wu, Q., Ding, T., & Mei, S. (2019). Optimal service pricing and charging scheduling of an electric vehicle sharing system. *IEEE Transactions on Vehicular Technology*, *69*(1), 78–89.

Xu, M., & Meng, Q. (2019). Fleet sizing for one-way electric carsharing services considering dynamic vehicle relocation and nonlinear charging profile. *Transportation Research, Part B (Methodological)*, *128*, 23–49.

Xu, M., Meng, Q., & Liu, Z. (2018). Electric vehicle fleet size and trip pricing for one-way carsharing services considering vehicle relocation and personnel assignment. *Transportation Research, Part B (Methodological)*, *111*, 60–82.