An Adaptive Large Neighborhood Search Method for Rebalancing Free-Floating Electric Vehicle Sharing Systems

Xufei Liu[1], Sang Won Kim[2], and Changhyun Kwon[*3,4]

[1]College of Management and E-Business, Zhejiang Gongshang University, Hangzhou, 310018, China, xufeiliu@zjgsu.edu.cn

[2]KAIST College of Business, Seoul, 02455 South Korea, sw.kim@kaist.ac.kr

[3]Department of Industrial and Management Systems Engineering, University of South Florida, Tampa, FL, 33620, U.S.A. chkwon@usf.edu

[4]Department of Industrial and Systems Engineering, KAIST, Daejeon, 34141, South Korea

January 17, 2023

Abstract

Free-floating electric vehicle sharing systems allow users to pick up an available electric vehicle (EV) and return it to any permissible parking location within a service area. Such service flexibility can drive a severe spatial imbalance between vehicle availability and trip demands. Hence, it is an important part of their operations to relocate the EV fleet to meet the next day's demand with sufficient battery levels. This relocation operation involves a complicated routing problem for a fleet of shuttles to transport the staff drivers who recharge, if necessary, and relocate the EVs to proper demand locations. Characterized by unique hierarchical and interdependent decisions, the EV relocation and shuttle routing problem poses significant computational challenges for large-scale problems. We devise an efficient algorithm that adapts the Adaptive Large Neighborhood Search (ALNS) metaheuristic framework to overcome such unique challenges. The algorithm is tested on two sets of data: randomly generated data and real-world EV-sharing usage data in Amsterdam. The results validate the efficiency and effectiveness of our ALNS algorithm. In addition, the analysis of total operational cost and waiting time percentage provides practical recommendations to decision-makers on choosing the mode of staff transportation (e.g., shuttles vs. personal mobility such as scooters). Lastly, our numerical results also highlight the usefulness of our ALNS method, which is quite flexible to be applied to a dynamic environment where some EV demands are removed or added in the course of EV relocation operations.

Keywords: transportation; electric vehicle; free-floating car sharing; adaptive large neighborhood search

1 Introduction

Electric vehicle (EV) sharing systems are a promising solution for smarter and more sustainable urban mobility. Vehicle sharing systems have the potential to become a viable alternative to private

*Corresponding author. Part of this research was done while visiting KAIST, Daejeon, South Korea.

car ownership, leading to more efficient utilization of vehicles and parking sites. Moreover, the use of EVs in vehicle sharing systems can offer an effective solution to curbing greenhouse gas emissions in urban transportation. Accordingly, many vehicle sharing companies, such as Zipcar and car2go, are developing rapidly with the traction of increasing vehicle rental demands. As of 2019, the global carsharing market exceeded USD 2.5 billion and is estimated to reach USD 9 billion in 2026 (Preeti and Prasenjit, 2019).

The carsharing systems are classified into *one-way* and *round-trip* according to the case of whether the customers are required to return the vehicles. Moreover, the one-way type is classified as *station-based* and *free-floating* according to the restricted parking locations (Boyacı et al., 2015). This paper considers the one-way free-floating EV sharing problem. Unlike traditional carsharing systems based on fixed stations, free-floating carsharing systems allow users to pick up any available vehicle wherever and whenever they need it and return them at any permissible parking location within a designated service area. As of 2021, free-floating car-sharing systems are operated up to roughly 14,000 cars which are available in 15 mainly large cities by five companies in Germany which is one of the leading markets for car sharing worldwide(Kolleck, 2021). An important feature of such vehicle-sharing systems is that they allow users to make a customized one-way trip with the mobile apps, providing more flexible service (Wielinski et al., 2015). It is reported that this type of carsharing system reduces traffic congestion in urban areas (Becker et al., 2018; Le Vine and Polak, 2019). In North America, it is estimated that 11–19% of carsharing participants have sold their private vehicles after joining a carsharing program (Shaheen and Cohen, 2007). Moreover, carsharing improves the quality of mobility service compared to public transportation (Mattia et al., 2019).

Despite such clear benefits of FFEVSS, they also pose multiple operational challenges. First, the vehicles should be available at the right location at the right time based on the customers' demand patterns, as one-way trip demands can cause a severe spatial imbalance between vehicle availability and trip demands. Second, the low charging level of EVs may trigger customers' range anxiety and lower the service demand, and hence, the EVs should be frequently and sufficiently charged (Weikl and Bogenberger, 2015; Noel et al., 2019). Indeed, for a successful operation of FFEVSS, it is essential to develop an efficient EV relocation plan to resolve the imbalance and charging issues.

Another important decision associated with the operation of a free-floating one-way carsharing system is how to relocate EVs. EV relocations are achieved by staffs (Kek et al., 2006) or customers (Barth et al., 2004). We consider the staff-based vehicle relocations, and the carsharing company arranges the shuttles to dispatch the staff. There are two key decisions in the FFEVSS rebalancing problem: (1) EV relocation and (2) shuttle routing. EV relocation decides which EV should be relocated from its origin (i.e., supply location) to which demand location and also whether and where it has to be charged. This operation can be particularly complex and costly, because EVs cannot be moved in a batch on one carrier, unlike other shared mobility systems such as bikes or scooters. In addition, an EV relocation operation requires a driver to drive the vehicle to its destination. Therefore, supporting vehicles, which we call shuttles, are employed to drop off and pick

up the drivers. Hence, a shuttle routing problem arises. Specifically, the shuttle routing problem determines how such shuttles pick up and drop off drivers to satisfy the planned EV relocations. For an efficient operation, these two decisions should be made simultaneously rather than sequentially, which creates additional layers of computational challenge.

In this paper, we develop an adaptive large neighborhood search (ALNS) algorithm to solve the FFEVSS rebalancing problem. ALNS, first proposed by Ropke and Pisinger (2006), is a well-known iterative metaheuristic framework that has been popularly applied to solving various vehicle routing problems. One of the key characteristics of ALNS is that it partially destroys an incumbent solution and repairs it to construct a better solution in each iteration. Normally, a small number of destroy and repair methods are used, and the choice of the method is determined adaptively as the algorithm proceeds. Applying ALNS to our problem setting is not at all trivial or straightforward. As the two decisions of EV relocation and shuttle routing are tangled in one problem, the destroy and repair methods should also be able to handle such complexity. We propose multiple destroy and repair methods that are adequate to the problem setting that we focus in this paper. We then propose a way to modify our ALNS algorithm to demonstrate how it can accommodate various operational environments. For instance, we demonstrate how our ALNS algorithm can be adopted for the case where each driver carries a small personal mobility vehicle such as a scooter or Segway instead of being transported by a shuttle. In this case, our method routes the drivers directly, rather than routing the shuttles. We present how we may modify our main algorithm to create a variant.

One important strength of our ALNS algorithm, compared to other types of computational methods available for the FFEVSS rebalancing problems in the literature, is its flexibility to be applied in a dynamic environment. Because ALNS algorithms continuously improve the current route by destroying and repairing solutions, any change in the current system can be easily considered within the destroy and repair procedures. However, the ALNS framework applied to the EV relocation and shuttle routing problem should consider the hierarchy and interdependency of the two decisions to be made. Our key methodological contributions include developing efficient methods to handle such hierarchy and interdependency within the popular ALNS framework.

We also conduct numerical experiments for which we use randomly generated instances of several sizes as well as actual data from an FFEVSS operated in the city of Amsterdam. Haider et al. (2019) and Bogyrbayeva et al. (2022) proposed an exchange-based neighborhood-search method (EBNSM) and a Reinforcement Learning (RL) approach to solve the same problem, respectively. With EBNSM and RL being benchmark methods, we demonstrate the effectiveness and efficiency of our ALNS algorithm.

The remainder of the paper is organized as follows. In Section 2, the related literature is reviewed. In Section 3, the problem statement and mathematical formulation are presented in detail. Two benchmark methods are described briefly in Section 6.1. The computational method based on ALNS, our main methodological contribution, is presented in Section 4. The modification of ALNS in problem variants is presented in Section 5. In Section 6, the experiment results validate the performance of ALNS. We conclude this paper in Section 7.

2 Literature Review

Since the inception of the Car2go operation, which started a free-floating carsharing system in Ulm, Germany in April 2009 (Firnkorn and Müller, 2011), many models and approaches have been proposed to optimize the operations in FFEVSS. In this section, we review the literature in this stream within the past decade.

Earlier research focuses only on vehicle relocation without taking personnel allocation into account (Weikl and Bogenberger, 2013; Jorge et al., 2014; Weikl and Bogenberger, 2015; Boyacı et al., 2015). This line of research only considers vehicles relocations schemes to decide where to relocate shared vehicles. They emphasize the demand estimation and the customer service level when making relocation decisions. An inevitable practical problem in vehicle relocation is, however, that each vehicle movement requires one worker who drives from the current vehicle location to the demand location. Thus, EV relocation assignment requires a simultaneous personnel assignment.

More recent research considers the integration of vehicle relocation and personnel assignment. The personnel who drives and relocates shared vehicles—called a worker or a driver—is assumed to move between vehicles by folding bicycles (Bruglieri et al., 2017, 2019; Cai et al., 2022), by walk (Kypriadis et al., 2020), or by shuttles (Haider et al., 2019; Folkestad et al., 2020; Bogyrbayeva et al., 2022). The recharging of EVs or refueling of conventional vehicles is also considered jointly Kypriadis et al. (2020). In this paper, we assume that drivers use shuttles to move from one shared EV to another.

Related to our algorithmic development, Bruglieri et al. (2019) proposed an ALNS for relocating vehicles in electric carsharing services. Hellem et al. (2021) proposed an Adaptive Large Neighborhood Search heuristic to solve the dynamic electric carsharing relocation problem with consideration of recharging and the service employee. Cai et al. (2022) combined adaptive large neighborhood search and tabu search algorithms to solve a similar problem with the objective of maximizing the total profit. In the above three papers, since they assumed that the workers move by folding bicycles that are put in the back when driving EVs, the paths of EV relocations are simply part of worker routes. Therefore, the solution can be represented only by a set of decision variables: i.e., the worker routes. If we consider the shuttle that transports workers between EVs, however, two sets of decisions—shuttle routing and EV relocation—need to be made separately but jointly. The tangled decisions greatly increase computational complexity. In this paper, we address this challenge and show that our computational method can handle personal mobility option cases, such as folding bicycles, as a special case.

The shuttle routing decision is considered for the relocation problem in the following papers. Folkestad et al. (2020) developed a MIP model and a Hybrid Genetic Search with an Adaptive Diversity Control algorithm for relocating cars to charging stations and the routing of staff and service vehicles in the free-floating carsharing system. While final demand locations are assumed to be all charging stations in Folkestad et al. (2020), we assume that final demand locations may not be charging stations in general; hence some EVs need to be recharged in a separate intermediate charging station location. This general consideration certainly increases computational

Table 1: A Summary of the Literature (CO = Combinatorial Optimization; IP = Integer Programming; MIP = Mixed Inter Programming)

Reference	Vehicle Relocation	Personnel Allocation	Charge	Model	Method
Weikl and Bogenberger (2013)	✓	×	×	MIP	Mesoscopic two-step relocation algorithm
Jorge et al. (2014)	✓	×	×	IP	Real-time relocation policies
Weikl and Bogenberger (2015)	✓	×	✓	MIP	Mesoscopic two-step relocation algorithm
Boyacı et al. (2015)	✓	×	✓	MIP	CPLEX
Kypriadis et al. (2020)	✓	walking	✓	CO	Heuristic
Bruglieri et al. (2017)	✓	folding bicycles	✓	MIP	Randomized Heuristic, Ruin-and-Recreate Heuristic
Bruglieri et al. (2019)	✓	folding bicycles	✓	MIP	Adaptive Large Neighborhood Search
Hellem et al. (2021)	✓	folding bicycles	✓	MIP	Adaptive Large Neighborhood Search
Cai et al. (2022)	✓	folding bicycles	✓	MIP	Adaptive Large Neighborhood Search with Tabu Search
Haider et al. (2019)	✓	shuttles	✓	MIP	Exchange-Based Neighborhood Search
Folkestad et al. (2020)	✓	shuttles	✓	MIP	Hybrid Genetic Search with Adaptive Diversity Control
Bogyrbayeva et al. (2022)	✓	shuttles	✓	MDP	Deep Reinforcement Learning
This paper	✓	shuttles or folding bicycles	✓	MIP	Adaptive Large Neighborhood Search

complexity. For such a more general problem, Haider et al. (2019) formulated a MIP model for the relocation operations in the sequential and synchronized approach and proposed the exchange-based neighborhood-search method (EBNSM) for large-scale problems. However, EBNSM is only suitable when the number of charging stations is not greater than the number of EV relocations. In this paper, we try to solve a more real problem without this restriction. Bogyrbayeva et al. (2022) proposed a reinforcement learning approach for rebalancing EVs by considering charging in the free-floating electric vehicle sharing systems (FFEVSS) for a similar problem. They focused on the shuttle routing problem and formulated the shuttle routes using a multi-agent reinforcement learning framework. EVs are relocated to the nearest available charger or demand node. This paper intends to develop an optimization heuristic algorithm that can improve solution quality and show that the algorithm is flexible enough to handle personnel mobility options like folding bicycles as well as shuttles. The literature is summarized in Table 1.

Our joint EV-relocation and shuttle-routing problem is related to the pickup-and-delivery problem (PDP) in the literature for vehicle routing problems (VRP); If we make the EV-relocation decision and the shuttle-routing decision sequentially, once the EV-relocation decision is made, we need to solve a shuttle-routing problem. The EV-relocation decision corresponds to the pickup-and-delivery requests in PDPs. For solving various PDPs, ALNS has been successfully applied (Ropke and Pisinger, 2006; Masson et al., 2013; Ghilas et al., 2016; Sun et al., 2020). Our problem is much more complicated than regular PDPs since we need to make the EV-relocation decision simultaneously with the routing decision. Motivated by the success of ALNS for solving PDPs, we extend findings from the PDP literature and devise an ALNS algorithm for solving the joint EV-relocation and shuttle-routing problem.

To some extent, the bike or scooter sharing systems have similar properties to our problem. The bike or scooter sharing systems also require mobility relocation and personnel allocation. The critical difference is that the bikes or scooters can be relocated in groups since of their small size (Zhu et al., 2020; Du et al., 2020; Osorio et al., 2021), while EVs only can move individually (Shaheen et al.,

2020). Moreover, the charging time for EVs is much longer than for bikes or scooters. Relocation routes for bikes or scooters coincide with a part of the routes of a fleet of trucks that transport the bikes (Pal and Zhang, 2017) or scooters (Carrese et al., 2021), while the relocation routes and shuttle routes are distinct decisions for EVs.

3 Problem Statement

Before we formally define the problem, we introduce the following assumptions:

Assumption 1. We consider the static environment where the carsharing service is closed while the EVs are being relocated. After the EV relocations are done, the carsharing system is open to the customers. Alternatively, we can assume the carsharing service demand is zero during the EV relocation period.

Assumption 2. We consider a deterministic problem. EV relocations only happen between the predefined supply and demand nodes. The supply nodes are decided based on the locations of EVs at the beginning of the time horizon. The demand nodes, to which we move the shared EVs, are decided by a demand estimation model. The number of supply nodes is not less than that of demand nodes. These supply and demand nodes are exogenous to our problem and predetermined before we solve the problem.

Assumption 3. For the sake of simplicity, the charging time is assumed to be proportional to the residual level of the battery, as done in the literature (Schneider et al., 2014; He et al., 2015). If the initial battery level exceeds the required level, EVs do not need to be charged and directly move to one demand node; otherwise, the EV moves to a charging station to be charged fully.

Assumption 4. The battery consumption during the EV relocation is ignored.

Assumption 5. There is no limit on the number of chargers in a charging station. Based on this assumption, an EV can charge immediately when it arrives at a charging station. The assumption is made because the number of required chargers at a charging station is low and not worth considering. That is shown in the experimental results in Section 6.3.

We formulate our problem with the notation listed in Table 2. The service area is constrained in a region with $|\mathcal{N}|$ nodes. One node means a parking lot for an EV. The supply nodes are the places where the excess available EVs park and no customers will pick up EVs at those nodes the next day. Based on the booking orders, the customers will pick up EVs at the demand nodes. Now there are no available EVs at these nodes. Thus, the sharing system company is required to relocate EVs from supply nodes to demand nodes to satisfy the customers' needs.

In this problem, two decisions are made: EV relocation and shuttle routing. The EV relocation is divided into two cases: (1) when an EV can move directly from a supply node $s \in \overline{\mathcal{S}}$ to a demand node when it has sufficient battery energy (more than the minimum battery level); and (2) when an EV does not have enough battery energy (less than minimum battery level). The EV at supply

Table 2: Mathematical Notation

Indices	
s	Representing supply nodes, $s \in \mathcal{S}$
c	Representing charging nodes, $c \in \mathcal{C}' \cup \mathcal{C}^+$
d	Representing demand nodes, $d \in \mathcal{D}$
k	Representing shuttles, $k \in \mathcal{K}$
p	Representing paths that EVs are relocated along, $p \in \mathcal{P}$
Sets	
$\mathcal{N}$	Set of all real nodes, $\mathcal{N} = \{0, 1, 2, .., N\} = \{0\} \cup \mathcal{S} \cup \mathcal{D} \cup \mathcal{C}$, where the depot is node 0
$\mathcal{N}'$	Set of all nodes including all real and dummy nodes, $\mathcal{N}' = \mathcal{N} \cup \{N+1\} \cup \mathcal{C}' \cup \mathcal{C}^+$, where the dummy depot node is $N+1$.
$\mathcal{S}$	Set of supply nodes, $\mathcal{S} = \overline{\mathcal{S}} \cup \widehat{\mathcal{S}}$
$\overline{\mathcal{S}}$	Set of supply nodes where EVs do not require charging
$\widehat{\mathcal{S}}$	Set of supply nodes where EVs require charging
$\mathcal{D}$	Set of demand nodes
$\mathcal{C}$	Set of real charge nodes
$\mathcal{C}'$	Set of real and paired dummy charge nodes
$\mathcal{C}^+$	Set of dummy charge process nodes
$\mathcal{K}$	Set of shuttles
$\mathcal{P}$	Set of all feasible EV relocation paths
$\phi(i)$	Set of paths that contain node i, $\{p \in \mathcal{P} : i \in p\}$
Parameters	
V_{EV}	Average EV speed
V_{shuttle}	Average shuttle speed
I_s	Initial battery percentage of EV at supply s
β	Charging time for one battery percentage
K	Number of shuttles $K = \lvert\mathcal{K}\rvert$
Q	Number of all workers
W	Capacity of workers onboard each shuttle, excluding the worker who drives the shuttle
d_{ij}	Distance of arc (i, j)
M	A sufficient large positive number
H	A sufficient large positive number for creating dummy charge nodes
Variables	
x_p	1, EV is relocated along path $p \in \mathcal{P}$; Otherwise, 0.
y_{ij}	1, if a shuttle comes through arc (i, j) as part of a shuttle route; otherwise, 0.
z_i	The number of workers onboard a shuttle after this shuttle leaves node i.
τ_i	The time when a shuttle arrives node i
e_i	The time when an EV arrives node i

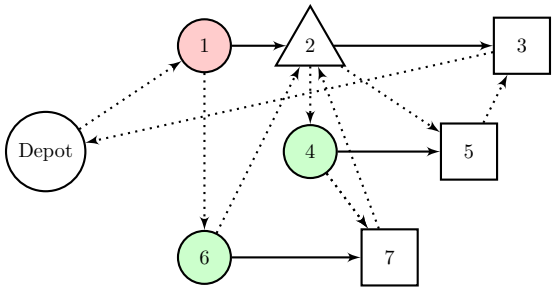


Figure 1: FFEVSS Example (circles, triangles, and squares represent supply nodes, charging stations, and demand nodes, respectively. Red circles mean that EV requires charging; green circles mean EV does not require charging; The solid lines represent EV relocation; the dotted lines represent a single shuttle route)

node $s \in \widehat{\mathcal{S}}$ is required to go to a charging station $c \in \mathcal{C}$ to be charged fully before going to the demand node $d \in \mathcal{D}$.

Shuttle routing determines the sequence of visiting nodes. Shuttles transport workers to supply nodes and charging stations, and the workers will be picked up at demand nodes and charging stations. Since the charging process takes a long time, it is assumed that the worker has two choices. First, the worker can be picked up by one shuttle to do other EV relocation assignments. In this case, another worker will come to this charging station and drive the EV to the demand node. Second, the worker can stay at the charging station and wait for charging. Then, the worker drives an EV to the demand node.

The number of shuttles is K, and the capacity of workers onboard each shuttle is W. All shuttles leave the depot with a full load and return to the depot with all workers. The objective function is to minimize the total time spent in the system, i.e., the makespan. The makespan is calculated as the time between shuttles leaving and all returning to the depot.

For example, in Figure 1, there is one shuttle and two workers. EV relocation contains 1-2-3, 4-5, and 6-7. The shuttle route is depot-1-6-2-4-7-2-5-3-depot. The shuttle starts from the depot and drops off 1st worker at supply 1. Then it goes to supply 6 and drops off 2nd worker. It picks up 1st worker at charging station 2 and transports him to supply 4. It continues to pick up 2nd worker at demand 7. The shuttle returns to charging station 2 and dispatches the 2nd worker to drive the EV from node 2 to demand 3. Then, it picks up 1st and 2nd workers at demands 5 and 3. Finally, the shuttle carries all workers back to the depot.

The dummy charge node-set is introduced to deal with multiple visits to each charging station. Because charging stations can serve many EVs and be visited many times, each charging station needs sufficient enough dummy nodes. These dummy nodes have the same location coordinates as the real charging station. Define H as the number of dummy charge nodes for each real charge node. The depot end node is labeled as $N+1$, so the dummy charge nodes are labeled starting from $N+2$. For example, the dummy charge node for the first charge node is $\{N+2, N+3, .., N+1+H\}$. Let the total charge nodes set be $\mathcal{C}' = \mathcal{C} \cup \{N+2, N+3, ..., N+1+|\mathcal{C}| \times H\}$.

The paired dummy charge process node-set $\mathcal{C}^+ = \{c^+ = \text{copy}(c) : c \in \mathcal{C}'\}$ is introduced to deal

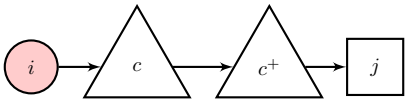


Figure 2: Relocation Decision - In a relocation with charge model, one relocation operation involves a stopover at a pair of charger-dummy charge process nodes

with the charging process, where c^+ represents a copy of c. The charge time for EV at s is calculated from the initial battery percentage $I_s\%$ to 100% and is assumed to equal $\beta \times (100 - I_s)$. Because the charging takes a long time, the workers can be dispatched to do other tasks. The charging process is described in Figure 2, EV goes to charging station c for charging, and after finishing the charge, it goes to a dummy charge process node c^+. The pair of charger and dummy charge process nodes $c \in \mathcal{C}', c^+ \in \mathcal{C}^+$ are designated for receiving workers and dispatching shuttles.

To model the EV relocation decisions, we can enumerate all the possible EV relocation paths, rather than using arc-based formulations, as the EV relocation decisions only have two or three layers in their decision-making tree. If an EV does not require charging, the path is defined as directly from any supply node to any demand node $p = \{s \to d : s \in \overline{\mathcal{S}}, d \in \mathcal{D}\}$ and the number of paths is $|\mathcal{S}| \times |\mathcal{D}|$. If an EV requires charging, the path is defind as $p = \{s \to c \to d : s \in \widehat{\mathcal{S}}, c \in \mathcal{C}', d \in \mathcal{D}\}$ and the number of paths is $|\mathcal{S}| \times |\mathcal{C}'| \times |\mathcal{D}|$. The set of all possible EV relocation paths $\mathcal{P}$ is defined as

$$\mathcal{P} = \{s \to d : s \in \overline{\mathcal{S}}, d \in \mathcal{D}\} \cup \{s \to c \to d : s \in \widehat{\mathcal{S}}, c \in \mathcal{C}', d \in \mathcal{D}\}$$

with $|\mathcal{S}| \times |\mathcal{D}| + |\mathcal{S}| \times |\mathcal{C}'| \times |\mathcal{D}|$ paths. This path enumeration enables a faster computation compared to arc-based formulations (Haider et al., 2019).

The mathematical model for the FFEVSS is formulated as follows:

$$\text{minimize} \quad \tau_{N+1} \tag{1}$$

subject to

$$\sum_{p \in \phi(i)} x_p = 1 \qquad \forall i \in \mathcal{D} \tag{2}$$

$$\sum_{p \in \phi(i)} x_p \leq 1 \qquad \forall i \in \mathcal{S} \cup \mathcal{C}' \tag{3}$$

$$e_d \geq e_s + \frac{d_{sd}}{V_{\text{EV}}} \qquad \forall p = \{s \to d\} \in \mathcal{P} : s \in \overline{\mathcal{S}} \tag{4}$$

$$e_c \geq e_s + \frac{d_{sc}}{V_{\text{EV}}} \qquad \forall p = \{s \to c \to d\} \in \mathcal{P} : s \in \widehat{\mathcal{S}} \tag{5}$$

$$e_{c^+} \geq e_c + \beta(100 - I_s) \qquad \forall p = \{s \to c \to d\} \in \mathcal{P} : c \in \mathcal{C}' \tag{6}$$

$$e_d \geq e_{c^+} + \frac{d_{cd}}{V_{\text{EV}}} \qquad \forall p = \{s \to c \to d\} \in \mathcal{P} \tag{7}$$

$$\sum_{(0,j) \in \mathcal{A}} y_{0j} = \sum_{(i,N+1) \in \mathcal{A}} y_{i,N+1} = K \tag{8}$$

$$\sum_{(i,j)\in\mathcal{A}} y_{ij} - \sum_{(j,i)\in\mathcal{A}} y_{ji} = 0 \qquad \forall j \in \mathcal{N}' \setminus \{0, N+1\} \tag{9}$$

$$\sum_{(i,j)\in\mathcal{A}} y_{ij} \leq 1 \qquad \forall j \in \mathcal{S} \cup \mathcal{C}' \tag{10}$$

$$\sum_{(i,j)\in\mathcal{A}} y_{ij} = 1 \qquad \forall j \in \mathcal{D} \tag{11}$$

$$\sum_{(i,c)\in\mathcal{A}} y_{ic} = \sum_{(i,c^+)\in\mathcal{A}} y_{ic^+} \qquad \forall c \in \mathcal{C}' \tag{12}$$

$$\tau_j \geq \tau_i + \frac{d_{ij}}{V_{\text{shuttle}}} - M(1 - y_{ij}) \qquad \forall (i,j) \in \mathcal{A} \tag{13}$$

$$z_0 = W \tag{14}$$

$$0 \leq z_i \leq W \qquad \forall i \in \mathcal{N}' \tag{15}$$

$$z_j = z_i - y_{ij} \qquad \forall (i,j) \in \mathcal{A}, j \in \mathcal{S} \cup \mathcal{C}^+ \tag{16}$$

$$z_j = z_i + y_{ij} \qquad \forall (i,j) \in \mathcal{A}, j \in \mathcal{D} \cup \mathcal{C}' \tag{17}$$

$$\sum_{(i,s)\in\mathcal{A}} y_{is} = \sum_{p\in\phi(s)} x_p \qquad \forall s \in \mathcal{S} \tag{18}$$

$$\sum_{(i,c)\in\mathcal{A}} y_{ic} \leq \sum_{p\in\phi(c)} x_p \qquad \forall c \in \mathcal{C}' \tag{19}$$

$$e_s \geq \tau_s - M(1 - \sum_{p\in\phi(s)} x_p) \qquad \forall s \in \mathcal{S} \tag{20}$$

$$\tau_c \geq e_c - M(1 - \sum_{(i,c)\in\mathcal{A}} y_{ic}) \qquad \forall c \in \mathcal{C}' \tag{21}$$

$$e_{c^+} \geq \tau_{c^+} - M(1 - \sum_{(i,c^+)\in\mathcal{A}} y_{ic^+}) \qquad \forall c^+ \in \mathcal{C}^+ \tag{22}$$

$$\tau_d \geq e_d \qquad \forall d \in \mathcal{D} \tag{23}$$

$$x_p \in \{0,1\} \qquad \forall p \in \mathcal{P} \tag{24}$$

$$y_{ij} \in \{0,1\} \qquad \forall (i,j) \in \mathcal{A} \tag{25}$$

$$e_i, \tau_i \geq 0 \qquad \forall i \in \mathcal{N}' \tag{26}$$

$$z_i \in \mathbb{Z}_+ \qquad \forall i \in \mathcal{N}' \tag{27}$$

Constraints (2)–(7) describe the EV relocation operations. Constraints (2) ensure that each demand node is visited exactly once, so the inflow of each demand node $i \in \mathcal{D}$ is equal to 1. Constraints (3) ensure that each supply node and charging station is visited less than or equal to 1. Constraints (4) show that if an EV is assigned from s to d, the time when EV arrives node d is more than or equal to EV's arrival time at node s plus the movement time. Constraints (5)–(7) set the arrival time if an EV moves along the path $s \to c \to d$. EV's arrival time at node c is the arrival time at node s plus movement time from s to c; EV arrives at dummy charge node c^+ after adding a charging time $\beta(100 - I_s)$; EV's arrival time at demand d is arrival time at dummy charge node c^+ plus movement time from c to d.

The shuttles routing problem is described by Constraints (8)–(17). Constraint (8) makes sure that K shuttles leave depot start node 0 and enter depot end node $N+1$. Constraints (9) ensure the flow balance between intermediate nodes. Except for the depot node, supply nodes and charging stations can be visited once or not, as in Constraints (10), and demand nodes must be visited once, as in Constraints (11). Constraints (12) show that if a charge node $c \in \mathcal{C}'$ is visited by a shuttle, its paired dummy charge process node $c^+ \in \mathcal{C}^+$ must be visited by a shuttle. Constraints (13) restrict the times when a shuttle arrives at node j. Constraint (14) means that all shuttles leave the depot with W workers. Constraints (15) ensure that the number of workers on a shuttle must be nonnegative and less than or equal to the capacity W. Constraints (16) and (17) show that one shuttle drops off one worker on a supply node or a dummy charge process node and picks up one worker on a demand node or a charge node.

Constraints (18)–(23) are formulated to synchronize EV relocation and shuttle routing problems. Constraints (18) show that when an EV at node s is assigned to relocate to node d, the shuttle must pass through node s. Constraints (19) show that the shuttle passes through charging station c (the worker is picked by this shuttle) or does not pass charging station node c (the worker chooses to wait for EV to be charged). The shuttle routes restrict EVs' time window. Assume the times of getting on and off shuttles are not considered. Constraints (20) and (22) restrict that when a shuttle drop off a worker at supply s or a dummy charge process node $c^+ \in \mathcal{C}^+$, EV starts to leave. Constraints (21) and (23) show that the shuttle picks up a driver and leaves charging station c or demand d after EV has already arrived.

Constraints (24) and (25) restrict that EV relocation path decision variable x^p and routing decision y are binary variables. Constraints (26) show that EV arrival times e_i and shuttle arrival time τ are nonnegative continuous variables. Constraints (27) restrict that the number of drivers on board shuttles is an integer variable.

4 Adaptive Large Neighborhood Search

In this section, we develop an Adaptive Large Neighborhood Search (ALNS) for the free-floating EV sharing system. Ropke and Pisinger (2006) developed ALNS as an extension of the Large Neighborhood Search method (Shaw, 1998). The idea of ALNS is to search in a large neighborhood using multiple destroy and repair methods and to choose the destroy and repair methods based on their adaptive probabilities.

To the best of our knowledge, this is the first paper using ALNS on the free-floating carsharing system problem that considers EV relocation and shuttle routing jointly. In the original ALNS (Ropke and Pisinger, 2006), the feasible solution only has one decision (request routes). However, in this paper, the feasible solution has two decisions: EV relocation X and shuttle routes Y. The main challenge is to jump from one solution to a new solution in the neighborhood. Because two decisions affect each other, the processes of destroy and repair become more complicated. So, the repair methods are divided into two stages. In the first stage, greedy and probabilistic methods are

proposed to match suppliers and demanders to repair EV relocations. In the second stage, greedy and regret methods are presented to reconstruct shuttle routes.

Algorithm 1: Pseudocode for ALNS

Input: $\mathcal{D}$, $\mathcal{S}$, $\mathcal{C}$, DM, RM, $N_{\max}, Z_{\max}$
Output: $X_{\text{best}}, Y_{\text{best}}$
Initialize EV relocation X_0 and shuttle routes Y_0 (Sec 4.1);
Initialize destroy methods probability $\mathbf{P}_D^0$ and repair methods probability $\mathbf{P}_R^0$ (Sec 4.4);
$X_{\text{best}} \leftarrow X_{\text{current}} \leftarrow X_0$, $Y_{\text{best}} \leftarrow Y_{\text{current}} \leftarrow Y_0$;
Calculate the makespan of current best solution $t_{\text{best}} \leftarrow f(X_{\text{best}}, Y_{\text{best}})$;
$N \leftarrow 1, Z \leftarrow 0$;
while $N \leq N_{\max}$, $Z \leq Z_{\max}$ **do**
    Select a destroy method $d \in$ DM with probability P_D^N;
    Select a repair method $r \in$ RM with probability P_R^N;
    Let X_{new} and Y_{new} be the new solution obtained by apply destroy d and repair r;
    if $f(X_{\text{new}}, Y_{\text{new}}) < t_{\text{best}}$ **then**
        $X_{\text{best}} \leftarrow X_{\text{new}}$, $Y_{\text{best}} \leftarrow Y_{\text{new}}$, $t_{\text{best}} \leftarrow f(X_{\text{new}}, Y_{\text{new}})$, $Z \leftarrow 0$;
    else
        $Z \leftarrow Z + 1$;
        $v = e^{-(f(X_{\text{new}}, Y_{\text{new}}) - f(X_{\text{current}}, Y_{\text{current}}))/T}$;
        Generate a random number $\epsilon \in [0, 1]$;
        if $\epsilon < v$ **then**
            $X_{\text{current}} \leftarrow X_{\text{new}}$, $Y_{\text{current}} \leftarrow Y_{\text{new}}$;
    $T \leftarrow hT$;
    Update $\mathbf{P}_D^N$ and $\mathbf{P}_R^N$ (Sec 4.4);
    $N \leftarrow N + 1$;

The procedure of the proposed ALNS is shown in Algorithm 1. In each iteration, the neighborhood of a solution is produced by destroy and repair. We let DM and RM denote the sets of the destroy and repair methods, respectively. The destroy process is to remove part of EV relocation solution X and shuttle routes Y. Next, the repair process can reconstruct the partially incomplete solution. The destroy and repair methods are chosen according to their past successes. When a better solution is obtained by applying the methods, the counts of the corresponding methods are added by one, and their probabilities will increase, as described in detail in Section 4.4. The roulette-wheel selection principle is used for the selection of destroy and repair. In this paper, a simple acceptance rule is applied: the new best solution is accepted if its objective value is better than the current best solution. The new solution with a higher objective value is accepted by a simulated annealing acceptance criterion. T denotes the value of the temperature and gradually decreases at each iteration by a rate $h \in [0, 1]$. The stop criteria are the maximum iterations $N_{\max}$ and non-improving iteration $Z_{\max}$.

4.1 Finding an Initial Solution

The nearest neighborhood (NNH) search (Algorithm 2) is applied to generate an initial solution as follows. We randomly select a demand node d in undecided demand node-set UD and match its nearest supply node s in undecided supply node-set US. If an EV at the node s requires being charged, then insert the nearest charging station c in the middle. Then, delete s and d in the undecided set US and UD. The EV relocation ω is assigned on the shuttles in sequence. An initial

Algorithm 2: Pseudocode for NNH

Input: $\mathcal{D}, \mathcal{S}, \widehat{\mathcal{S}}, \mathcal{C}$
Output: X, Y
Initialize EV relocation set: $X_1 \leftarrow \emptyset$, $X_2 \leftarrow \emptyset$, $X \leftarrow X_1 \cup X_2$;
Initialize shuttle routes: $Y_k \leftarrow \emptyset\ \forall k \in \mathcal{K}$, $Y \leftarrow \cup_{k \in \mathcal{K}} Y_k$;
Initialize search sets: $\mathsf{UD} \leftarrow \mathcal{D}$, $\mathsf{US} \leftarrow \mathcal{S}$;
$k \leftarrow 1$;
while $\mathsf{UD} \neq \emptyset$ **do**
    $\forall d \in \mathsf{UD}$, find the nearest supplier $s = \arg\min\{d_{sd}, \forall s \in \mathsf{US}\}$;
    if $s \in \widehat{\mathcal{S}}$, **then**
        Create a EV relocation $\omega_{sd} \leftarrow \{s \rightarrow c \rightarrow d\}$, where $c = \arg\min\{d_{sc} + d_{cd}, \forall c \in \mathcal{C}\}$;
        $X_2 \leftarrow X_2 \cup \{\omega_{sd}\}$;
    else
        Create a EV relocation $\omega_{sd} \leftarrow \{s \rightarrow d\}$;
        $X_1 \leftarrow X_1 \cup \{\omega_{sd}\}$;
    $\mathsf{US} \leftarrow \mathsf{US} \setminus \{s\}$, $\mathsf{UD} \leftarrow \mathsf{UD} \setminus \{d\}$;
    Add s and d to the end of kth shuttle route list Y_k;
    if $k = K$ **then**
        $k \leftarrow 1$;
    else
        $k \leftarrow k + 1$;

solution consists of two decisions: EV relocation denoted as X and shuttle routes denoted as Y. EV relocation decision X comprises two distinct types: (1) an EV directly moves from a supply node to a demand node, and (2) an EV moves to a charging station and then goes to a demand node. Shuttle routes serve these EV relocation operations, and the corresponding shuttle route set Y consists of K lists, namely $Y_1, Y_2, ..., Y_K$. The list Y_k means the sequence nodes visited by the k-th shuttle.

4.2 Destroy Methods

Three destroy methods are applied to destroy a completely feasible solution into a partial one. The destroy methods consisting of Random Removal, Worst Route Removal, and Cluster Removal, are presented as follows.

4.2.1 Random Removal

Randomly remove $\lfloor \alpha\% \times |D| \rfloor$ EV relocation ω_{sd} in X, where α is the destroy percentage. Put these supply nodes and demand nodes into the undecided supplier set (US) and undecided demand set (UD). Meanwhile, delete them in shuttle routes Y.

4.2.2 Worst Route Removal

Given a solution (X, Y), a demand d is matched with the supply s in EV relocation solution X. we define the cost for EV relocation pair ω_{sd} as

$$\text{cost}(\omega_{sd}, X, Y) = f(X, Y) - f_{-\omega_{sd}}(X, Y) \tag{28}$$

where $f_{-\omega_{sd}}(X, Y)$ is the objective value without EV relocation pair ω_{sd} in (X, Y). It is reasonable to remove the supply-demand pair with the high cost and perhaps create new EV relocation to obtain a better solution.

Sort all supply-demand pairs ω in descending costs. Remove first $\lfloor \alpha\% \times |D| \rfloor$ EV relocation ω with larger costs. Put these supply nodes and demand nodes into the undecided supplier set (US) and undecided demand set (UD). Meanwhile, delete them in shuttle routes Y.

4.2.3 Cluster Removal

The idea of cluster removal is to remove the similar demand nodes (Shaw, 1998). Since a new and better solution is expected to be created, the current solution needs to be destroyed more heavily. This allows the farther neighborhood to be searched.

The relatedness between node i and j is used to measure how close node j is to node i. For any demand node i, the measure of relatedness is calculated as the following equation.

$$R(i, j) = w_1 \frac{d_{ij}}{\max\{d_{ik}, \forall k \in \mathcal{D}\}} + w_2 \frac{|e_i - e_j|}{\max\{e_k, \forall k \in \mathcal{D}\} - \min\{e_k, \forall k \in \mathcal{D}\}} \quad \forall j \neq i \in \mathcal{D} \tag{29}$$

where w_1 and w_2 are weights with sum of 1. d_{ij} is the distance between node i and j. e_i is the time when an EV arrives at demand node i. The smaller $R(i, j)$ is, the more related the demands i and j are.

The following steps are followed: randomly select a demand node $d \in \mathcal{D}$ and calculate $R(d, j), \forall j \neq d \in \mathcal{D}$. Sort all $R(d, j)$ in descending order. Remove demands with the first $\lfloor \alpha\% \times |D| \rfloor$ in the sequence and their corresponding supplier nodes. Put these supply nodes and demand nodes into undecided supplier (US) and undecided demand sets (UD). Meanwhile, delete them in shuttle routes Y.

4.3 Repair Methods

The following repair process must conform to two constraints:

1. Sequence constraint: supply s must be visited before demand d, when supply s is paired with demand d in EV relocation. Also, charging station c must be visited between s and d, and its paired dummy charger c^+ must be visited after c.

2. Personnel constraint: Since there are a limited number of workers (capacity) onboard each shuttle, in any shuttle route Y_k, the difference between the numbers of visiting supply and demand nodes must stay within $[0, W]$. For example, given a shuttle route $Y_k = \{a_1, a_2, a_3, ...\}$.

For each positive integer $n = 1, 2, ..., |Y_k|$, we enforce

$$0 \leq \sum_{i=1}^{n} \mathbb{1}[a_i \in \mathcal{S} \cup \mathcal{C}^+] - \sum_{i=1}^{n} \mathbb{1}[a_i \in \mathcal{D} \cup \mathcal{C}] \leq W$$

where $\mathbb{1}[\cdot]$ equals one if the condition inside the bracket holds and equals zero otherwise.

We have two repair rules for EV relocation and two repair rules for route insertion. Four repair methods are formed by combining these repair rules.

1. Sequential Greedy: Greedy s-d matching with Greedy routes insertion

2. Sequential Regret: Greedy s-d matching with Regret routes insertion

3. Probabilistic Greedy: Probabilistic s-d matching with Greedy routes insertion

4. Probabilistic Regret: Probabilistic s-d matching with Regret routes insertion

4.3.1 Repair Rules for EV Relocation

Greedy s-d matching For $\forall d \in \mathsf{UD}$, demand d is matched with an undecided supply s where $s = \text{argmin}\{d_{sd}, s \in \mathsf{US}\}$. If $s \in \widehat{\mathcal{S}}$, $\omega_{sd} = \{s \to c \to d\}$, where $c = \text{argmin}\{d_{sc} + d_{cd}, c \in \mathcal{C}\}$; else $\omega_{sd} = \{s \to d\}$. The new supply-demand pair ω_{sd} is formed by the nearest-neighbor rule.

Probabilistic s-d matching s and d are matched as a new pair based on the probability that is related to their distance. The s-d matching probability is expressed as

$$P(\omega_{sd}) = \frac{\exp(-\lambda d_{\omega_{sd}})}{\sum_{\omega_{sd} \in \mathcal{U}} \exp(-\lambda d_{\omega_{sd}})} \tag{30}$$

where $\mathcal{U}$ is the set of all combinations of undecided $s \in \mathsf{US}$ and undecided $d \in \mathsf{UD}$. λ is a constant within $[0, 1]$. If $s \in \widehat{\mathcal{S}}$, each charge station is inserted in the middle of s and d. There are $|\mathcal{C}|$ combinations for s and d. For each $\omega_{sd} = \{s \to c \to d\}$, we set $d_{\omega_{sd}} = d_{sc} + d_{cd}$. If $s \in \overline{\mathcal{S}}$, we set $d_{\omega_{sd}} = d_{sd}$.

4.3.2 Repair Rules for Routes Insertion

Greedy routes insertion A concept of Insertion Cost is introduced. Let $\Delta f(n, p, Y)$ denote the change in the objective value when inserting node n into partial shuttle routes Y at position p. The

insertion cost is expressed as

$$I(n,p,Y) = \Delta f(n,p,Y) \tag{31}$$

After any new supply-demand ω_{sd}, insert s and d into the current partial shuttle routes Y, separately. Select the position p to insert s with the least insertion cost $I(s,p,Y)$. The new partial shuttle routes are formed after inserting s, denoted as Y_{+s}. Then, insert d at the position with the least insertion cost $I(d,p,Y_{+s})$ and get the new partial routes Y. If $s \in \widehat{\mathcal{S}}$, insert c and c^+ with the least insertion cost sequentially. Repeat the above steps until all demands are satisfied.

Regret routes insertion The regret routes insertion is improved by incorporating look-ahead information when selecting the supply-demand pair to insert. For any supply-demand pair ω_{sd}, the regret-k cost is defined as

$$R(\omega_{sd},Y) = \sum_{j=1}^{k} \{\Delta f_j(\omega_{sd},Y) - \Delta f_1(\omega_{sd},Y)\} \tag{32}$$

where $\Delta f(\omega_{sd},Y)$ is the increased value in the objective value after inserting s and d. Sort $\Delta f(\omega_{sd},Y)$ for all possible insertion positions in the increasing order. The best insertion position has the least $\Delta f_1(\omega_{sd},Y)$. Note that $\Delta f_k(\omega_{sd},Y)$ means the increased value in the objective for the k-th best insertion position. The regret routes insertion is the reconstruction heuristic that chooses to insert the supply-demand pair ω_{sd} with the maximum $R(\omega_{sd},Y)$. The ω_{sd} is inserted at its minimum cost position. If $s \in \widehat{\mathcal{S}}$, insert c and c^+ with the least insertion cost sequentially. Repeat the above steps until all demands are satisfied.

4.4 Adaptive Probability Update Procedure

The adaptivity of ALNS is achieved by selecting the destroy and repair methods based on their previous successes. The initial probabilities of destroys and repairs are set to 1 divided by the number of available destroy and repair methods; that is, $\frac{1}{|\mathsf{DM}|}$ and $\frac{1}{|\mathsf{RM}|}$. In each iteration i, if destroy method d and repair method r create a new best solution X_{best}, the count n_d^i and n_r^i of destroy d and repair r is increased by 1, respectively. Then, the probability values of destroy and repair methods in iteration N are updated by the multiplication of two sets of ratios σ_d and σ_r as follows:

$$\mathbf{P}_D^N = \left(\frac{\sigma_d n_d^N}{\sum_{d\in\mathsf{DM}} n_d^N} : d \in \mathsf{DM} \right) \tag{33}$$

$$\mathbf{P}_R^N = \left(\frac{\sigma_r n_r^N}{\sum_{r\in\mathsf{RM}} \sigma_r n_r^N} : r \in \mathsf{RM} \right) \tag{34}$$

The destroy and repair methods are adaptively chosen based on the new probabilities by using the roulette-wheel selection principle.

5 Modification of ALNS in Problem Variants

We consider two variants of the rebalancing problem. First, we consider the case when EV drivers use their own personal mobility option instead of shuttles. Second, we consider the dynamic environments wherein EV supply or demand locations change while shuttles and drivers are already executing an operational plan. In both cases, we show that ALNS can be easily modified.

5.1 Routing with Personal Mobility Options

In some cases, the workers can be moved not only by shuttles but also by personal mobility vehicles such as scooters. Each worker has one personal mobility vehicle. When a worker arrives at a supply node, this worker puts the scooter in the back and drives the EV to the demand node or goes to a charging station. When the EV arrives at a charging station, the worker can ride his mobility tool to another node to accomplish other tasks. In the sequence constraint, $s \to c$ and $c \to d$ are binded as one unit. Also, $s \to c$ must be inserted before $c \to d$ in the routes. The personnel constraint is necessary to be taken into account.

EV relocation is the same as stated in Section 4. It has two types: $\omega_{sd} = \{s \to d\}$ and $\omega_{sd} = \{s \to c \to d\}$. The destroy and repair methods for EV relocation do not change.

The repair rules for routes insertion need some changes as follows. Because the scooters and workers move synchronously with the EVs, the worker routing is consistent with EV relocation from $s \to d$, $s \to c$ and $c \to d$. We can regard $s \to d$, $s \to c$ and $c \to d$ as one unit and insert them into the partial routes. For an example, some EV relocations are $s1 \to d4$, $s2 \to c3$, and $c5^+ \to d6$. One worker route can be described as $(1 \to 4) \to (2 \to 3) \to (5^+ \to 6)$. That means that this worker drives an EV from supply 1 to demand 4 and then goes to node 2 by riding his scooter; he drives the EV from supply 2 to charging station 3; Finally, he rides the scooter to a dummy charging station 5^+ and drives the EV to demand 6.

5.2 EV Relocation and Routing in Dynamic Environments

In dynamic environments, the number of EV relocation assignments can change in the middle of the relocation operations. It happens in cases when some of the current available EVs are assigned to the arriving customers, or some additional demands are added to the system. Instead of solving the new routing problem from scratch, we can use the ALNS algorithm to destroy and repair the current solution to adapt to the dynamic environment. When we repair, we just ignore those EVs which have already been served. Remove the decreased demands (or suppliers) or add the additional demands (or suppliers) to the undecided demand set UD (or the undecided supply set US).

The shuttles have departed the depot, and some EV relocation demands $\widetilde{\mathcal{D}}$ have been served already. The changes in demand and supply sets are described as sets themselves and denoted as $\mathcal{D}^*$ and $\mathcal{S}^*$. There are two cases.

1. When the number of EV demands decreases: The decreased demand nodes and corresponding supply nodes are removed from the current solution. Then, partial EV relocation X and

Algorithm 3: Pseudocode for ALNS in Dynamic Environment

Input: $\mathcal{D}, \widetilde{\mathcal{D}}, \mathcal{D}^*, \mathcal{S}, \widetilde{\mathcal{S}}, \mathcal{S}^*, X^0, Y^0$

Output: $X_{\text{best}}, Y_{\text{best}}$

The partial EV relocation $X \leftarrow X^0 \setminus \{\omega_{sd}, \forall \omega_{sd} \in X^0 : d \in \widetilde{\mathcal{D}}\}$;

The partial shuttle routing $Y \leftarrow Y^0 \setminus \{s, d, \forall s \in \widetilde{\mathcal{S}}, d \in \widetilde{\mathcal{D}}\}$;

Initialize undecided demand set $\mathsf{UD} \leftarrow \mathcal{D} \setminus \widetilde{\mathcal{D}} \cup \mathcal{D}^*$ and undecided supply set $\mathsf{US} \leftarrow \mathcal{S} \setminus \widetilde{\mathcal{S}} \cup \mathcal{S}^*$;

Initialize destroy methods probability $\mathbf{P}_D^0$ and repair methods probability $\mathbf{P}_R^0$ (Sec 4.4);

Apply a repair method $r \in \mathsf{RM}$ with probability P_R^0 on X, Y and $X_{\text{new}}, Y_{\text{new}}$ are obtained;

$X_{\text{best}} \leftarrow X_{\text{current}} \leftarrow X_{\text{new}}$, $Y_{\text{best}} \leftarrow Y_{\text{current}} \leftarrow Y_{\text{new}}$;

Calculate the makespan of current best solution $t_{\text{best}} \leftarrow f'(X_{\text{best}}, Y_{\text{best}}, \Psi)$;

$N \leftarrow 1, Z \leftarrow 0$;

while $N \leq N_{\max}$, $Z \leq Z_{\max}$ **do**

  Select a destroy method $d \in \mathsf{DM}$ with probability P_D^N;

  Select a repair method $r \in \mathsf{RM}$ with probability P_R^N;

  Let X_{new} and Y_{new} be the new solution obtained by apply destroy d and repair r;

  if $f'(X_{\text{new}}, Y_{\text{new}}, \Psi) < t_{\text{best}}$ **then**

    $X_{\text{best}} \leftarrow X_{\text{new}}$, $Y_{\text{best}} \leftarrow Y_{\text{new}}$, $t_{\text{best}} \leftarrow f'(X_{\text{new}}, Y_{\text{new}}, \Psi)$, $Z \leftarrow 0$;

  else

    $Z \leftarrow Z + 1$;

    $v = e^{-(f'(X_{\text{new}}, Y_{\text{new}}, \Psi) - f'(X_{\text{current}}, Y_{\text{current}}, \Psi))/T}$;

    Generate a random number $\epsilon \in [0, 1]$;

    if $\epsilon < v$ **then**

      $X_{\text{current}} \leftarrow X_{\text{new}}$, $Y_{\text{current}} \leftarrow Y_{\text{new}}$;

  $T \leftarrow hT$;

  Update $\mathbf{P}_D^N$ and $\mathbf{P}_R^N$ (Section 4.4);

  $N \leftarrow N + 1$;

partial shuttle routes Y are obtained. This process is just like a destroy operation. The next step is to repair the current partial sets X and Y by four repair methods in Section 4.3.

2. When additional EV demands are added in the middle of relocation operations: The additional EV demands and supply sets $\mathcal{D}^*$ and $\mathcal{S}^*$ are added to the undecided demand set UD, and the undecided supply set US. Then a new solution is obtained after repairing based on the new demand and supply nodes.

The process to create a new solution in the dynamic environment is described in Algorithm 3. We let X^0 and Y^0 denote current EV relocation and shuttle routing solutions, respectively. $\widetilde{\mathcal{D}}$ and $\widetilde{\mathcal{S}}$ are defined as the finished demands and suppliers, respectively. Let the tuple of lists

$$\Psi = \left(\left[(e_i, \tau_i) : i \in \widetilde{\mathcal{D}} \cup \widetilde{\mathcal{S}} \right], \left[\mathcal{L}(k) : k \in \mathcal{K} \right], \left[\mathcal{O}(k) : k \in \mathcal{K} \right] \right)$$

denote the current state of the system, which includes EVs' and shuttles' arrival time e_i, τ_i at

node i; the locations $\mathcal{L}(k)$ of shuttle k; and the number of workers $\mathcal{O}(k)$ onboard shuttle k. Because the finished EV relocation and shuttle routing cannot be changed, the following operations are done to reorganize the unfinished demands and suppliers. The value of makespan is calculated based on the current state of the system Ψ.

6 Numerical Experiments

The experiments are done on a computer that runs 64-bit Windows 10 with a 2.60 GHz Intel Core (TM i5-7300U) CPU and 8 GB RAM. ALNS and EBNSM are coded in Julia 1.6.1. For the same instance, ALNS and EBNSM run 10 times, and the average of the 10 objective values is regarded as the result for this instance. The Reinforcement Learning approach of Bogyrbayeva et al. (2022) is implemented in Python 3.6.

6.1 Benchmark Methods

Two benchmark methods, exchange-based neighborhood-search method (EBNSM) (Haider et al., 2019) and reinforcement learning approach (RL) (Bogyrbayeva et al., 2022) are used in the experiments.

EBNSM is an iterative heuristic algorithm that requires that the number of available charging stations is no less than the number of EVs that need to be recharged. Such a requirement can be too restrictive in real problems. Our ALNS lifts such a restriction by allowing multiple EVs to visit the same charging station.

Bogyrbayeva et al. (2022) focus on solving the shuttle routing by a trained neural network and making the EV relocation by using the nearest-neighbor rule. After a time-consuming training phase of the RL approach, the trained neural network can solve the problem in a fraction of a second. The main difference between RL and ALNS is that ALNS can search further neighborhoods of both EV relocations and shuttle routes, while RL only relocates EV to the nearest available charging station or demand node. Thus, ALNS may obtain a better solution than RL.

6.2 Small-Scale Instances

First, we test the accuracy of ALNS. The optimal solutions can be obtained by solving the MIP formulation using the Gurobi solver for small-scale instances. The objective values of EBNSM, RL, ALNS, and MIP are shown in Table 3. ALNS can get optimal solutions for 25% 10-nodes instances. ALNS solutions are very close to the optimality with an average gap of 0.24%. RL can also get good solutions that are 2.34% greater than the optimal solutions on average. EBNSM performs the worst within these three methods, with an average gap of 3.00%. From these small-scale experiments, we can see that ALNS finds near-optimal solutions, while other benchmark methods also find competitive solutions.

When the problem size increases, the computation difficulty grows very quickly. Gurobi can not solve the larger test instances to optimality in a short amount of time. Solve the larger test

Table 3: Objective Values of EBNSM, RL, ALNS and MIP on $|\mathcal{N}| = 10$ Instances

$\lvert\mathcal{D}\rvert$	$\lvert\mathcal{C}\rvert$	$\lvert\mathcal{S}\rvert$	$\lvert\widehat{\mathcal{S}}\rvert$	Instance No.	EBNSM	RL	ALNS	MIP
4	1	4	1	1	34.34	33.10	32.61	32.60
				2	36.12	35.98	35.12	35.11
				3	30.42	30.23	29.30	29.30
				4	27.11	26.85	26.34	26.28
				5	32.76	32.76	32.04	32.04
				6	28.31	27.98	27.61	27.51
				7	35.41	35.33	34.97	34.97
				8	39.02	38.78	38.51	38.50
				9	24.00	23.56	22.83	22.74
				10	38.02	37.71	37.45	37.45
Average					32.35	32.13	31.57	31.55
4	1	4	2	1	-	37.03	36.84	36.82
				2	-	39.12	38.52	38.41
				3	-	39.03	38.10	38.10
				4	-	27.22	26.44	26.28
				5	-	33.41	32.10	32.06
				6	-	34.87	34.01	33.96
				7	-	36.42	35.16	35.13
				8	-	40.07	38.87	38.50
				9	-	23.42	22.98	22.74
				10	-	37.98	37.67	37.45
Average					-	34.62	33.78	33.63

Table 4: Comparision between ALNS, Incumbent Feasible Solution, and Lower Bound by Gurobi

$\lvert\mathcal{N}\rvert$	$\lvert\mathcal{D}\rvert$	$\lvert\mathcal{C}\rvert$	$\lvert\mathcal{S}\rvert$	$\lvert\widehat{\mathcal{S}}\rvert$	Gap-Opt[a](%)	Gap-FS[b](%)	Gap-LB[c](%)
14	5	3	5	2	0.00	-0.41	0.41
				3	0.00	-1.13	1.13
				4	0.00	-0.98	0.98
Average					0.00	-0.84	0.84
16	5	4	6	3	99.01	4.43	98.97
				4	99.53	3.92	99.52
				5	99.61	6.65	99.59
Average					99.38	5.00	99.36
18	6	5	6	3	99.45	8.36	99.41
				5	99.32	9.45	99.40
				6	99.50	10.66	99.45
Average					99.42	9.49	99.42

a Gap-Opt is the optimality gap by Gurobi within 2-hour computational time
b Gap-FS $= \frac{\text{(Obj of Incumbent Feasible Solution by Gurobi) } - \text{ (Obj of ALNS)}}{\text{Obj of ALNS}} \times 100\%$
c Gap-LB $= \frac{\text{(Obj of ALNS) } - \text{ (Obj of Lower Bound by Gurobi)}}{\text{Obj of ALNS}} \times 100\%$

instances with $|\mathcal{N}| = 14, 16, 18$ by Gurobi in the limited computational time of 2 hours. Set the number of shuttles $K = 2$, the number of workers onboard each shuttle $W = 3$, and the number of dummy charging stations $H = 1$. The comparison between the ALNS solution, the incumbent feasible solution by Gurobi, and the lower bound by Gurobi are summarized in Table 4. In the table, Gap-Opt, Gap-FS, and Gap-LB are the average percentages of 10 test instances of each type. When the problem has 14 nodes, Gurobi can get the optimal solutions. ALNS is able to obtain a near-optimal solution with an average gap of 0.84%. When the problem size increases to 16 and 18 nodes, the growing difficulty makes it hard to narrow the optimality gap. The lower bounds are very low. Although such a long computational time (2 hours) is given to Gurobi, its incumbent feasible solutions are worse than ALNS, with average gaps of 5.00% and 9.49%.

Moreover, we also test larger instances. Gurobi could not find a feasible solution for a 20-node instance within 3 hours and a 23-node within 5 hours. Therefore, it is necessary to develop a fast metaheuristic ALNS. In what follows, the performance of ALNS is shown by the comparison with some other existing methods.

6.3 Randomly Generated Instances

We consider a 10×10 kilometers square network, which is proximate to one urban city area. There are supplier, charger, and demand nodes within the network. We fix the total number of nodes and the number of suppliers, chargers, and demand nodes. The x and y coordinates of each node are generated by a uniform distribution from 0 to 10. The initial residual charge level of the battery

Table 5: Types of Random Instances

Nodes Num	Easy				Medium				Hard			
$\lvert\mathcal{N}\rvert$	$\lvert\mathcal{D}\rvert$	$\lvert\mathcal{C}\rvert$	$\lvert\mathcal{S}\rvert$	$\lvert\widehat{\mathcal{S}}\rvert$	$\lvert\mathcal{D}\rvert$	$\lvert\mathcal{C}\rvert$	$\lvert\mathcal{S}\rvert$	$\lvert\widehat{\mathcal{S}}\rvert$	$\lvert\mathcal{D}\rvert$	$\lvert\mathcal{C}\rvert$	$\lvert\mathcal{S}\rvert$	$\lvert\widehat{\mathcal{S}}\rvert$
23	7	7	8	4	7	4	8	4	7	2	8	4
50	17	15	17	10	17	10	17	10	17	5	17	10
150	50	50	50	30	50	30	50	30	50	20	50	30

for each supplier node is generated randomly between 0 to 100%. Assume that the speed of each EV and the speed of each shuttle are equal to 30 km/h. We assume that EVs do not need to be charged and directly move to one demand node if the initial battery level is more than 50%. Otherwise, the EV moves to a charging station at first and is required to be charged fully. The charge time is assumed as 40 minutes from empty to full. The charging energy is directly proportional to the time. For example, an EV with 40% remaining battery level will spend 24 minutes to be charged fully. We do not consider the battery consumption of EV movement. In ALNS, the stop criteria is set as the maximum iteration $N_{\max}$ is 3000 and the Maximum non-improving iteration $Z_{\max}$ is 300.

The test instances are generated by three difficulty types. When the number of charging stations is more than the number of suppliers requiring recharged, the instance is easy; When they are the same, it is medium; otherwise, it is hard. The number of nodes $|\mathcal{N}|$, the number of demand nodes $|\mathcal{D}|$, the number of charging stations $|\mathcal{C}|$, the number of supply nodes $|\mathcal{S}|$, and the number of supply nodes requiring charging $|\widehat{\mathcal{S}}|$ are summarized in Table 5.

The algorithm parameters of ALNS affect the performance. To calibrate the parameters in ALNS, we tested various values using a separate set of random instances and chose the following values: 1) destroy percentage $\alpha\% = 30\%$; 2) the constant in Probabilistic s-d matching $\lambda = 0.5$; and 3) ratios for destroy $\sigma_d = (0.35, 0.4, 0.25)$ and ratios for repair methods $\sigma_r = (0.3, 0.2, 0.3, 0.2)$.

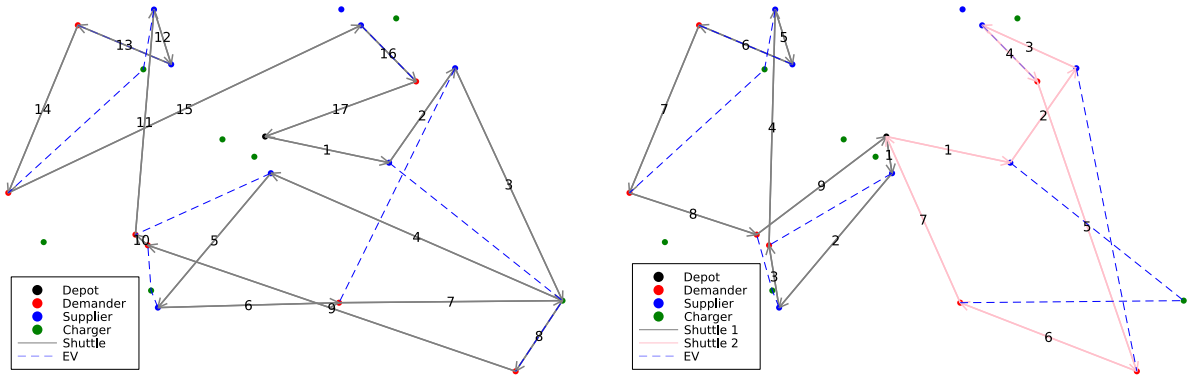
ALNS, EBNSM, and RL are implemented on 10 instances for each type. The example solutions for single shuttle and multiple shuttles are shown in Figures 3a and Figure 3b, respectively. The solutions are obtained by using ALNS to solve one easy instance with $|\mathcal{N}| = 23$.

The averages of objective values are summarized in Table 6. The percentage deviations between RL and ALNS are calculated as follows.

$$\text{PD} = \frac{\text{Objective of RL} - \text{Objective of ALNS}}{\text{Objective of ALNS}} \times 100\% \tag{35}$$

For all instances, ALNS can obtain the best average objective values. In the EBNSM structure, the charging stations are not allowed to be visited more than one time, so a feasible solution cannot be found in the hard-type instances. The average percentage deviations between RL and ALNS for easy, medium, and hard instances are 6.14%, 6.42%, and 9.10%, respectively. Therefore, ALNS can perform better than EBNSM and RL in the solution quality.

The average computational times are shown in Table 7. After training on the dataset, the trained RL model can take less than 1 second to get the solution. The training time is not included in the



(a) single-shuttle case, $K = 1$ (b) multi-shuttle case, $K = 2$

Figure 3: Example solutions for $|\mathcal{N}| = 23$ and $W = 3$

Table 6: Average Objective Values of EBNSM, RL and ALNS on Random Instances

Instances			Easy				Medium				Hard			
$\|\mathcal{N}\|$	K	W	EBNSM	RL	ALNS	PD (%)	EBNSM	RL	ALNS	PD (%)	EBNSM	RL	ALNS	PD (%)
23	1	3	15.41	13.55	**13.08**	2.45	21.81	17.93	**17.83**	1.61	-	21.93	**20.67**	3.22
	2	3	10.07	9.36	**8.83**	5.67	13.37	10.88	**10.46**	6.40	-	14.72	**14.19**	2.32
	3	2	9.22	9.08	**8.52**	6.77	12.25	11.48	**10.55**	6.03	-	13.17	**12.44**	9.11
50	1	3	30.35	23.96	**21.45**	11.88	43.10	31.20	**26.04**	12.61	-	32.92	**29.89**	11.09
	2	3	16.36	14.71	**14.33**	2.66	21.80	20.11	**18.41**	8.69	-	21.05	**17.71**	18.51
	3	2	12.18	10.48	**10.09**	1.99	16.95	16.52	**16.11**	1.32	-	16.99	**16.43**	3.38
150	1	3	53.70	38.73	**37.08**	4.21	67.34	53.98	**48.53**	10.89	-	64.27	**57.24**	12.89
	2	3	25.99	22.61	**20.37**	10.43	37.85	31.87	**30.36**	3.87	-	31.50	**29.01**	6.95
	3	2	20.96	17.87	**16.33**	9.23	27.98	23.33	**22.25**	6.37	-	25.99	**22.71**	14.39
Average						6.14				6.42				9.10

Table 7: Average Computational Times of EBNSM, RL and ALNS on Random Instances(unit: second)

Instances			Easy			Medium			Hard		
$\|\mathcal{N}\|$	K	W	EBNSM	RL	ALNS	EBNSM	RL	ALNS	EBNSM	RL	ALNS
23	1	3	6.93	0.02	**6.07**	11.28	0.02	**8.12**	-	0.02	15.22
	2	3	4.60	0.03	**3.01**	3.99	0.03	**3.04**	-	0.08	8.99
	3	2	9.34	0.06	**7.98**	5.01	0.05	**3.88**	-	0.10	7.84
50	1	3	45.32	0.05	**39.89**	55.90	0.05	**46.79**	-	0.06	50.42
	2	3	31.87	0.22	**27.24**	38.27	0.17	**30.14**	-	0.25	34.23
	3	2	21.02	0.19	**19.22**	28.20	0.22	**22.75**	-	0.31	53.81
150	1	3	100.51	0.16	**77.27**	104.01	0.17	**88.14**	-	0.20	109.98
	2	3	89.22	0.43	**74.33**	76.25	0.41	**63.74**	-	0.56	96.19
	3	2	129.53	0.89	**105.74**	182.32	1.01	**171.48**	-	1.02	163.22
Average			48.70	0.23	40.08	56.14	0.24	48.68	-	0.29	59.99

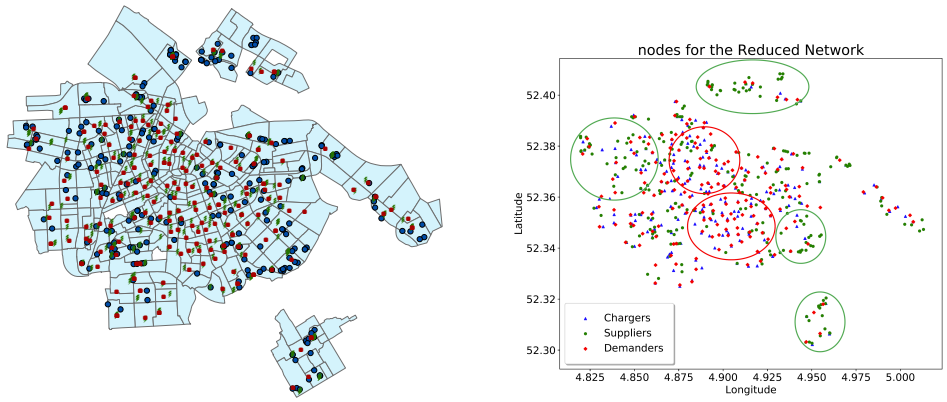
computational times in Table 7. So, the computational times of RL cannot be directly compared with the computational times of ALNS. We also observe that ALNS can run faster (21.5%) and (15.3%) than EBNSM for easy and medium instances in terms of percentage deviations of average computational times.

The numerical experiments on randomly generated instances demonstrate that ALNS outperformed both EBNSM and RL in terms of the solution quality in all problem sizes. The computational time of ALNS was shorter than EBNSM but was much longer than RL. While RL executes in less than a second in most cases, RL requires a very long training time, several days to a couple of weeks, depending on the hardware used. On the other hand, ALNS can be applied to each problem directly without lengthy training.

Moreover, the number of chargers required at each charging station is easy to calculate as the number of arrivals or departures of EVs to this charging station. Based on the ALNS results of Random Instances, the number of required chargers is not greater than 3. The percentages of 3, 2, 1, and 0 required chargers at each charging station are 7.05%, 37.86%, 47.34%, and 7.75%, respectively. The number of chargers is low and not worth considering, which supports making Assumption 5.

6.4 Case Study: car2go in Amsterdam

We apply our approach to a fully operational system of car2go in Amsterdam, the Netherlands, where the FFEVS service is operational using more than 300 EVs. From the actual data, we take the initial and target locations of EVs that need to be relocated and test the performance of our computational method. The nodes are depicted in Figure 4-(b), which clearly shows that the current EV locations (suppliers) and the desired locations (demanders) are concentrated in different areas, hence necessitating relocation operations. In order to avoid biased results and make the results meaningful in the real case study, we multiply the Euclidean distances with a detour index of 1.324



(a) Map for the Neighborhoods (b) All Nodes for Reduced Network

Figure 4: Left: Map for the Neighborhoods together with Suppliers (blue circle), Demanders (red square), Chargers (green volt sign), and Depot node (Green Pentagon), Right: All Nodes for Reduced Network

to approximate the real network distances (Chen and Chen, 2021).

To create a case study, we sample test data from 12:00 am to 7:00 am on each day. Within this period, if one EV moves from one place s to another place d, place s is set as a supply node, and place d is set as a demand node. When one EV stays at one location and the battery level increases, the location c is set as a charging station. The 143 days data are sampled, and their dates ranged from May to October in 2016. The numbers of suppliers, chargers and demanders in the final network stay within the range of [46, 251], [21, 64] and [46, 251]. The average numbers of suppliers, chargers and demanders are 85.3, 39.2, and 85.3, respectively. Since the typical speed limit in the city of Amsterdam is 30 km/h in urban residential areas, the speed of a shuttle and an EV is assumed as 30 km/h.

Two scenarios are considered in this section:

Scenario 1: No parking lots are equipped with chargers.

Scenario 2: Some parking lots are equipped with chargers.

Considering these two scenarios, we also test additional solution strategies:

RL: Using the trained neural network by RL, we can select the most probable solution by the *greedy* decoding strategy as done in the previous section.

RL-Sampling: We can generate solution samples from the trained neural network by the *probabilistic* decoding strategy instead of the greedy decoding. Among all solution samples, we choose the best solution.

RL-ALNS: We can feed the greedy RL solution as the initial solution to ALNS.

Table 8: Average makespan (unit:min) and computational times (unit:sec) in Scenario 1

Q	(K,W)	EBNSM		RL		RL-Sampling		ALNS		RL-ALNS	
		obj	time	obj	time	obj	time	obj	time	obj	time
6	(1,5)	372.35	148.6	355.82	0.9	351.47	**82.3**	352.34	155.2	**350.26**	89.7
	(2,2)	383.37	154.3	373.15	1.2	369.82	95.3	363.88	144.4	**360.64**	**91.1**
	(3,1)	380.14	169.1	367.84	1.3	364.26	114.2	360.46	151.3	**357.96**	**88.4**
8	(1,7)	365.83	176.7	346.57	1.5	341.31	152.3	341.83	149.4	**334.83**	**98.1**
	(2,3)	341.47	142.7	320.29	0.9	318.12	104.1	309.35	100.5	**306.82**	**54.3**
	(4,1)	333.28	111.9	317.33	1.6	310.00	153.2	292.16	98.2	**287.39**	**69.3**
10	(1,9)	318.60	109.9	294.54	1.3	288.06	130.7	286.75	76.3	**285.55**	**60.2**
	(2,4)	316.18	117.3	306.07	1.0	303.35	123.2	291.46	78.2	**283.75**	**55.8**
	(5,1)	299.29	102.9	276.86	1.5	271.63	140.6	258.98	92.6	**251.64**	**69.5**

6.4.1 Scenario 1

In this scenario, no parking lots are equipped with chargers. The demand, supply, and charge nodes are located at different places. The average objective values (makespan) and computational times on 143 days data are summarized in Table 8. For all instances, ALNS can perform better than EBNSM in both solution quality and computational times. ALNS can get less average objective values than RL for all instances. However, when each trained RL model is sampled 100 times in the RL-Sampling approach, a better solution can be obtained for instances $(K,W) = (1,5)$ and $(1,7)$. The computational times of RL-ALNS are calculated as the summation of two parts. The first part is the computational time of the trained RL with the greedy decoding strategy. The second part is the computational time of ALNS with the greedy RL solutions as the initial solution.

We observe that RL-ALNS can obtain the best solution in all cases, while the computational times are decreased by up to 35.05%, compared to the RL-Sampling approach. Only in the instance (1,5), RL-ALNS consumes less computational time but produces a larger objective value.

6.4.2 Scenario 2

In this scenario, many parking lots are equipped with chargers. Small changes are made to let ALNS adapt to Scenario 2. There are two cases:

1. Many demand nodes are equipped with chargers.

   All EVs can directly move to one demand node that is equipped with a charger. The charge time is not necessary to be considered. The changes in ALNS only happen in finding EV relocation solution X as follows.

   (a) In Algorithm 2, remove Lines 7-12 and only keep Lines 11-12.

   (b) In the Greedy and Probabilistic s-d matching, $\widehat{\mathcal{S}} = \emptyset$.

Table 9: Average objective values and computational times in Scenario 2

Q	(K, W)	EBNSM		RL		RL-Sampling		ALNS		RL-ALNS	
		obj	time	obj	time	obj	time	obj	time	obj	time
6	(1,5)	288.57	147.9	270.01	1.3	267.52	130.8	247.17	124.3	**243.55**	**87.2**
	(2,2)	283.55	153.2	262.54	1.2	259.96	112.2	253.06	124.4	**248.99**	**82.1**
	(3,1)	274.38	120.4	269.37	1.4	262.13	135.6	249.25	145.2	**247.27**	**104.2**
8	(1,7)	223.08	145.2	200.14	1.5	196.08	149.3	189.62	137.1	**184.03**	**89.6**
	(2,3)	284.18	172.3	256.88	1.8	246.44	156.4	245.37	103.8	**243.57**	**79.5**
	(4,1)	286.66	100.4	274.79	2.1	268.71	178.5	254.09	95.3	**247.92**	**87.3**
10	(1,9)	161.37	99.8	155.59	2.2	153.51	188.7	144.63	78.1	**138.85**	**53.7**
	(2,4)	214.07	104.2	200.33	1.6	198.24	183.4	193.37	80.6	**188.76**	**57.3**
	(5,1)	180.58	103.1	166.64	1.9	164.20	208.1	157.14	78.2	**150.51**	**60.8**

2. Many supply nodes are equipped with chargers.

   EVs at the supply nodes with chargers do not need to move to a charging station. EVs at these supply nodes are set into $\overline{\mathcal{S}}$.

Besides the changes stated above, the calculation of the makespan also changes. When a shuttle arrives at a supply node equipped with a charger and drops off a worker, this worker has to wait for EV to complete fully charging. So the EV's arrival time at supply $s \in \widehat{\mathcal{S}}$ becomes:

$$e_s \geq \tau_s + \frac{1}{\beta}\Big(100 - I_s\Big) \quad \forall s \in \widehat{\mathcal{S}}$$

The average objective values and computational times on Scenario 2 are summarized in Table 9. When the trained RL model is sampled 100 times, the average objective values of RL-Sampling become better than RL for all instances. When the RL solution is used as the initial solution in ALNS, the average objective values and computational times decrease by 2.29% and 26.77%.

6.5 Routing with Personal Mobility Vehicle

When a personal mobility vehicle is used instead of a shuttle, ALNS can solve the problem as well by following the changes in Section 5.1. The results are summarized in Table 10. The speed of scooters is assumed as 15 km/h. Since riding a scooter is much slower than a shuttle, the makespans for personnel 6, 8, and 10 are all longer. The workers spend more time in the movement. Moreover, makespan is not the only factor that affects the decision. In this section, the analysis of total operation cost and the waiting times are discussed between using shuttles or scooters.

6.5.1 Analysis on Total Operation Cost

Besides the total time spent in the system, the operation cost of EV relocation is also important in the FFEVSS. The cost consists of operating the fleet of shuttles or scooters and the labor cost of

Table 10: Average objective values and computational times of ALNS using scooters

Q	Scenario1		Scenario2	
	obj	time	obj	time
6	538.15	110.2	446.34	99.6
8	370.36	122.9	273.73	104.8
10	302.23	152.9	248.11	142.3

workers. Let T be the makespan value. Let Γ_w be the hourly cost for each worker. The median pay in 2020 for one vehicle driver was \$16.67 per hour (U.S. Bureau of Labor Statistics, 2021). Assume the per-hour labor cost Γ_w is \$17/hr. Let Γ_{sh} and Γ_{sc} be the hourly cost to operate each shuttle and each scooter. The total cost when using shuttles is calculated as $C_{sh} = Q \times T \times \Gamma_w + K \times T \times \Gamma_{sh}$. When using scooters, the total cost is calculated as $C_{sc} = Q \times T \times (\Gamma_w + \Gamma_{sc})$. Assume per hour cost of a shuttle Γ_{sh} is \$24/hr.

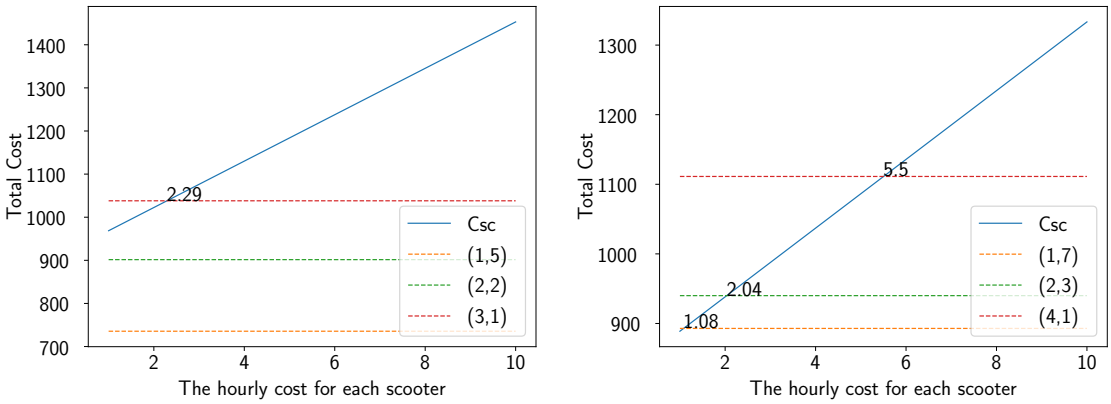
The total costs for shuttles and scooters on Scenario 1 data are analyzed in the following. The total costs for the given number of personnel 6, 8, 10, and 12 are illustrated in Figure 5. When Γ_{sc} is less than the hourly cost at the cross point, the total cost of scooters is lower. It is better to choose the scooter as the movement tool. The higher the cost Γ_{sc} at the cross point is, the better to choose scooters. It is better to choose scooters as the movement tool when the shuttle combinations are (2,2), (3,1) for personnel 6; (4,1) and (5,1) for personnel 8 and 10, respectively. For personnel 12, using scooters is better when the shuttle combinations are (6,1) and (4,2). Even though the more number of shuttles results in a short makespan, the cost of shuttles becomes expensive. Given the certain number of personnel, it is not a good choice for more shuttles with a small capacity. If the per-hour labor cost increases, the makespan will also further impact the total cost. It is important to balance makespan and the cost of movement tools. This suggests not using very large shuttles or scooters for the system with the high hourly labor cost.

6.5.2 Analysis on Waiting Times

Makespan values are influenced by both waiting times and movement times. When the mobility tool is the shuttle, the waiting times happen at charging stations and demand nodes. If a shuttle arrives earlier than EV's arrival time, the shuttle has to wait for picking up the worker; otherwise, if an EV arrives earlier than the shuttle's arrival time, the worker has to wait for a shuttle. So, the waiting time per shuttle is calculated as $\frac{\sum_{i \in \mathcal{C}' \cup \mathcal{C}^+ \cup \mathcal{D}} |\tau_i - e_i|}{K}$.

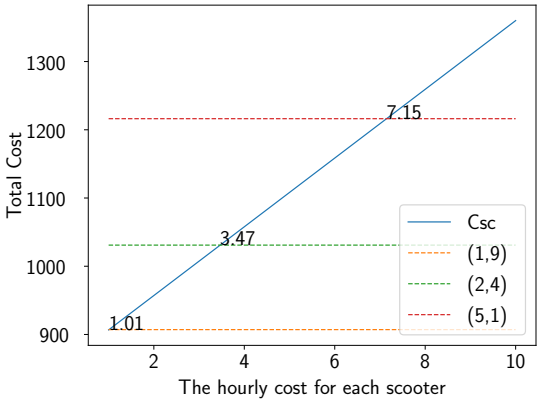
When the mobility tool is the scooter, workers can directly leave charging stations and demands by themselves. The waiting times only happen at dummy charger nodes $i \in \mathcal{C}^+$. If a worker arrives earlier, he has to wait for the EV to complete charging; otherwise, the EV has to wait for a worker to drive it. The waiting time per scooter is calculated as $\frac{\sum_{i \in \mathcal{C}^+} |\tau_i - e_i|}{Q}$.

The waiting time percentage is used to standardize the waiting times as a percentage of the

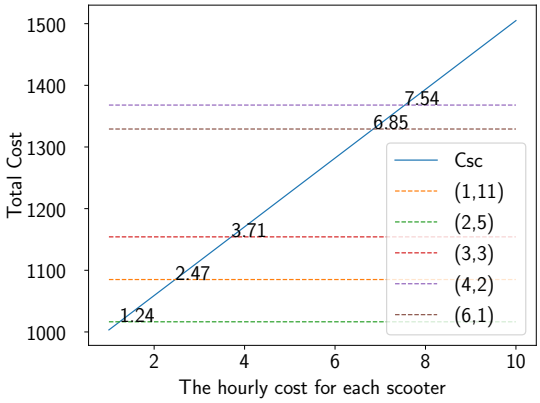


(a) Personnel = 6

(b) Personnel = 8

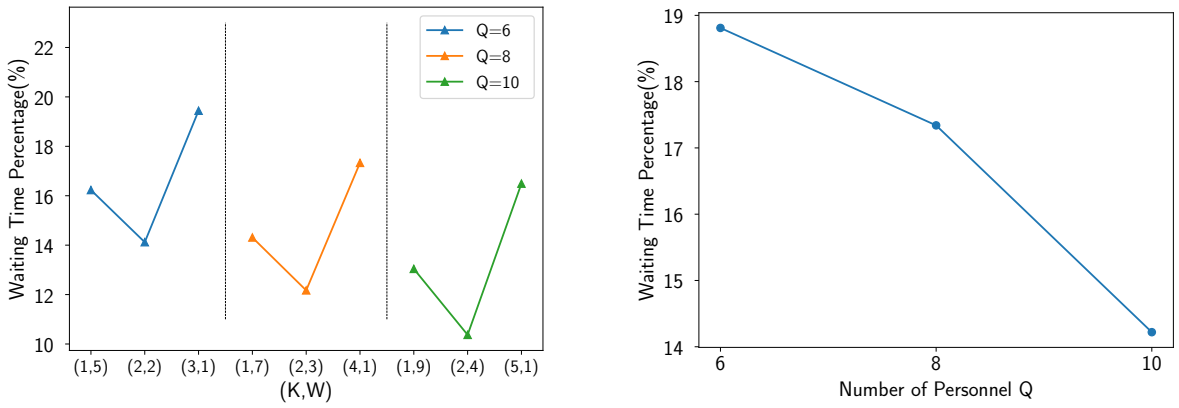


(c) Personnel = 10



(d) Personnel = 12

Figure 5: Total Cost for using Shuttles and Scooters



(a) Shuttles

(b) Scooters

Figure 6: Average Waiting Time Percentage

total time. The waiting time percentage is calculated as

$$\text{Waiting Time Percentage} = \frac{\text{Total Waiting Time}}{\text{Makespan}} \times 100\%$$

The average waiting time percentages for different combinations of shuttles and workers and 6, 8, and 10 personnel for scooters are illustrated in Figure 6. As shown in Figure 6a, given a certain number of personnel, when the number of shuttles is 2, the waiting time percentage is the lowest. When the number of shuttles is 1, the workers spend more time waiting for the shuttle to pick them up. When the number of shuttles is more than 2, the shuttles have only one worker on board and have to pick up and drop off workers frequently. With the increasing number of personnel, the workers can do the relocation simultaneously, so the waiting times also decrease. Thus, in order to decrease the waiting time, it is important to choose the shuttles with the proper capacity. As shown in Figure 6b, the waiting time percentages become smaller with the increasing number of personnel. Because the scooter speed is slow, the workers spend more time on the movement, and EVs need to wait for a worker to drive them to their destinations after charging. So, the more workers participate in the relocation assignment, the less waiting time is.

6.6 EV Relocation and Routing in Dynamic Environment

This section compares the quality of the solution with the benchmark, assuming that the upcoming change is foreseen at the beginning of the planning. We test the dynamic environment when 50% EV relocations have been done. Two cases are considered: 1) Randomly remove 10% EV demands. 2) Randomly add 10% additional EV demands. When the stop criteria are set as the maximum iteration, $N_{\max}$ is set at 3000 while for maximum non-improving iteration, $Z_{\max}$ is set at 300. The average makespan and computational times of 143 days Amsterdam data are summarized in the Column **obj** and **time** in Table 11. The objective values are calculated from the time when the shuttle departs the depot at the beginning. The computational times are the running times of

Table 11: Results of tests on Amsterdam data in Dynamic Environment

Q	(K,W)	Remove 10% EVs						Add 10% EVs					
		obj	time	5s	10s	20s	30s	obj	time	5s	10s	20s	30s
6	(1,5)	318.46	43.11	364.28	332.50	320.74	320.72	403.07	64.13	458.01	416.52	411.90	406.26
	(2,2)	312.43	32.43	364.32	326.39	316.97	313.65	388.18	53.87	438.85	412.94	392.92	390.43
	(3,1)	340.86	47.87	385.75	351.62	345.21	341.64	423.46	65.95	480.07	446.42	431.03	427.56
8	(1,7)	298.77	34.80	337.19	312.92	301.06	299.81	350.95	51.34	401.33	373.53	357.10	354.03
	(2,3)	265.33	45.73	302.46	278.07	268.73	266.73	310.62	47.27	352.48	330.21	316.07	313.52
	(4,1)	240.89	48.32	272.26	251.59	244.22	242.27	313.09	57.21	354.89	324.62	317.81	316.04
10	(1,9)	206.13	32.69	239.28	211.96	208.15	207.35	258.46	50.83	306.90	272.35	263.87	260.60
	(2,4)	234.48	27.53	264.47	244.06	237.95	234.48	284.50	43.27	332.10	297.86	289.79	287.08
	(5,1)	208.98	39.77	240.85	217.31	211.33	209.72	268.91	35.56	316.58	284.95	273.77	271.18
Avg. PD*				14.24	4.10	1.16	0.41			14.94	5.29	1.78	0.85

* Avg. PD is the Average Percentage Deviation between obj values and those under the limited time of 5s, 10s, 20s, and 30s.

getting a new solution when removing or adding EVs. When the stop criterion is set as a limited running time of 5s, 10s, 20s, and 30s, the average objective values are shown in Table 11.

ALNS can solve the dynamic case when adding or removing EVs in the middle of EV relocation operations. ALNS can also provide a new routing for shuttles in short computational times, less than 48.32s for removing 10% EVs and less than 65.95s for adding 10% EVs. When 10% EV demands are removed, the average percentage deviations are 14.24%, 4.10%, 1.16%, and 0.41% when ALNS is limited to computational times 5s, 10s, 20s, and 30s, respectively. When 10% EV demands are added, average percentage deviations are 14.94%, 5.29%, 1.78%, and 0.85%. It shows that a relatively good new solution (less than 2%) can be obtained in a short time. Therefore, ALNS is a flexible method to adapt to the dynamic environment where some EVs are assigned to external drivers or additional EV demands are added in the middle of relocation operations.

7 Concluding Remarks and Future Works

This paper considers the EV relocation and shuttle routing problem for the rebalancing operation of free-floating EV sharing systems. One of the key operational decisions for the carsharing company is how to relocate the EV fleet to meet the next day's demand with sufficient battery levels.

We developed a metaheuristic ALNS algorithm for the EV relocation problem that determines where to relocate each EV and how to route the shuttles that transport the staff drivers synchronously. We applied our method to conduct numerical experiments using both randomly generated data and actual FFEVSS data in Amsterdam. We found that ALNS outperforms EBNSM both in the solution quality and the computational time. Our ALNS also produced better solutions than the RL approach but requires much longer computational time than RL. Our experiments revealed that providing the RL solution as the initial solution for ALNS is an effective and efficient solution strategy that can take advantage of both approaches, achieving the best solution quality and reducing the computational time significantly.

We also demonstrated how our ALNS can be modified to solve the problem where staff drivers carry a personal mobility vehicle such as a scooter. Our further analysis provided practical

recommendations on which mode of transportation will be more efficient—i.e., a small number of shuttles with large capacity or a large number of shuttles with small capacity (or even personal mobility)—in terms of total operational cost as well as waiting times.

Lastly, we showed that our ALNS that destroys an incumbent solution partially and repairs to a new solution in each iteration is quite flexible to be applied to a dynamic environment. Specifically, our numerical results highlighted the usefulness of our flexible ALNS method for an environment where some EV demands are removed or added in the course of EV relocation operations.

Future research works could consider day-time dynamic relocation operations, which need to consider dynamic changes of supply and demand during the relocation operations. Then an important factor in the successful implementation of static repositioning is the accuracy of the demand forecast. The demand faced by a car-sharing system is highly sensitive to various external factors. Sophisticated data mining and demand prediction models should be devised. Instead of moving shared EVs to charging stations to recharge, mobile charging vehicles (MCVs) can be also used (Cui et al., 2022). The joint decision problems for relocations, shuttle routing, and MCV routing will be challenging future research directions.

Acknowledgments

This research was partially supported by the National Research Foundation of Korea (Project No. 2021H1D3A2A01039401). Any opinions, findings, conclusions or recommendations expressed in this manuscript are those of the authors and do not necessarily reflect the views of the National Research Foundation of Korea.

References

Barth, M., W. Li, M. Todd. 2004. Interoperability options for shared-use vehicle systems. *Transportation Research Record* **1887**(1) 137–144.

Becker, H., F. Ciari, K. W. Axhausen. 2018. Measuring the car ownership impact of free-floating car-sharing–a case study in Basel, Switzerland. *Transportation Research Part D: Transport and Environment* **65** 51–62.

Bogyrbayeva, A., S. Jang, A. Shah, Y. J. Jang, C. Kwon. 2022. A reinforcement learning approach for rebalancing electric vehicle sharing systems. *IEEE Transactions on Intelligent Transportation Systems* **23**(7) 8704–8714.

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

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

Bruglieri, M., F. Pezzella, O. Pisacane. 2019. An adaptive large neighborhood search for relocating vehicles in electric carsharing services. *Discrete Applied Mathematics* **253** 185–200.

Cai, L., X. Wang, Z. Luo, Y. Liang. 2022. A hybrid adaptive large neighborhood search and tabu search algorithm for the electric vehicle relocation problem. *Computers & Industrial Engineering* **167** 108005.

Carrese, S., F. d'Andreagiovanni, T. Giacchetti, A. Nardin, L. Zamberlan. 2021. A beautiful fleet: Optimal repositioning in e-scooter sharing systems for urban decorum. *Transportation Research Procedia* **52** 581–588.

Chen, X., Y. Chen. 2021. Quantifying the relationships between network distance and straight-line distance: applications in spatial bias correction. *Annals of GIS* **27**(4) 351–369.

Cui, S., X. Ma, M. Zhang, B. Yu, B. Yao. 2022. The parallel mobile charging service for free-floating shared electric vehicle clusters. *Transportation Research Part E: Logistics and Transportation Review* **160** 102652.

Du, M., L. Cheng, X. Li, F. Tang. 2020. Static rebalancing optimization with considering the collection of malfunctioning bikes in free-floating bike sharing system. *Transportation Research Part E: Logistics and Transportation Review* **141** 102012.

Firnkorn, J., M. Müller. 2011. What will be the environmental effects of new free-floating car-sharing systems? the case of car2go in Ulm. *Ecological Economics* **70**(8) 1519–1528.

Folkestad, C. A., N. Hansen, K. Fagerholt, H. Andersson, G. Pantuso. 2020. Optimal charging and repositioning of electric vehicles in a free-floating carsharing system. *Computers & Operations Research* **113** 104771.

Ghilas, V., E. Demir, T. Van Woensel. 2016. An adaptive large neighborhood search heuristic for the pickup and delivery problem with time windows and scheduled lines. *Computers & Operations Research* **72** 12–30.

Haider, Z., H. Charkhgard, S. W. Kim, C. Kwon. 2019. Optimizing the relocation operations of free-floating electric vehicle sharing systems. *Available at SSRN: http://dx.doi.org/10.2139/ssrn.3480725* .

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

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

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

Kek, A. G., R. L. Cheu, M. L. Chor. 2006. Relocation simulation model for multiple-station shared-use vehicle systems. *Transportation Research Record* **1986**(1) 81–88.

Kolleck, A. 2021. Does car-sharing reduce car ownership? empirical evidence from germany. *Sustainability* **13**(13) 7384.

Kypriadis, D., G. Pantziou, C. Konstantopoulos, D. Gavalas. 2020. Optimizing relocation cost in free-floating car-sharing systems. *IEEE Transactions on Intelligent Transportation Systems* **21**(9) 4017–4030.

Le Vine, S., J. Polak. 2019. The impact of free-floating carsharing on car ownership: Early-stage findings from london. *Transport Policy* **75** 119–127.

Masson, R., F. Lehuédé, O. Péton. 2013. An adaptive large neighborhood search for the pickup and delivery problem with transfers. *Transportation Science* **47**(3) 344–355.

Mattia, G., R. G. Mugion, L. Principato. 2019. Shared mobility as a driver for sustainable consumptions: The intention to re-use free-floating car sharing. *Journal of Cleaner Production* **237** 117404.

Noel, L., G. Z. de Rubens, B. K. Sovacool, J. Kester. 2019. Fear and loathing of electric vehicles: the reactionary rhetoric of range anxiety. *Energy Research & Social Science* **48** 96–107.

Osorio, J., C. Lei, Y. Ouyang. 2021. Optimal rebalancing and on-board charging of shared electric scooters. *Transportation Research Part B: Methodological* **147** 197–219.

Pal, A., Y. Zhang. 2017. Free-floating bike sharing: Solving real-life large-scale static rebalancing problems. *Transportation Research Part C: Emerging Technologies* **80** 92–116.

Preeti, W., S. Prasenjit. 2019. Car sharing market size by model (p2p, station-based, free-floating), by business model (round trip, one way), by application (business, private), industry analysis report, regional outlook, application potential, price trend, competitive market share & forecast, 2020 – 2026. `https://www.gminsights.com/industry-analysis/carsharing-market`.

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

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

Shaheen, S., A. Cohen, N. Chan, A. Bansal. 2020. Sharing strategies: carsharing, shared micromobility (bikesharing and scooter sharing), transportation network companies, microtransit, and other innovative mobility modes. *Transportation, land use, and environmental planning*. Elsevier, 237–262.

Shaheen, S. A., A. P. Cohen. 2007. Growth in worldwide carsharing: An international comparison. *Transportation Research Record* **1992**(1) 81–89.

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

Sun, P., L. P. Veelenturf, M. Hewitt, T. Van Woensel. 2020. Adaptive large neighborhood search for the time-dependent profitable pickup and delivery problem with time windows. *Transportation Research Part E: Logistics and Transportation Review* **138** 101942.

U.S. Bureau of Labor Statistics. 2021. Occupational Outlook Handbook, Passenger Vehicle Drivers. `https://www.bls.gov/ooh/transportation-and-material-moving/passenger-vehicle-drivers.htm`.

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

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

Wielinski, G., M. Trépanier, C. Morency. 2015. What about free-floating carsharing? A look at the Montreal, Canada, Case. *Transportation Research Record* **2563**(1) 28–36.

Zhu, R., X. Zhang, D. Kondor, P. Santi, C. Ratti. 2020. Understanding spatio-temporal heterogeneity of bike-sharing and scooter-sharing mobility. *Computers, Environment and Urban Systems* **81** 101483.