A Reinforcement Learning Approach for Rebalancing Electric Vehicle Sharing Systems

Aigerim Bogyrbayeva[†], Sungwook Jang[♭], Ankit Shah[†], Young Jae Jang[♭], Changhyun Kwon[†]

***Abstract*— This paper proposes a reinforcement learning approach for nightly offline rebalancing operations in free-floating electric vehicle sharing systems (FFEVSS). Due to sparse demand in a network, FFEVSS require relocation of electrical vehicles (EVs) to charging stations and demander nodes, which is typically done by a group of drivers. A shuttle is used to pick up and drop off drivers throughout the network. The objective of this study is to solve the shuttle routing problem to finish the rebalancing work in the minimal time. We consider a reinforcement learning framework for the problem, in which a central controller determines the routing policies of a fleet of multiple shuttles. We deploy a policy gradient method for training recurrent neural networks and compare the obtained policy results with heuristic solutions. Our numerical studies show that unlike the existing solutions in the literature, the proposed methods allow to solve the general version of the problem with no restrictions on the urban EV network structure and charging requirements of EVs. Moreover, the learned policies offer a wide range of flexibility resulting in a significant reduction in the time needed to rebalance the network.**

***Index Terms*— shared mobility, reinforcement learning, neural combinatorial optimization, vehicle routing**

I. Introduction

The advent of electric vehicles (EVs) and car-sharing services provides a sustainable option to move people and goods across dense urban areas. Car sharing services with EVs have the potential to increase the utilization of resources and offer a unique opportunity to the urban population in the form of free-floating EV sharing systems (FFEVSS). With the FFEVSS, customers no longer need to own a vehicle and can conveniently pick up/drop off any EV, on-demand, from the parking lots of designated service areas. However, there are some critical operational challenges to bring this on-demand service into the mainstream.

Before the start of the day, an operating company needs to relocate EVs to the ideal demand locations to establish supply-demand balance in the system. Furthermore, to provide a certain level of service, EVs need to be charged before they can be used by the customers. There are two major issues: i) there exists a sparse demand in the service area network and hence it is not trivial to find the ideal locations to relocate the EVs; ii) there needs to be an efficient routing plan to drop off the drivers for picking up the EVs and taking the EVs to the charging stations for charging, and then pick up the drivers from their respective locations. It is evident that without efficient solutions for the above operational challenges, the sustainable existence of the FFEVSS is uncertain. Therefore, we propose a decision-making framework designed to solve the above-mentioned relocation problem of the EVs.

We consider a static, nightly rebalancing problem similar to [1]–[4], where a group of drivers is used to relocate and recharge the EVs based on the predicted demand for the next day, assuming the utilization level of FFEVSS is minimal. Shuttles are used to support the movements of drivers. In this setting, rebalancing operations require two key decisions to be made: i) how to route shuttles to pick up and drop off the drivers (shuttle routing decision) and ii) where to relocate each of the EVs (EV relocation decision). In this paper, focusing on solving the shuttle routing decision problem, we propose a reinforcement learning approach, in which the EV relocation decisions are made by a rule-based approach.

The proposed RL approach possesses several advantages, compared to optimization-based approaches. First, unlike solutions coming from the static optimization techniques such as [3, 4], which need to be re-solved each time an input changes, the RL agent learns robust solutions that can be applied to any input coming from the same distribution [5]. Second, while static optimization approaches can take significant time to solve a problem, a trained RL agent can be invoked to produce quality solutions instantaneously. Third, many practical considerations can be flexibly incorporated within the simulator in the training phase.

The shuttle routing to rebalance FFEVSS with its variety of trade-offs is not a trivial problem. For instance, as depicted in Figure 1, one may allow or disallow the reuse of charging stations in the derivation of solutions. The former choice offers more flexibility, but it also increases the complexity of exploring solutions. Therefore, the existing methods do not allow reuse of the charging stations [4]. On the other hand, such choice results in the opportunity loss. Another trade-off is depicted in Figure 2, where the first supplier node has an EV that needs to be recharged while the second supplier has an EV with a sufficient charging level. Then one needs to balance between traveling time and waiting time related to routing a shuttle to supplier nodes. The complexity of such routing decisions increases with the network size, its structure and the number of shuttles and drivers deployed. Hence, it may not be possible to efficiently explore potential solutions with human-driven heuristics. With the proven ability of neural networks in recognizing patterns in graph-based representations, the utilization of a neural network architecture with the proposed RL approach

[†]Department of Industrial and Management Systems Engineering, University of South Florida, Tampa, Florida, USA, Email: {aigerimb, ankitshah, chkwon}@usf.edu

[♭]Department of Industrial and Systems Engineering, KAIST, Daejeon, South Korea, Email: {jedi829, yjang}@kaist.ac.kr

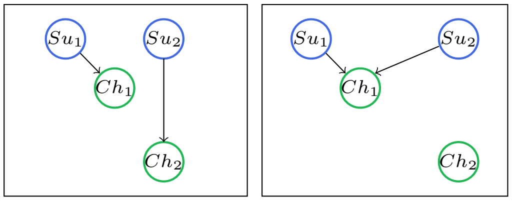


Fig. 1: Assign supplier-charger pairs or reuse charger nodes? Ch and Su denotes charger and supplier nodes respectively.

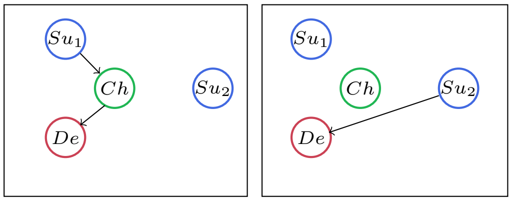


Fig. 2: How to balance traveling time and waiting time trade-off? De, Ch and Su denotes demander, charger and supplier nodes respectively.

will provide better approximations and assist in obtaining efficient solutions that can be generalized.

In recent years, there has been a surge of studies that apply reinforcement learning to solve various vehicle routing problems (VRPs) [6]–[8]. The proposed solution approaches mainly apply to the traditional VRP settings such as capacity constraints, time windows and stochastic demand. The shuttle routing problem, taken under this study, possesses significant differences with other VRPs. First, in a VRP setting, demand is independent of the routing decisions. However, in the shuttle routing problem, locations of drivers to be picked up are determined by preceding routing decisions, highlighting a *strong interdependence* between demand and routing. Second, unlike VRP, the shuttle routing problem is characterized by *delayed rewards*. As shown in Figure 3, the actual relocations of EVs from a node happen after the execution of shuttle routing to the node. As a result, we observe delayed rewards with respect to the shuttle routing decision only after EVs reach their designated nodes. Consequently, such differences require a new approach to finding solutions for the shuttle routing problem.

We consider two settings of rebalancing FFEVSS. In the first setting, we focus on a single shuttle problem, where we train a single agent to learn routing policies. In the second setting, we aim to train a fleet of shuttles through a single-agent reinforcement learning, where a central controller is responsible for routing multiple shuttles. In both cases, we deploy policy gradient methods along with recurrent neural networks for training. The shuttle routing problem under both of the above-mentioned settings possesses significant challenges that prohibit the direct use of the existing solution methods. For instance, in routing a single shuttle, we must train an agent not only to find efficient routes, but at the same time maintain the feasibility of the solutions related to the precedence of the visiting nodes. As for routing the fleet of shuttles, in the training, we must promote learning policies to route multiple shuttles that will contribute to a common goal.

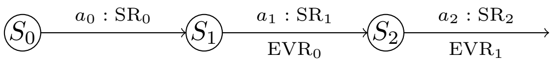


Fig. 3: State transitions: SR$_i$ - shuttle routing decisions, EVR$_i$ - EV relocation decisions, a_i - selected action

The main contributions of this study are as follows. First, to the best of our knowledge, this study is the first to present an RL-based approach for handling *multiple* vehicles explicitly in the context of VRPs, while focusing on the shuttle routing problem for rebalancing the FFEVSS. Second, within the RL framework, we propose the utilization of deep neural network architecture to process the complex and high dimensional observations from an urban service area network to help train the RL agent in its decision-making. In particular, we adopt sequence to sequence models with attention mechanism to fit the unique challenges of the rebalancing FFEVSS. Third, we present a novel training algorithm to route efficiently a fleet of shuttles to rebalance FFEVSS by utilizing policy gradient methods. Our training algorithm does not require splitting an urban network into sub-clusters for each shuttle, but instead allows developing policies that efficiently utilize shuttles and drivers in a whole network. Fourth, we develop a simulator to mimic real-world FFEVSS, which serves as the environment for training an RL-agent and allows efficient exploration of joint actions of multiple shuttles.

Moreover, unlike the solutions obtained using the methods from the literature, the empirical results obtained from this study show that the proposed method allows solving the general version of the problem with no restrictions on the urban network structure and charging levels of EVs. Moreover, the learned policies offer a wide range of flexibility resulting in a significant reduction in the time needed to rebalance the network.

The remainder of the paper will proceed as follows. In Section II we provide an overview of relevant literature and outline the unique challenges of the rebalancing FFEVSS. In Section III we present the problem formulation. In Section IV we introduce the proposed reinforcement learning model. In Section V we demonstrate the results of our computational studies. Lastly, in Section VI we provide concluding remarks.

II. Related Work

Even though the problem of rebalancing FFEVSS has been recognized as essential for their sustainable existence in the literature [9, 10], most of the studies focus on high-level approaches to address the issue. One category of studies falls on incentive-based methods that aim to rebalance the system through influencing customer behavior [11]. Another set of papers study the deployment of personnel and offer rule-based high-level decision-making frameworks [12, 13]. There are only a few studies that specifically focus on the shuttle routing problem to rebalance FFEVSS, thus offering detailed solutions for day to day operational challenges.

One of such studies is [3], which aims to solve both for EV relocation and shuttle routing problems jointly. However, the proposed model does not enforce relocation of EVs directly to demander nodes, but indeed permits leaving EVs in charger nodes. As a result, charger stations will be blocked and cannot be reused requiring the postponing of charging for the remaining set of EVs. Similarly, in a recent study [4] presents novel approaches in addressing EV relocation and shuttle routing problems simultaneously. Even though the study aims at relocating EVs directly to demander nodes, it assumes the abundance of charger stations in an urban network. Thus, again reusing charger stations is not considered and the postponement of charging for EVs requiring it is allowed. Since charging infrastructure is often limited [14], the reuse of charging stations must be an integral part of solutions to rebalance FFEVSS in real-world urban networks.

Recently reinforcement learning approaches gained popularity to solve various problems in transportation including fleet management and rebalancing in ride-hailing services [15]–[18]. However, none of the existing studies focus on FFEVSS specifically and do not address the unique issue of charging and relocation together. For solving VRPs, deep reinforcement learning has been first applied in [6], which utilizes sequence to sequence methods [19] and an attention mechanism [20]. Later [21] adopted the transformer model [22] to solve VRPs without recurrent neural networks. [8] proposes a novel model to solve online VRPs by utilizing neural combinatorial optimization and deep reinforcement learning. Similarly, [23] presents a hybrid model that combines local search with attention mechanism. However, these studies focus on routing a single capacitated vehicle, where the main goal is to minimize the distance traveled. While multiple loops of a single capacitated vehicle can be interpreted as multiple vehicles, this paper is the first to present an explicit modeling of multiple vehicles within an RL framework.

Although this study also adopts sequence to sequence models with attention mechanism similar to [6], the significant differences in the nature of the rebalancing FFEVSS problem and VRP dictate the development of novel solution techniques. For instance, in the given problem, shuttles need to leave a depot, drop off, pick up drivers who relocate EVs, and return back to the depo, highlighting two sets of constraints. First, the precedence of visited nodes needs to be maintained when charging stations are visited after nodes with EVs and nodes that require EVs are visited after either charging stations or nodes with EVs. Second, the capacity constraint must be satisfied when nodes with EVs are visited only when there is a driver in a shuttle and nodes with drivers are visited only if there is seating available for a driver in the shuttle. In addition to feasibility constraints, since both charging and relocations of EVs are involved in the shuttle routing problem, only considering factors that affect the total distance traveled is not sufficient. Moreover, the dynamics of an urban network due to routing a shuttle is more complex compared to the VRP due to the delayed movements of EVs relocation. Also, routing multiple shuttles requires a novel training algorithm. In particular, when several shuttles are present in an urban network and each of their movement influence the state of the network, we need a novel framework that enables the application of reinforcement learning tools based on Markov Decision Process (MDP).

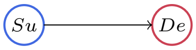


Fig. 4: EV relocation Case I.

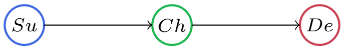


Fig. 5: EV relocation Case II.

III. Problem Statement and Formulations

A. Network

Let us consider a network $\mathcal{N}$ consisting of N number of nodes and a depot. We define a node as a supplier, if it has an excess EV, and a demander, if it requires an EV. The network also has charger nodes. Each node in the network can store at most one EV. Depending on the charging levels of EVs there are two possibilities of the EVs relocation. In Case I, EVs are relocated from supplier nodes directly to demander nodes as shown in Figure 4. In Case II, EVs first need to be taken to charger nodes and after charging is complete, they need to be relocated to the demander nodes as shown in Figure 5. We consider discrete charging levels of EVs, where a threshold-based rule is applied to decide whether to charge an EV or not. We consider two settings of the problem when a single shuttle or a fleet of shuttles is deployed for rebalancing the system. We formulate the routing problem for a single shuttle as MDP and utilize a central controller to route a fleet of shuttles.

B. Multi-shuttle Routing as MDP

Even though it is possible to formulate the routing of a fleet of shuttles using a multi-agent reinforcement learning framework, such an approach suffers from several drawbacks. Firstly, in the presence of several shuttles, each of which is treated as an autonomous agent, the stationary assumption of MDP is no longer valid [24]. In particular, in the presence of other agents in the environment, the Markovian property, which states that reward and current state only depends on individual action and previous state, does not hold. Therefore, a multi-agent reinforcement learning framework works under partially observable MDP [25]–[27], when each agent can observe only a local view of the network [28]. Then each agent can only visit nodes visible from its local view, which imposes significant restrictions on developing an efficient routing. Secondly, training autonomous agents is challenging without making strong assumptions about constant communication between agents. For instance, if at the current time step one agent selects a node to visit, then such information must be shared among other agents to avoid the presence of several agents at the same node. Lastly, under a static network, when the state of the network is constant and well-known, a centralized approach will help navigate a fleet of shuttles efficiently. Therefore, we formulate routing multiple shuttles to rebalance FFEVSS using a central controller, that is responsible for making routing decisions of all shuttles. Then, we can formulate the problem using a single-agent reinforcement learning framework and MDP.

A fleet of shuttle with drivers leaves a depot and visits nodes in the network to relocate EVs from supplier nodes to demander nodes. Shuttles must return to a depot after fulfilling demand at all demander nodes and picking up all the drivers. These sequential decisions of a central controller for routing shuttles under uncertain demand (locations of drivers) can be formulated as a finite horizon MDP, where the future dynamics of the system depend only on the current state. We define the RL framework for the problem as tuple $\mathcal{M} = \langle X, \mathcal{A}, P, R, T\rangle$ representing states, actions, transition probabilities, reward function, and time horizon, respectively. The definitions are as follows:

- $\mathcal{I} = \{1, ..., I\}$ is the set of I shuttles that are controlled by a central controller;
- State set X represents the network, where for each node it shows its location, the relative distance, the number of EVs, the number of drivers, the charging levels of EVs' and indicators for the expected transitions. We utilize binary vectors to indicate if there is an expected EV coming to a node. We denote state as x_t at time t.
- $\mathcal{A}$ is the set of joint actions such that $\mathcal{A}_t = \mathcal{A}_t^1 \times \mathcal{A}_t^2 \times \cdots \times \mathcal{A}_t^I$, where $\mathcal{A}_t^i$ is the action set of shuttle i at time t and action a_t^i indicates a node number to be visited next by shuttle i. Then a central controller's action set consists of joint actions of all shuttles, $\mathcal{A}_t$, at time t.
- Transition Probabilities function, P, determines state transitions probabilities $p(x_{t+1}|x_t, a_t)$ at time t with respect to taken action a_t. In the given problem, transitions are deterministic, but often delayed. After an action is taken, the relocations of EVs are scheduled. However, the actual state transitions related to the movements of EVs occur later as shown in Figure 3.
- All shuttles share a common reward R and immediate reward r_t, which are assigned based on the joint actions of all shuttles at time t denoted by a_t and state x_t;
- Instead of defining the specific time value of T, we define one episode rollout for the problem based on the experiment outcomes. One episode is terminated either if all demander nodes are fulfilled and all drives are picked up back to a depot or if the total number of time steps exceeds the predefined maximum time steps, the value of which is set based on the size of a network.
- Each time step t is determined by the earliest fulfilled action among all shuttles. Thus, each time step starts when a central controller takes an action and finishes whenever any action is fully executed.

IV. Reinforcement Learning Model

We adopt policy gradient methods, that are similar to those popularly used in routing problems [6, 8, 21], to learn the complex routing policies of shuttles *directly*. In general, policy gradient methods consist of two separate networks: an actor and a critic. The critic estimates a value function given a state according to which the actor's parameters are set to generate policies in the direction of improvement We train an agent and a central controller to route a single shuttle and multiple shuttles in an urban network by simulating the FFEVSS environment. The simulator is developed to handle EVs relocations through rule-based decisions and utilizing sequence to sequence models to generate policies.

A. The FFEVSS Simulator

The main function of the FFEVSS simulator is to represent the dynamics in an urban network caused by movements of shuttles. There are immediate and delayed transitions related to routing shuttles. In an immediate update to the environment at each time step, we consider locations of shuttles, drivers, EVs, the number of drivers in a shuttle and fulfillment of scheduled transitions either related to charging or relocation of EVs. Also, at each time step, we schedule transitions related to movements of EVs that have started, but unfulfilled. In particular, starting at current clock time $t_c = 0$, we update the environment according to movements of a shuttle:

$$t_c \leftarrow \begin{cases} t_c + \tau(n_{t-1}, n_t) & \text{if } n_{t-1} \neq n_t \\ t_c + w_t & \text{if } n_{t-1} = n_t \end{cases}$$

where τ represents traveling time between nodes visited by a shuttle at time $t-1$ and t and w_t denotes waiting time at node n. We define waiting time at node n as the difference between the time when a delayed transition at node n occurs and the time when a shuttle reaches node n. To account for delayed transitions we introduce a time vector, which keeps track of remaining times until either EVs arrive at designated nodes or their charging completes. In the case of multiple shuttles, the environment is updated with the earliest movements of shuttles.

Another function of the FFEVSS simulator is to update a masking scheme according to the current state of the urban network. The masking scheme helps to maintain the feasibility of solutions related to the precedence of visited nodes and the number of drivers in a shuttle. Also, having an efficient masking scheme expedites the exploration of action space. We deploy the following masking scheme, where $\mathcal{A}_t = \emptyset$ stores the set of available nodes/actions to visit at time t and the rest of the nodes are masked. For each $n \in \mathcal{N}$, we update:

$$\mathcal{A}_t \leftarrow \begin{cases} \mathcal{A}_t \cup \{n\} & \text{if } l_t > 0 \quad \text{and } n \in \mathcal{D}_t \cup \mathcal{E}_t \\ \mathcal{A}_t \cup \{n\} & \text{if } l_t = 0 \quad \text{and } n \in \mathcal{D}_t \end{cases}$$

Here set $\mathcal{E}_t$ denotes nodes with an EV or nodes with the expected EV due to delayed transitions, set $\mathcal{D}_t$ denotes nodes with a driver or nodes with the expected drivers, and l_t denotes the number of drivers in a shuttle at time t.

B. EV relocation decisions

As described earlier, our focus in this study is to solve for the shuttle routing problem. Hence, we are using a rule-based approach for EVs' relocation decisions. The rule-based approach is as follows: every time a supplier node with an EV has a driver, that EV is relocated to the nearest available either charger or demander node. The decision of whether to relocate an EV to a demander or charger node is predetermined in the settings of a simulator. We apply a threshold-based rule; that is, if the charging level of an EV exceeds the threshold, then it can be directly relocated to a demander node or must be charged, otherwise.

We maintain a binary vector in the simulator to indicate if a charger node is available or not. This representation helps in deciding the relocation of an EV from a supplier node to an available charger node. We determine the closest available charger node by multiplying the binary vector by a time matrix that indicates time to travel among any pair of nodes. To decide EVs' relocations from either supplier or charger nodes to demander nodes, we maintain a demand matrix that keeps track of demander nodes that still need an EV at time t. In particular, in the simulator we store time needed to move from all nodes to each demander node and increase those values to large numbers, if a demander node is satisfied. Then, if an EV needs to be relocated to a demander node, we compute the minimum time from a node to the closest demander nodes.

C. A sequence-to-sequence model for the shuttle routing problem

Motivated by [6], we propose using a sequence to sequence model for rebalancing FFEVSS, which typically consists of an encoder and a decoder. Given urban network $\mathcal{N}$, we aim to generate a sequence of nodes to be visited by either a shuttle or a fleet of shuttles. In other words, we are interested in learning the following conditional probability or parametrized policy π_θ:

$$\pi_\theta(Y_T|x_0) = \prod_{t=0}^{T-1} \phi(y_{t+1}|x_t, Y_t) \tag{1}$$

In (1), we let $x_t = \{x_t^1, \ldots, x_t^N\}$, where x_t^n denotes static and dynamic states of node n at time t. Unlike in machine translation, the state of nodes in the network status changes dynamically with shuttles movement; thus, we need to consider both static and dynamic states for each node. Also, we let y_t denote a node to be visited at time t and $Y_t = \{y_1, \ldots, y_t\}$ with $Y_0 = \emptyset$. Then to select a next node to visit y_{t+1}, we are interested in learning $\phi(y_{t+1}|x_t, Y_t)$.

However, a set of nodes in the network does not convey any sequential information. Therefore, it is common in literature [6], to omit recurrent neural network for encoding. Indeed, due to the sparse nature of networks, graph embedding is deployed in encoder to build their continuous vector representation as they suit better for statistical learning [29]. The following equation describes embedding for each $n \in \mathcal{N}$:

$$\bar{x}_s^n = b^s + W^s x_s^n \tag{2}$$

$$\bar{x}_{d_t}^n = b^d + W^d x_{d_t}^n \tag{3}$$

where, $\bar{x}_s^n$ and $\bar{x}_{d_t}^n$ are embedded static and dynamic states of node n at time t and b, W represent the trainable parameters of a neural network. We denote by $\bar{x}_t^n = (\bar{x}_s^n; \bar{x}_{d_t}^n)$ concatenation of embedded static and dynamic states of nodes.

For decoding we use recurrent neural networks (RNN), that takes static state of the last visited node and stores the sequence as follows:

$$h_t = W^h f(h_{t-1}) + W^x \bar{x}_s^n \tag{4}$$

where h_t is a memory state of RNN, f represents nonlinear transformation and x_s^n is a static state of node n visited at time t. Trainable weight matrices W^h and W^x represent connections between hidden state to hidden state and hidden state to an input respectively.

In addition to encoder and decoder, we also utilize content based attention mechanism as in [6]. Content based attention tries to mimic associative memory and is designed to handle cases when an input to the sequence to sequence model is a set [20]. In particular, the current state of an urban network is coupled with the memory state of RNNs about the sequence to calculate an alignment vector c_t that assigns the probabilities of nodes to visit next:

$$u_t^n = v \tanh(W(\bar{x}_t^n; h_t)) \qquad \forall n \in \mathcal{N} \tag{5}$$

$$c_t = \text{softmax}(u_t) \tag{6}$$

where v and W are trainable weight matrices.

For the problem under study, we define the static state of nodes as their location coordinates and the initial charging levels of EVs at supplier nodes. Even though charging levels of EVs will change as EVs are taken to charging stations, only their initial values determine charging times. Therefore, we consider them as a static state of nodes. For dynamic representation of the states of nodes we use the number of EVs, the number of drivers in a shuttle and the distance from the current node to other nodes. Our experimental studies show that passing distance information as a dynamic state of nodes substantially reduces training time. Figures 6 summarizes the sequence to sequence model of the shuttle routing problem. In routing a fleet of shuttles, we also deploy a single actor network, where a sequence of visited nodes Y_t, includes nodes visited by all shuttles up to time t.

D. Reward Function

Reward function along with sets of available actions reflects our aim to maintain the feasibility and efficiency of routing decisions. Since the shuttle routing problem considers both charging and relocation of EVs, reward function must not only reflect traveling times between nodes, but also include waiting times. Therefore, we define reward function as the negative of total time spent in the system starting when a shuttle or a fleet of shuttles leaves a depot and

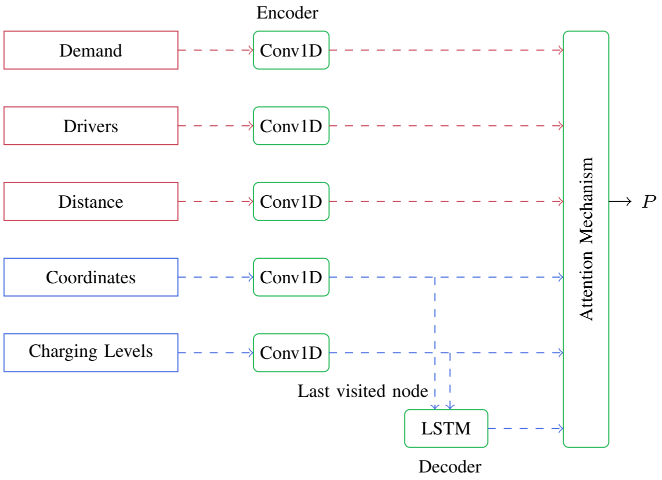


Fig. 6: Sequence to sequence model for the shuttle routing problem, red represents dynamic and blue represents static states of nodes. Conv1D represents an 1-dimensional convolution neural network for embedding and $P \approx \phi(y_{t+1}|x_t, Y_t)$.

ending when all shuttles are returned back to the depot with all drivers after fulfilling all demander nodes. Then our aim is to maximize the negative of total time spent in the system denoted by R. More formally we define reward function as follows, using immediate rewards r_t:

$$R = \sum_{t=1}^{T} r_t \tag{7}$$

where

$$r_t = \begin{cases} -\tau(n_{t-1}, n_t) & \text{if } n_{t-1} \neq n_t \\ -w_t & \text{if } n_{t-1} = n_t \end{cases}$$

and τ_t is traveling time and w_t is waiting time at time t.

E. Training Algorithm

In training we are interested in finding policy parameters θ that maximize the total expected reward:

$$\theta_\pi = \underset{\theta_\pi}{\operatorname{argmax}} \, \mathbb{E}_{\pi_\theta}[R] \tag{8}$$

Given the state of network X, we can rewrite the expression as follows:

$$J(\theta_\pi|x) = \mathbb{E}_{\pi(\theta_\pi|x)}[R(\pi|x)] \tag{9}$$

Then we update values of policy parameters θ in the direction of the following gradient using Advantage function A^π:

$$\nabla_\theta J(\theta_\pi|x) = \mathbb{E}_\pi[A^\pi \nabla_\theta \log p_{\theta_a}(\pi|x)] \tag{10}$$

$$A^\pi = R(\pi|x) - V(x_0) \tag{11}$$

Even though it is possible to estimate Advantage function using temporal difference methods by utilizing n-step returns, the nature of the problem under study dictates considering a full episode to minimize the total time spent in the system. Therefore we use the REINFORCE algorithm with a baseline [30] as the value of the initial state of an urban network estimated by a critic. Algorithm 1 represents our training procedure. The details of Data Generator can be found in the experiments section. Unlike in the existing literature [4], the algorithm does not require splitting an urban network into sub-clusters for each shuttle, but instead deploys all shuttles to serve the whole network. Also, utilizing a central controller that observes the entire urban network state along with the masking scheme in the simulator, allows efficiently exploring joint action of all shuttles. For instance, if a node has been assigned to be visited by a shuttle then that node is masked for other shuttles.

Algorithm 1 Training Algorithm

```
Initialize network parameters $\theta^a$ and $\theta^c$ for actor and critic networks respectively. Set the maximum number of epochs, a batch size and the maximum number of steps denoted as $M_{\text{epochs}}$, $M_{\text{epis}}$ and $T$ respectively;
for epochs = 1 to $M_{\text{epochs}}$ do
    Reset gradients $d\theta^a, d\theta^c$;
    for $m$ = 1 to $M_{\text{epis}}$ do
        data $\sim$ DataGenerator($\rho$);
        $x_0^m$, $\mathcal{A}_0$ = simulator.reset(data);
        Store initial state $x_0^m$ in $X_0$, set $R^m = 0$;
        Add index of each shuttle $i$ to $L$;
        for t=0 to $T$ do
            for each $i \in L$ do
                $a_t^i$, $p_t^i$ = actor-network($x_t$, $\mathcal{A}_t^i$);
                Store $p_t^i$ in $p^m$, remove $a_t^i$ from $\mathcal{A}_t$;
            end for
            $x_{t+1}$, $\mathcal{A}_{t+1}$, $r_t$, $t_c$ = simulator.step($a_t$);
            Empty set $L$;
            for each $i \in \mathcal{I}$ do
                if $a_t^i$ is complete at $t_c$ then
                    add $i$ to $L$
                else
                    $a_{t+1}^i = a_t^i$ and remove $a_t^i$ from $\mathcal{A}_{t+1}$
                end if
            end for
            $R^m = R^m + r_t$;
        end for
        calculate $V^m(x_0^m; \theta_c)$ using critic
    end for
    $d\theta^a = \frac{1}{M_{\text{epis}}} \sum_{m=1}^{M_{\text{epis}}} (R^m - V^m(x_0^m; \theta_c)) \nabla_{\theta^a} \log p^m$;
    $d\theta^c = \frac{1}{M_{\text{epis}}} \sum_{m=1}^{M_{\text{epis}}} \nabla_{\theta^c} (R^m - V^m(x_0^m; \theta_c))^2$;
end for
```

V. Computational Studies

A. Data Generation and Configurations

We consider 1×1 square mile network consisting of demander, supplier, and charger nodes. We first specify the total number of nodes in the network and the number of demander and charger nodes. We sample x, y coordinate of each node from a uniform distribution with values ranging from 0 to 1. Similarly, we sample demander, charger, and supplier nodes from a uniform distribution. For each supplier

TABLE I: Hyperparamter values

Hyperparameter	Value
Convolution 1D hidden dimensions	128
LSTM hidden dimensions	128
Critic hidden dimensions	128
Feed-forward network hidden dimension in critic	128
Learning rate for actor and critic	0.0001

node we set the initial charging levels of EVs randomly between 1 and 5. We assume that EVs do not need charging and can be directly taken to demander nodes if their charging levels exceed 3. Otherwise, EVs first need to be taken to charger nodes, where all of them charged until the charging level of 5 is reached. For each charging level, we assign the charging time equal to average traveling time between all pairs of nodes in the network. We do not consider discharging rates in the movements of EVs. We assume the constant velocity for EVs equal to 45 miles/hour.

Computational experiments are conducted with 2 Intel Xeon E5-2630 2.2 GHz 20-Core Processors (30MB), 32GB RAM, and the Ubuntu 18.04.4 LTS operating system. All implementations are done in Python 3.7 using PyTorch 1.5. In our implementations of critic network has similarities to the actor network structure except using RNN to store sequence information. We first embed the initial static state of the urban network using 1D convolution networks and then pass it to attention mechanism, where RNN hidden state is replaced by a matrix of zeros. We repeat the process three times and pass the output of attention mechanism through a sequence of feed-forward networks to obtain the final estimate for a value function. Table I represents the hyperparameters used for training.

We train RL agents on networks of various sizes and difficulty levels. For each problem class defined by the size of a network, we consider instances with 3 different levels of difficulty. Cases when there is an abundant presence of charging stations than the number of EVs requiring charging we call *easy* instances. Similarly, cases when there is an exact number of charging stations as the number of demander nodes we call *medium* difficulty instances. Finally, cases when there is a less number of charging stations than the number of demander nodes we call them *hard* instances. The descriptions of difficulty levels are found in Table II.

B. RL agents and Benchmarks

We train three types of agents using the proposed RL models. The first agent denoted as *gen-RL* is trained on all three difficulty levels, but on a fixed network size. The second agent denoted as *net-RL* is trained on networks of various sizes, but it is tailored to a specific difficulty level. The last agent denoted as *RL* is trained on a fixed network size and on a specific difficulty level.

For our computational studies, we consider a benchmark from [4]. The benchmark model denotes as *Sim* represents a joint model that solves for EVs relocation and the shuttle routing problem simultaneously. To solve multi-shuttle routing problems, the heuristic splits an urban network into some clusters and solve a single-shuttle routing problem for each cluster. However, there are some limitations of the method. One of them is related to the inflexibility of the solutions when drivers that have been dropped off from one shuttle cannot be picked up by other shuttles. Another disadvantage is related to charger nodes. The heuristic can only handle the cases of the problem when the number of charger nodes is not less than the number of EVs that must be charged.

C. Results

Figure 7 shows training rewards for the single-shuttle and multi-shuttle problems on the network with 23 nodes and 3 drivers. Overall, training time depends on the network size, its structure and the features passed to the actor network. In both cases, using distance information from the current node to other nodes in the actor network results in better rewards compared to when not passing such information.

To compare different RL agents' performances we conduct experiments on various network sizes and the degree of difficulty of instances and measure the mean of the total time spent in the system out of 128 instances. Table III shows the experiments' results. In most instances RL agent trained on a specific size and a specific instance difficulty level tend to perform the best. We observe that net-RL agents, trained on various network sizes, tend to perform better on larger network sizes, while gen-RL agents, trained on various difficulty levels, can be competitive on medium sized networks. As the network size increases, the results show that using net-RL and gen-RL agents can be beneficial. For the rest of experiments, we use RL agent.

Table IV illustrates the performance of the RL solutions with those of the heuristic optimization method, labeled Sim. The reinforcement learning approach can solve all instances of the problem, while the optimization method can handle only easy and medium cases. Moreover, for easy and medium cases measured in the mean of total time spent in the system, the RL solutions perform better than the heuristic optimization solutions. We also, note that the derived RL solutions do not solve for optimal relocation of EVs and only based on predefined rules, while the simultaneous approach of the heuristic optimization solves for both the shuttle routing and EV relocation problems. Also, Table IV shows the performance comparison of Sim and RL models in terms of percentages of winning instances. For instance, in a RL-Sim pair comparison, the value of cells under the column indicates the percentages of instances when RL model performed at least equally to Sim model out of 128 test instances. As shown in Table IV the RL model performs better than the heuristic method in at least 50% of all instances, except one instance.

To show the generation of the instantaneous solutions using RL models, we measured computation time. Table V demonstrates the computation time it takes to derive a solution under Sim and RL models. We report an average time to solve an instance out of 128 instances in total.

TABLE II: Difficulty levels description, where De, Ch, Su, and $Su^{'}$ denote the set of demanders, chargers, suppliers, and suppliers with EVs that require charging, respectively.

	Easy				Medium				Hard			
$\lvert\mathcal{N}\rvert$	$\lvert De\rvert$	$\lvert Ch\rvert$	$\lvert Su\rvert$	$\lvert Su^{'}\rvert$	$\lvert De\rvert$	$\lvert Ch\rvert$	$\lvert Su\rvert$	$\lvert Su^{'}\rvert$	$\lvert De\rvert$	$\lvert Ch\rvert$	$\lvert Su\rvert$	$\lvert Su^{'}\rvert$
23	7	7	8	4	7	7	8	8	8	6	8	8
50	16	16	17	8	16	16	17	17	17	15	17	17
100	33	33	33	16	33	33	33	33	33	32	34	34

TABLE III: Comparison of RL agents in terms of total time spent in the system, the average of 128 test instances are reported. In bold are the best results.

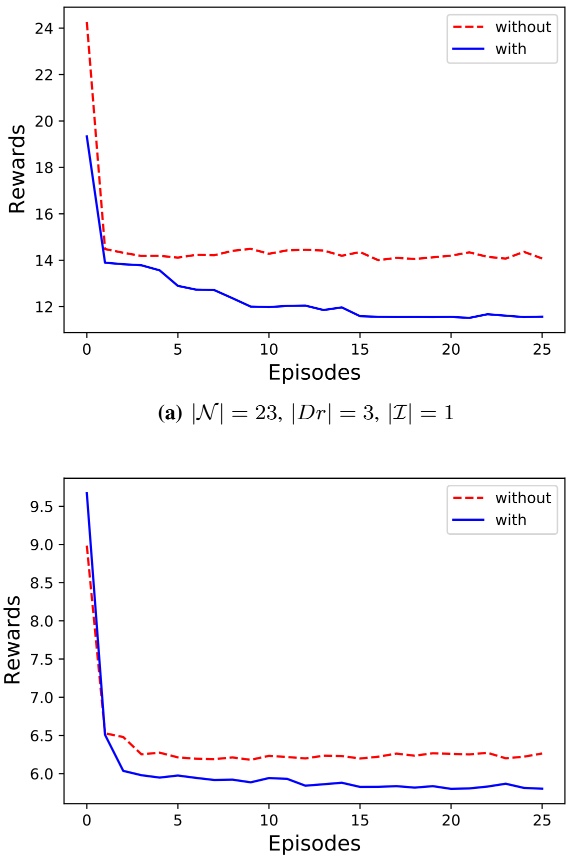
			Easy			Medium			Hard		
$\lvert\mathcal{N}\rvert$	$\lvert\mathcal{I}\rvert$	$\lvert Dr\rvert$	net-RL	gen-RL	RL	net-RL	gen-RL	RL	net-RL	gen-RL	RL
23	1	3	9.29	8.34	**7.70**	14.63	11.75	**10.27**	16.01	13.73	**12.32**
	2	3	6.01	5.79	**5.40**	7.93	8.45	**6.93**	8.89	9.00	**8.34**
	3	2	5.48	5.28	**5.21**	7.02	7.58	**6.38**	8.33	8.11	**7.79**
50	1	3	14.97	13.96	**13.77**	20.35	19.36	**17.93**	22.60	20.05	**18.92**
	2	3	8.54	**8.21**	8.41	11.81	**10.81**	11.23	12.15	11.76	**11.96**
	3	2	7.22	6.90	**6.89**	9.58	9.41	**9.23**	10.33	9.89	**9.77**
100	1	3	23.16	22.98	**22.18**	**30.62**	32.33	30.67	**31.53**	32.30	36.67
	2	3	**12.91**	14.25	12.92	17.55	18.42	**17.54**	**17.21**	18.79	17.90
	3	2	10.23	**10.21**	**10.21**	13.39	13.73	**13.33**	13.69	**13.67**	14.94

TABLE IV: RL model vs. the heuristic optimization in terms of total time spent in the system and the percentages of winning instances, the average of 128 test instances are reported. In bold are the best results.

			Easy			Med			Hard	
			Mean		Win %	Mean		Win %	Mean	
$\lvert\mathcal{N}\rvert$	$\lvert\mathcal{I}\rvert$	$\lvert Dr\rvert$	Sim	RL	RL-Sim	Sim	RL	RL-Sim	Sim	RL
23	1	3	8.81	**7.70**	85.94%	12.39	**10.27**	94.53%	–	12.32
	2	3	5.72	**5.40**	65.63%	7.43	**6.93**	73.44%	–	8.34
	3	2	5.27	**5.21**	51.56%	6.39	**6.38**	48.44%	–	7.79
50	1	3	17.34	**13.77**	96.09%	24.59	**17.93**	100.00%	–	18.92
	2	3	9.19	**8.41**	74.22%	12.25	**11.23**	73.44%	–	11.96
	3	2	6.96	**6.89**	53.13%	9.25	**9.23**	50.00%	–	9.77
100	1	3	34.20	**22.18**	100.00%	45.97	**30.67**	100.00%	–	36.67
	2	3	16.11	**12.92**	100.00%	21.63	**17.54**	96.09%	–	17.90
	3	2	11.71	**10.21**	86.72%	15.63	**13.33**	92.97%	–	14.94

TABLE V: Computation time in seconds to derive solutions, the average of 128 test instances are reported.

			Easy		Medium		Hard	
$\lvert\mathcal{N}\rvert$	$\lvert\mathcal{I}\rvert$	$\lvert Dr\rvert$	Sim	RL	Sim	RL	Sim	RL
23	1	3	7.07	0.01	13.98	0.02	–	0.02
	2	3	1.61	0.04	3.31	0.04	–	0.09
	3	2	0.91	0.06	1.71	0.05	–	0.11
50	1	3	48.43	0.05	98.61	0.05	–	0.06
	2	3	11.62	0.21	23.47	0.16	–	0.25
	3	2	5.35	0.20	10.75	0.21	–	0.35
100	1	3	152.15	0.16	599.36	0.17	–	0.26
	2	3	66.13	0.44	118.68	0.44	–	0.55
	3	2	28.23	0.92	53.86	1.03	–	1.02



(a) $|\mathcal{N}| = 23$, $|Dr| = 3$, $|\mathcal{I}| = 1$

(b) $|\mathcal{N}| = 23$, $|Dr| = 3$, $|\mathcal{I}| = 2$

Fig. 7: Training rewards with and without distance as an input

The difference in deriving solutions between Sim and RL models increases up to 585 times in the case of a single shuttle routing in a network with 100 nodes for Easy instance difficulty level.

We also compare the effects of the number of drivers and difficulty levels on the trained models. In particular, we train models with a specific number of drivers on easy, medium and hard instances on a fixed network size and check these models' performances against the models with varying a number of drivers and difficulty levels. For example, in Tables VI and VII rows indicate the problems' configurations in testing and columns indicate the problems' configurations in training datasets. The cells corresponding to a row and column show the percentages of instances when a trained model outperformed the model specifically trained for a test dataset. As we observe for both single and multi-shuttle problems, models trained on specific difficulty levels tend to perform better on similar instances with a different number of drivers compared to on test models with the same number of drivers, but different difficulty levels.

The sample solution for a single-shuttle case, where 4 EVs in an urban network require charging is shown in Figure 8. A shuttle with 3 drivers leaves the depot and visits supplier nodes first followed by a charger node. By

TABLE VI: The number of drivers vs. difficulty levels, a single shuttle case.

Tested On \ Trained On	E, $dr = 2$	M, $dr = 2$	H, $dr = 2$	E, $dr = 3$	M, $dr = 3$	H, $dr = 3$
E, $dr = 2$	0	33.6	24.2	32.8	21.9	5.5
M, $dr = 2$	10.2	0	11.7	0	27.3	7.8
H, $dr = 2$	21.1	3.9	0	0	2.3	32.0
E, $dr = 3$	22.7	4.7	8.6	0	15.6	3.9
M, $dr = 3$	12.5	7.8	10.9	0	0	8.6
H, $dr = 3$	19.5	1.6	31.3	0	1.6	0

TABLE VII: The number of drivers vs. difficulty levels, a multiple-shuttle case.

Tested On \ Trained On	E, $dr = 2$	M, $dr = 2$	H, $dr = 2$	E, $dr = 3$	M, $dr = 3$	H, $dr = 3$
E, $dr = 2$	0	14.8	16.4	53.9	3.1	10.9
M, $dr = 2$	6.3	0	21.9	0	8.6	14.8
H, $dr = 2$	10.9	3.9	0	0	6.2	40.6
E, $dr = 3$	23.4	11.7	12.5	0	6.3	7.0
M, $dr = 3$	14.8	39.8	14.1	0	0	14.8
H, $dr = 3$	17.2	0.8	37.5	0	3.1	0

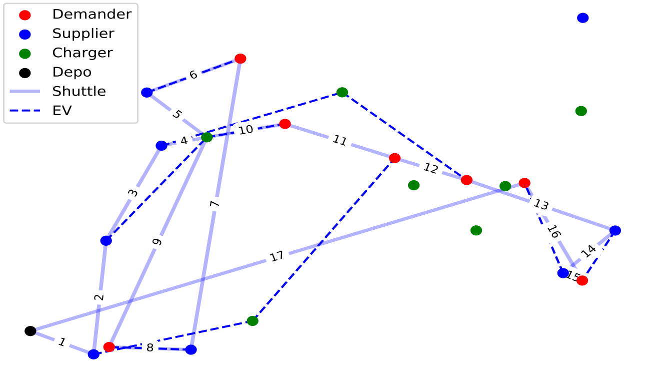


Fig. 8: Example solution for a single-shuttle case, $|\mathcal{N}| = 23$, $|Dr| = 3$ and $|\mathcal{I}| = 1$.

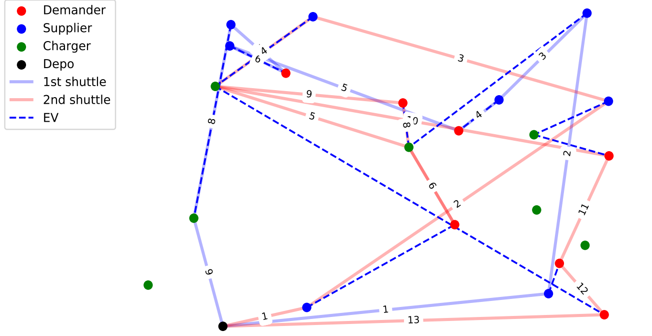


Fig. 9: Example solution for a multiple-shuttle cases, $|\mathcal{N}| = 23$, $|Dr| = 3$ and $|\mathcal{I}| = 2$.

interchangeably visiting nodes thorough the network, the shuttle returns back to the depot after picking up drivers from demander nodes. We can observe the versatility of the produced solutions looking at the charging stations. For instance, a driver dropped off at the first visited supplier node relocates the EV to a charging station, waits there until the EV is charged and then relocates it to a demander node. Only then the driver is picked up by a shuttle. In another example, the driver dropped off at the second visited supplier node is picked up immediately at a designated charging station by a shuttle. Similarly, Figure 9 represents the sample solution for the case with 2 shuttles. Each shuttle visits first supplier nodes until it runs out of drivers. Then each of them interchangeably visits charger, supplier and demander nodes and returns back to a depo. The flexibility of the produced solutions can be observed when a driver originally dropped at the second visited supplier node by the first shuttle is picked up at a charging station by the second shuttle.

VI. Conclusion

This study solves the shuttle routing problem for FFEVSS. We consider a static network, in which a group of drivers is deployed to relocate EVs from supplier nodes to charger and demander nodes. We propose a reinforcement learning approach to learn routing policies for single-shuttle and multi-shuttle cases. The proposed solution methods allow solving the new class of problem instances, while demonstrating improved results on instances solvable by existing methods in the literature. We also present several RL agents that generalize on various network structures or network sizes and we demonstrate that the RL agent specifically trained on a network produces superior results.

References

[1] D. Kypriadis, G. Pantziou, C. Konstantopoulos, and D. Gavalas, "Minimum walking static repositioning in free-floating electric car-sharing systems," in *2018 21st international conference on intelligent transportation systems (ITSC)*, pp. 1540–1545, IEEE, 2018.

[2] A. G. Santos, P. G. Cândido, A. F. Balardino, and W. Herbawi, "Vehicle relocation problem in free floating carsharing using multiple shuttles," in *2017 IEEE Congress on Evolutionary Computation (CEC)*, pp. 2544–2551, IEEE, 2017.

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

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

[5] I. Bello, H. Pham, Q. V. Le, M. Norouzi, and S. Bengio, "Neural combinatorial optimization with reinforcement learning," *arXiv preprint arXiv:1611.09940*, 2016.

[6] M. Nazari, A. Oroojlooy, L. Snyder, and M. Takác, "Reinforcement learning for solving the vehicle routing problem," in *Advances in Neural Information Processing Systems*, pp. 9839–9849, 2018.

[7] A. K. Kalakanti, S. Verma, T. Paul, and T. Yoshida, "Rl solver pro: Reinforcement learning for solving vehicle routing problem," in *2019 1st International Conference on Artificial Intelligence and Data Sciences (AiDAS)*, pp. 94–99, IEEE, 2019.

[8] J. James, W. Yu, and J. Gu, "Online vehicle routing with neural combinatorial optimization and deep reinforcement learning," *IEEE Transactions on Intelligent Transportation Systems*, vol. 20, no. 10, pp. 3806–3817, 2019.

[9] F. Schulte and S. Voß, "Decision support for environmental-friendly vehicle relocations in free-floating car sharing systems: The case of car2go," *Procedia CIRP*, vol. 30, pp. 275–280, 2015.

[10] S. Herrmann, F. Schulte, and S. Voß, "Increasing acceptance of free-floating car sharing systems using smart relocation strategies: a survey based study of car2go hamburg," in *International conference on computational logistics*, pp. 151–162, Springer, 2014.

[11] S. Weikl and K. Bogenberger, "Relocation strategies and algorithms for free-floating car sharing systems," *IEEE Intelligent Transportation Systems Magazine*, vol. 5, no. 4, pp. 100–111, 2013.

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

[13] M. Zhao, X. Li, J. Yin, J. Cui, L. Yang, and S. An, "An integrated framework for electric vehicle rebalancing and staff relocation in one-way carsharing systems: Model formulation and lagrangian relaxation-based solution approach," *Transportation Research Part B: Methodological*, vol. 117, pp. 542–572, 2018.

[14] L. He, G. Ma, W. Qi, and X. Wang, "Charging an electric vehicle-sharing fleet," *Manufacturing & Service Operations Management*, 2020.

[15] J. Shi, Y. Gao, W. Wang, N. Yu, and P. A. Ioannou, "Operating electric vehicle fleet for ride-hailing services with reinforcement learning," *IEEE Transactions on Intelligent Transportation Systems*, 2019.

[16] K. Lin, R. Zhao, Z. Xu, and J. Zhou, "Efficient large-scale fleet management via multi-agent deep reinforcement learning," in *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pp. 1774–1783, 2018.

[17] J. Wen, J. Zhao, and P. Jaillet, "Rebalancing shared mobility-on-demand systems: A reinforcement learning approach," in *2017 IEEE 20th International Conference on Intelligent Transportation Systems (ITSC)*, pp. 220–225, IEEE, 2017.

[18] N. Sadeghianpourhamami, J. Deleu, and C. Develder, "Achieving scalable model-free demand response in charging an electric vehicle fleet with reinforcement learning," in *Proceedings of the Ninth International Conference on Future Energy Systems*, pp. 411–413, 2018.

[19] I. Sutskever, O. Vinyals, and Q. V. Le, "Sequence to sequence learning with neural networks," in *Advances in neural information processing systems*, pp. 3104–3112, 2014.

[20] O. Vinyals, S. Bengio, and M. Kudlur, "Order matters: Sequence to sequence for sets," *arXiv preprint arXiv:1511.06391*, 2015.

[21] W. Kool, H. Van Hoof, and M. Welling, "Attention, learn to solve routing problems!," *arXiv preprint arXiv:1803.08475*, 2018.

[22] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," in *Advances in neural information processing systems*, pp. 5998–6008, 2017.

[23] J. Zhao, M. Mao, X. Zhao, and J. Zou, "A hybrid of deep reinforcement learning and local search for the vehicle routing problems," *IEEE Transactions on Intelligent Transportation Systems*, 2020.

[24] L. Buşoniu, R. Babuška, and B. De Schutter, "Multi-agent reinforcement learning: An overview," in *Innovations in multi-agent systems and applications-1*, pp. 183–221, Springer, 2010.

[25] R. Lowe, Y. Wu, A. Tamar, J. Harb, O. P. Abbeel, and I. Mordatch, "Multi-agent actor-critic for mixed cooperative-competitive environments," in *Advances in neural information processing systems*, pp. 6379–6390, 2017.

[26] J. K. Gupta, M. Egorov, and M. Kochenderfer, "Cooperative multi-agent control using deep reinforcement learning," in *International Conference on Autonomous Agents and Multiagent Systems*, pp. 66–83, Springer, 2017.

[27] J. N. Foerster, G. Farquhar, T. Afouras, N. Nardelli, and S. Whiteson, "Counterfactual multi-agent policy gradients," in *Thirty-second AAAI conference on artificial intelligence*, 2018.

[28] F. A. Oliehoek, C. Amato, *et al.*, *A concise introduction to decentralized POMDPs*, vol. 1. Springer, 2016.

[29] B. Perozzi, R. Al-Rfou, and S. Skiena, "Deepwalk: Online learning of social representations," in *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 701–710, 2014.

[30] R. J. Williams, "Simple statistical gradient-following algorithms for connectionist reinforcement learning," *Machine learning*, vol. 8, no. 3-4, pp. 229–256, 1992.