This article has been accepted for inclusion in a future issue of this journal. Content is final as presented, with the exception of pagination.



# Operating Electric Vehicle Fleet for Ride-Hailing Services With Reinforcement Learning

Jie Shi, *Student Member, IEEE*, Yuanqi Gao, *Student Member, IEEE*, Wei Wang, *Student Member, IEEE*, Nanpeng Yu, *Senior Member, IEEE*, and Petros A. Ioannou, *Fellow, IEEE*

***Abstract*— Providing ride-hailing services with electric vehicles can help reduce greenhouse gas emissions and solve the last mile problem. This paper develops a reinforcement learning based algorithm to operate a community owned electric vehicle fleet, which provides ride-hailing services to local residents. The goals of operating the electric vehicle fleet are to minimize customer waiting time, electricity cost, and operational costs of the vehicles. A novel framework characterized by decentralized learning and centralized decision making is proposed to solve the electric vehicle fleet dispatch problem. The decentralized learning process allows the individual vehicles to share their operating experiences and deep neural network model for state-value function estimation, which mitigates the curse of dimensionality of state and action domains. The centralized decision making framework converts the vehicle fleet coordination problem into a linear assignment problem, which has polynomial time complexity. Numerical study results show that the proposed approach outperforms the benchmark algorithms in terms of societal cost reduction.**

***Index Terms*— Assignment problem, electric vehicle, reinforcement learning, ride-hailing services.**

## I. Introduction

ELECTRIC vehicles (EVs) are gaining widespread adoption because of their low greenhouse gas (GHG) emissions and zero tailpipe pollution. EVs on average produce less than half the life-cycle GHG emissions of their internal combustion engine counterparts [1]. It is projected by the International Energy Agency (IEA) that the number of EVs will grow from 3 million in 2017 to 125 million by 2030 [2]. Meanwhile, ride-hailing platforms such as Uber and DiDi have experienced phenomenal growth in the past ten years. A total of 7.43 billion and 5 billion rides were completed on DiDi and Uber, respectively, in 2017. To help reduce their carbon footprint and vehicle operating costs, most of the major ride-hailing companies have launched electric car initiatives.

Operating a community or city owned EV fleet to provide ride-hailing services can be a great solution to delivering low cost and low emission mobility services to the local residents. The EV ride-hailing service could complement public transportation services to help bridge the 'last mile', when one's pickup location or final destination is too far from a public transit node.

The key difference between dispatching conventional gasoline vehicles and EVs lies in the relatively frequent recharging processes of EV batteries. Although the ranges of EVs have been growing rapidly in recent years, the recharging process still needs to be considered for the following three reasons. First, the majority of EVs on the market currently have much lower ranges than their gasoline counterparts. For example, the Nissan Leaf has a range of 151 to 226 miles. It would run out of power after only a few hours of continuous operation. Second, recharging an EV is still significantly slower than refueling a conventional gasoline vehicle. For example, charging a Nissan Leaf from empty to full would take around one hour even using a 50 kW fast charger. Third, even though the ranges of EVs will continue to grow in the future, the modeling of recharging process would still be important for many applications. For instance, the recharging process of an EV fleet can be coordinated with the smart grid control systems to provide frequency regulation services [3], which will bring additional benefits to both the ride-hailing service provider and the power grid.

In this paper, we develop an EV fleet operating algorithm to provide ride-hailing services, which minimizes the total customer waiting time, electricity consumption, and vehicle operational costs. Specifically, given a fleet of EVs with random initial locations and remaining battery levels, our task is to make sequential decisions to dispatch the EV fleet to serve an initially unknown set of customer trip requests within the operating time horizon. The information of each customer's trip request is revealed in real-time. The EVs stop where they are at the end of the operating horizon. This problem formulation belongs to the family of vehicle routing problems (VRPs) [4]. Specifically, it can be categorized as a dynamic VRP because the information of the customer trip requests is initially unknown and revealed over time during the operating process.

The unknown customer trip requests and additional complexity of recharge demands make it difficult to develop a model-based optimization algorithm to solve this dynamic VRP. In this work we propose a reinforcement learning based approach to solve the online EV fleet dispatch problem. The goal of our EV fleet operating algorithm is to minimize

Manuscript received March 26, 2019; revised August 9, 2019; accepted October 2, 2019. The Associate Editor for this article was H. B. Celikoglu. *(Corresponding author: Jie Shi.)*

J. Shi, Y. Gao, W. Wang, and N. Yu are with the Department of Electrical and Computer Engineering, University of California at Riverside, Riverside, CA 92501 USA (e-mail: jshi005@ucr.edu; nyu@ece.ucr.edu).

P. A. Ioannou is with the Department of Electrical Engineering, University of Southern California, Los Angeles, CA 90007 USA (e-mail: ioannou@usc.edu).

Digital Object Identifier 10.1109/TITS.2019.2947408

1524-9050 © 2019 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission.
See http://www.ieee.org/publications_standards/publications/rights/index.html for more information.

This article has been accepted for inclusion in a future issue of this journal. Content is final as presented, with the exception of pagination.



residents' waiting time, electricity cost, and vehicle operational costs. Note that this objective is very different from the goal of a typical ride-hailing service provider, which is maximization of gross merchandise volume (GMV) [5]. A decentralized learning and centralized decision making framework is proposed in this paper to implement the reinforcement learning algorithm. The decentralized learning component allows individual EVs to share their experiences and learned model to overcome the curse of dimensionality in the multi-agent reinforcement learning environment. The adoption of experience replay and target network ideas helps stabilize the reinforcement learning process. The centralized decision making component enables seamless coordination of EV fleet to avoid scheduling conflicts. The centralized decision making problem is converted into a linear assignment problem, which can be solved in polynomial time for online implementation.

The contributions of this paper are listed as follows:

- We develop a reinforcement learning based algorithm to provide ride-hailing services to communities with an EV fleet.
- We develop a scalable off-policy reinforcement learning framework with decentralized learning and centralized decision making processes, which has polynomial time complexity in real-time execution.
- We demonstrate that the proposed EV fleet dispatch algorithm outperforms the benchmark algorithms in terms of reducing societal costs for mobility service provision.

The rest of this paper is organized as follows. Section II briefly summarizes the related works. Section III describes the overall framework and the formulation of the reinforcement learning problem. Section IV presents the technical methods used in the decentralized learning and centralized decision making processes. Section V carries out numerical simulations to validate the effectiveness of our proposed method. The conclusions are stated in Section VI.

## II. Related Works

Our problem falls into the family of vehicle routing problems according to the generic definition given by [4]. The original VRP and its variants have been extensively investigated since the seminal work [6]. In this section, we first present an overview of the general vehicle routing problems. Then, a comprehensive review of the recent development on the electric vehicle routing problem (EVRP) is carried out. We close the literature review section by discussing a few papers, which adopted the reinforcement learning framework to solve similar problems.

### A. Overview of Vehicle Routing Problems

The family of vehicle routing problems have been studied for more than 50 years. The original VRP was introduced in 1959 to solve the dispatch schedule of a fleet of trucks for delivering gasoline from a single depot to different stations [6]. Numerous variants and extensions of the original VRP have been formulated and studied by researchers since then. Most of them belong to the combinatorial optimization problems that are notoriously difficult to solve. Meanwhile, the VRP family receives a lot of interests not only from the academic society but also from the industry. In general, the VRPs can be categorized into the following four groups based on the problem inputs and the optimization framework [4], [7].

- *Static and Deterministic*: All the inputs such as transportation requests and vehicle parameters are known and deterministic prior to the operating process. None of the inputs are defined as random variables. The routes are determined in advance and can not be re-optimized during the operating process.
- *Static and Stochastic*: Same as *Static and Deterministic* group except either all or parts of the inputs are stochastic with certain distributions.
- *Dynamic and Deterministic*: Either all or parts of the inputs are unknown prior to the operating process, which are revealed to the dispatcher during the execution. The routes can be re-optimized or evolve as a function of the inputs over time. In other words, the routes are not determined in advance.
- *Dynamic and Stochastic*: Same as *Dynamic and Deterministic* group except either all or parts of the inputs are stochastic with certain distributions.

The first two and the last two groups are called static VRPs and dynamic VRPs, respectively. Our problem belongs to the dynamic VRPs since the customer trip requests are not known in advance but received during the operating process. A large number of approaches have been proposed to solve different VRPs. Most of them formulate vehicle routing as mixed integer programming problems, which are then solved through heuristics. Refer to [4] for a comprehensive discussion of the formulations of different VRPs and solution methods.

### B. Electric Vehicle Routing Problems

With the rapid development and growing popularity of electric vehicles, the EV routing problem has attracted strong interests from researchers. In this study, we distinguish EVRP from traditional VRP based on two criteria. First, the vehicle fleet includes EVs. Second, the EVs can be recharged at the charging stations during the operating process. As a pioneering work, [8] first introduced dummy vertices[1] to allow refueling the vehicles at the charging stations along their routes. Following this idea, [10] extended the traditional VRP with customer time windows to its EV version (EVRPTW). Both of these two works assumed full recharges in their formulations. However, fully recharging EVs each time might not be the optimal solution in many cases. To mitigate this constraint, [11] first introduced partial recharge in the EVRP formulation. Unfortunately, the introduction of partial recharge can complicate the problem and make it more difficult to solve. Later, a number of other extensions and modifications of the original EVRP were proposed. For instance, [12] considered the effect of vehicle load on electricity consumption. Reference [13] extended EVRPTW by using a mixed fleet of electric and conventional vehicles as well as a more realistic energy consumption model. This mixed fleet routing problem

[1]Dummy vertices was originally introduced by [9] to model stops at intermediate depots in a routing problem with traditional vehicles.

This article has been accepted for inclusion in a future issue of this journal. Content is final as presented, with the exception of pagination.



was further examined by [14] considering different vehicle capacities, battery sizes, and acquisition costs. Reference [15] introduced additional constraints on the EV speed for different time intervals based on the corresponding traffic flow forecasts. Reference [16] first proposed to capture the nonlinear behavior of the charging process using a piecewise linear approximation in the EVRPs. Inspired by this idea, [17] proposed an improved formulation of EVRP with nonlinear charging functions, in which an arc-based tracking of the time and the state of charge is proposed. Reference [18] incorporated the queuing time of EVs at the charging stations into the EVRP formulation by enforcing the constraint of limited capacities of the charging stations.

All the EVRPs mentioned above share the following two assumptions. First, all the vehicles start and return to the same depot. Second, the transportation request information is fully known in the beginning of the operating process. These two strong assumptions hinder the above methods from being directly applied in the ride-hailing applications. Recently, [19] and [20] removed the first assumption by extending the EVRP formulation to a multiple depot version. Both of these two works and our problem fall into the pickup-and-delivery problems for passenger transportation. Nevertheless, [19] and [20] still belong to the static VRPs due to the second assumption. Thus, they are significantly different from our dynamic VRP.

Several papers have studied the dynamic electric vehicle routing problem (DEVRP), which has a similar formulation to our paper. In these papers, the transportation requests are initially unknown and the initial EV locations are picked arbitrarily. These studies are often called EV operating or dispatch in the literature. To the best of our knowledge, [21] is one of the earliest works on operating an EV fleet to serve real-time customer trip requests. It was designed to minimize the total customer waiting and traveling time cost. Every new customer's trip request is assigned to an EV that yields the least incremental time cost. EVs are sent to recharge once their remaining battery levels fall below some predefined threshold. References [22] and [23] investigated the operations of a shared autonomous EV fleet. Both of them use simple greedy algorithms that assign new customer trip requests to the EVs that are either closest [22] or lead to the least waiting time [23]. More recently, [24] solved a DEVRP in the context of ride-hailing using approximate dynamic programming. In [24], the operating region is divided into small square zones and every zone is assumed to have one charging station.

### C. Related Reinforcement Learning Works

Recently, researchers started applying reinforcement learning algorithms to solve VRPs. See [25] for an example of a single vehicle routing problem. Reinforcement learning based approaches are also taken to either dispatch or reposition vehicles for ride-hailing services in real-time [5], [26], [27]. [5] is closely related to our work. To overcome the curse of dimensionality, a decentralized learning and centralized decision making approach was proposed. However, our work differs from [5] in the following aspects. First, [5] focuses on the traditional vehicle operating problem instead of the EVs. Thus, they do not need to model the charging process. Second, we design an entirely different reinforcement learning algorithm. [5] uses an on-policy RL algorithm with the value function represented by a table. In this work, we construct an off-policy RL algorithm with the value function approximated by a neural network. Third, the problem formulations have different objectives. [5] tries to maximize the gross merchandise volume (GMV) while the goal of this work is to minimize the customer waiting time and EV operational costs.

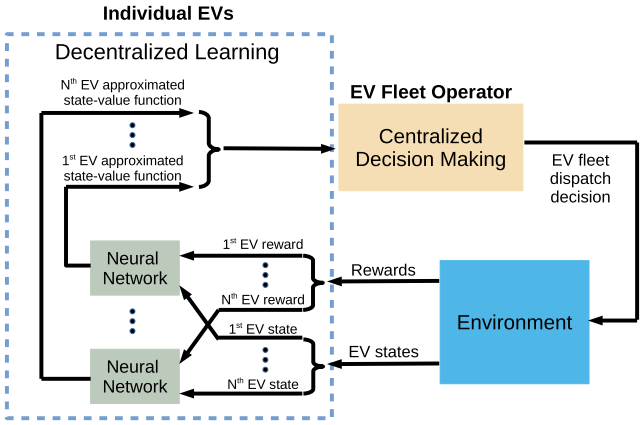



Fig. 1. Overall reinforcement learning framework.

Reinforcement learning algorithms can be classified into two groups, on-policy learning and off-policy learning. In the on-policy approach, the learned policy (target policy) and the policy that generates behaviors (behavior policy) are the same. On the other hand, the target policy and behavior policy in the off-policy methods are different. In other words, the learning is from data "off" the target policy. The off-policy learning approach is more powerful and general. It includes on-policy approach as the special case in which the target and behavior policies are the same. More importantly, the off-policy methods can exploit the historic data or data generated from other conventional non-learning methods.

## III. Overall Framework and Problem Formulation

In this section, we present the overall framework of our proposed EV fleet operating algorithm for ride-hailing services. The operating problem for a fleet of EVs to provide ride-hailing services is formulated as a Markov decision process (MDP). We propose a reinforcement learning framework with decentralized learning and centralized decision making to solve this MDP problem. The overall reinforcement learning framework is illustrated in Fig. 1. In the learning process, we treat EVs as individual agents with shared state-value function. The parameters of the common state-value function approximator are trained and updated based on the collection of individual EVs' experiences of interacting with the environment. In the decision making process, the EV fleet operating problem is solved in a centralized manner by leveraging state-value function estimates from the learning process.

This article has been accepted for inclusion in a future issue of this journal. Content is final as presented, with the exception of pagination.



Before formulating the EV fleet operating for ride-hailing service problem as an MDP, we first introduce some preliminary information about MDP and reinforcement learning. Within an MDP, we call the learner or decision maker an agent. The agent interacts with the environment it lives in at each of a sequence of discrete time steps, $t = 0, 1, 2, \cdots$. At each time step $t$, the agent senses the environment's state $S_t \in \mathcal{S}$ and takes a control action $A_t \in \mathcal{A}(s)$. When reaching the next time step, the agent receives a reward $R_{t+1} \in \mathcal{R} \subset \mathbb{R}$ based on previous state-action pair and the current state. Then the agent reaches a new state $S_{t+1}$. The numerical reward and the new state depend on the preceding state and control action according to the following transition model [28]:

$$p(s', r|s, a) \doteq \Pr\{S_{t+1} = s', R_{t+1} = r|S_t = s, A_t = a\} \quad (1)$$

where $s, s' \in \mathcal{S}$ are the current and the next states. $r \in \mathcal{R}$ is the reward, and $a \in \mathcal{A}(s)$ is the action taken by the agent. The sequence of the interactions between the agent and the environment from the starting state until the terminal state is called an episode.

At time step $t$, the agent tries to achieve the maximum discounted return $G_t = \sum_{k=t+1}^{T} \gamma^{k-t-1} R_k$, where $\gamma$ is the discount factor. $T$ is the time step when the terminal state of an episode is reached. When interacting with the environment, the agent follows a policy $\pi(s) = p(a|s)$, which maps a given state $s$ to a probability distribution of taking action $a$. In order to evaluate how good it is for the agent to be in a given state or take an action starting from a state, we define the state-value function $v_\pi(s)$ and action-value function $q_\pi(s, a)$ under a policy $\pi$ as:

$$v_\pi(s) = \mathbb{E}_\pi[G_t|S_t = s] \quad (2)$$

$$q_\pi(s, a) = \mathbb{E}_\pi[G_t|S_t = s, A_t = a] \quad (3)$$

A policy is optimal if its expected return is greater than or equal to that of any other policy for all states. The corresponding action-value function of an optimal policy is called the optimal action-value function. The goal of the agent is to find an optimal policy for the MDP problem.

### A. Framing the Problem of Operating an EV Fleet for Ride-Hailing Services as MDP

Assume that a community operates an EV fleet for ride-hailing services to meet its residents' trip requests. The goals of the EV fleet operator are to minimize customers' waiting time, electricity costs from charging, and vehicle operational costs. It is assumed that EVs only receive dispatch signals every $T^I$ (time interval). An EV's status is *available* if it is idle, being charged, or on its way to the charging station. Otherwise, the EV's status will be *in-service*. It is assumed that all the EVs are *available* at the initial time step. The status of an EV changes from *available* to *in-service* once a customer's trip request is assigned to it. At each time step, an EV can be assigned to at most one more customer trip request. Once the customer trip assignment is made, it can not be changed. All EVs are required to serve the trip requests assigned to them according to the 'first-come-first-serve' rule.

Next we will define the state domain, action domain, and reward function of the MDP describing the problem of operating an EV fleet for ride-hailing services.

*1) State Domain:* Let $\boldsymbol{s}^{EF} \in \mathcal{S}^{EF}$ denote the state vector of the entire EV fleet. $\boldsymbol{s}^{EF} = [\boldsymbol{s}_1, \cdots, \boldsymbol{s}_{N_{EV}}]$ is a tuple of state vectors of individual EVs, where $N_{EV}$ represents the total number of EVs. $\boldsymbol{s}_i \in \mathcal{S}^{IE}$ denotes the state vector of the $i$th EV, which consists of three self states and one global state:

- $E_i$: Remaining battery level of the $i$th EV when it becomes *available*.
- $L_i$: Location of the $i$th EV when it becomes *available*.
- $T_i^A$: Time length before the $i$th EV becomes *available*.
- $t$: Global state for time.

Note that in real-world applications, we can use high quality estimates for $E_i$ and $T_i^A$ as part of the state vector.

*2) Action Domain:* Let $\boldsymbol{a}^{EF} \in \mathcal{A}^{EF}(\boldsymbol{s}^{EF})$ denote the action vector of the entire EV fleet. $\boldsymbol{a}^{EF} = [a_1, \cdots, a_{N_{EV}}]$ consists of actions of individual EVs, where $a_i \in \mathcal{A}^{IE}(s_i)$ is the action of $i$th EV. Each EV can take one of the following three actions:

- $a_i = Pass$: If the $i$th EV is idle, then $Pass$ action keeps it idle during the next time interval. If the $i$th EV is *in-service*, then $Pass$ action has no effect on its existing assignment.
- $a_i = Charge$: If the $i$th EV is in a charging station, then it starts charging or keeps being charged. If the $i$th EV is not in a charging station, then it moves toward the nearest charging station during the next time interval.
- $a_i = Assign\ c$: This action means that a new customer trip request $c \in \mathcal{C}(t)$ is assigned to the $i$th EV, where $\mathcal{C}(t)$ is the set of customer trip requests to be processed at the current time step ($\mathcal{C}(t)$ has a maximum capacity of $N_C$). Note that there is a set of assignment actions to choose from, each of which corresponds to a different customer trip request $c \in \mathcal{C}(t)$.

An action is classified as a *feasible* action if the EV has enough electricity left in the battery to return to the nearest charging station after serving all its assigned customer trip requests. Otherwise, the action is classified as *infeasible*.

*3) Reward Function:* Let $r^{EF}(\boldsymbol{s}^{EF}, \boldsymbol{a}^{EF}) \in \mathbb{R}$ denote the immediate reward received by the entire EV fleet when taking action $\boldsymbol{a}^{EF}$ in state $\boldsymbol{s}^{EF}$. Let $r(\boldsymbol{s}_i, a_i) \in \mathbb{R}$ denote the immediate reward received by the $i$th EV when taking action $a_i$ in state $\boldsymbol{s}_i$. The reward received by the entire EV fleet equals the sum of rewards received by the individual EVs $r^{EF}(\boldsymbol{s}^{EF}, \boldsymbol{a}^{EF}) = \sum_{i=1}^{N_{EV}} r(\boldsymbol{s}_i, a_i)$. The reward functions of individual EVs are designed as follows:

- The reward for the pass action is:

$$r(\boldsymbol{s}_i, a_i = Pass) = 0 \quad (4)$$

- The reward for the charge action is:

$$r(\boldsymbol{s}_i, a_i = Charge) = \begin{cases} \epsilon_C, & \text{if the } i\text{th EV is in} \\ & \text{a charging station} \\ -w_C, & \text{otherwise} \end{cases} \quad (5)$$

where $\epsilon_C$ is a small positive number and $w_C$ is a positive tunable parameter.

This article has been accepted for inclusion in a future issue of this journal. Content is final as presented, with the exception of pagination.



- The reward for assignment action is:

$$r(\boldsymbol{s}_i, a_i = Assign\ c) = r_S - d_i^c \cdot w_S \tag{6}$$

where $r_S$ and $w_S$ are positive tunable parameters. $d_i^c$ is the distance between $L_i$ and the pickup location of customer trip request $c$. Note that the traveling costs (including electricity consumption costs and operational costs) are in proportion to $d_i^c$. $r_S$ corresponds to the incentive for an EV to serve a customer's trip request. The design of the reward function strikes a balance between incentives and costs.

The design philosophy for the reward function is explained below. Recall that the goals of the EV fleet scheduler are to minimize customer waiting time, electricity costs, and EV operational costs. The $Pass$ action does not contribute to achieving any of the three goals. Thus, the reward for the $Pass$ action should be 0. An EV with higher battery charge level is capable of serving customer trip requests for a longer ride. Hence, we encourage an EV to select $Charge$ rather than $Pass$ when it is at a charging station by offering a small positive reward. If an EV is not at a charging station, then we use a negative reward $-w_C$ to discourage the EV to travel to a charging station that is far away. In this way, we could reduce the travel cost related to the charging activities, which includes the electricity cost. Regarding the reward of the assignment action, the second term $d_i^c \cdot w_S$ is a penalty for EVs to pick up faraway customers. In this way, we implicitly reduce the electricity consumption and operational costs of the EV fleet.

## IV. Technical Methods

### A. Decentralized Learning Process

The learning process aims to acquire a high quality estimate of the state-value function for the EV fleet. It is assumed that we have a homogeneous EV fleet, where each EV has the same fuel economy rating, the same battery size, and the same travel speed. To address the curse of dimensionality of dealing with the entire EV fleet's state domain, we adopt a decentralized learning framework with the following two assumptions:

*Assumption 1:* We assume that state-value function of the entire EV fleet $v_\pi(\boldsymbol{s}^{EF})$ equals the summation of the individual EVs' state-value functions, i.e., $v_\pi(\boldsymbol{s}^{EF}) = \sum_{i=1}^{N_{EV}} v_{\pi,i}(\boldsymbol{s}_i)$, where $v_{\pi,i}(\boldsymbol{s}_i)$ denotes the state-value function of the $i$th EV under policy $\pi$.

*Assumption 2:* We assume all the individual EVs have the same state-value function $v_\pi^{EV}(\boldsymbol{s})$, i.e., $v_{\pi,i}(\boldsymbol{s}_i) = v_\pi^{EV}(\boldsymbol{s}_i), \forall i$. In other words, the state-value functions of all EVs share the same form.

Note that these two simplifying assumptions have been shown to be effective in multi-agent cooperative environments [5], [27], [29].

*1) State-Value Function Approximation With Neural Network:* The size of the state space $[E_i, L_i, T_i^A, t]$ of the $ith$ EV grows quickly when the resolutions of the map's grid, time, and remaining battery level increase. Hence, it is infeasible to represent the value functions through tables [28]. Recently, neural networks have been widely employed to approximate the value functions to overcome the curse of dimensionality. This idea has gained successes across various reinforcement learning tasks [30]–[32]. In this study, we use a parameterized deep feed-forward neural network (FNN) to approximate the shared state-value function $v_\pi^{EV}(\boldsymbol{s})$ of individual EVs. A deep FNN is composed of three parts: the input layer, the hidden layers, and the output layer. Different layers are interconnected in a feed-forward way. In this study, the input layer carries the state vector $\boldsymbol{s}_i$ and the output layer represents the value of the approximated state-value function.

The feed-forward neural network approximator $v_{NN}(\boldsymbol{s}, \boldsymbol{\theta})$ is trained by using the gradient descent based method to minimize a loss function. In this work, the loss function is defined as the mean squared error (MSE) between the approximated value function and the target value function:

$$loss = \frac{1}{N_s} \sum_{\boldsymbol{s} \in \mathcal{S}^s} \left( \hat{v}_\pi^{EV}(\boldsymbol{s}) - v_{NN}(\boldsymbol{s}, \boldsymbol{\theta}) \right)^2 \tag{7}$$

where $\mathcal{S}^s$ is a set of sample states. $N_s$ is its cardinality. Since the true state-value function $v_\pi^{EV}(\boldsymbol{s})$ is unknown, we use an estimate of it, i.e., $\hat{v}_\pi^{EV}(\boldsymbol{s})$, as the training target. Specifically, we use temporal-difference (TD) prediction to calculate $\hat{v}_\pi^{EV}(\boldsymbol{s})$ [33] as follows:

$$\hat{v}_\pi^{EV}(\boldsymbol{s}) = \begin{cases} r(\boldsymbol{s}, a), & \boldsymbol{s}' \text{ is terminal} \\ r(\boldsymbol{s}, a) + \gamma\, v_{NN}(\boldsymbol{s}', \boldsymbol{\theta}), & \boldsymbol{s}' \text{ is non-terminal} \end{cases} \tag{8}$$

where $\boldsymbol{s}'$ is the next state of the corresponding EV given the current state-action pair $(\boldsymbol{s}, a)$. This particular approach is called the TD(0) method since the estimate is updated immediately after state transition from $\boldsymbol{s}$ to $\boldsymbol{s}'$. TD(0) is widely used in deep reinforcement learning frameworks for its simplicity and efficiency. It is worth noting that TD(0) prediction is a biased estimate and the corresponding gradient includes only part of the true gradient. Nevertheless, using TD(0) often makes the learning process faster, and more importantly, allows continual and online learning.

*2) Stabilize Reinforcement Learning With Experience Replay and Target Network:* It has been shown that reinforcement learning tends to be unstable when a deep neural network is used to approximate the state-value function [30]. To stabilize the reinforcement learning process, we adopt two innovative mechanisms called experience replay and target network, which were originally proposed in [34] and [30], respectively. The experience replay mechanism stabilizes the learning process by removing correlations in the learner's past *experiences* and smoothing over variations of the data distribution. The basic idea of experience replay is to keep a fixed-sized memory that stores the historic *experiences* and sample the training dataset from it for each training step. More precisely, an *experience* is defined as a 4-tuple made up of the current state, the action, the reward, and the next state $(\boldsymbol{s}, a, r, \boldsymbol{s}')$. The agent's most recent $N_D$ *experiences* are stored in a memory. At each training step, we use samples of *experiences* drawn uniformly from the memory to update the parameters of the neural network approximating the state-value function.

To further stabilize the online reinforcement learning process, we use two separate neural networks, the evaluation

This article has been accepted for inclusion in a future issue of this journal. Content is final as presented, with the exception of pagination.



network $v_{NN}(\boldsymbol{s}, \boldsymbol{\theta}^E)$ and the target network $v_{NN}(\boldsymbol{s}, \boldsymbol{\theta}^T)$. The loss function (7) and the estimate of training target for state-value function (8) are reformulated with these two neural networks as follows:

$$loss = \frac{1}{N_s} \sum_{\boldsymbol{s} \in \mathcal{S}^s} \left( \hat{v}_\pi^{EV}(\boldsymbol{s}) - v_{NN}(\boldsymbol{s}, \boldsymbol{\theta}^E) \right)^2 \tag{9}$$

$$\hat{v}_\pi^{EV}(\boldsymbol{s}) = \begin{cases} r(\boldsymbol{s}, a), & \boldsymbol{s}' \text{ is terminal} \\ r(\boldsymbol{s}, a) + \gamma\, v_{NN}(\boldsymbol{s}', \boldsymbol{\theta}^T), & \boldsymbol{s}' \text{ is non-terminal} \end{cases} \tag{10}$$

The evaluation network and the target network share the same structure but have different parameter update strategies. The evaluation network's parameters are updated during each training iteration while the parameters of the target network are only periodically updated. Specifically, the target network's parameters are replaced by the evaluation network's parameters every $C$ steps. In other words, $v_{NN}(\boldsymbol{s}, \boldsymbol{\theta}^T)$ is a clone of $v_{NN}(\boldsymbol{s}, \boldsymbol{\theta}^E)$ with less frequent parameter updates. In this way, the correlations between the evaluation and target are reduced, thereby making the training process more stable.

### *B. Centralized Decision Making Process*

The goal of the decision making process is to find the optimal EV fleet dispatch policy $\pi^*$, which achieves the maximum expected reward among all feasible policies for all states. The state-value function of the optimal policy is called the optimal state-value function $v_*$, which is defined as $v_* \doteq \max_\pi v_\pi(s)$, for all $s \in \mathcal{S}$. Similarly, the action-value function of the optimal policy is called the optimal action-value function $q_*$, which is defined as $q_* \doteq \max_\pi q_\pi(s, a)$, for all $s \in \mathcal{S}$ and $a \in \mathcal{A}$. For a traditional reinforcement learning task, once the optimal action-value function is found, then the action $a$ to be taken in a given state $s$ is the one that yields the highest optimal action-value $q_*(s, a)$. However, in the EV fleet operating problem for ride-hailing services, the following fleet-wide constraint needs to be satisfied at all times. Any customer trip request can only be assigned to at most one EV. Therefore, the dispatch of individual EVs should be coordinated in a centralized decision making process.

There are two technical obstacles to be addressed in the centralized decision making process. First, how do we approximate the optimal action-value and state-value functions for the EV fleet? Second, how do we determine the optimal dispatch for the entire EV fleet considering the operating constraints? These two problems are dealt with separately below.

*1) Approximation of the Optimal Action-Value and State-Value Functions for the EV Fleet:* Let $v_*(\boldsymbol{s}^{EF})$ and $q_*(\boldsymbol{s}^{EF}, \boldsymbol{a}^{EF})$ denote the optimal state-value function and action-value function of the entire EV fleet, respectively. Note that the optimal action-value function gives the expected return for taking an action in a state and following an optimal policy thereafter. Thus, the optimal action-value function can be derived in terms of the optimal state-value function as [28]:

$$\begin{aligned} &q_*(\boldsymbol{s}^{EF}, \boldsymbol{a}^{EF}) \\ &\quad = \mathbb{E}[R_{t+1}^{EF} + \gamma\, v_*(S_{t+1}) | S_t = \boldsymbol{s}^{EF}, A_t = \boldsymbol{a}^{EF}] \\ &\quad = \sum_{\boldsymbol{s}^{EF'} \in \mathcal{S}^{EF}} p(\boldsymbol{s}^{EF'} | \boldsymbol{s}^{EF}, \quad \boldsymbol{a}^{EF})(r^{EF} + \gamma\, v_*(\boldsymbol{s}^{EF'})) \end{aligned} \tag{11}$$

where $\boldsymbol{s}^{EF'}$ is the next state vector of the entire EV fleet. In this particular MDP formulation, the next state $\boldsymbol{s}^{EF'}$ and reward $r^{EF}$ are assumed to be determined once the current state-action pair $(\boldsymbol{s}^{EF}, \boldsymbol{a}^{EF})$ is given. Therefore, the above equation can be further reduced to

$$q_*(\boldsymbol{s}^{EF}, \boldsymbol{a}^{EF}) = r^{EF}(\boldsymbol{s}^{EF}, \boldsymbol{a}^{EF}) + \gamma\, v_*(\boldsymbol{s}^{EF'}) \tag{12}$$

With *Assumption 1* and *2*, the optimal state-value function can be estimated as the summation of individual EVs' state-value function approximators:

$$\hat{v}_*(\boldsymbol{s}^{EF'}) = \sum_{i=1}^{N_{EV}} v_{NN}(\boldsymbol{s}_i', \boldsymbol{\theta}^E) \tag{13}$$

where $\boldsymbol{s}_i'$ is the next state of $i$th EV. Let $\hat{q}_*(\boldsymbol{s}^{EF}, \boldsymbol{a}^{EF})$ denote the approximator of the optimal action-value function of the EV fleet. By substituting (13) into (12), we can express $\hat{q}_*(\boldsymbol{s}^{EF}, \boldsymbol{a}^{EF})$ as a function of individual EVs' rewards and state-value function approximators:

$$\hat{q}_*(\boldsymbol{s}^{EF}, \boldsymbol{a}^{EF}) = \sum_{i=1}^{N_{EV}} r(\boldsymbol{s}_i, a_i) + \gamma \sum_{i=1}^{N_{EV}} v_{NN}(\boldsymbol{s}_i', \boldsymbol{\theta}^E) \tag{14}$$

Next, we explain how to find the optimal EV fleet dispatch, which maximizes $\hat{q}_*(\boldsymbol{s}^{EF}, \boldsymbol{a}^{EF})$ while satisfying the operational constraints.

*2) EV Fleet Dispatch Problem:* The problem of finding the optimal EV fleet dispatch with the maximum action-value function can be converted into a linear assignment problem. In other words, solving $\text{argmax}_{\boldsymbol{a}^{EF}} \hat{q}_*(\boldsymbol{s}^{EF}, \boldsymbol{a}^{EF})$ is equivalent to finding a maximum weight matching between EVs and actions in a weighted bipartite graph.

The formulation of the linear assignment problem is given as follows. First, let $\mathcal{M} = \{m_1, \cdots, m_{N_{EV}}\}$ denote the set of vertices that represent individual EVs. Let $\mathcal{N} = \{n_1, \cdots, n_{N_A}\}$ denote the set of vertices that represent different actions, where $N_A = 2 \times N_{EV} + |\mathcal{C}(t)|$. $\mathcal{N}$ includes $N_{EV}$ *Pass* actions, $N_{EV}$ *Charge* actions, and $|\mathcal{C}(t)|$ *Assign c* actions. We need to reserve $N_{EV}$ vertices for both *Pass* and *Charge* actions because each of them might be taken by all the EVs simultaneously. Next, we define $\{x_{ij} | i = 1, \cdots, N_{EV}; j = 1, \cdots, N_A\}$ as the decision variables, where $x_{ij}$ is a binary variable that equals 1 if $m_i$ and $n_j$ is connected, 0 otherwise. Connecting $m_i$ and $n_j$ means the $i$th EV takes the action corresponding to $n_j$. Define $b(i, j)$ as the weight of the edge between $m_i$ and $n_j$. It represents the contribution of the $i$th EV to the objective function (14) by taking action $n_j$. The weights between EVs and their *infeasible* actions should be sufficiently small such that no connections would exist between them. Formally, $b(i, j)$ is calculated as:

$$b(i, j) = \begin{cases} r(\boldsymbol{s}_i, a_i = n_j) + \gamma\, v_{NN}(\boldsymbol{s}_i', \boldsymbol{\theta}^E), & n_j \text{ is } \textit{feasible} \\ -W, & \text{otherwise} \end{cases} \tag{15}$$

where $W$ is a positive number that is sufficiently large.

Now, the problem of finding the optimal EV fleet dispatch schedule with the maximum action-value function can be

This article has been accepted for inclusion in a future issue of this journal. Content is final as presented, with the exception of pagination.



formulated as the following optimization problem:

$$\max_{x_{ij}} \sum_{i=1}^{N_{EV}} \sum_{j=1}^{N_A} b(i,j) \cdot x_{ij} \tag{16}$$

$$\text{subject to: } \sum_{i=1}^{N_{EV}} x_{ij} \leq 1, \quad \forall j = 1, \cdots, N_A \tag{17}$$

$$\sum_{j=1}^{N_A} x_{ij} = 1, \quad \forall i = 1, \cdots, N_{EV} \tag{18}$$

$$x_{ij} \in \{0,1\}, \quad \forall i = 1, \cdots, N_{EV} \quad \forall j = 1, \cdots, N_A \tag{19}$$

The above linear assignment problem can be solved in polynomial time. In this work, we solve the linear assignment problem with an adapted Munkres algorithm [35], which has a time complexity of $O(n^3)$.

To improve convergence of the reinforcement learning algorithm, we need to ensure that all actions are selected infinitely often. This can be implemented via the $\epsilon$-$greedy$ policy during the online training session. In other words, with probability $1-\epsilon$, the EV fleet's dispatch will be determined by solving the optimization problem (16)-(19), but with probability $\epsilon$, each EV will be assigned a *feasible* action at random. The initial value of $\epsilon$ is 1 and then gradually decreases to 0.1 at the speed of $\Delta\epsilon$ per training step.

### C. Summary of the Overall Algorithm

The decentralized learning and centralized decision making processes are discussed in the previous two subsections. During the online training process, the steps for solving the problem of operating an EV fleet for ride-hailing services are summarized and shown in Algorithm 1. During the testing session, we can use the same algorithm by removing the neural network training and the $\epsilon$-$greedy$ components. Note that graphical processing units (GPUs) can be used to speed up the process of solving the linear assignment problem and the training of deep neural networks.

Note that the proposed reinforcement learning algorithm is an off-policy algorithm. The target policy being learned is the optimal policy $\pi_*$, while the behavior policy that generates real actions is the $\epsilon$-$greedy$ policy $\pi_\epsilon$.

**Algorithm 1** Complete Algorithm for the Training Process

Initialize $\epsilon = 1$;
Initialize replay memory $D$ to capacity $N_D$;
Initialize evaluation neural network $v_{NN}(\boldsymbol{s}, \boldsymbol{\theta}^E)$ with random weights $\boldsymbol{\theta}^E$;
Initialize target neural network $v_{NN}(\boldsymbol{s}, \boldsymbol{\theta}^T)$ with weights $\boldsymbol{\theta}^T = \boldsymbol{\theta}^E$;
**for** *episode=1:N* **do**
  Initialize the EV fleet's state;
  **for** *t=1:T* **do**
    Update the list of customer trip requests $\mathcal{C}(t)$;
    **for** *EV* $i = 1 : N_{EV}$ **do**
      Calculate $r_i(\boldsymbol{s}_i, a_i)$ and $v_{NN}(\boldsymbol{s}_i', \boldsymbol{\theta}^E)$ for each feasible action $a_i \in \mathcal{A}^{IE}(\boldsymbol{s}_i)$;
    Calculate the EV fleet dispatch solution $\boldsymbol{a}_{sol}^{EF}$ by solving optimization problem (16)-(19);
    Draw a sample $p$ from uniform distribution $U(0,1)$;
    **if** $p \leq \epsilon$ **then**
      Draw a feasible sample action $\boldsymbol{a}^{EF}$ at random and use it to dispatch the EV fleet;
    **else**
      Use $\boldsymbol{a}_{sol}^{EF}$ to dispatch individual EVs of the fleet;
    **for** *EV* $i = 1 : N_{EV}$ **do**
      Calculate $v_{NN}(\boldsymbol{s}_i', \boldsymbol{\theta}^T)$ where $\boldsymbol{s}_i'$ is the next state given state-action pair $(\boldsymbol{s}_i, \boldsymbol{a}_{sol}^{EF}[i])$;
    Update replay memory $D$ with transitions $\{(\boldsymbol{s}_i, r(\boldsymbol{s}_i, \boldsymbol{a}_{sol}^{EF}[i]), \boldsymbol{a}_{sol}^{EF}[i], \boldsymbol{s}_i')\}$ where $i = 1, \cdots, N_{EV}$;
    Sample a minibatch of *experiences* uniformly at random from $D$;
    Update $\boldsymbol{\theta}^E$ by using gradient descent method with the sampled minibatch of *experiences*;
    **if** $\epsilon > 0.1$ **then**
      $\epsilon = \epsilon - \Delta\epsilon$;
    Reset $\boldsymbol{\theta}^T = \boldsymbol{\theta}^E$ every $C$ time steps;

## V. Numerical Studies

Numerical studies with three test cases are carried out to validate the performance of the proposed reinforcement learning based EV fleet operating algorithm for ride-hailing services. The first test case represents a single region/city where the majority of the customer trip requests come from the center of the region/city (downtown area). The second test case represents a scenario where people commute between two regions on a daily basis. The third test case is a small-scale single region case used to evaluate the optimality of the proposed framework.

The proposed reinforcement learning algorithm will be compared with two benchmark algorithms. The first one is the greedy algorithm adapted from [22] and [23]. The second one is a state-of-the-art optimization-based algorithm proposed by [36], which performs well in solving a typical dynamic VRP for meal delivery applications. We call it the optimization-based algorithm in the numerical studies section. The greedy algorithm can be implemented via the following three steps:

- Step 1: Select the first $N_C$ unassigned customer trip requests sorted by the request time.
- Step 2: Process the customers trip requests on a first come first serve basis. Assign customer trip requests to the EVs that result in the least waiting time.[2]
- Step 3: Send the EVs to charge if they do not have assigned customer trip requests.

[2]The trip requests can only be assigned to EVs that have enough energy to return to the charging station after service.

This article has been accepted for inclusion in a future issue of this journal. Content is final as presented, with the exception of pagination.



The optimization-based algorithm tackles the dynamic VRP with a rolling horizon matching-based framework that solves a linear assignment problem every few time intervals. The weights in the assignment problem are designed to balance the "throughput" and the customer waiting time. It is similar to our centralized decision making process without considering future rewards. Specifically, the weights are adapted as follows to make this approach compatible with our problem.

- For the pass action:

$$b(i, j) = 0 \tag{20}$$

- For the assignment action:

$$b(i, j) = \frac{1}{d_i^c + d_c} - \theta w_i^c \tag{21}$$

where $d_i^c$ is the distance between the $i$th EV's available location $L_i$ and the pickup location of customer request $c$. $d_c$ is the distance between the pickup location and the drop-off location of customer request $c$. $w_i^c$ is the waiting time of customer request $c$ between the current time and the pickup time if the request is assigned to the $i$th EV.

- For the charge action:

$$b(i, j) = \begin{cases} \dfrac{r_M}{p_i^{CS} + p_i^E + 0.1}, & p_i^E < 0.5 \\ -r_M, & else \end{cases} \tag{22}$$

where $r_M$ is a positive tunable parameter. $p_i^{CS}$ is the ratio between the $i$th EV's current distance to the charging station and the maximum possible distance from any point to the charging station. $p_i^E$ is the ratio between the $i$th EV's current remaining battery level and its full battery level.

We set $\theta = 0.01$ and $r_M = 0.008$ for all the test cases.

The societal costs over all the operating horizon of the proposed reinforcement learning approach will be compared with that of the above two benchmarks. The societal costs of operating an EV fleet for ride-hailing services of an episode $e$ equals the summation of traveling costs of all EVs and the waiting costs of all customers:

$$C_s(e) = \sum_{i=1}^{N_{EV}} d_i(e) C_p + \sum_{c=1}^{N_{TC}(e)} w_c(e) C_w \tag{23}$$

where $d_i(e)$ denotes the total distance traveled by the $i$th EV in episode $e$. $w_c(e)$ is the waiting time of the $c$th customer in episode $e$. $N_{TC}(e)$ is the total number of customer requests in episode $e$. Note that the cost of electricity is implicitly modeled as part of the EV traveling cost. $C_p$ and $C_w$ are the costs of EV traveling and customer waiting, respectively. The equivalent traveling cost of an electric taxi in New York is given in [37], which ranges from \$0.29 to \$0.61 per mile. The traveling cost of an EV in Austin, Texas ranges from \$0.392 to \$0.876 per mile as shown in [22]. Therefore, we set the EV traveling cost $C_p$ to \$0.5/mile in this numerical study. The customer waiting cost $C_w$ is set to \$2/hour. Note that these parameters can be easily adjusted by the user for their specific community and operating condition.

### A. Single Region Test Case

*1) Simulation Settings:* The single region/city test case simulates a scenario where the majority of the customer trip requests come from the region/city center. A fleet of 50 EVs are dispatched to offer ride-hailing services in a square region consisting of a $10 \times 10$ grid. Each small square in the grid has a dimension of $2 \times 2$ miles. An EV can either stay at its current grid point or move to an adjacent grid point in the next time step. The grid points can be located by their coordinates. For example, (3,4) is the coordinates of the grid point located at the 3rd column and the 4th row. The charging station is assumed to be located at grid point (1,1). The time interval $T^I$ used in the simulation and decision making process is assumed to be 0.1 hours. The maximum number of customer trip requests being processed at each time step is 65, i.e., $|\mathcal{C}(t)| \leq N_C = 65$. The battery capacity of each EV is set at $B_{EV} = 80$ kWh. The EVs are randomly placed across the test region at the initial time step with random remaining battery levels. Note that the remaining battery levels should be sufficient for the corresponding EVs to reach the charging station.

The parameters of the reinforcement learning and testing process are set up as follows. One training episode represents an operating day, which consists of 240 time steps. The state-value functions of individual EVs are estimated by a three-layer feed-forward neural network with two fully connected hidden layers. Each hidden layer has 200 neurons. The input variables of the neural network are scaled as follows. Remaining battery level $E_i$ is normalized by battery capacity $B_{EV}$. The *available* location of the $i$th EV $L_i$ is normalized by $L$, which is the number of grid points in a column of the test region. $T_i^A$ is normalized by the number of time steps it takes for an EV to travel from the bottom left corner of the region to the upper right corner. $t$ is normalized by the time length of one episode. The replay memory of the reinforcement learner has a capacity of 2,000 *experiences* and the size of training minibatch is 10. The target network's parameters are updated every 5 time steps, i.e., $C = 5$. The annealing step $\Delta\epsilon$ is fixed at $4 \times 10^{-6}$. The discount factor $\gamma$ is 0.9999. The four parameters of the reward functions $\epsilon_C$, $w_C$, $r_S$, and $w_S$ are set to be 0.0001, 0.01, 2, and 0.06, respectively. We use a gradient descent based algorithm called Adam [38] to update the parameters of evaluation network at each training iteration. The learning rate of the Adam optimizer is set at $2 \times 10^{-5}$. We train the evaluation network for 4,000 episodes and save the network parameters every 80 episodes for out-of-sample testing and evaluation.

The customer trip requests are produced randomly for every training episode. The time stamps of customers' trip requests are generated from a Poisson process with an expectation of 20 trip requests per hour. The coordinates of the trip requests' pick up locations are generated by Algorithm 2. Note that more trip requests are generated from the region/city center in this test case. The drop-off locations of the trip requests are sampled uniformly across the whole region. The pickup location and the drop-off location of a trip request must be different. Otherwise, they will be re-sampled.

*2) Performance of State-Value Function Estimation:* The effectiveness of the proposed RL method highly depends

This article has been accepted for inclusion in a future issue of this journal. Content is final as presented, with the exception of pagination.

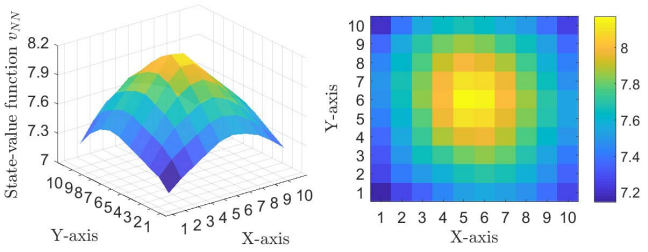


**Algorithm 2** Pickup Location Generation

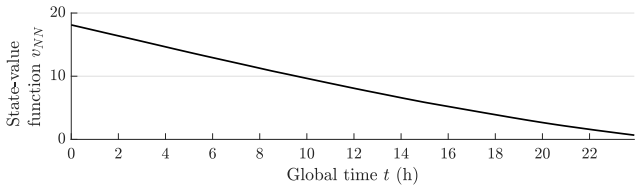
Draw a sample pair $(x, y)$ from multivariate normal distribution $\mathcal{N}(\begin{bmatrix}0\\0\end{bmatrix}, \begin{bmatrix}\frac{0.5L}{3} & 0\\ 0 & \frac{0.5L}{3}\end{bmatrix})$;
**if** $x < -0.5L$ **then**
  $x = 1$;
**else if** $x > 0.5L$ **then**
  $x = L$;
**else**
  $x = \lceil x \rceil + 0.5L$;
Repeat lines 2-7 for $y$;
Return $(x, y)$

on the performance of the state-value function estimation. In this subsection, we will assess the quality of the state-value function estimation by varying certain variables of the state vector and observing if the corresponding changes in $v_{NN}$ are reasonable. Specifically, we vary one state variable at a time and leave the other scaled state variables fixed at 0.5. The parameters of the state-value function estimator at the end of all training episodes are used in the evaluation. The value-function estimation results under different scenarios are recorded and plotted in Fig. 2. Fig. 2a depicts the state-values of different EV locations. As an EV moves closer to the center of the region, its state-value increases. This pattern is consistent with the fact that both traveling cost and customer waiting time can be reduced when an EV is strategically located close to the region center, which is the preferred customer pickup location. Fig. 2b shows that the estimated state-value decreases monotonically as the global time $t$ increases. Recall that the state-value function equals the summation of expected discounted rewards up to the end of the episode. Thus, the state-value should, in theory, continue to decrease as we approach the terminal time state. Fig. 2c and 2d suggest that EVs with a higher remaining battery level and less service load tend to have higher state-value functions. This pattern is again consistent with our expectations.

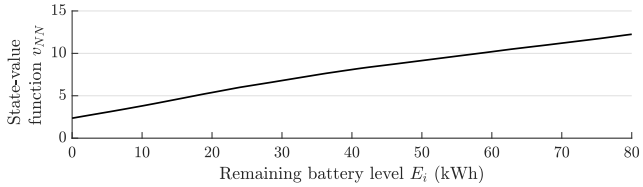
*3) Out-of-Sample Testing:* We save the trained reinforcement learning (RL) model every 80 training episodes as the training process proceeds. For each saved RL model, we evaluate its performance on a fixed out-of-sample testing set containing 50 different episodes. The performance of the proposed RL model is evaluated by measuring the average societal cost of operating the EV fleet of all the testing episodes. Fig. 3 shows both the average episodic return (red line) and the average societal cost of the testing episodes (blue line). The black horizontal solid line and the dash line represent the average societal cost of operating the EV fleet following the benchmark greedy algorithm and the optimization-based algorithm, respectively. As the training process proceeds, the average episodic return of the proposed RL based method increases and the average episodic societal cost decreases. After a few hundred episodes of training, the proposed RL based EV fleet dispatch algorithm outperforms both the benchmark greedy algorithm and the optimization-based algorithm. At the end of the training process, the proposed RL based algorithm yields a 20.73% and a 10.17% societal saving on the testing episodes compared with the greedy algorithm and the optimization-based algorithm, respectively.



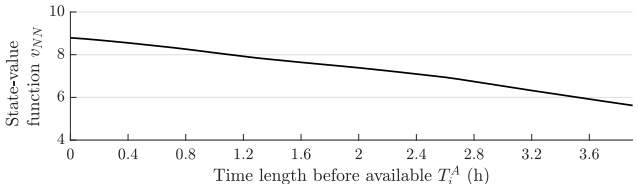

(a) Estimated state-value function of different EV locations.




(b) Estimated state-value function with different global time.




(c) Estimated state-value function with different remaining battery levels.




(d) Estimated state-value function with different time steps before available.

Fig. 2. Performance of the state-value function estimation in the single region test case.

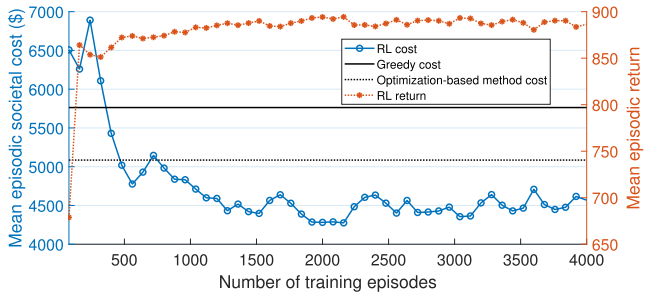


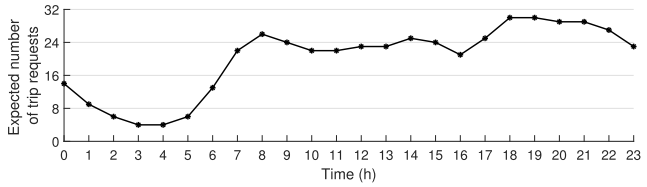

Fig. 3. Model performance as the training process proceeds for the single region test case.

## B. Commute Test Case

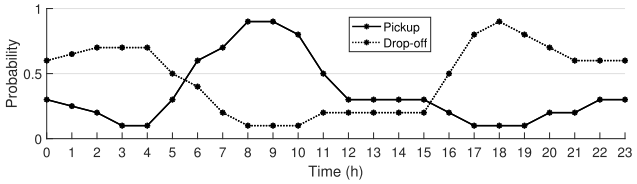
*1) Simulation Settings:* In the commute test case, we simulate a scenario where residents commute between a dwelling region and a working region. Specifically, we assume that each of these two adjacent regions consists of a 8 × 8 grid.

This article has been accepted for inclusion in a future issue of this journal. Content is final as presented, with the exception of pagination.






(a) Expected number of trip requests of different hours in a day.

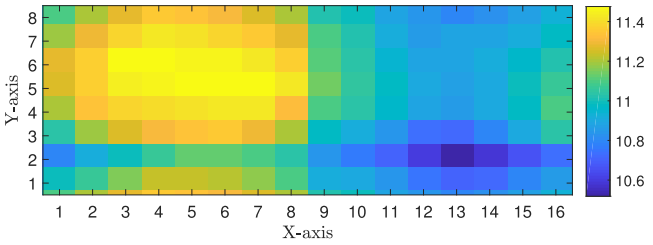



(b) Probabilities of trip pickup and drop-off locations belonging to the dwelling region of different hours in a day

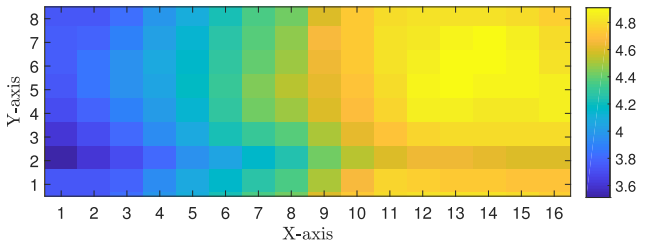
Fig. 4. Temporal and spatial distributions of trip requests.

Together they form a rectangular area that consists of a $8 \times 16$ grid. The charging station is placed at grid point (8, 4), which is in the middle of the two regions. The time stamps of customer trip requests are generated through a Poisson process with a time-varying expectation. Fig. 4a shows the expected number of customer trip requests for different hours in a day. This temporal distribution of trip requests is derived by scaling down the average hourly customer trip requests served by New York yellow cabs in a weekday [39]. The pickup and drop-off locations of each trip request are sampled independently through two steps. The first step is determining the pickup/drop-off region of the customer trip request through a Bernoulli trial. The probabilities of the trip pickup and drop-off locations belonging to the dwelling region of different hours in a day are shown in Fig. 4b. In the commute test case, the majority of the trip requests' pickup locations are in the dwelling region during the morning rush hours, while most of the trip requests' pickup locations are in the working region during the evening rush hours. The second step is determining the specific coordinates of the pickup/drop-off locations. We assume that both pickup and drop-off coordinates are generated from Gaussian distributions according to Algorithm 2 for the corresponding regions. The other simulation settings of the commute test case are the same as that of the single region test case.

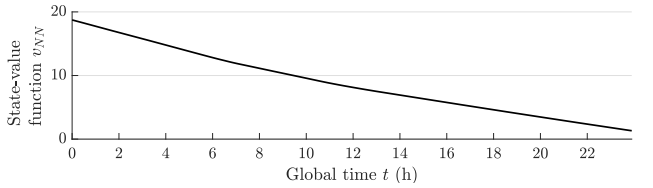
*2) Performance of State-Value Function Estimation:* We assess the quality of the state-value function estimation in the commute test case by varying certain variables of the state variables using the neural network with parameters obtained at the end of the training session. Fig. 5a and 5b show the estimated state-values of different locations at 8 am (morning rush hour) and 6 pm (evening rush hour) when the scaled $E_i$ and $T_i^A$ are fixed at 0.5 and 0, respectively. As expected, the dwelling region (left side of the rectangular area) has higher state-values during the morning rush hours, while the working region (right side of the rectangular area) has higher state-values during the evening rush hours. Fig. 5c, 5d, and 5e show the change in estimated state-value with respect to global time $t$, remaining battery level $E_i$, and time before



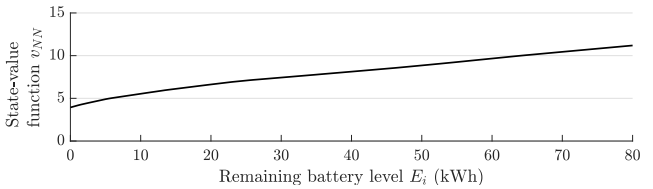

(a) Estimated state-value function with different EV locations at 8 am.



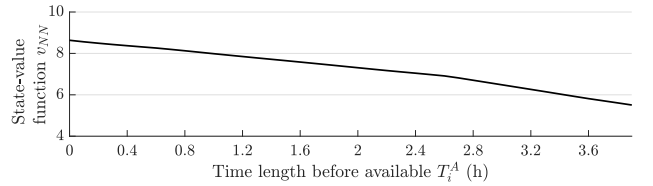

(b) Estimated state-value function with different EV locations at 6 pm.




(c) Estimated state-value function with different global time.




(d) Estimated state-value function with different remaining EV battery level.




(e) Estimated state-value function with different time steps before available.

Fig. 5. Performance of the state-value function estimation in the commute test case.

available $T_i^A$. For each case, the EV location is fixed at grid point (8,4) and the scaled values of the other input variables are set at 0.5. As depicted in these subfigures, the changes in state-values with respect to these three state variables are similar to that of the single region case, which are consistent with our expectations.

*3) Out-of-Sample Testing:* Similar to the single region case, we save the trained RL model every 80 training episodes as the training process proceeds. We evaluate the performance of each saved RL model on a fixed out-of-sample testing set containing 50 different episodes. The performance of the proposed RL model is evaluated by measuring the average

This article has been accepted for inclusion in a future issue of this journal. Content is final as presented, with the exception of pagination.



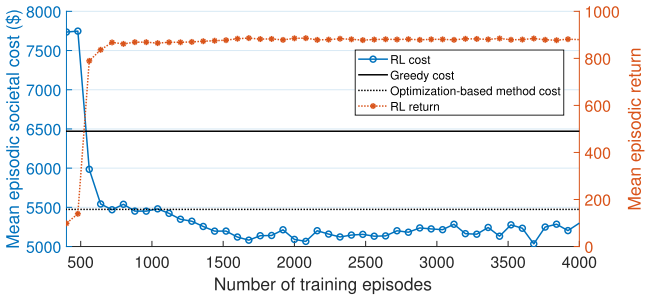



Fig. 6. Model performance as the training session proceeds for the commute test case.

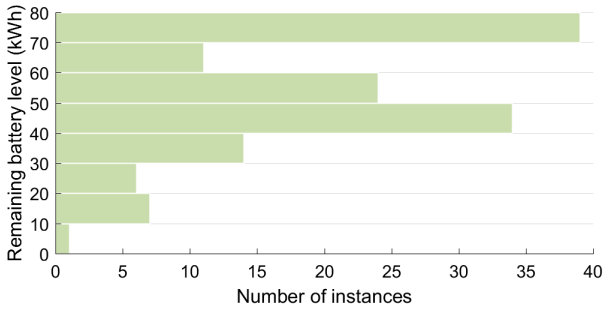



Fig. 7. Distribution of EV remaining battery levels when leaving the charging station in one testing episode.

societal costs of operating the EV fleet for all of the testing episodes. Fig. 6 shows the average episodic return (red line), the average societal cost of RL model (blue line), the benchmark greedy algorithm (black solid line), and the optimization-based algorithm (black dash line). As the training session proceeds, the average episodic return of the RL model increases and the average episodic societal cost decreases. After about 1,000 training episodes, the proposed RL-based EV fleet dispatch algorithm outperforms both the greedy algorithm and the optimization-based algorithm. At the end of the training session, the proposed RL-based EV fleet dispatch method achieves a 18.12% and a 3.20% societal cost saving on the testing episodes compared with the greedy algorithm and the optimization-based method, respectively.

It is worth noting that the proposed RL framework does allow partial charging. We show this by recording the remaining battery levels of EVs whenever they depart from the charging station in one testing episode. The distribution of the recorded EV remaining battery levels is shown in Fig. 7. As shown in the histogram, the majority of the EV charging sessions are not full recharge.

### C. Optimality Evaluation

It is interesting to consider a baseline, where all the customers' trip requests are known in advance. In this case, we could find the global optimal solution for a small-scale test case within reasonable time. In this way, we can quantify the performance gap between this optimal baseline, our proposed RL-based algorithm, and the benchmark algorithms. In this small-scale test case, we have a single region with 3 EVs and 6 customers. The simulation settings are the same as the

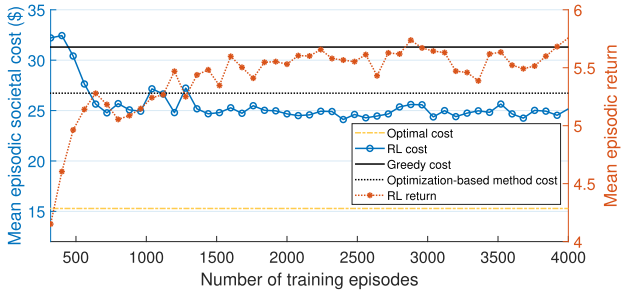



Fig. 8. Model performance as the training process proceeds for the single region test case.

previous single region test case except for the followings. The test region is a $5 \times 5$ grid. The operating time horizon is assumed to be 3 hours. The time stamps of the customers' trip requests are generated uniformly across the whole operating time horizon. The EV travelling cost $C_p$ is \$0.2/mile. The customer waiting cost $C_w$ is \$5/hour. $w_S$ is 0.5. The battery capacity $B_{EV}$ is 10 kWh.

Similar to previous test cases, we train the RL model for 4,000 episodes and save the intermediate trained RL model every 80 episodes. For each saved RL model, we evaluate its performance on a fixed out-of-sample testing set containing 50 different sets of customer requests and initial EV states. The results are shown in Fig. 8. The yellow dash-dot line is the average optimal cost of all the scenarios in the testing set, which is obtained by solving a mixed-integer programming problem following the procedures from [20]. Note that the optimal baseline is found by assuming that all of the customer trip requests are known at the initial time. In this case, the EVs can be routed to the exact pickup locations even before the corresponding requests come up. However, our proposed algorithm and the benchmark algorithms do not have perfect foresight about the upcoming customer trip requests. Thus, the optimal baseline solution is better than the solutions found by the benchmark algorithms and our proposed method.

## VI. Conclusion

This paper develops a RL based algorithm to dispatch an EV fleet for ride-hailing services. The proposed RL based algorithm is built on a novel framework with decentralized learning and centralized decision making components. The decentralized learning component allows the entire EV fleet to share their experiences and parameters of the approximated state-value function in the training process, which greatly improves the scalability of the algorithm. The centralized decision making process enables coordination of the individual EVs by formulating the EV fleet dispatch problem as a linear assignment problem, which maximizes the EV fleet's action-value function. A comprehensive numerical study is carried out to evaluate the performance of the proposed RL based algorithm. The simulation results show that the RL agent quickly learns how to dispatch an EV fleet to provide ride-hailing services. Our proposed RL algorithm outperforms the benchmark algorithms in terms of societal costs, which include the EV operational costs and the customer waiting time.

This article has been accepted for inclusion in a future issue of this journal. Content is final as presented, with the exception of pagination.



## REFERENCES


[1] R. Nealer, D. Reichmuth, and D. Anair, "Cleaner cars from cradle to grave: How electric cars beat gasoline cars on lifetime global warming emissions," *Union Concerned Sci.*, 2015.

[2] T. Bunsen *et al.*, *Global EV Outlook 2018: Towards Cross-Modal Electrification*. Paris, France: International Energy Agency, 2018.

[3] C. Peng, J. Zou, and L. Lian, "Dispatching strategies of electric vehicles participating in frequency regulation on power grid: A review," *Renew. Sustain. Energy Rev.*, vol. 68, pp. 147–152, Feb. 2017.

[4] P. Toth and D. Vigo, *Vehicle Routing: Problems, Methods, and Applications*. Philadelphia, PA, USA: SIAM, 2014.

[5] Z. Xu *et al.*, "Large-scale order dispatch in on-demand ride-hailing platforms: A learning and planning approach," in *Proc. 24th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, Jul. 2018, pp. 905–913.

[6] G. B. Dantzig and J. H. Ramser, "The truck dispatching problem," *Manage. Sci.*, vol. 6, no. 1, pp. 80–91, Oct. 1959.

[7] H. N. Psaraftis, M. Wen, and C. A. Kontovas, "Dynamic vehicle routing problems: Three decades and counting," *Networks*, vol. 67, no. 1, pp. 3–31, Jan. 2016.

[8] S. Erdoğan and E. Miller-Hooks, "A green vehicle routing problem," *Transp. Res. E, Transp. Rev.*, vol. 48, no. 1, pp. 100–114, Jan. 2012.

[9] J. Bard, L. Huang, P. Jaillet, and M. Dror, "A decomposition approach to the inventory routing problem with satellite facilities," *Transp. Sci.*, vol. 32, no. 2, pp. 189–203, May 1998.

[10] M. Schneider, A. Stenger, and D. Goeke, "The electric vehicle-routing problem with time windows and recharging stations," *Transp. Sci.*, vol. 48, no. 4, pp. 500–520, Mar. 2014.

[11] A. Felipe, M. T. Ortuño, G. Righini, and G. Tirado, "A heuristic approach for the green vehicle routing problem with multiple technologies and partial recharges," *Transp. Res. E, Logistics Transp. Rev.*, vol. 71, pp. 111–128, Nov. 2014.

[12] J. Lin, W. Zhou, and O. Wolfson, "Electric vehicle routing problem," *Transp. Res. Proc.*, vol. 12, pp. 508–521, Jan. 2016.

[13] D. Goeke and M. Schneider, "Routing a mixed fleet of electric and conventional vehicles," *Eur. J. Oper. Res.*, vol. 245, no. 1, pp. 81–99, Aug. 2015.

[14] G. Hiermann, J. Puchinger, S. Ropke, and R. F. Hartl, "The electric fleet size and mix vehicle routing problem with time windows and recharging stations," *Eur. J. Oper. Res.*, vol. 252, no. 3, pp. 995–1018, Aug. 2016.

[15] R. Basso, P. Lindroth, B. Kulcsár, and B. Egardt, "Traffic aware electric vehicle routing," in *Proc. IEEE 19th Int. Conf. Intell. Transp. Syst. (ITSC)*, Nov. 2016, pp. 416–421.

[16] A. Montoya, C. Guéret, J. E. Mendoza, and J. G. Villegas, "The electric vehicle routing problem with nonlinear charging function," *Transp. Res. B, Methodol.*, vol. 103, pp. 87–110, Sep. 2017.

[17] A. Froger, J. E. Mendoza, O. Jabali, and G. Laporte, "Improved formulations and algorithmic components for the electric vehicle routing problem with nonlinear charging functions" *Comput. Oper. Res.*, vol. 104, pp. 256–294, Apr. 2019.

[18] M. Keskin, G. Laporte, and B.Çatay, "Electric vehicle routing problem with time-dependent waiting times at recharging stations," *Comput. Oper. Res.*, vol. 7, pp. 77–94, Jul. 2019.

[19] T. Chen, B. Zhang, H. Pourbabak, A. Kavousi-Fard, and W. Su, "Optimal routing and charging of an electric vehicle fleet for high-efficiency dynamic transit systems," *IEEE Trans. Smart Grid*, vol. 9, no. 4, pp. 3563–3572, Jul. 2018.

[20] J. Shi, Y. Gao, and N. Yu, "Routing electric vehicle fleet for ride-sharing," in *Proc. IEEE 2nd Conf. Energy Internet Energy Syst. Integr. (EI2)*, Oct. 2018, pp. 1–6.

[21] J. Jung, R. Jayakrishnan, and K. Choi, "Shared-taxi operations with electric vehicles," *Inst. Transp. Stud. Work. Paper Ser.*, Irvine, CA, USA, May 2012.

[22] T. D. Chen, K. M. Kockelman, and J. P. Hanna, "Operations of a shared, autonomous, electric vehicle fleet: Implications of vehicle & charging infrastructure decisions," *Transp. Res. A, Policy Pract.*, vol. 94, pp. 243–254, Dec. 2016.

[23] N. Kang, F. Feinberg, and P. Papalambros, "Autonomous electric vehicle sharing system design," *J. Mech. Des.*, vol. 139, no. 1, Jan. 2017, Art. no. 011402.

[24] L. Al-Kanj, J. Nascimento, and W. B. Powell, "Approximate dynamic programming for planning a ride-sharing system using autonomous fleets of electric vehicles," Oct. 2018, *arXiv:1810.08124*. [Online]. Available: https://arxiv.org/abs/1810.08124

[25] M. Nazari, A. Oroojlooy, L. V. Snyder, and M. Takáč, "Reinforcement learning for solving the vehicle routing problem," *Adv. Neural Inf. Process. Syst.*, Dec. 2018, pp. 9861–9871.

[26] J. Holler *et al.*, "Deep Q-learning approaches to dynamic multi-driver dispatching and repositioning," in *Proc. NeurIPS Deep Reinforcement Learn. Workshop*, Dec. 2018.

[27] K. Lin, R. Zhao, Z. Xu, and J. Zhou, "Efficient large-scale fleet management via multi-agent deep reinforcement learning," Feb. 2018, *arXiv:1802.06444*. [Online]. Available: https://arxiv.org/abs/1802.06444

[28] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*. Cambridge, MA, USA: MIT Press, 2018.

[29] J. K. Gupta, M. Egorov, and M. J. Kochenderfer, "Cooperative multi-agent control using deep reinforcement learning," in *Proc. Int. Conf. Auto. Agents Multiagent Syst.*, May 2017, pp. 66–83.

[30] V. Mnih *et al.*, "Human-level control through deep reinforcement learning," *Nature*, vol. 518, no. 7540, pp. 529–533, Feb. 2015.

[31] D. Silver *et al.*, "Mastering the game of Go without human knowledge," *Nature*, vol. 550, no. 7676, pp. 354–359, Oct. 2017.

[32] T. P. Lillicrap *et al.*, "Continuous control with deep reinforcement learning," Sep. 2015, *arXiv:1509.02971*. [Online]. Available: https://arxiv.org/abs/1509.02971

[33] R. S. Sutton, "Learning to predict by the methods of temporal differences," *Mach. Learn.*, vol. 3, no. 1, pp. 9–44, Aug. 1988.

[34] L.-J. Lin, "Self-improving reactive agents based on reinforcement learning, planning and teaching," *Mach. Learn.*, vol. 8, nos. 3–4, pp. 293–321, May 1992.

[35] F. Bourgeois and J.-C. Lassalle, "An extension of the Munkres algorithm for the assignment problem to rectangular matrices," *Commun. ACM*, vol. 14, no. 12, pp. 802–804, Dec. 1971.

[36] D. Reyes, A. Erera, M. Savelsbergh, S. Sahasrabudhe, and R. O'Neil, "The meal delivery routing problem," *Optim. Online*, Mar. 2018.

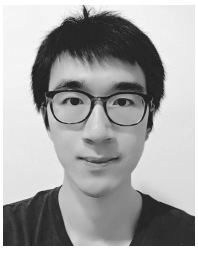
[37] G. S. Bauer, J. B. Greenblatt, and B. F. Gerke, "Cost, energy, and environmental impact of automated electric taxi fleets in Manhattan," *Environ. Sci. Technol.*, vol. 52, no. 8, pp. 4920–4928, Mar. 2018.

[38] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," Dec. 2014, *arXiv:1412.6980*. [Online]. Available: https://arxiv.org/abs/1412.6980

[39] NYC Taxi and Limousine Commission. *2017 Yellow Taxi Trip Data*. Accessed: Aug. 2018. [Online]. Available: https://data.cityofnewyork.us/Transportation/2017-Yellow-Taxi-Trip-Data/biws-g3hs


**Jie Shi** (S'19) received the B.S. degree in automation from the Shenyang University of Technology, Shenyang, China, in 2012, and the M.S. degree in control theory and engineering from Southeast University, Nanjing, China, in 2015. He is currently pursuing the Ph.D. degree in electrical and computer engineering with the University of California, Riverside, CA, USA. His research interests include smart city, machine learning, and time series analysis.

**Yuanqi Gao** (S'16) received the B.S. degree in electrical engineering from Donghua University, Shanghai, China, in 2015. He is currently pursuing the Ph.D. degree in electrical and computer engineering with the University of California, Riverside, CA, USA. His research interests include big data analytics in smart grids and machine learning with application to power distribution system.



**Wei Wang** (S'16) received the B.S. degree in electrical engineering from the Huazhong University of Science and Technology, Wuhan, China, in 2012, and the M.S. degree in electrical and computer engineering from the University of Michigan, Ann Arbor, USA, in 2014. He is currently pursuing the Ph.D. degree in electrical and computer engineering with the University of California, Riverside, CA, USA. His research interests include power system optimization and data analytics in smart grid.

This article has been accepted for inclusion in a future issue of this journal. Content is final as presented, with the exception of pagination.



**Nanpeng Yu** (M'11–SM'16) received the B.S. degree from Tsinghua University, Beijing, China, in 2006, and the M.S. and Ph.D. degrees from Iowa State University, Ames, IA, USA, in 2007 and 2010, respectively, all in electrical engineering. He is currently an Associate Professor with the Department of Electrical and Computer Engineering, University of California, Riverside, CA, USA. His current research interests include electricity market design and optimization, machine learning in smart grid, and smart energy communities. He is also an Editor of the IEEE TRANSACTIONS ON SMART GRID and the *International Transactions on Electrical Energy Systems*.

**Petros A. Ioannou** (F'97) received B.S. degree from University College, London, in 1978, and the M.S. and Ph.D. degrees from the University of Illinois at Urbana–Champaign, Illinois, in 1980 and 1982, respectively. In 1982, he joined the Department of Electrical Engineering, University of Southern California, where he is currently the A. V. 'Bal' Balakrishnan Chair Professor, the Director of the Center of Advanced Transportation Technologies, and the Associate Director for Research of METRANS, University Transportation Center. He also holds a courtesy appointment with the Department of Aerospace and Mechanical Engineering and the Department of Industrial Systems Engineering. He is also the author/coauthor of eight books and over 400 research articles in the areas of controls and intelligent transportation systems. His research interests are in the areas of adaptive control and vehicle dynamics, Intelligent Transportation Systems (ITS), and urban transportation for freight and people. He is also a fellow of IFAC and IET. He was the General Chair of IV 2017, Redondo Beach, CA, USA. He is also the VP for publications of the IEEE TRANSACTIONS ON ITS.