Joint Charging Scheduling and Computation Offloading in EV-Assisted Edge Computing: A Safe DRL Approach

Yongchao Zhang, Jia Hu, Geyong Min, *Member, IEEE,* Xin Chen, and Nektarios Georgalas, *Member, IEEE*

Abstract—Electric Vehicle-assisted Multi-access Edge Computing (EV-MEC) is a promising paradigm where EVs share their computation resources at the network edge to perform intensive computing tasks while charging. In EV-MEC, a fundamental problem is to jointly decide the charging power of EVs and computation task allocation to EVs, for meeting both the diverse charging demands of EVs and stringent performance requirements of heterogeneous tasks. To address this challenge, we propose a new joint charging scheduling and computation offloading scheme (OCEAN) for EV-MEC. Specifically, we formulate a cooperative two-timescale optimization problem to minimize the charging load and its variance subject to the performance requirements of computation tasks. We then decompose this sophisticated optimization problem into two sub-problems: charging scheduling and computation offloading. For the former, we develop a novel safe deep reinforcement learning (DRL) algorithm, and theoretically prove the feasibility of learned charging scheduling policy. For the latter, we reformulate it as an integer non-linear programming problem to derive the optimal offloading decisions. Extensive experimental results demonstrate that OCEAN can achieve similar performances as the optimal strategy and realize up to 24% improvement in charging load variance over three state-of-the-art algorithms while satisfying the charging demands of all EVs.

Index Terms—Electric vehicles, edge computing, charging scheduling, computation offloading, safe reinforcement learning.

1 Introduction

Electric vehicles (EVs), as an environment-friendly alternative to conventional petroleum-based vehicles, have been receiving considerable attention from the automobile industry and governments. According to a report from the International Energy Agency, the number of EVs will reach 250 million globally by 2030 [1]. Meanwhile, with the emergence of diverse delay-sensitive vehicular applications such as autonomous driving and vehicular video streaming [2], modern vehicles are expected to be equipped with performant computation and storage devices to effectively run these applications. However, these rich resources on EVs are mostly underutilized when they are parked somewhere (e.g., in a parking lot), overlooking the fact that their idle resources could be used to support various computing tasks at the network edge [3], [4]. Nevertheless, due to the limited battery capacities, parked EVs may be reluctant to share their computation resources for task execution.

To address this issue, a promising way is to effectively utilize the vast idle resources of EVs to extend the computational and storage capabilities of the network edge while EVs are charging [5], which we refer to as the EV-assisted multi-access edge computing (EV-MEC). Given a certain incentive mechanism (e.g., monetary reward and free parking), the EVs while charging are generally willing to share their computation resources to assist task processing as long as their charging demands can be satisfied before leaving [6]. In the EV-MEC, a group of EVs as a whole acts as a static network infrastructure that can provide effective computing and storage services. For example, a pool of EVs charged in the parking lot of shopping malls can be envisioned as an edge computing platform to serve a myriad of customers inside the mall [7]. Thus, EV-MEC can alleviate the huge economic and time costs of deploying massive edge servers to meet the ever-increasing computing demands of various delay-sensitive applications.

In the EV-MEC, a fundamental problem is to jointly determine the charging power of EVs and allocation of computation tasks among EVs, which has attracted increasing attention in recent years [5], [8], [9]. Specifically, it is crucial to efficiently schedule the charging power of EVs to meet the diverse EV charging demands. Meanwhile, the computation offloading decisions should be jointly optimized to minimize the energy consumption of task processing and satisfy the stringent performance requirements (e.g., service delay) of computation tasks. Therefore, an efficient joint charging scheduling and computation offloading strategy is necessary for the EV-MEC.

However, the design of such a strategy faces two critical challenges. First, the lack of knowledge on the system dynamics, including the diverse EV charging demands,

- *Yongchao Zhang, Jia Hu and Geyong Min are with the Department of Computer Science, University of Exeter, Exeter EX4 4QF, U.K. E-mails: yz737@exeter.ac.uk; j.hu@exeter.ac.uk; g.min@exeter.ac.uk*
- *Xin Chen is with the Computer School, Beijing Information Science and Technology University, Beijing 100101, China. E-mail: chenxin@bistu.edu.cn*
- *Nektarios Georgalas is with Applied Research Department, British Telecom, London EC1A 7AJ, U.K. E-mail: nektarios.georgalas@bt.com*
- *This work was supported in part by UKRI Grant No. EP/X038866/1 and Horizon EU Grant No. 101086159. For the purpose of open access, the author has applied a Creative Commons Attribution (CC BY) license to any Author Accepted Manuscript version arising.*
- *Corresponding authors: Jia Hu and Geyong Min.*

dynamic task arrivals, and heterogeneous performance requirements, causes huge complexity to the joint charging scheduling and offloading optimization problem in the EV-MEC. Specifically, since these system dynamics are heavily influenced by various uncertain factors (e.g., traffic conditions, vehicle behaviors, and user preferences), it is extremely difficult to acquire their complete knowledge in advance, posing significant challenges in finding the optimal charging scheduling and computation offloading decisions. Second, even if this knowledge is known as a priori, the varied target battery levels, charging time, computation abilities of EVs, and processing demands of computation tasks (e.g., computation density and service delay) as well as the mutual coupling between charging scheduling and computation offloading create significant challenges in the problem-solving.

Many efforts have been devoted to the charging scheduling for EVs [10]–[12] and computation offloading in MEC [13]–[15], respectively. Meanwhile, some recent studies have paid attention to the joint charging scheduling and computation offloading in EV-MEC [5], [8], [9]. However, they did not take the dynamics of both charging demands and computation task arrivals into account, which are important characteristics of the practical EV-MEC systems. To fill this gap, we develop a novel joint charging scheduling and computation offloading (OCEAN) algorithm for the EV-MEC based on safe model-free deep reinforcement learning (DRL), which can automatically learn the system dynamics and make reliable and optimal decisions accordingly. Our major contributions are summarized as follows.

- We formulate a cooperative two-timescale optimization problem that is composed of the charging scheduling of EVs at a long timescale (e.g., in minutes) and the offloading of computation tasks at a short timescale (e.g., in seconds). The optimization objective is to minimize the charging load and its variance while satisfying the constraints of the performance requirements of computation tasks and charging demands of EVs.
- We model the charging scheduling of EVs as a constrained Markov decision process (CMDP), and develop a novel Lyapunov-based safe DRL algorithm to efficiently solve this CMDP. Specifically, we apply the Lyapunov approach to transform the long-term constraints into a sequence of single-step *state-wise* safety constraints, and devise a new action projection approach to obtain the safe actions that satisfy the above *state-wise* safety constraints.
- We theoretically prove the feasibility of learned charging scheduling strategy that can satisfy the charging demands of EVs. Given the charging scheduling decisions, we formulate the offloading of computation tasks as an integer non-linear programming problem to determine the task assignment and computation resource allocation, with the aim of minimizing the energy consumption of task processing while meeting the strict delay requirements of tasks.
- Extensive simulation results and performance analysis demonstrate that OCEAN can guarantee the selected charging scheduling and computation offloading decisions to meet the charging demands of EVs. Furthermore, OCEAN can achieve similar performances as the optimal strategy (which knows ahead all system uncertainties such as EV charging demands and task arrivals), and realize up to 24% improvement in charging load variance over three state-of-the-art algorithms while guaranteeing the charging demands of all EVs.

The rest of this paper is organized as follows. Section 2 describes the related work. The system model and problem formulation are presented in Section 3. In Section 4, we propose the OCEAN algorithm. Simulation results are presented and analyzed in Section 5. Finally, Section 6 concludes this paper.

2 Related Work

In this section, we discuss the existing studies on charging scheduling for EVs and computation offloading in MEC as well as summarize the differences between this work and previous studies.

There exist many studies focusing on the charging scheduling for EVs. Liu *et al.* [10] introduced software-defined networking into vehicular edge computing, based on which they proposed a scalable EV charging scheduling approach to jointly optimize the charging station selection and route planning decisions for minimizing the charging time and fares. Yan *et al.* [11] focused on the dynamic wireless charging problem for EVs. They developed an offline deployment approach for mobile energy disseminators and proposed a DRL method to adjust the deployment decisions in an online manner based on real-time road traffic. Li *et al.* [12] formulated a constrained EV charging and discharging scheduling problem to minimize the charging cost and proposed a reinforcement learning-based method to learn the scheduling strategy. Cao *et al.* [16] studied the charging scheduling problem to reduce the electricity bill of EV fleets under unknown future information on EV arrival time, departure time, and charging demand, and designed an actor-critic learning-based charging approach that improves computational efficiency through reducing the state dimension during training processes.

The computation offloading in MEC has also attracted widespread interest in recent years. Tütüncüoğlu *et al.* [13] proposed a queuing network model of task graph execution and designed an online distributed rate-adaptive task offloading strategy to solve Nash equilibrium. Yan *et al.* [14] formulated a mixed non-linear programming problem to jointly optimize the computation offloading and resource allocation decisions in MEC, and developed two imitation learning approaches to learn the near-optimal policy and fast adapt to the changes in network environments, respectively. Li *et al.* [15] modelled the quality-of-service driven task offloading problem as a mixed integer non-linear programming and presented a convex optimization and Gibbs sampling-based algorithm that can converge to the global optimal solution with a high probability. Teng *et al.* [17] formulated a multi-server multi-task allocation and scheduling problem with the objective of maximizing MEC system profit as well as designed both centralized and distributed greedy-based algorithms to solve this problem.

Meanwhile, some efforts have been devoted to joint charging scheduling and computation offloading in EV-

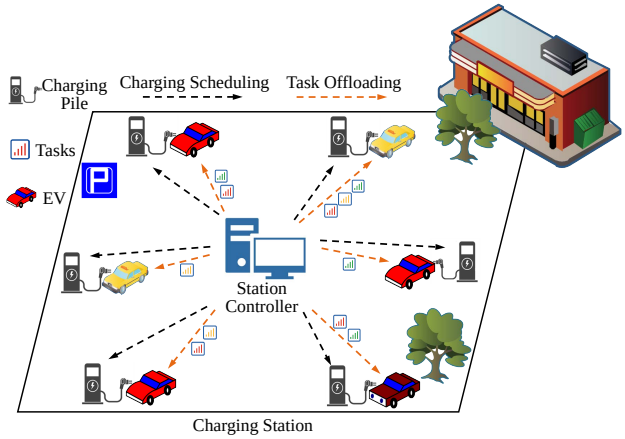


Fig. 1: An illustration of the studied EV-MEC.

MEC. Wei *et al.* [5] developed a multi-attribute contract-based charging and computation offloading scheme to maximize the utility of charging stations. Huang *et al.* [8] formulated a bi-level optimization problem to determine the charging/discharging power of EVs with the constraint of service delay, and took advantage of duality theory and linear relaxation method to solve the above problem. Zhang *et al.* [9] studied the joint optimization of task allocation and charging scheduling of mobile charging vehicles to minimize the total energy consumption. However, these works all focused on a static optimization problem, ignoring the dynamics of charging demands and task arrivals. This paper differs significantly from all the above studies. Both the dynamic charging demands and task arrivals are fully considered in our formulated optimization problem. We formulate a cooperative two-timescale optimization problem, rather than the single-timescale problem in all aforementioned works. In addition, we propose a new safe DRL-based algorithm that can theoretically guarantee that the charging demands of EVs can be met.

3 Charging Scheduling and Computation Offloading for EV-MEC

3.1 System Overview

We consider a charging station equipped with N charging piles denoted by $\mathcal{N} = \{1, 2, \cdots, N\}$, as shown in Fig. 1. The entire time horizon is divided into T time slots $\mathcal{T} = \{0, 1, \cdots, T-1\}$ with an equal length of τ. Due to the high time-sensitivity of computation tasks, their offloading is performed at each time slot $t \in \mathcal{T}$, e.g., every second. However, as it is impractical to adjust the charging power of EVs at such a short timescale, the charging scheduling decisions are updated every $\tilde{T}$ time slots ($1 < \tilde{T} < T$), e.g., every 15 minutes, as shown in Fig. 2.

3.2 Charging Scheduling of EVs

For the EV using the i-th charging pile (i.e., EVi)[1], let $t_i^a \in \mathcal{T}$ denote its arrival time, and SoC_i^a denote its initial state of charge (SoC). Due to the high mobility of EVs, the arrival rates of incoming EVs at different time slots of the day

1. For simplicity, we use EVi to represent the EV that occupies the i-th charging pile at time slot t.

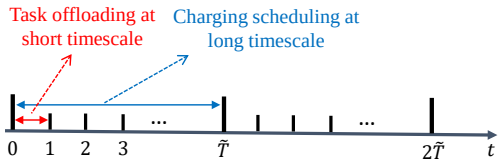


Fig. 2: Charging scheduling and task offloading at two different timescales.

$\mathcal{T}$ are different, which means the EV arrival at charging stations has a time-varying distribution. In this paper, we consider that when an EV arrives at the charging station, it will report its departure time and target SoC to the charging station controller [18], [19][2], which are denoted by $t_i^d \in \mathcal{T}$ and SoC_i^d, respectively. In addition, we denote the SoC of EVi at the beginning of time slot t as SoC_i^t, which should satisfy

$$0 \leq SoC_i^t \leq 1, \forall i \in \mathcal{N}, \tag{1}$$

where $SoC_i^t = 0$ indicates that the battery of EVi is empty, while $SoC_i^t = 1$ represents a full battery.

Considering the dynamic arrival time of EVs and diverse charging demands, the total energy demand of all EVs while charging at the charging station also varies over time. At each time slot t, the charging station controller needs to determine the charging power for each EVi, i.e., p_i^t, which is limited by

$$0 \leq p_i^t \leq p_i^{max}, \forall i \in \mathcal{N}, \tag{2}$$

where p_i^{max} is the maximum charging power of EVi. Note that as mentioned before, the charging power of each EV is updated every $\tilde{T}$ time slot.

For each EVi, in order to meet its charging demand before leaving, we have the following constraint

$$SoC_i^d \leq SoC_i^{t_i^d}, \forall i \in \mathcal{N}. \tag{3}$$

In this work, we investigate the dynamics of charging demands by taking into account the uncertain EV arrival times along with the varying both initial and target SoC levels, and make an assumption that EVs would leave as scheduled (as in many prior works, e.g., [18], [21]–[23]). It is worth noting that our proposed algorithm is also adaptable to scenarios where EV users change their plans and leave earlier or later. EVs can notify the charging station of a new departure time, and our learned scheduling policy can accordingly adjust the charging power, in order to meet their charging demands. However, if EV users drive away before the predefined time, they need to accept that their desired charging demands may not be satisfied, like [24], [25].

3.3 Offloading of Computation Tasks

In this paper, we can consider the scenario of a roadside charging station, where EVs charging at this station collaboratively share their computation resources to support the

2. In case this setting is not applicable or available, predictive analytics techniques can be used to forecast the departure time of EVs based on historical charging data of EVs, which has been validated in previous studies and shown to obtain excellent prediction results, e.g., [20].

traffic management for smart cities [26]–[28]. The tasks include traffic prediction, video analysis for congestion monitoring, traffic control, etc. The need for offloading computing tasks arises due to the rapidly growing computational demands of modern urban traffic systems. A centralized system handling all the above tasks would necessitate vast computation resources. By leveraging the computation capabilities of parked EVs at roadside charging stations, we can distribute the computational load to enhance efficiency and reduce response times, thereby optimizing overall traffic management performance.

At each time slot t, we assume there are total M_t computation tasks to be computed, each of which is represented by a tuple $(b_j^t, \chi_j^t, d_{j,max}^t)$, where b_j^t is the data size of task $j \in \{1, 2, \cdots, M_t\}$, χ_j^t is the required CPU cycles that can be obtained by using the call-graph analysis method [29], and $d_{j,max}^t$ is the maximal tolerable delay to accomplish this task. In this paper, we consider delay-sensitive computation tasks [30]–[32], assuming that each task is required to be completed within one time slot, i.e., $d_{j,max}^t \leq \tau$. Without loss of generality, M_t varies across different time slots due to the highly dynamic task arrivals. At the beginning of each time slot, the charging station allocates these computation tasks to different EVs for computing, and then the computation results are returned after the task execution is completed. Since the output of computation tasks is often much smaller than their input size, the transmission delay and energy consumption of result returning are neglected, which is a common setting in the related literature [33]. Let a binary variable $\alpha_{i,j}^t$ denote the assignment decision of task j at time slot t. If task j is assigned to EVi, $\alpha_{i,j}^t = 1$; otherwise, $\alpha_{i,j}^t = 0$. Since each task can only be assigned to one EV for processing, it yields

$$\sum_{i=1}^{N} \alpha_{i,j}^t = 1, \forall j \in \{1, 2, \cdots, M_t\}. \tag{4}$$

To encourage and compensate EVs for sharing their idle computation resources, we adopt a discounted charging price as the monetary reward, which is modelled as a function of the total CPU cycles required for all tasks processed by each EV during its charging period. In more detail, the discounted charging price received by EVi is defined as $v_i = \varkappa \cdot \max\{0, 1 - \lambda \cdot x_i\}$, where $\varkappa$ represents the standard price, λ is a positive coefficient, and x_i denotes the total CPU cycles required for all tasks processed by EVi throughout its charging period, calculated by

$$x_i = \sum_{t=t_i^a}^{t_i^d} \sum_{j=1}^{M_t} \alpha_{i,j}^t \chi_j^t. \tag{5}$$

This pricing function ensures that EVs contributing greater computation resources benefit from a lower charging price, thereby guaranteeing the fairness across all participating EVs while providing a monetary incentive for joining in resource sharing. Note that the function mentioned above is just one example to formulate the discounted charging price. Other decreasing functions with respect to x_i can also be employed to define the discounted price.

Then, the transmission delay $d_j^{t,u}$ of task j is

$$d_j^{t,u} = b_j^t / \sum_{i=1}^{N} \alpha_{i,j}^t \varpi_i^t, \tag{6}$$

where ϖ_i^t is the wireless transmission rate of EVi. Since EVs are stationary while charging, we consider a quasi-static wireless channel model, where the transmission rate ϖ_i^t remains unchanged during one time slot, but varies among different time slots.

Let $f_{i,j}^t$ denote the amount of computation resource (i.e., CPU frequency) allocated to compute task j. Note that if $\alpha_{i,j}^t = 0$, then $f_{i,j}^t$ also equals to zero. Due to the limited computation capacity of EVi, we have

$$\sum_{j=1}^{M_t} \alpha_{i,j}^t f_{i,j}^t \leq f_i^{max}, \forall i \in \mathcal{N}, \tag{7}$$

where f_i^{max} is the maximum CPU frequency of EVi. Then, the task computing delay is

$$d_j^{t,c} = \sum_{i=1}^{N} \alpha_{i,j}^t \cdot \chi_j^t / f_{i,j}^t. \tag{8}$$

Thus, the total service delay of task j is $d_j^t = d_j^{t,u} + d_j^{t,c}$. To meet the delay requirement of each task, the following constraint must be satisfied,

$$d_j^t \leq d_{j,max}^t, \forall j \in \{1, 2, \cdots, M_t\}. \tag{9}$$

In addition, for each EVi, its energy consumption for task computing can be calculated as,

$$e_i^t = \sum_{j=1}^{M_t} \alpha_{i,j}^t \vartheta_i \chi_j^t (f_{i,j}^t)^2, \tag{10}$$

where ϑ_i is the effective switched capacitance that depends on the chip architecture of each EV. Based on the above definitions, the dynamics of EV battery energy from time slot t to $t+1$ can be expressed as

$$SoC_i^{t+1} = SoC_i^t + \frac{\eta_i \cdot p_i^t \cdot \tau - e_i^t}{b_i^{max}}, \tag{11}$$

where b_i^{max} is the battery capacity of EVi, and η_i is the charging efficiency coefficient. To guarantee that each EV has enough energy to compute the assigned tasks, we have the following constraint

$$SoC_i^t \cdot b_i^{max} + \eta_i \cdot p_i^t \cdot \tau \geq e_i^t. \tag{12}$$

3.4 Problem Formulation

With the large integration of EVs into our power grids in the future, we here seek to minimize the long-term load variance of the charging station, so as to avoid sudden load peaks and thus improve the stability of power systems. To this end, for a charging station, we define its load variance function as the sum of squares of its energy consumption over time [34], which is $\sum_{t=0}^{T-1} (\sum_{i=1}^{N} p_i^t \cdot \tau)^2$. By squaring the values, we emphasize the significance of large deviations from the average load, while downplaying smaller fluctuations. This approach allows to effectively evaluate the impact of extreme load peaks and their contribution to the overall load variance. Based on the above definitions and

models, we jointly optimize the charging power of EVs (i.e., p_i^t), task assignment (i.e., $\alpha_{i,j}^t$), and computation resource allocation (i.e., $f_{i,j}^t$), to minimize the load variance of the charging station while meeting the delay requirements of computation tasks and charging demands of EVs. Thus, the problem of joint charging scheduling and computation offloading (**CSCO**) can be formulated as

$$\begin{aligned} \textbf{CSCO:} \min_{p_i^t, \alpha_{i,j}^t, f_{i,j}^t} \ & \mathbb{E}[\sum_{t=0}^{T-1}(\sum_{i=1}^{N} p_i^t \tau)^2], \\ \text{s.t. } & (1), (2), (3), (4), (7), (9), (12). \end{aligned} \tag{13}$$

In (13), it can be observed that the task assignment decision $\alpha_{i,j}^t$ is the binary variable, whereas the allocated computation resource $f_{i,j}^t$ and charging power p_i^t are continuous variables. Additionally, the constraints defined in (3), (9), and (12) are non-linear. Therefore, our formulated problem **CSCO** can be classified as a mixed-integer non-linear programming problem, which is typically NP-hard. This implies that it is challenging to find an optimal solution to **CSCO** in polynomial time. In addition, considering the long-term optimization goal in **CSCO**, it requires prior knowledge of the system dynamics, such as charging demands of EVs and computation task arrivals, to derive the optimal solutions. However, it is extremely difficult to acquire such knowledge accurately, resulting in significant challenges in solving **CSCO**. Meanwhile, the charging demand constraint (3) involves the charging scheduling and offloading decisions across multiple time slots, which makes the optimization of these decisions tightly coupled. This also brings many difficulties in solving **CSCO**.

4 Algorithm Design

In this section, we develop a novel safe DRL-based algorithm, called OCEAN, to effectively solve **CSCO**. We decompose **CSCO** into two subproblems: long timescale charging scheduling and short timescale computation offloading. For the former, we develop a Lyapunov-based safe DRL algorithm to learn the optimal charging scheduling strategy. For the latter, we reformulate it as an integer non-linear programming problem to derive the computation offloading decisions.

4.1 Long Timescale Charging Scheduling Subproblem

Considering that the charging demand constraint (3) involves the decisions across multiple time slots, we first model the long timescale charging scheduling subproblem as a CMDP, which is an extension of the standard MDP augmented with long-term constraints on expected cumulative cost. The CMDP can typically be described by a five-tuple $(\mathcal{S}, \mathcal{A}, \mathcal{P}, R, C)$. $\mathcal{S}$ denotes the set of states, $\mathcal{A}$ is the set of feasible actions, $\mathcal{P}$ represents the state transition probability, which is the probability of next state s' when selecting action a at state s, R is the reward function to define the immediate reward received after performing action a at state s, and C represents the constraint cost function. Let π denote the policy that is a decision rule mapping from a state to an action. The goal in a CMDP is to find an optimal policy π that not only maximizes the expected cumulative reward but also satisfies the additional long-term constraints. In this paper, the main components of our formulated CMDP are defined as follows,

- *State*: At each decision time of charging scheduling $t = \kappa\tilde{T}(\kappa = 0, 1, 2, \cdots)$, the system state $s_t \in \mathcal{S}$ consists of the target SoC of each EV SoC_i^d, the remaining charging duration before leaving $t_i^d - t$, and current SoC SoC_i^t. In addition, since the charging scheduling and computation offloading are tightly coupled, information about the computation tasks during the last $\tilde{T}$ time slots $[(\kappa-1)\tilde{T},\ \kappa\tilde{T}-1]$ is also included. However, it would lead to an extremely large state space if the input data size, required CPU cycles, and tolerance delay of all computation tasks during the last $\tilde{T}$ time slots are included one by one. Thus, we use $(W_t, \bar{b}_t, \bar{\chi}_t, \bar{d}_t)$ to represent these computation tasks, where W_t is the total number of computation tasks generated during last $\tilde{T}$ time slots, $\bar{b}_t$ is average data input size, $\bar{\chi}_t$ is the average required CPU cycles, and $\bar{d}_t$ is the average maximal tolerable delay. Thus, the system state s_t at each decision time t is
$$s_t = \{[SoC_i^d, t_i^d - t, SoC_i^t]_{\forall i \in \mathcal{N}}, W_t, \bar{b}_t, \bar{\chi}_t, \bar{d}_t\}. \tag{14}$$
- *Action*: The action $a_t \in \mathcal{A}$ is composed of the charging powers of all EVs, which is $a_t = \{a_{1,t}, a_{2,t}, \cdots, a_{N,t}\}$ where $a_{i,t} = p_i^t$. Note that the action a_t should satisfy the constraint (2), and $a_{i,t} = 0$ if the i-th charging pile is not in use.
- *State Transition*: $\mathcal{P} : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to [0, 1]$ denotes the state transition function, which describes the distribution of next state s_{t+1} given the current state s_t and the selected action a_t.
- *Reward*: In the standard RL, the goal of the RL agent is to maximize the expected cumulative reward. However, in our formulated optimization problem (13), the charging station aims to minimize its load variance. Therefore, in order to model our proposed optimization problem as an RL problem and minimize the load variance, the reward function r_t is defined as the negative of load variance during the period $[\kappa\tilde{T},\ \kappa\tilde{T} + \tilde{T} - 1]$, which is
$$r_t = -\tilde{T}(\sum_{i=1}^{N} p_i^t \tau)^2. \tag{15}$$
- *Constraint Cost*: The immediate constraint cost function $c_i(s_t), \forall i \in \mathcal{N}$ is defined as
$$c_i(s_t) = \begin{cases} 1 - SoC_i^t, & \text{if } t = t_i^d; \\ 0, & \text{Otherwise.} \end{cases} \tag{16}$$
Then, to satisfy the charging demand of each EV, which is (3), the long-term cumulative constraint cost should satisfy
$$\mathbb{E}\{\sum_{t=t_i^a}^{t_i^d} c_i(s_t)\} \leq 1 - SoC_i^d, \forall i \in \mathcal{N}. \tag{17}$$

Compared to standard MDP, the long-term constraint (17) in the above CMDP, also called the safety constraint, makes the actions of multiple time slots highly coupled, which renders this CMDP difficult to solve directly. To make it more tractable, we first transform the long-term safety constraint (17) into a sequence of single-step *state-wise*

constraints by taking advantage of the Lyapunov approach [35].

4.1.1 Lyapunov-based Safety Constraint Transformation

We denote $\Delta(s) = \{\pi(\cdot|s)|\sum_{a\in\mathcal{A}}\pi(a|s) = 1, \forall s \in \mathcal{S}\}$ as a set of Markov stationary policies. Let $T_{\pi,h}[V](s)$ denote the generic Bellman operator w.r.t. policy $\pi \in \Delta$ and generic cost function h, which is

$$T_{\pi,h}[V](s) = \sum_a \pi(a|s)[h(s,a) + \sum_{s'\in\mathcal{S}} \mathcal{P}(s'|s,a)V(s')]. \quad (18)$$

Then, we define a set of Lyapunov functions as

Definition 1. *Given the immediate constraint cost function $c_i(s)$ in (16) and letting $\hat{q}_i = 1 - SoC_i^d$ denote the safety threshold, a set of Lyapunov functions is defined as*

$$\mathcal{L}_{i,\pi_B}(\hat{q}_i) = \{L_i : \mathcal{S} \to \mathbb{R}_{\geq 0} : T_{\pi_B,c_i}[L_i](s) \leq L_i(s) \leq \hat{q}_i, \forall s \in \mathcal{S}; L_i(x) = 0, \forall s \notin \mathcal{S}\}, \quad (19)$$

where π_B is a feasible (i.e., safe) policy of the above CMDP that satisfies (17).

For any arbitrary Lyapunov function $L_i \in \mathcal{L}_{i,\pi_B}(q_i)$, we denote $\mathcal{F}_{L_i}(s) = \{\pi(\cdot|s) \in \Delta : T_{\pi,c_i}[L_i](s) \leq L_i(s)\}$ as the set of L_i-induced Markov stationary policies. Due to the contraction mapping property of the Bellman operator T_{π,c_i}, it is clear that any L_i-induced policy π satisfies $\mathcal{D}_{i,\pi}(s) = \mathbb{E}\{\sum_{t=t_i^a}^{t_i^d} c_i(s_t)|\pi, s\} \leq \hat{q}_i$ because $\mathcal{D}_{i,\pi}(s) = \lim_{k\to\infty} T^k_{\pi,c_i}[L_i](s) \leq L_i(s)$ and $L_i(s) \leq \hat{q}_i$. Thus, we can find that $\mathcal{F}_{L_i}(s)$ is a set of feasible policies of the above CMDP. In this context, we need to find out a Lyapunov function $L_i \in \mathcal{L}_{i,\pi_B}(\hat{q}_i)$ whose L_i-induced policies include the optimal policy π^*. To this end, the following Lemma 1 is first presented to demonstrate that given an optimal policy π^*, with the proper cost shaping, $\mathcal{D}_{i,\pi^*}(s)$ can be transformed into a Lyapunov function induced by any feasible policy π_B, which is $L_i^\epsilon \in \mathcal{L}^i_{i,\pi_B}(\hat{q}_i)$.

Lemma 1. *For any feasible policy π_B, there exists an auxiliary function $\epsilon_i(s) : \mathcal{S} \to \mathbb{R}$ such that the Lyapunov function given by $L_i^\epsilon(s) = \mathbb{E}[\sum_{t=t_i^a}^{t_i^d} c_i(s_t) + \epsilon_i(s_t)|\pi_B, s], \forall s \in \mathcal{S}, \forall i \in \mathcal{N}$ satisfies $L_i^\epsilon(s) = 0, \forall s \notin \mathcal{S}$ and $L_i^\epsilon(s) = \mathcal{D}_{i,\pi^*}(s)$.*

The proof of Lemma 1 can be found in [35]. In addition, this auxiliary constraint cost $\epsilon_i(s)$ is proved to be uniformly bounded by $\hat{\epsilon}_i(s) = 2(t_i^d - t_i^a)D_{i,max}D_{TV}(\pi^*\|\pi_B)(s)$, where $D_{i,max}$ is the maximum immediate constraint cost, and $D_{TV}(\pi^*\|\pi_B)(s) = \frac{1}{2}\sum_{a\in\mathcal{A}}|\pi_B(a|s) - \pi^*(a|s)|$. With this upper bound $\hat{\epsilon}_i(s)$, we can construct a Lyapunov function as

$$L_i^{\hat{\epsilon}}(s) = \mathbb{E}[\sum_{t=t_i^a}^{t_i^d} c_i(s_t) + \hat{\epsilon}_i(s_t)|\pi_B, s]. \quad (20)$$

Then, it is proved in [35] that if the feasible baseline policy π_B satisfies $\max_{s\in\mathcal{S}}\hat{\epsilon}_i(s) \leq D_{i,max} \cdot \min\{\frac{\hat{q}_i - \mathcal{D}_{i,\pi_B}(s)}{(t_i^d - t_i^a)D_{i,max}}, \frac{(t_i^d - t_i^a)D_{i,max} - \bar{D}_i}{(t_i^d - t_i^a)D_{i,max} + \bar{D}_i}\}$ where $\bar{D}_i = \max_{s\in\mathcal{S}}\max_\pi \mathcal{D}_{i,\pi}(s)$, then its $L_i^{\hat{\epsilon}}$-induced feasible set of policies $\mathcal{F}_{L_i^{\hat{\epsilon}}}$ contains an optimal policy. However, it is extremely difficult to obtain $\hat{\epsilon}_i(s)$ since it requires calculating the distance between π_B and optimal policy π^* that is unknown.

To address this issue, we approximate $\hat{\epsilon}_i(s)$ with an auxiliary constraint cost $\tilde{\epsilon}_i$. In order to allow $\mathcal{F}_{L_i^{\tilde{\epsilon}}}$ to include as many policies as possible so that it is more likely to contain the optimal policy π^*, $\tilde{\epsilon}_i$ is chosen as the *largest* auxiliary constraint cost while satisfying the Lyapunov condition $T_{\pi_B,c_i}[L_i^{\tilde{\epsilon}}](s) \leq L_i^{\tilde{\epsilon}}(s), \forall s \in \mathcal{S}$ and the safety constraint $L_i^{\tilde{\epsilon}}(s) \leq \hat{q}_i$. Thus, $\tilde{\epsilon}_i$ is calculated by

$$\tilde{\epsilon}_i = (\hat{q}_i - \mathcal{D}_{i,\pi_B}(s))/(t_i^d - t_i^a), \forall i \in \mathcal{N}. \quad (21)$$

Then, we can construct the Lyapunov function $L_i^{\tilde{\epsilon}}(s)$ as

$$L_i^{\tilde{\epsilon}}(s) = \mathbb{E}[\sum_{t=t_i^a}^{t_i^d} c_i(s_t) + \tilde{\epsilon}_i|\pi_B, s], \forall s \in \mathcal{S}, \forall i \in \mathcal{N}. \quad (22)$$

By using the Lyapunov function $L_i^{\tilde{\epsilon}}(s)$, the original long-term cumulative safety constraint (17) can be transformed into a *state-wise* Lyapunov safety constraint, which is

$$T_{\pi,c_i}[L_i^{\tilde{\epsilon}}](s) \leq L_i^{\tilde{\epsilon}}(s), \forall s \in \mathcal{S}. \quad (23)$$

That is any policy π satisfying $T_{\pi,c_i}[L_i^{\tilde{\epsilon}}](s) \leq L_i^{\tilde{\epsilon}}(s)$ holds that $\mathbb{E}[\sum_{t=t_i^a}^{t_i^d} c_i(s_t)|\pi] \leq 1 - SoC_i^d, \forall i \in \mathcal{N}$. This is because, based on (23) and the definition of Lyapunov function, we have $\mathcal{D}_{i,\pi}(s) = \lim_{k\to\infty} T^k_{\pi,c_i}[L_i^{\tilde{\epsilon}}](s) \leq L_i^{\tilde{\epsilon}}(s)$. Combining (20) and (21), it yields $L_i^{\tilde{\epsilon}}(s) \leq \hat{q}_i = 1 - SoC_i^d$. Hence, we can obtain $\mathcal{D}_{i,\pi}(s) = \mathbb{E}\{\sum_{t=t_i^a}^{t_i^d} c_i(s_t)|\pi, s\} \leq 1 - SoC_i^d$.

Consequently, given a feasible policy π_B, the charging scheduling subproblem can be reformulated as

$$\textbf{SP1-1:} \min_\pi \mathbb{E}[\sum_{t=0}^{T-1}(\sum_{i=1}^{N} p_i^t \cdot \tau)^2|\pi], \quad \text{s.t. (2) and } T_{\pi,c_i}[L_i^{\tilde{\epsilon}}](s) \leq L_i^{\tilde{\epsilon}}(s), \forall s \in \mathcal{S}, \forall i \in \mathcal{N}. \quad (24)$$

4.1.2 Proposed Safe DRL Algorithm

In the following, we devise a novel safe DRL algorithm to solve **SP1-1**. We first define $Q_r(s,a)$ as the state-action reward function to represent the cumulative reward after executing action a at state s. In addition, let $Q_c^i(s,a)$ denote the state-action constraint cost function, which is

$$Q_c^i(s,a) = c_i(s) + \sum_{a\in\mathcal{A}} \pi(a|s)P(s'|s,a)\mathcal{D}_{i,\pi_B}(s'). \quad (25)$$

Then, according to (22), the state-action Lyapunov function is denoted as $Q_l^i(s,a)$, which is $Q_l^i(s,a) = Q_c^i(s,a) + \tilde{\epsilon}_i$. Given this, we have $L_i^{\tilde{\epsilon}}(s) = \sum_a \pi_B(a|s)^\top Q_c^i(s,a) + \tilde{\epsilon}_i$. Moreover, based on the definition of $T_{\pi,c_i}[L_i^{\tilde{\epsilon}}](s)$ in (18), we can obtain $T_{\pi,c_i}[L_i^{\tilde{\epsilon}}](s) = \sum_a \pi(a|s)^\top Q_c^i(s,a)$. Thus, **SP1-1** can be rewritten as

$$\textbf{SP1-2:} \max_\pi \sum_a \pi(a|s)^\top Q_r(s,a), \quad \text{s.t.}(2), \sum_a(\pi(a|s) - \pi_B(a|s))^\top Q_c^i(s,a) \leq \tilde{\epsilon}_i, \forall s \in \mathcal{S}, \forall i \in \mathcal{N}.$$

In the context of EV charging scheduling, the charging power of EVs is considered as a continuous variable. Thus, we employ a Gaussian distribution to model the continuous charging power for each EV. This modelling approach is widely used in policy gradient and actor-critic

methods with continuous action space [36]–[38]. The Gaussian distribution offers several advantages. Firstly, Gaussian distributions are differentiable, which is important when applying optimization techniques like gradient descent to update the policy parameters based on the policy gradient. Secondly, the standard deviation of Gaussian distributions can effectively characterize the uncertainty about the chosen actions by the RL agent, which allows for a principled way to balance exploration and exploitation. In this case, the charging power of each EVi is sampled from a Gaussian distribution, which is $\pi(a_i|s) \sim \mathcal{N}(\mu_i, \sigma^2), \forall i \in \mathcal{N}$. In this paper, σ is set to be fixed or independent of the state [39], which is used to control the action exploration. Considering that the stochastic Gaussian policy $\pi(a_i|s)$ is parameterized by the deterministic parameters s, μ and σ, according to identical machinery [40], **SP1-2** is equivalent as

$$\textbf{SP1-3:} \max_{\boldsymbol{\mu}} Q'_r(s, \boldsymbol{\mu}),$$
$$\text{s.t. } (2) \text{ and } Q'^i_c(s, \boldsymbol{\mu}) - Q'^i_c(s, \boldsymbol{\mu}_B) \leq \tilde{\epsilon}_i, \forall s \in \mathcal{S}, \forall i \in \mathcal{N}.$$

where $\boldsymbol{\mu} = \{\mu_1, \mu_2, \cdots, \mu_N\}$, $Q'_r(s, \boldsymbol{\mu}) = r(s, \boldsymbol{\mu}) + \sum_{a \in \mathcal{A}} \pi(a|s; \boldsymbol{\mu}, \sigma) P(s'|s, a) \mathcal{V}_{\pi_B}(s')$, $\mathcal{V}_{\pi_B}(s) = \mathbb{E}\{\sum_{t=0}^{T-1} r_t | s, \pi_B\}$, and $Q'^i_c(s, \boldsymbol{\mu}) = c_i(s) + \sum_{a \in \mathcal{A}} \pi(a|s; \boldsymbol{\mu}, \sigma) P(s'|s, a) \mathcal{D}_{i,\pi_B}(s')$. For **SP1-3**, we can find that it requires the accurate $Q'_r(s, a)$, $Q'^i_c(s, a)$ and $\boldsymbol{\mu}_B$ to derive the optimal solution. Due to the highly complex non-linear relationship making it extremely hard to directly derive an explicit mathematical expression, as well as the continuous state and action space, we here adopt deep neural networks (DNNs) to approximate their actual values. Specifically, let $\tilde{Q}'_r(s, a; \theta_r)$ denote the critic network, and $\tilde{Q}'_c(s, a; \theta_c)$ denote the constraint cost network, where θ_r and θ_c are the network parameters. In addition, let $\omega(s; \theta_\omega)$ denote the policy network, which outputs the mean of Gaussian policy for each action dimension, i.e., $\{\mu_1, \mu_2, \cdots, \mu_N\}$.

However, it is very challenging to directly derive the optimal solutions of **SP1-3** since the continuous action space makes it impossible to traverse all feasible actions to find the optimum. To tackle this issue, we apply the idea of safety layer [41] to solve **SP1-3**. The unconstrained actions are first calculated to maximize the reward through standard policy gradient algorithms, such as deep deterministic policy gradient (DDPG) and proximal policy optimization (PPO), without taking the Lyapunov safety constraint into account. Then, these unconstrained actions are passed through a safety layer (which is implemented by a DNN) to project them onto the feasibility set induced by the Lyapunov constraints. Specifically, let $\boldsymbol{\mu}_{unc}$ denote the unconstrained action to solve **SP1-3** without the Lyapunov constraints. Then, we project this unconstrained action $\boldsymbol{\mu}_{unc}$ into a constrained (i.e., feasible) one. In order to minimize the impact of action perturbation on reward degradation, we seek to perturb the unconstrained actions as little as possible. Toward this aim, the feasible action is obtained by solving the following action projection problem at each state $s \in \mathcal{S}$:

$$\arg\min_{\boldsymbol{\mu}} \frac{1}{2} \|\boldsymbol{\mu} - \boldsymbol{\mu}_{unc}\|^2, \tag{26}$$
$$\text{s.t. } Q'^i_c(s, \boldsymbol{\mu}) - Q'^i_c(s, \boldsymbol{\mu}_B) \leq \tilde{\epsilon}_i, \forall s \in \mathcal{S}, \forall i \in \mathcal{N}. \tag{27}$$

To solve (26), Chow *et al.* [40] proposed to approximate the Lyapunov constraint (27) with its first-order Taylor series at action $\boldsymbol{\mu}_B$. However, this approximation is inaccurate in practice as the action may not have a linear relationship with the cost function, hence it cannot guarantee the derived solutions satisfy the Lyapunov constraints, as shown in Section 5.2.

To overcome this drawback, we develop a novel approach based on the state-cost action function to solve the action projection problem (26). In detail, let $A(s, q)$ denote the state-cost action function to represent the action to be executed at state s whose cumulative constraint cost is q. Based on $A(s, q)$, we can then derive the feasible actions given a state s and the target cumulative constraint cost q. Thus, to solve (26), we first discretize $\tilde{\epsilon}_i$ with an equal interval $\frac{\tilde{\epsilon}_i}{K}$ and obtain a set of discrete values, i.e., $\{0, \frac{\tilde{\epsilon}_i}{K}, \frac{2\tilde{\epsilon}_i}{K}, \cdots, \tilde{\epsilon}_i\}$. Then, a set of feasible state-action constraint costs can be constructed by

$$\begin{aligned}\mathbf{G} = \{&Q'^i_c(s, \boldsymbol{\mu}_B), Q'^i_c(s, \boldsymbol{\mu}_B) - \frac{\tilde{\epsilon}_i}{K}, \\ &Q'^i_c(s, \boldsymbol{\mu}_B) - \frac{2\tilde{\epsilon}_i}{K}, \cdots, Q'^i_c(s, \boldsymbol{\mu}_B) - \tilde{\epsilon}_i\}.\end{aligned} \tag{28}$$

Given $\mathbf{G}$, we can derive a set of safe actions using the state-cost action function $A(s, q)$, which is

$$\begin{aligned}\boldsymbol{U} = \{\boldsymbol{\mu}^k | \boldsymbol{\mu}^k = A(s, Q'^i_l(s, \boldsymbol{\mu}_B) - \frac{k \cdot \tilde{\epsilon}_i}{K}), \\ \forall k \in \{1, 2, \cdots, K\}\}.\end{aligned} \tag{29}$$

Based on the set of safe actions $\boldsymbol{U}$, the action projection problem (26) can be simplified as

$$\textbf{SP1-4:} \arg\min_{\boldsymbol{\mu} \in \boldsymbol{U}} \frac{1}{2} \|\boldsymbol{\mu} - \boldsymbol{\mu}_{unc}\|^2. \tag{30}$$

SP1-4 is a convex problem that can be easily solved by the standard convex optimization techniques such as CVX solver [42]. Let $\boldsymbol{\mu}^*$ denote its optimal solution. Similar to the above critic network and constraint cost network, as it is extremely hard to explicitly model the highly complex non-linear relationship of $A(s, q)$, we also utilize a DNN to approximate its accurate value, denoted by $\tilde{A}(s, q; \theta_A)$ where θ_A is the network parameter. However, there inevitably exists an approximation error between $\tilde{A}(s, q; \theta_A)$ and its exact value, thus $\boldsymbol{\mu}^*$ might not be able to satisfy the Lyapunov constraint in some cases. To address this issue, we next give a safe policy π'. Compared to the policy obtained by solving (30), although π' has smaller improvements against π_B, it always guarantees the safety. To do this, we first give the following Theorem 1.

Theorem 1. ***(Consistent Feasibility)*** *Given a safe baseline policy π_B, suppose the successive policy π' satisfies that $\int_a |\pi'(a_i|s) - \pi_B(a_i|s)| \leq \frac{\tilde{\epsilon}_i}{2\hat{q}_i - \mathcal{D}_{i,\pi_B}(s)}, \forall s \in \mathcal{S}$, then the policy update is consistently feasible, i.e., if $\mathcal{D}_{i,\pi_B}(s) = \mathbb{E}\{\sum_{t=t_i^a}^{t_i^d} c_i(s_t) | \pi_B, s\} \leq \hat{q}_i$, then $\mathcal{D}_{i,\pi'}(s) = \mathbb{E}\{\sum_{t=t_i^a}^{t_i^d} c_i(s_t) | \pi', s\} \leq \hat{q}_i$.*

The proof of Theorem 1 can be found in Appendix A.

According to Theorem 1, we can find that when the distance between polices π' and π_B is within a bound, i.e., $\xi(s) = \frac{\tilde{\epsilon}_i}{2\hat{q}_i - \mathcal{D}_{i,\pi_B}(s)}$, this new policy π' is also feasible.

Recall that we adopt Gaussian distribution to represent the probability distribution of the agent's actions. Here, we let $\mathcal{N}(\mu', \sigma^2)$ denote the parameterized policy $\pi'(a|s)$, and $\mathcal{N}(\mu_B, \sigma^2)$ denote the parameterized policy $\pi_B(a|s)$. In this case, we can further derive the relationship between μ' and μ_B to guarantee the distance between policies π' and π_B within the bound $\xi(s)$, which is presented in the following Theorem 2.

Theorem 2. *Given* $\pi'(\mathbf{a}|s) = \frac{1}{(\sqrt{2\pi}\sigma)^N} e^{-\frac{\sum_{i=1}^{N}(a_i-\mu_i')^2}{2\sigma^2}}$ *and* $\pi_B(\mathbf{a}|s) = \frac{1}{(\sqrt{2\pi}\sigma)^N} e^{-\frac{\sum_{i=1}^{N}(a_i-\mu_{B,i})^2}{2\sigma^2}}$, *letting* x^* *and* ς^* *be the solutions of* $x^N - (\sqrt{2\pi}\sigma - x)^N = \frac{\xi(\sqrt{2\pi}\sigma)^N}{2^N}$ *and* $\frac{x^*}{\sigma} = \frac{\sqrt{2\pi}}{\sqrt{\frac{\pi}{2}} e^{\frac{(-\frac{\varsigma}{2\sigma})^2}{2}} (-\frac{\varsigma}{2\sigma} + \sqrt{(-\frac{\varsigma}{2\sigma})^2 + \frac{8}{\pi}})}$, *respectively, if* $|\mu_i' - \mu_{B,i}| \leq \varsigma^*, \forall i \in \{1, 2, \cdots, N\}$, *it holds* $\int_{\mathbf{a}} |\pi'(\mathbf{a}|s) - \pi_B(\mathbf{a}|s)| d\mathbf{a} \leq \xi$.

The proof of Theorem 2 can be found in Appendix B.

Based on Lemma 1 and Theorem 2, we can transform (26) into the following problem:

$$\begin{aligned} &\arg\min_{\boldsymbol{\mu}} \frac{1}{2} \|\boldsymbol{\mu} - \boldsymbol{\mu}_{\theta,unc}\|^2, \\ &\text{s.t. } -\varsigma^* \leq \mu_i' - \mu_{B,i} \leq \varsigma^*, \forall i \in \mathcal{N}, \forall s \in \mathcal{S}. \end{aligned} \tag{31}$$

Problem (31) is convex, thus its optimal solution $\mu_i'^*$ can be derived by using the CVX solver [42]. The above analysis proves that the solution $\mu_i'^*$ can guarantee to satisfy the Lyapunov safety constraint.

Compared to other constrained policy optimization methods, one of the key benefits of our proposed Lyapunov-based safe reinforcement learning approach is its ability to provide theoretical safety guarantees during both the learning and execution phases, meaning that the long-term constraints in the formulated CMDP can be consistently satisfied in theory. This is accomplished by ensuring the system stays within a safety region defined by the Lyapunov function. However, maintaining safety throughout the learning phase can pose challenges when employing some other constrained policy optimization methods. For instance, the work presented in [43] employs a risk function to indicate whether the selected actions violate the constraints and aims to learn a policy to minimize the violation risk. Given such a strategy, during the learning phase, the RL agent will perform a variety of actions to collect information about the environment. Some of these actions could be risky and potentially violate the constraints. Furthermore, although this approach strives to minimize the probability of constraint violations, it cannot fully guarantee that the learned policy always meets the safety constraints. In contrast, our proposed Lyapunov-based safe reinforcement learning approach can offer a theoretical guarantee of safety for the learned policy, as given by Theorem 1 in Appendix A, thus presenting a compelling advantage over other methods.

4.2 Short Timescale Computation Offloading Subproblem

Recall that the objective in (13) is to minimize the energy consumption by EV charging while flattening the load profile. Toward this aim, at each time slot t, based on the EV charging decisions p_i^t obtained by solving the above charging scheduling subproblem, we optimize the task assignment $\alpha_{i,j}^t$ and computation resource allocation $f_{i,j}^t$ to minimize the sum of energy consumption by task processing of all EVs under the task delay constraints. Thus, we formulate the short timescale computation offloading subproblem as follows:

$$\begin{aligned} &\textbf{SP2-1:} \min_{\alpha_{i,j}^t, f_{i,j}^t} \sum_{i=1}^{N} e_i^t, \\ &\text{s.t. } (4), (7), (9), (12), \\ &SoC_i^t + \frac{p_i^{max} \cdot (t_i^d - t) - e_i^t}{b_i^{max}} \geq SoC_i^d, \forall i \in \mathcal{N}. \end{aligned} \tag{32}$$

Constraint (32) guarantees to satisfy the charging demands of all EVs. **SP2-1** includes a crucial constraint to ensure the completion of all computation tasks before their deadlines, as detailed in (9). As a result, we can guarantee that all tasks assigned to EVs will be computed and finished on time. We here give an example to better illustrate these constraints. Considering a scenario with three EVs sharing their computation resources, namely EV-1, EV-2, and EV-3. They are equipped with computation capacities of 1.8 GHz, 2 GHz, and 2.3 GHz, respectively [44]. There are 30 computation tasks that need to be processed, and for simplicity, we categorize them into three different types, each with specific characteristics (b^t, χ^t, d_{max}^t): (300 KB, 60 Megacycles, 0.5s), (500 KB, 70 Megacycles, 0.6s), and (700 KB, 80 Megacycles, 0.7s), respectively [44]. The number of computation tasks for each type is set to 10. The transmission rate to each EV is set at 3 Mbps [44]. Based on these settings, we can calculate that the minimum computation resources required by all three types of tasks to meet their deadlines are 0.15 GHz, 0.16 GHz, and 0.17 GHz, respectively. We then consider a task allocation scheme where all tasks are assigned to EV-1. However, it is impossible for EV-1 to complete all tasks before their maximum tolerance delays since EV-1 does not have enough computing capacity. To address this issue, we introduce a crucial constraint (7) into our optimization problem, ensuring that the tasks allocated to each EV must not exceed its computation capacity. Under such a constraint, one feasible solution to our problem is to allocate all type-1 tasks to EV-1, type-2 tasks to EV-2, and type-3 tasks to EV-3. This ensures that each EV possesses sufficient computation resources to complete its received tasks within the specified delay requirements. Therefore, solving our optimization problem can provide an effective task offloading strategy that efficiently considers both the computation capacities of the EVs in (7) and the task performance requirements in (9). As a result, all tasks assigned to the EVs will be computed and finished before their respective deadlines.

Next, according to the Lemma 1 in [45], we transform **SP2-1** into the following problem:

$$\begin{aligned} &\textbf{SP2-2:} \min_{\alpha_{i,j}^t} \sum_{i=1}^{N} e_i^t, \\ &\text{s.t. } (4), (12), (32), \sum_{j=1}^{M_t} f_{i,j}^{*,t} \leq f_{i,max}, \forall i \in \mathcal{N}, \end{aligned} \tag{33}$$

where $f_{i,j}^{*,t} = \frac{\alpha_{i,j}^t \cdot \chi_j}{d_{j,max} - \frac{b_j}{r_i^t}}$. **SP2-2** is an integer nonlinear programming problem, which is solved by using CPLEX [46]. In the context of the joint charging scheduling and computation offloading problem addressed in this paper, the dimensionality of the above computation offloading subproblem (i.e., **SP2-2**) primarily depends on the number of EVs involved, which in turn is determined by the number of charging piles within the charging station. In cases where the charging station is with a great many charging piles, the partitioning methods can be incorporated into our proposed algorithm. By partitioning the large charging station into smaller subregions, we can schedule EV charging and allocate tasks for each subregion independently. This partitioning strategy can effectively reduce the complexity of the problem in large-scale charging station scenarios, allowing for efficient problem-solving with CPLEX. However, due to the considerations of power grid stability and economic factors, charging stations typically have only a few dozen charging piles [47], helping ensure power grid stability and prevent excessive strain on the infrastructure. Given this setup where charging stations have a relatively modest number of charging piles, it is unlikely to encounter extremely high-dimensional spaces when solving the above task offloading subproblem. Therefore, CPLEX remains a suitable solver for efficiently addressing the optimization problem **SP2-2**.

Fig. 3 shows the implementation architecture of the proposed task offloading algorithm in a charging station. This system can be divided into two main sections including the Charging Station Controller and the EVs. The Charging Station Controller plays an important role in the task offloading and includes the following components:

- Task Profiling Module: This module is in charge of collecting and analyzing the incoming computation tasks to extract their task attributes such as data size, computational requirements, and deadlines. Acquiring these attributes is necessary for task offloading.
- Resource Monitoring Module: It is used to continuously monitor the CPU, memory, battery level, and wireless channel status of each EV, and provide the real-time EV resource status to the Task Scheduler, for making the task offloading decisions.
- Task Scheduler: As the core of the system, the Task Scheduler employs our learned task offloading policy to intelligently allocate tasks to the proper EVs. By carefully evaluating both the task profiles and EV resource status, it aims to optimize the task offloading while guaranteeing all tasks are completed by their deadlines.
- Data Transmission Unit: This unit sends tasks to the EVs according to the decisions made by the Task Scheduler and receives the computing results from EVs. It acts as the communication bridge between the charging station and the EVs.

For the EVs, each of them is equipped with two main components:

- Data Transmission Unit: It is responsible for receiving computation tasks from the Charging Station Controller and returning the computing results. It ensures smooth communication between the EV and the Controller.

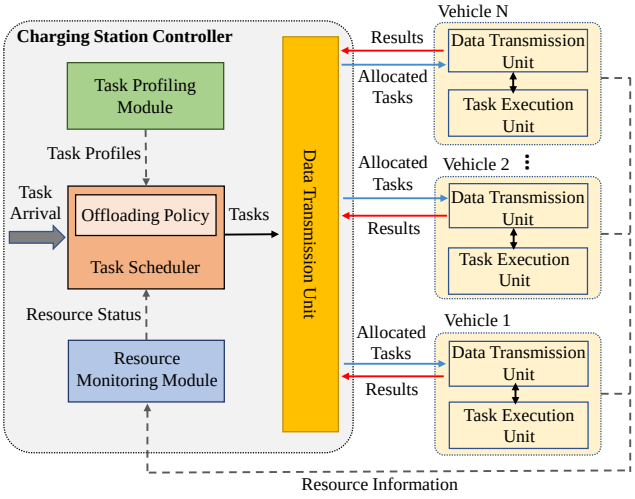


Fig. 3: The implementation architecture of task offloading in OCEAN.

- Task Execution Unit: This unit takes charge of carrying out the computation tasks received from the Task Scheduler. It ensures that tasks run securely and efficiently, constrained by the computation ability of the EV.

The task offloading workflow begins with the arrival of computation tasks at the Charging Station Controller. The Task Profiling Module creates detailed profiles for each task, while the Resource Monitoring Module acquires information about the resource status of the EVs. Utilizing the above information, the Task Scheduler intelligently allocates tasks to suitable EVs by using our learned offloading policy. The Controller then offloads these tasks to the EVs through the Data Transmission Unit. On the EV side, the Data Transmission Unit receives the allocated tasks, and the Task Execution Unit securely executes the computation tasks within their deadlines. Once the tasks are completed, EVs transmit the computing results back to the Controller, completing the task offloading process.

4.3 Joint Charging Scheduling and Computation Offloading

Based on the above analysis, we now formally specify the procedure of OCEAN, as shown in Algorithm 1. OCEAN can be divided into three major parts: trajectory generation, network training, and action projection.

Firstly, we initialize the DNN parameters $\theta_{r,0}$, $\theta_{c,0}$, $\theta_{A,0}$, $\theta_{\omega,0}$, and $\theta_{\phi,0}$, where $\theta_{\phi,0}$ denotes the parameters of the safety layer $\phi(s,\mu)$. In addition, let $\pi_B(s) = \pi_0(s) = \mathcal{N}(\phi(s,\omega(s;\theta_{\omega,0});\theta_{\phi,0}),\sigma^2)$. In the stage of trajectory generation, at the time slot $t = \kappa\tilde{T}(\kappa = 0,1,2,\cdots)$, based on state s_t, we can calculate $\mu_t = \phi(s_t,\omega(s_t;\theta_{\omega,k});\theta_{\phi,k})$, and select the charging power based on π_k. Then, at each time slot t, the task assignment and resource allocation decisions are obtained by solving (33). Next, the selected charging scheduling and offloading decisions are performed. At the end of the time slot $t = \kappa\tilde{T} + \tilde{T} - 1$, we can receive the reward r_t and constraint cost $c_i(s_t)$. At the end of time slot $T-1$, the trajectory $\xi_{z,k} = \{s_t,\mu_t,r_t,c_i(s_t)\}_{t=0}^{T-1}$ is stored for the following network training.

Algorithm 1: Joint Charging Scheduling and Computation Offloading Algorithm (OCEAN)

Initialize the number of trajectories Z and DNN parameters $\theta_{r,0}$, $\theta_{c,0}$, $\theta_{A,0}$, $\theta_{\omega,0}$, $\theta_{\phi,0}$, let $\pi_B(s) = \pi_0(s) = \mathcal{N}(\phi(s, \omega(s;\theta_{\omega,0});\theta_{\phi,0}), \sigma^2)$;
for $k = 0, 1, 2, \cdots$ **do**
  // **Trajectory Generation**;
  for $z = 0$ *to* $z = Z-1$ **do**
    Let $\kappa = 0$;
    for $t = 0$ *to* $t = T-1$ **do**
      At the time slot $t = \kappa\tilde{T}$, observe the system state s_t, obtain $\mu_t = \phi(s_t, \omega(s_t;\theta_{\omega,k});\theta_{\phi,k})$, and select the charging power of each EV based on policy $a_{i,t} \sim \pi_k$;
      Obtain the task assignment decision $\alpha_{i,j}^t$ and resource allocation decision $f_{i,j}^t$ by solving (33);
      Perform actions $a_{i,t}, \alpha_{i,j}^t$ and $f_{i,j}^t$;
      At the end of time slot $t = \kappa\tilde{T} + \tilde{T} - 1$, obtain the reward r_t, constraint cost $c_i(s_t)$, and $\kappa = \kappa + 1$;
    At the end of time slot $T-1$, store the trajectory $\xi_{z,k} = \{s_t, \mu_t, r_t, c_i(s_t)\}_{t=0}^{T-1}$;
  // **Network Training**;
  Using the collected Z trajectories $\{\xi_{j,k}\}_{z=1}^{Z}$, update the network parameters $\theta_{r,k}$, $\theta_{c,k}$, $\theta_{A,k}$, $\theta_{\omega,k}$ according to (34), (35), (36), and (37);
  // **Action Projection**;
  For any given state $s \in \{\xi_{z,k}\}_{z=1}^{Z}$, compute the auxiliary constraint cost $\tilde{\epsilon}_i$;
  Obtain the candidate action μ_i^c by solving (30);
  if $\tilde{Q}_c'^i(s, \mu_i^c) - \tilde{Q}_c'^i(s, \mu_{i,B}) \leq \tilde{\epsilon}_i$ **then**
    Let $\mu_i^* = \mu_i^c$;
  else
    Obtain μ_i^v by solving (31), and let $\mu_i^* = \mu_i^v$;
  Update the parameters of the safety layer according to (38);
  Set $\pi_{k+1}(s) = \mathcal{N}(\phi(s, \omega(s;\theta_{\omega,k});\theta_{\phi,k}), \sigma^2)$, and let $\pi_B = \pi_{k+1}$.

In the stage of network training, as shown in Fig. 4, using the collected Z trajectories, we update the critic network $\tilde{Q}_r'$, constraint cost network $\tilde{Q}_c'$, state-cost action network $\tilde{A}(s,q)$, and policy network $\omega(s, \theta_{\omega,k})$. Specifically, the parameter $\theta_{r,k}$ of the critic network is updated to minimize the following loss function:

$$Loss(\theta_{r,k}) = \mathbb{E}[y_r - \tilde{Q}_r'(s_t, \mu_t; \theta_{r,k})]^2, \tag{34}$$

where $y_r = \sum_{\iota=t}^{T-1} r_\iota$ is the target value.

Similarly, the parameter θ_c of the constraint cost network is updated by minimizing the following loss function:

$$Loss(\theta_{c,k}) = \mathbb{E}\{\sum_{i=1}^{N}[y_{i,c} - \tilde{Q}_c'^i(s_t, \mu_t; \theta_{c,k})]^2\}, \tag{35}$$

where $y_{i,c} = \sum_{\iota=t}^{T-1} c_i(s_\iota)$ is the target value.

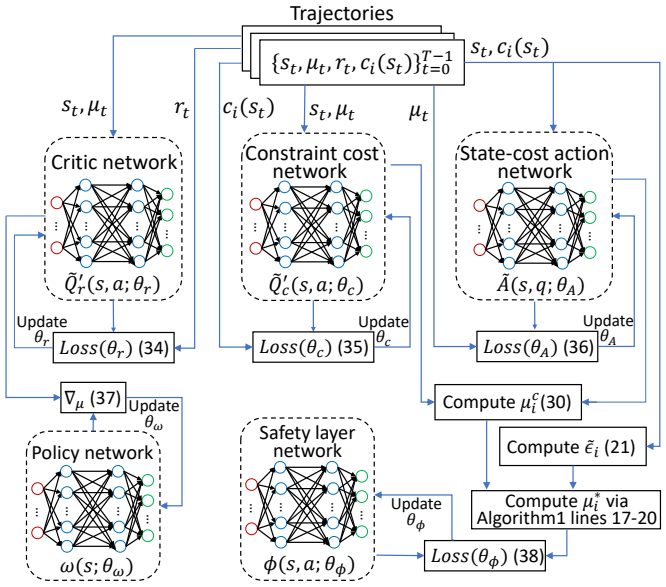


Fig. 4: Training framework of the proposed safe DRL approach.

Then, the parameter θ_A of the state-cost action network is updated to minimize the following loss function:

$$Loss(\theta_{A,k}) = \mathbb{E}[\mu_t - \tilde{A}(s_t, \{y_{i,c}\}_{i=1}^{N}; \theta_{A,k})]^2. \tag{36}$$

In addition, for the policy network $\omega(s;\theta_{\omega,k})$, its parameter $\theta_{\omega,k}$ is updated by following the objective gradient, so as to maximize the action-value function:

$$\theta_{\omega,k} = \theta_{\omega,k} - \vartheta_k \mathbb{E}_{s_t \in \{\xi_{j,k}\}_{z=1}^{Z}}[\nabla_{\theta_\omega}\omega(s_t;\theta_\omega)|_{\theta_\omega=\theta_{\omega,k}} \cdot \nabla_\mu \tilde{Q}_r(s_t,\mu)|_{\mu=\omega(s_t;\theta_{\omega,k})}]. \tag{37}$$

After the network training, we perform action projection to map the actions generated by the policy network into feasible ones that adhere to given safety constraints. This projection procedure is implemented by the safety layer network $\phi(s,\mu)$, which takes as input the system states as well as unconstrained actions generated by the policy network, and then transforms the input actions into feasible ones induced by the Lyapunov constraint in (23), ultimately guaranteeing that the charging demand constraints of EVs are always met. Toward this aim, at any given state s in the trajectories $\{\xi_{z,k}\}_{z=1}^{Z}$, we first compute the auxiliary constraint cost $\tilde{\epsilon}_i$. Based on derived $\tilde{\epsilon}_i$, we can obtain a candidate action μ_i^c at state s by solving (30). In order to check the safety of this candidate action, we calculate $\tilde{Q}_c'^i(s,\mu_i^c) - \tilde{Q}_c'^i(s,\mu_{i,B})$ and compare it with $\tilde{\epsilon}_i$. If $\tilde{Q}_c'^i(s,\mu_i^c) - \tilde{Q}_c'^i(s,\mu_{i,B}) \leq \tilde{\epsilon}_i$, it means action μ_i^c can satisfy the Lyapunov safety constraint, so we let $\mu_i^* = \mu_i^c$. Otherwise, another feasible action μ_i^v can be obtained by solving (31), and let $\mu_i^* = \mu_i^v$. Based on the derived feasible action $\boldsymbol{\mu}^* = \{\mu_1^*, \mu_2^*, \cdots, \mu_N^*\}$ at any state s in $\{\xi_{z,k}\}_{z=1}^{N}$, the safety layer is trained to minimize the following loss function:

$$Loss(\theta_{\phi,k}) = \mathbb{E}[\boldsymbol{\mu}^* - \phi(s, \omega(s;\theta_{\omega,k});\theta_{\phi,k}))]^2. \tag{38}$$

According to the policy network and safety layer, we can obtain an improved and feasible charging scheduling policy $\pi_{k+1}(s) = \mathcal{N}(\phi(s, \omega(s;\theta_{\omega,k});\theta_{\phi,k}), \sigma^2)$. We repeat steps (3)-(22) in Algorithm 1 until convergence.

TABLE 1: Electric Specifications of Four EV Models [12], [50]

Model	b_i^{max} [kWh]	p_i^{max} [kW]
Tesla Model 3	55	0-11
Nissan Leaf	24	0-6
BMW i3	33	0-7.7
Hyundai Ioniq Elec.	28	0-7

5 Performance Evaluation

In this section, we first describe the parameter settings in our experiments, and then present simulation results to demonstrate the effectiveness of OCEAN.

5.1 Experimental Settings

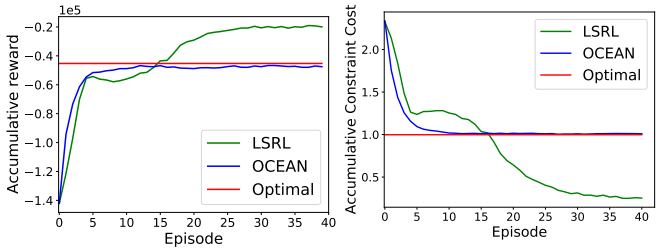
We consider a charging station consisting of $N = 25$ charging piles. Four different types of EVs are used in the experiments, which are Nissan Leaf, BMW i3, Tesla Model 3, and Hyundai Ioniq Elec. The electric specifications of these EVs are listed in Table 1. For each EV arriving to charge, its initial SoC is sampled from the truncated normal distribution $\mathcal{N}(0.5, 0.1^2)$ within the interval $[0.2, 0.8]$, and the target SoC is sampled from $\mathcal{N}(0.9, 0.1^2)$ with the bound $[0.85, 0.95]$ [36]. The charging efficiency coefficient is set as 0.95. The charging time of each EV follows a Weibull distribution where the scale parameter is 2.57 and the shape parameter is 1.27 [21]. The standard charging price $\varkappa$ is 0.10015 \$/kWh [48], and the coefficient λ is $5 \cdot 10^{-14}$. The charging scheduling decisions are updated every 15 minutes. For the computation offloading, we consider the number of computation tasks to be randomly distributed in $[70, 110]$. The task size, required CPU cycles, and task delay follow the uniform distributions $U[100, 900]$ KB, $U[50, 90]$ Megacycles, and $U[0.5, 1]$ s, respectively [44]. The length of each time slot is $\tau = 1$s. The wireless transmission rate is set to 3 Mbps [44], and the CPU energy consumption coefficient $\vartheta_i = 5 \cdot 10^{-26}$. In addition, the computation capacities of four different EVs are set as 1.8 GHz (Nissan leaf), 2 GHz (BMW i3), 2.3 GHz (Tesla Model 3), and 1.8 GHz (Hyundai Ioniq Elec.) [49]. The constraint cost network consists of four hidden layers where the numbers of neurons are 128, 256, 128, and 64, respectively. The safety layer network has three hidden layers, where the numbers of neurons are 256, 128, and 64, respectively. Both the critic and actor networks have two hidden layers each with 128 neurons and 64 neurons. The learning rate of the critic network is set to 0.001, and that of the actor network is 0.0001.

5.2 Performance of OCEAN

We first study the convergence and safety performance of OCEAN. Specifically, OCEAN is compared with classic Lyapunov-based safe reinforcement learning (LSRL) [40] to show its advantage. We also implement the optimal policy to show the upper bound of performance and the safe condition. In order to obtain the optimal policy, it is assumed that all system uncertainties such as EV charging demands and task arrivals are known in advance. In this case, **CSCO** in (13) can be considered as a deterministic problem, whose optimal solutions are solved by the CPLEX solver [46]. To deal with the various safety constraint thresholds of different EVs, we normalize the cumulative constraint cost (17) as

$$\varrho_i = \frac{1 - SoC_i^d}{\mathbb{E}[\sum_{t=0}^{T-1} c_{i,t}]}, \forall i \in \mathcal{N}. \tag{39}$$

When $\varrho_i \geq 1$, the safety constraint of EVs can be satisfied.



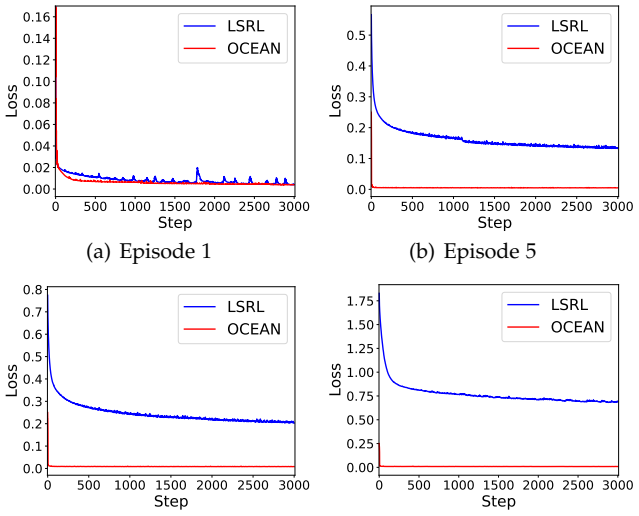
(a) Accumulative reward (b) Accumulative constraint cost

Fig. 5: The accumulative reward and constraint cost of OCEAN, LSRL, and Optimal during training.

Figs. 5(a) and 5(b) plot the accumulative reward and constraint cost achieved by OCEAN, classic LSRL, and Optimal varying with the increase of training episodes, respectively. From Fig. 5(a), we can observe that the accumulative reward of OCEAN rises with the number of training episodes, and converges after around 15 episodes. It means that a better policy can be learned by OCEAN during training to improve the system reward, and a stable policy can be learned at the end of training. It can also be seen that OCEAN can gradually approach the optimal policy, which validates its superior performance. In addition, we can observe that the classic LSRL converges after around 30 episodes, which is much slower than OCEAN. Notice that although classic LSRL can achieve a higher reward than the optimal policy, this is at the expense of safety violation, as shown in Fig. 5(b). Specifically, OCEAN can keep guaranteeing the safety of learned policy and gradually approach the optimum, while classic LSRL violates the safety condition after around 16 episodes. It is because in order to solve (26), classic LSRL approximates the Lyapunov safety constraint with its first-order Taylor series, but this approximation is inaccurate in practice, which cannot guarantee that the derived solutions meet the safety constraint.

To look into the reason behind the improvement by OCEAN over classic LSRL, we plot the loss curves of the safety layer network obtained by OCEAN and classic LSRL at different training episodes in Fig. 6. At episode 1, the loss curves of both OCEAN and classic LSRL can rapidly decrease to zero. At episode 5, the loss curve of OCEAN can still reduce to around zero, but that of classic LSRL reduces to about 0.2 and does not decrease any more. Similar results can also be observed at episodes 10 and 15. In addition, as the increase of training episodes, the loss value obtained by classic LSRL also rises. This is because classic LSRL cannot guarantee the safety of the selected actions, causing inefficiency in the training of the safety layer network. In contrast, since OCEAN is able to learn the safe policy during training, the loss value obtained by OCEAN can rapidly reduce to around zero.

Then, we study the monetary benefits received by EVs for sharing their idle computation resources. Fig. 7 plots the total CPU cycles required for all tasks processed by each



(a) Episode 1 (b) Episode 5

(c) Episode 10 (d) Episode 15

Fig. 6: The loss curves of safety layer network obtained by OCEAN and classic LSRL at different training episodes.

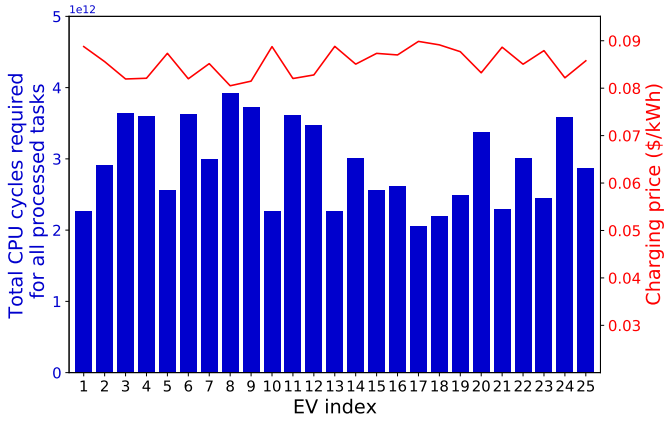


Fig. 7: The total CPU cycles required for all tasks processed by each EV and their charging prices.

EV and their charging prices. It can be observed that the charging price is inversely correlated with the total CPU cycles required for all processed tasks, indicating that the EVs with a larger computational workload have a smaller charging price. In other words, EVs that contribute more to assisting with the task processing during their charging periods are rewarded a lower charging price, which serves as an effective incentive and compensation for their resource sharing.

Moreover, we analyze the impact of various parameters of offloaded tasks in their scheduling. To facilitate the analysis, we consider four different types of computation tasks and three types of EVs [44], [49], as detailed in Tables 2 and 3. The number of tasks for each type is set to 30 and the number of EVs for each type is set to 7. Fig. 8 shows the scheduling of computation tasks to respective types of EVs. It can be observed that Type-1 tasks are only assigned to Type-2 and Type-3 EVs, with no allocation to Type-1 EVs. This is because Type-1 tasks have substantial data sizes and stringent latency requirements, which require a short communication latency, making them not fit for the offloading to Type-1 EVs whose communication rates are the

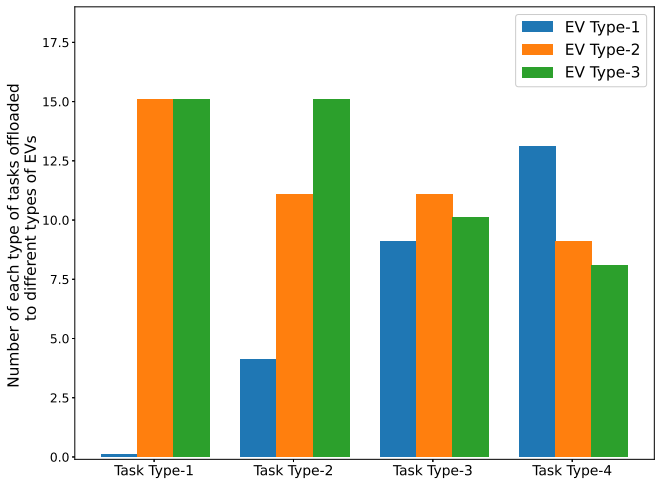


Fig. 8: Scheduling of computation tasks to EVs.

TABLE 2: Parameters of computation tasks

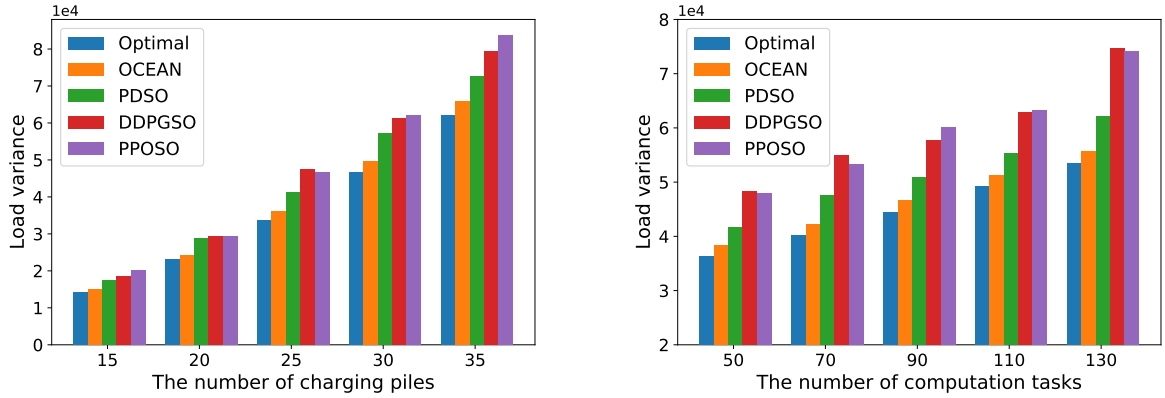
Task Type	Data size (KB)	CPU cycles (Megacycles)	Delay (s)
1	700	70	0.6
2	500	90	0.6
3	300	70	0.6
4	700	70	1.0

lowest. In contrast, Type-2 tasks, having the same latency constraints but smaller data sizes compared to Type-1 tasks, exhibit diversification in their allocation. Some of them are assigned to Type-1 EVs, owing to their reduced communication requirements resulting from diminished data sizes. The majority, however, are still scheduled to Type-2 and Type-3 EVs, characterized by enhanced communication and computation capacities in line with these tasks' requirements of substantial CPU resources and strict latency constraints. Similarly, we can observe that Type-3 tasks are distributed across all EV types for processing. This is due to their moderate communication and computation requirements, making them compatible for processing across all types of EVs. Lastly, Type-4 tasks have more relaxed delay constraints compared to the other types of tasks, which allows for their effective offloading and processing on Type-1 EVs as they do not need fast responses like the other tasks.

5.3 Comparison Results

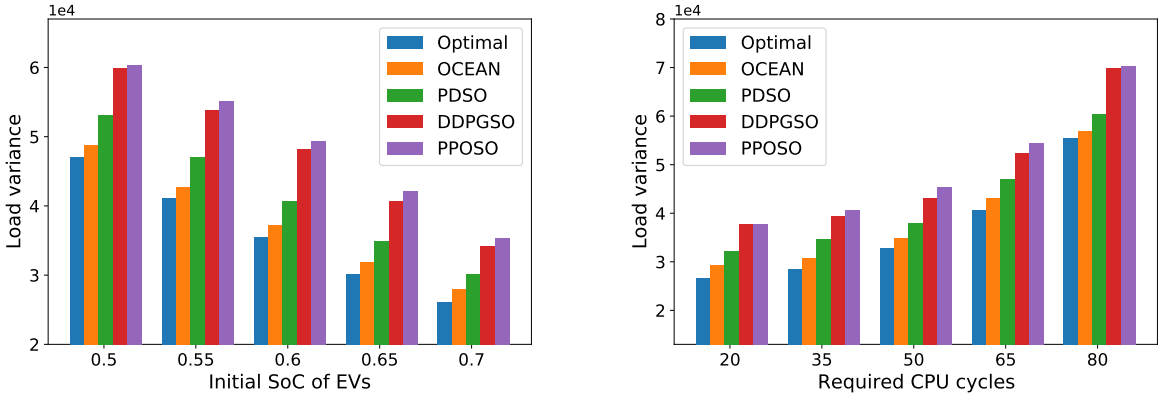
Next, we carry out comparison experiments to further demonstrate the effectiveness of our algorithm. In addition to the above optimal policy, we implement three DRL-based state-of-the-art algorithms.

- **Optimal**: As mentioned before, assuming all system uncertainties such as EV charging demands and task arrivals are known ahead, the CPLEX solver is used to derive the optimal solutions to problem **CSCO**. However, the dynamic nature of charging demands is influenced by various factors, e.g., user driving habits, traffic conditions, and weather, making their precise predictions difficult. It is also challenging to accurately predict uncertain task arrivals and varying task characteristics, such as data sizes, required computation resources, and delays. Thus, the optimal solution that requires complete



(a) Numbers of charging piles

(b) Numbers of computation tasks



(c) Initial SoC of EVs

(d) Required CPU cycles

Fig. 9: The load variances of all five algorithms under different parameters.

TABLE 3: Parameters of EVs

EV Type	Wireless Communication Rate (Mbps)	Computing Capacity (GHz)
1	1	1.8
2	2	2
3	3	2.3

knowledge of these uncertainties is impractical as such information cannot be accurately known in advance.

- **DDPG-based joint charging scheduling and computation offloading (DDPGSO)**: Many works in the literature have employed DDPG algorithm to address the charging scheduling for EVs [51] and computation offloading in MEC [52]. We implement this technique to jointly optimize the charging scheduling and computation offloading.
- **PPO-based joint charging scheduling and computation offloading (PPOSO)**: Some recent studies adopted PPO algorithm to solve the charging scheduling [37] and computation offloading [53] problems. We use the same method as in the above studies to learn the joint optimization strategy.
- **Primal-dual RL based joint charging scheduling and computation offloading (PDSO)**: The primal-dual RL approach has also been used recently to address the safe learning problem in many previous works, e.g., [54], [55]. We here leverage this approach to solve our joint charging scheduling and computation offloading problem.

To ensure the safety of learned joint charging scheduling and computation offloading policies, similar to the safe RL algorithms proposed in previous studies, e.g., [56], [57], DDPGSO and PPOSO utilize a penalty-augmented objective, which combines the expected reward objective with a penalty term associated with constraint violations for learning the safe policy.

Fig. 9(a) plots the load variances achieved by all five algorithms when the number of charging piles varies from 15 to 35. We can see that OCEAN can achieve a similar load variance as the optimal policy, and always has a better performance than PDSO, DDPGSO, and PPOSO. E.g., when $N = 25$, OCEAN achieves a load variance of 36165.7, compared to 41264.6 obtained by PDSO with a 12.4% reduction, 47547.2 obtained by DDPGSO with a 23.9% reduction, and 46530.1 obtained by PPOSO with a 22.3% reduction. On average, OCEAN can decrease the load variance by 12.7%, 19.3%, and 22.6% over PDSO, DDPGSO, and PPOSO, respectively. It indicates that OCEAN can effectively optimize the charging scheduling and offloading decisions to minimize the load variance.

Fig. 9(b) shows the load variance achieved by all five algorithms when the number of computation tasks varies from 50 to 130. We can also see that OCEAN always outperforms PDSO, DDPGSO, and PPOSO. On average, OCEAN can achieve 9.1%, 19.2%, and 22.1% load variance reduction over PDSO, DDPGSO, and PPOSO, respectively. Fig. 9(c) presents the load variance achieved by all five

TABLE 4: Comparison of all five algorithms on the normalized cumulative constraint cost with different parameters.

		Optimal	OCEAN	PDSO	DDPGSO	PPOSO			Optimal	OCEAN	PDSO	DDPGSO	PPOSO
Number of charging piles	15	1	1.001	1.099	1.222	1.234	Initial SoC of EVs	0.5	1	1.010	1.119	1.203	1.222
	20	1	1.003	1.123	1.171	1.173		0.55	1	1.007	1.116	1.243	1.252
	25	1	1.026	1.144	1.191	1.185		0.6	1	1.026	1.101	1.272	1.283
	30	1	1.032	1.138	1.316	1.322		0.65	1	1.006	1.174	1.350	1.375
	35	1	1.034	1.107	1.205	1.248		0.7	1	1.001	1.185	1.425	1.432
Number of computation tasks	50	1	1.004	1.120	1.225	1.218	Required CPU cycles	20	1	1.006	1.021	1.034	1.195
	70	1	1.015	1.103	1.189	1.161		35	1	1.030	1.114	1.208	1.218
	90	1	1.015	1.113	1.209	1.227		50	1	1.007	1.089	1.193	1.209
	110	1	1.011	1.101	1.202	1.218		65	1	1.024	1.109	1.210	1.222
	130	1	1.005	1.087	1.200	1.184		80	1	1.011	1.096	1.200	1.205

algorithms when the initial SoC of EVs varies from 0.5 to 0.7. Compared to PDSO, DDPGSO, and PPOSO, OCEAN can reduce 8.3%, 22.7%, and 24.5% load variance on average, respectively. Fig. 9(d) shows the load variance achieved by all five algorithms when the required CPU cycles vary from 20 Megacycles to 80 Megacycles. On average, OCEAN can achieve 8.4%, 19.1%, and 22.4% load variance reduction over PDSO, DDPGSO, and PPOSO, respectively. To show the safety of OCEAN, Table 4 lists the normalized cumulative constraint costs of all five algorithms at convergence under different parameters. We can see that the constraint costs of all five algorithms are larger than 1, which means all of them can guarantee the safety of policy. Nevertheless, the constraint cost of OCEAN is significantly smaller than those of PDSO, DDPGSO, and PPOSO, and OCEAN can achieve a similar constraint cost as the optimal policy. These comparison results demonstrate that OCEAN can effectively reduce the load variance while guaranteeing the feasibility of the learned policy.

Additionally, Table 5 showcases the execution times of all five algorithms across varying numbers of charging stations. We can observe that our proposed algorithm, OCEAN, exhibits longer execution times than PDSO, DDPGSO, and PPOSO. The reason is that OCEAN combines an actor network and a safety network for action generation, whereas PDSO, DDPGSO, and PPOSO all rely solely on an actor network for decision-making. Despite this, OCEAN can still maintain its execution time within a remarkably short duration, not exceeding 7 ms. Moreover, as the number of charging stations increases, all five algorithms show an increase in execution time. Notably, Optimal shows a significantly rapid increase in execution time due to its exponential growth in computational complexity. In particular, when the number of charging stations is 60, it is unable to find an optimal solution. Although the optimal solutions can be obtained when the number of charging stations is less than 60, the execution time of Optimal is vastly larger than those of the other four algorithms.

TABLE 5: Execution times (in Seconds) of all five algorithms across varying numbers of charging stations. (N/A denotes cases unable to find optimum.)

Number of charging stations	Optimal	OCEAN	PDSO	DDPGSO	PPOSO
20	96	0.006225	0.003856	0.004081	0.004569
30	385	0.006242	0.003881	0.004103	0.004593
40	1165	0.006278	0.003919	0.004124	0.004617
50	2792	0.006319	0.003935	0.004131	0.004631
60	N/A	0.006350	0.003954	0.004155	0.004660

6 Conclusion

This paper studies the joint charging scheduling and computation offloading for EV-MEC. First, we formulate a two-timescale optimization problem with the objective of minimizing the load variance while satisfying both the charging demands of EVs and the strict performance requirements of computation tasks. Next, we develop a novel safe DRL-based intelligent algorithm, called OCEAN. Specifically, a new safe DRL algorithm is proposed to optimize the charging scheduling for EVs, and the optimization of computation offloading is reformulated as an integer nonlinear programming problem. Extensive experiments and performance comparison results show the superiority of OCEAN in reaching similar performances as the optimal strategy and considerably reducing the charging load variance compared to three state-of-the-art algorithms, while ensuring that the learned policy can satisfy the charging demands of all EVs.

For future work, the uncertainties of EV departure time will be taken into account to extend the proposed approach, which could be modelled to follow a normal distribution with the mean representing the scheduled departure time. We will also explore leveraging chance-constrained planning methods in DRL to effectively manage uncertain vehicle departures.

References

[1] I. G. E. Outlook *et al.*, "Scaling-up the transition to electric mobility," *International Energy Agency: Paris, France*, 2019.

[2] L. Liu, C. Chen, Q. Pei, S. Maharjan, and Y. Zhang, "Vehicular edge computing and networking: A survey," *Mobile networks and applications*, vol. 26, no. 3, pp. 1145–1168, 2021.

[3] C. Ma, J. Zhu, M. Liu, H. Zhao, N. Liu, and X. Zou, "Parking edge computing: Parked-vehicle-assisted task offloading for urban vanets," *IEEE Internet of Things Journal*, vol. 8, no. 11, pp. 9344–9358, 2021.

[4] X.-Q. Pham, T. Huynh-The, E.-N. Huh, and D.-S. Kim, "Partial computation offloading in parked vehicle-assisted multi-access edge computing: A game-theoretic approach," *IEEE Transactions on Vehicular Technology*, vol. 71, no. 9, pp. 10 220–10 225, 2022.

[5] Z. Wei, B. Li, R. Zhang, and X. Cheng, "Contract-based charging protocol for electric vehicles with vehicular fog computing: An integrated charging and computing perspective," *IEEE Internet of Things Journal*, pp. 1–1, 2022.

[6] Y. Li, B. Yang, Z. Chen, C. Chen, and X. Guan, "A contract-stackelberg offloading incentive mechanism for vehicular parked-edge computing networks," in *Proceedings of 2019 IEEE 89th Vehicular Technology Conference (VTC2019-Spring)*, 2019, pp. 1–5.

[7] M. Sookhak, F. R. Yu, Y. He, H. Talebian, N. Sohrabi Safa, N. Zhao, M. K. Khan, and N. Kumar, "Fog vehicular computing: Augmentation of fog computing using vehicular cloud computing," *IEEE Vehicular Technology Magazine*, vol. 12, no. 3, pp. 55–64, 2017.

[8] X. Huang, W. Zhong, J. Nie, J. Kang, Z. Xiong, Y. Wu, and M. Guizani, "Joint parking and power management for electric vehicle edge computing: A bilevel optimization approach," in *Proceedings of 2022 International Wireless Communications and Mobile Computing (IWCMC)*, 2022, pp. 719–724.

[9] W. Zhang, R. Wang, C. Yi, and K. Zhu, "Joint optimization of computation task allocation and mobile charging scheduling in parked-vehicle-assisted edge computing networks," in *Proceedings of International Conference on Wireless Algorithms, Systems, and Applications (WASA)*, 2022, pp. 406–418.

[10] J. Liu, H. Guo, J. Xiong, N. Kato, J. Zhang, and Y. Zhang, "Smart and resilient ev charging in sdn-enhanced vehicular edge computing networks," *IEEE Journal on Selected Areas in Communications*, vol. 38, no. 1, pp. 217–228, 2020.

[11] L. Yan, H. Shen, L. Kang, J. Zhao, Z. Zhang, and C. Xu, "Mobicharger: Optimal scheduling for cooperative ev-to-ev dynamic wireless charging," *IEEE Transactions on Mobile Computing*, 2022, DOI: 10.1109/TMC.2022.3200414.

[12] H. Li, Z. Wan, and H. He, "Constrained ev charging scheduling based on safe deep reinforcement learning," *IEEE Transactions on Smart Grid*, vol. 11, no. 3, pp. 2427–2439, 2020.

[13] F. Tütüncüoğlu, S. Jošilo, and G. Dán, "Online learning for rate-adaptive task offloading under latency constraints in serverless edge computing," *IEEE/ACM Transactions on Networking*, 2022, DOI: 10.1109/TNET.2022.3197669.

[14] Z. Yan, P. Cheng, Z. Chen, B. Vucetic, and Y. Li, "Two-dimensional task offloading for mobile networks: An imitation learning framework," *IEEE/ACM Transactions on Networking*, vol. 29, no. 6, pp. 2494–2507, 2021.

[15] Q. Li, S. Wang, A. Zhou, X. Ma, F. Yang, and A. X. Liu, "Qos driven task offloading with statistical guarantee in mobile edge computing," *IEEE Transactions on Mobile Computing*, vol. 21, no. 1, pp. 278–290, 2022.

[16] Y. Cao, H. Wang, D. Li, and G. Zhang, "Smart online charging algorithm for electric vehicles via customized actor–critic learning," *IEEE Internet of Things Journal*, vol. 9, no. 1, pp. 684–694, 2022.

[17] H. Teng, Z. Li, K. Cao, S. Long, S. Guo, and A. Liu, "Game theoretical task offloading for profit maximization in mobile edge computing," *IEEE Transactions on Mobile Computing*, pp. 1–1, 2022.

[18] J. Jin and Y. Xu, "Optimal policy characterization enhanced actor-critic approach for electric vehicle charging scheduling in a power distribution network," *IEEE Transactions on Smart Grid*, vol. 12, no. 2, pp. 1416–1428, 2021.

[19] X. Chen, K.-C. Leung, A. Y. S. Lam, and D. J. Hill, "Online scheduling for hierarchical vehicle-to-grid system: Design, formulation, and algorithm," *IEEE Transactions on Vehicular Technology*, vol. 68, no. 2, pp. 1302–1317, 2019.

[20] O. Frendo, N. Gaertner, and H. Stuckenschmidt, "Improving smart charging prioritization by predicting electric vehicle departure time," *IEEE Transactions on Intelligent Transportation Systems*, vol. 22, no. 10, pp. 6646–6653, 2021.

[21] C. B. Saner, A. Trivedi, and D. Srinivasan, "A cooperative hierarchical multi-agent system for ev charging scheduling in presence of multiple charging stations," *IEEE Transactions on Smart Grid*, vol. 13, no. 3, pp. 2218–2233, 2022.

[22] L. Liu and K. Zhou, "Electric vehicle charging scheduling considering urgent demand under different charging modes," *Energy*, vol. 249, p. 123714, 2022.

[23] T. Long, Q.-S. Jia, G. Wang, and Y. Yang, "Efficient real-time ev charging scheduling via ordinal optimization," *IEEE Transactions on Smart Grid*, vol. 12, no. 5, pp. 4029–4038, 2021.

[24] S. Han, S. Han, and K. Sezaki, "Development of an optimal vehicle-to-grid aggregator for frequency regulation," *IEEE Transactions on Smart Grid*, vol. 1, no. 1, pp. 65–72, 2010.

[25] W. Shi and V. W. Wong, "Real-time vehicle-to-grid control algorithm under price uncertainty," in *2011 IEEE International Conference on Smart Grid Communications (SmartGridComm)*, 2011, pp. 261–266.

[26] Z. Ning, J. Huang, and X. Wang, "Vehicular fog computing: Enabling real-time traffic management for smart cities," *IEEE Wireless Communications*, vol. 26, no. 1, pp. 87–93, 2019.

[27] X. Wang, Z. Ning, and L. Wang, "Offloading in internet of vehicles: A fog-enabled real-time traffic management system," *IEEE Transactions on Industrial Informatics*, vol. 14, no. 10, pp. 4568–4578, 2018.

[28] S.-S. Lee and S. Lee, "Resource allocation for vehicular fog computing using reinforcement learning combined with heuristic information," *IEEE Internet of Things Journal*, vol. 7, no. 10, pp. 10 450–10 464, 2020.

[29] L. Yang, J. Cao, Y. Yuan, T. Li, A. Han, and A. Chan, "A framework for partitioning and execution of data stream applications in mobile cloud computing," *SIGMETRICS Perform. Eval. Rev.*, vol. 40, no. 4, p. 23–32, 2013.

[30] Y. Mao, J. Zhang, and K. B. Letaief, "Dynamic computation offloading for mobile-edge computing with energy harvesting devices," *IEEE Journal on Selected Areas in Communications*, vol. 34, no. 12, pp. 3590–3605, 2016.

[31] K. Guo and T. Q. Quek, "Dynamic computation offloading in multi-server mec systems: An online learning approach," in *Proceedings of IEEE Global Communications Conference (GLOBECOM)*, 2020, pp. 1–6.

[32] H. Hu, Q. Wang, R. Q. Hu, and H. Zhu, "Mobility-aware offloading and resource allocation in a mec-enabled iot network with energy harvesting," *IEEE Internet of Things Journal*, vol. 8, no. 24, pp. 17 541–17 556, 2021.

[33] X. Lyu, H. Tian, W. Ni, Y. Zhang, P. Zhang, and R. P. Liu, "Energy-efficient admission of delay-sensitive tasks for mobile edge computing," *IEEE Transactions on Communications*, vol. 66, no. 6, pp. 2603–2616, 2018.

[34] N. I. Nimalsiri, E. L. Ratnam, D. B. Smith, C. P. Mediwaththe, and S. K. Halgamuge, "Coordinated charge and discharge scheduling of electric vehicles for load curve shaping," *IEEE Transactions on Intelligent Transportation Systems*, vol. 23, no. 7, pp. 7653–7665, 2022.

[35] Y. Chow, O. Nachum, E. Duenez-Guzman, and M. Ghavamzadeh, "A lyapunov-based approach to safe reinforcement learning," *Advances in neural information processing systems*, vol. 31, 2018.

[36] L. Yan, X. Chen, J. Zhou, Y. Chen, and J. Wen, "Deep reinforcement learning for continuous electric vehicles charging control with dynamic user behaviors," *IEEE Transactions on Smart Grid*, vol. 12, no. 6, pp. 5124–5134, 2021.

[37] Y. Jiang, Q. Ye, B. Sun, Y. Wu, and D. H. Tsang, "Data-driven coordinated charging for electric vehicles with continuous charging rates: A deep policy gradient approach," *IEEE Internet of Things Journal*, vol. 9, no. 14, pp. 12 395–12 412, 2022.

[38] C. Dai, K. Zhu, and E. Hossain, "Multi-agent deep reinforcement learning for joint decoupled user association and trajectory design in full-duplex multi-uav networks," *IEEE Transactions on Mobile Computing*, pp. 1–15, 2022.

[39] H. Satija, P. Amortila, and J. Pineau, "Constrained markov decision processes via backward value functions," in *Proceedings of International Conference on Machine Learning (ICML)*, 2020, pp. 8502–8511.

[40] Y. Chow, O. Nachum, A. Faust, E. Duenez-Guzman, and M. Ghavamzadeh, "Lyapunov-based safe policy optimization for continuous control," *arXiv preprint arXiv:1901.10031*, 2019.

[41] G. Dalal, K. Dvijotham, M. Vecerik, T. Hester, C. Paduraru, and Y. Tassa, "Safe exploration in continuous action spaces," *arXiv preprint arXiv:1801.08757*, 2018.

[42] M. Grant, S. Boyd, and Y. Ye, "Cvx: Matlab software for disciplined convex programming," 2008.

[43] C. Zhou, W. Wu, H. He, P. Yang, F. Lyu, N. Cheng, and X. Shen, "Deep reinforcement learning for delay-oriented iot task scheduling in sagin," *IEEE Transactions on Wireless Communications*, vol. 20, no. 2, pp. 911–925, 2021.

[44] X. Wang, Z. Ning, S. Guo, and L. Wang, "Imitation learning enabled task scheduling for online vehicular edge computing," *IEEE Transactions on Mobile Computing*, vol. 21, no. 2, pp. 598–611, 2022.

[45] Y. Chen, N. Zhang, Y. Zhang, X. Chen, W. Wu, and X. S. Shen, "Toffee: Task offloading and frequency scaling for energy efficiency of mobile devices in mobile edge computing," *IEEE Transactions on Cloud Computing*, vol. 9, no. 4, pp. 1634–1644, 2021.

[46] I. I. Cplex, "V12. 1: User's manual for cplex," *International Business Machines Corporation*, vol. 46, no. 53, p. 157, 2009.

[47] Z. Zhao, C. K. M. Lee, J. Ren, and Y. P. Tsang, "Optimal ev fast charging station deployment based on a reinforcement learning framework," *IEEE Transactions on Intelligent Transportation Systems*, vol. 24, no. 8, pp. 8053–8065, 2023.

[48] C. B. Saner, A. Trivedi, and D. Srinivasan, "A cooperative hierarchical multi-agent system for ev charging scheduling in presence of multiple charging stations," *IEEE Transactions on Smart Grid*, vol. 13, no. 3, pp. 2218–2233, 2022.

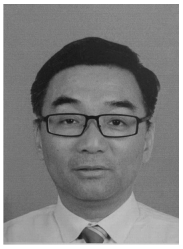
[49] Z. Ning, X. Wang, J. J. P. C. Rodrigues, and F. Xia, "Joint computation offloading, power allocation, and channel assignment

for 5g-enabled traffic management systems," *IEEE Transactions on Industrial Informatics*, vol. 15, no. 5, pp. 3058–3067, 2019.
[50] O. Fallah-Mehrjardi, M. H. Yaghmaee, and A. Leon-Garcia, "Charge scheduling of electric vehicles in smart parking-lot under future demands uncertainty," *IEEE Transactions on Smart Grid*, vol. 11, no. 6, pp. 4949–4959, 2020.
[51] R. Jin, Y. Zhou, C. Lu, and J. Song, "Deep reinforcement learning-based strategy for charging station participating in demand response," *Applied Energy*, vol. 328, p. 120140, 2022.
[52] Z. Sun, H. Yang, C. Li, Q. Yao, D. Wang, J. Zhang, and A. V. Vasilakos, "Cloud-edge collaboration in industrial internet of things: A joint offloading scheme based on resource prediction," *IEEE Internet of Things Journal*, vol. 9, no. 18, pp. 17 014–17 025, 2022.
[53] J. Wang, J. Hu, G. Min, W. Zhan, A. Y. Zomaya, and N. Georgalas, "Dependent task offloading for edge computing based on deep reinforcement learning," *IEEE Transactions on Computers*, vol. 71, no. 10, pp. 2449–2461, 2022.
[54] R. Yan, Q. Xing, and Y. Xu, "Multi agent safe graph reinforcement learning for pv inverter s based real-time de centralized volt/var control in zoned distribution networks," *IEEE Transactions on Smart Grid*, pp. 1–1, 2023.
[55] Q. Bai, A. S. Bedi, M. Agarwal, A. Koppel, and V. Aggarwal, "Achieving zero constraint violation for constrained reinforcement learning via primal-dual approach," *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 36, no. 4, pp. 3682–3689, Jun. 2022.
[56] P. Ning, H. Wang, T. Tang, L. Zhu, and X. Wang, "A service-oriented energy efficient resource allocation approach for wireless communications of the tunnel construction," *IEEE Transactions on Vehicular Technology*, vol. 72, no. 4, pp. 4948–4958, 2023.
[57] Q. Luo, T. H. Luan, W. Shi, and P. Fan, "Deep reinforcement learning based computation offloading and trajectory planning for multi-uav cooperative target search," *IEEE Journal on Selected Areas in Communications*, vol. 41, no. 2, pp. 504–520, 2023.
[58] L. Duembgen, "Bounding standard gaussian tail probabilities," *arXiv preprint arXiv:1012.2063*, 2010.

Yongchao Zhang received the M.S. degree in computer technology from the School of Computer, Beijing Information Science and Technology University, China, in 2020. He is currently working toward the PhD degree in computer science at the University of Exeter. His research interests include wireless communication, smart grid, edge computing, and deep reinforcement learning.

Jia Hu received the BEng and MEng degrees in electronic engineering from the Huazhong University of Science and Technology, China, in 2006 and 2004, respectively, and the PhD degree in computer science from the University of Bradford, UK, in 2010. He is an associate professor of computer science at the University of Exeter. His research interests include edge-cloud computing, resource optimization, applied machine learning, and network security.

Geyong Min received the BSc degree in computer science from the Huazhong University of Science and Technology, China, in 1995, and the PhD degree in computing science from the University of Glasgow, United Kingdom, in 2003. He is a professor of high performance computing and networking in the Department of Computer Science at the University of Exeter, United Kingdom. His research interests include computer networks, wireless communications, parallel and distributed computing, ubiquitous computing, multimedia systems, modeling and performance engineering.



Xin Chen received the PhD degree in computer science from the Beijing Institute of Technology, Beijing, China. He is currently a professor with the Computer School, Beijing Information Science and Technology University. His current research interests include the performance evaluation of wireless networks. He received the Postdoctoral Fellowship in Computer Architecture from Tsinghua University in 2006. He is a senior member of the China Computer Federation (CCF), a member of the CCF Technical Committee of Theoretical Computer Science, and the CCF Technical Committee of Petri Nets.

Nektarios Georgalas is a Principal Researcher with the Applied Research department of British Telecom. In his current role, he leads two collaborative research programmes with key BT partners delivering innovations in the areas of cloud, data centres, network virtualisation, smart cities, IoT and mobility. During his career with BT, since 1998, he has managed numerous collaborative and internal research projects in areas such as network management, market-driven data management systems, policy-based management, distributed information systems, SOA/Web services, model-driven design and development of telecoms OSS, cloud and NFV. He is the inventor and co-inventor of 11 patents. He has also authored more than 60 papers in international journals and conferences. He has served as general co-chair, programme cochair, programme committee and keynote speaker or invited panelist in top international IEEE academic and TMForum conferences.

APPENDIX A
PROOF OF THE THEOREM 1

Proof. Combining (21) and (22), for $\forall s \in \mathcal{S}$, we have

$$\begin{aligned} L_i^{\tilde{\epsilon}}(s) &= \mathbb{E}[\sum_{t=t_i^a}^{t_i^d} c_i(s_t) + \frac{\hat{q}_i - \mathcal{D}_{i,\pi_B}(s)}{t_i^d - t_i^a} | \pi_B, s], \\ &= \mathbb{E}[\sum_{t=t_i^a}^{t_i^d} c_i(s_t) | \pi_B, s] + \hat{q}_i - \mathcal{D}_{i,\pi_B}(s) \end{aligned} \tag{40}$$

Recall that $\mathbb{E}[\sum_{t=t_i^a}^{t_i^d} c_{i,t}(s_t)|\pi_B, s] \leq \hat{q}_i, \forall s \in \mathcal{S}$. Then, according to (40), we can obtain

$$L_i^{\tilde{\epsilon}}(s) \leq 2\hat{q}_i - \mathcal{D}_{i,\pi_B}(s), \forall s \in \mathcal{S}. \tag{41}$$

Rearranging the condition in Theorem 1, i.e., $\int_a |\pi'(a_i|s) - \pi_B(a_i|s)| \leq \frac{\tilde{\epsilon}_i}{2\hat{q}_i - \mathcal{D}_{i,\pi_B}(s)}$, we have

$$\int_a |\pi'(a_i|s) - \pi_B(a_i|s)|(2\hat{q}_i - \mathcal{D}_{i,\pi_B}(s)) \leq \tilde{\epsilon}_i. \tag{42}$$

Substituting (41) into (42), we can get

$$\int_a |\pi'(a_i|s) - \pi_B(a_i|s)| \sum_{s' \in \mathcal{S}} P(s'|s, a_i) L_i^{\tilde{\epsilon}}(s') \leq \tilde{\epsilon}_i. \tag{43}$$

Since $L_i^{\tilde{\epsilon}}(s) \geq 0, \forall s \in \mathcal{S}$, we have

$$\begin{aligned} \int_a \pi'(a_i|s) \sum_{s' \in \mathcal{S}} P(s'|s, a_i) L_i^{\tilde{\epsilon}}(s') - \\ \int_a \pi_B(a_i|s) \sum_{s' \in \mathcal{S}} P(s'|s, a_i) L_i^{\tilde{\epsilon}}(s') \leq \tilde{\epsilon}_i. \end{aligned} \tag{44}$$

Adding $c_i(s)$ to the both sides of (44), it holds

$$\begin{aligned} \int_a \pi'(a_i|s)[c_i(s) + \sum_{s' \in \mathcal{S}} P(s'|s, a_i) L_i^{\tilde{\epsilon}}(s')] \leq \\ \int_a \pi_B(a_i|s)[c_i(s) + \tilde{\epsilon}_i + \sum_{s' \in \mathcal{S}} P(s'|s, a_i) L_i^{\tilde{\epsilon}}(s')]. \end{aligned} \tag{45}$$

According to (18) and (22), (45) can be rewritten as,

$$T_{\pi', c_i}[L_i^{\tilde{\epsilon}}](s) \leq L_i^{\tilde{\epsilon}}(s) \tag{46}$$

As T_{π', c_i} is a contraction mapping, we have

$$\mathcal{D}_{i,\pi'}(s) = \lim_{k \to \infty} T_{\pi', c_i}^k [L_i^{\tilde{\epsilon}}](s) \leq L_i^{\tilde{\epsilon}}(s). \tag{47}$$

Then, based on $L_i^{\tilde{\epsilon}}(s) = \mathcal{D}_{i,\pi_B}(s) + \tilde{\epsilon}_i \cdot (t_i^d - t_i^a)$ and (21), we have

$$\mathcal{D}_{i,\pi'}(s) \leq \mathcal{D}_{i,\pi_B}(s) + \hat{q}_i - \mathcal{D}_{i,\pi_B}(s) = \hat{q}_i. \tag{48}$$

□

APPENDIX B
PROOF OF THEOREM 2

Proof. We first consider the case that $\mu_i' \geq \mu_{B,i}, \forall i \in \mathcal{N}$. According to the conditions of Theorem 2, we have

$$0 \leq \mu_i' - \mu_{B,i} \leq \varsigma^*, \forall i \in \{1, 2, \cdots, N\}. \tag{49}$$

Since ς^* is the solution of $\frac{x^*}{\sigma} = \frac{\sqrt{2\pi}}{\sqrt{\frac{\pi}{2}} e^{\frac{(-\frac{\varsigma}{2\sigma})^2}{2}} (-\frac{\varsigma}{2\sigma} + \sqrt{(-\frac{\varsigma}{2\sigma})^2 + \frac{8}{\pi}})}$, based on $\frac{1}{\sqrt{2\pi}} \int_{x_1}^{\infty} e^{-\frac{x^2}{2}} dx \leq \frac{1}{\sqrt{\frac{\pi}{2}} e^{\frac{x_1^2}{2}} (x_1 + \sqrt{x_1^2 + \frac{8}{\pi}})}$ [58], we can get

$$\sigma \int_{-\frac{\varsigma^*}{2\sigma}}^{\infty} e^{-\frac{\check{a}^2}{2}} d\check{a} \leq x^*. \tag{50}$$

As $\int_0^{\infty} e^{-\frac{(\check{a} - \frac{\varsigma^*}{2})^2}{2\sigma^2}} d\check{a} = \int_{-\frac{\varsigma^*}{2}}^{\infty} e^{-\frac{\check{a}^2}{2\sigma^2}} d\check{a} = \sigma \int_{-\frac{\varsigma^*}{2\sigma}}^{\infty} e^{-\frac{\check{a}^2}{2}} d\check{a}$ and according to (49),we have

$$\frac{1}{2} \leq \int_0^{\infty} e^{-\frac{(\check{a} - \frac{\mu_i' - \mu_{B,i}}{2})^2}{2\sigma^2}} d\check{a} \leq \int_0^{\infty} e^{-\frac{(\check{a} - \frac{\varsigma^*}{2})^2}{2\sigma^2}} d\check{a} \leq x^*. \tag{51}$$

As x^* is the solution of $x^N - (\sqrt{2\pi}\sigma - x)^N = \frac{\xi(\sqrt{2\pi}\sigma)^N}{2^N}$, we can obtain

$$\begin{aligned} (\int_0^{\infty} e^{-\frac{(\check{a} - \frac{\mu_i' - \mu_{B,i}}{2})^2}{2\sigma^2}} d\check{a})^N \\ -(\sqrt{2\pi}\sigma - \int_0^{\infty} e^{-\frac{(\check{a} - \frac{\mu_i' - \mu_{B,i}}{2})^2}{2\sigma^2}} d\check{a})^N \leq \frac{\xi(\sqrt{2\pi}\sigma)^N}{2^N}. \end{aligned} \tag{52}$$

Letting $a_i = \check{a}_i + \frac{\mu_i' + \mu_{B,i}}{2}$, it yields $\int_0^{\infty} e^{-\frac{(\check{a} - \frac{\mu_i' - \mu_{B,i}}{2})^2}{2\sigma^2}} d\check{a} = \int_{\frac{\mu_{B,i} + \mu_i'}{2}}^{\infty} e^{\frac{-(a_i - \mu_i')^2}{2\sigma^2}} da_i$. In addition, due to the symmetry of Gaussian distributions, we can obtain

$$\begin{aligned} 1 - \int_{\frac{\mu_{B,i} + \mu_i'}{2}}^{\infty} \frac{1}{\sqrt{2\pi}\sigma} e^{\frac{-(a_i - \mu_i')^2}{2\sigma^2}} da_i \\ = \int_{\frac{\mu_{B,i} + \mu_i'}{2}}^{\infty} \frac{1}{\sqrt{2\pi}\sigma} e^{\frac{-(a_i - \mu_{B,i})^2}{2\sigma^2}} da_i \end{aligned} \tag{53}$$

Thus, (52) can be rewritten as,

$$\begin{aligned} \int_{\frac{\mu_{B,1} + \mu_1'}{2}}^{\infty} \cdots \int_{\frac{\mu_{B,N} + \mu_N'}{2}}^{\infty} (e^{-\frac{\sum_{i=1}^N (a_i - \mu_i')^2}{2\sigma^2}} \\ - e^{-\frac{\sum_{i=1}^N (a_i - \mu_{B,i})^2}{2\sigma^2}}) d\mathbf{a} \leq \frac{\xi(\sqrt{2\pi}\sigma)^N}{2^N}. \end{aligned} \tag{54}$$

Rearranging (54), we can obtain

$$\begin{aligned} 2^N \cdot \int_{\frac{\mu_{B,1} + \mu_1'}{2}}^{\infty} \cdots \int_{\frac{\mu_{B,N} + \mu_N'}{2}}^{\infty} \frac{1}{(\sqrt{2\pi}\sigma)^N} (e^{-\frac{\sum_{i=1}^N (a_i - \mu_i')^2}{2\sigma^2}} \\ - e^{-\frac{\sum_{i=1}^N (a_i - \mu_{B,i})^2}{2\sigma^2}}) d\mathbf{a} \leq \xi. \end{aligned} \tag{55}$$

Then, we have

$$\frac{1}{(\sqrt{2\pi}\sigma)^N} \int |e^{-\frac{\sum_{i=1}^N (a_i - \mu_i')^2}{2\sigma^2}} - e^{-\frac{\sum_{i=1}^N (a_i - \mu_{B,i})^2}{2\sigma^2}}| d\mathbf{a} \leq \xi. \tag{56}$$

Thus, we can find that at the case that $\mu_i' \geq \mu_{B,i}, \forall i \in \{1, 2, \cdots, N\}$, if $\mu_i' - \mu_{B,i} \leq \varsigma^*, \forall i \in \{1, 2, \cdots, N\}$, then $\int_{\mathbf{a}} |\pi'(\mathbf{a}|s) - \pi_B(\mathbf{a}|s)| d\mathbf{a} \leq \xi$. Thanks to the symmetry of Gaussian distributions, other cases when $\mu_i' \leq \mu_{B,i}$ can also be easily proved. The detailed processes are omitted here to be brief. □