Efficient Task Offloading in Double Roadside RIS-Assisted Vehicular Edge Computing Networks Using Deep Reinforcement Learning

Yibin Xie, Lei Shi, *Member, IEEE*, Zhehao Li, Xu Ding, *Member, IEEE*, and Yuqi Fan

***Abstract*—The concept of vehicular edge computing (VEC) has recently been envisioned as a promising paradigm to satisfy the quality of service (QoS) requirement of delay-sensitive intelligent applications in future networks. However, the limited radio frequency communication range necessitates the dense deployment of communication infrastructures to accommodate the increasing number of connected vehicles and data traffic. This could lead to growing equipment and energy costs, hindering the full realization of the VEC system. To address such limitations, we introduce a double roadside reconfigurable intelligent surface (RIS) assisted VEC network in this paper, where RISs are deployed inside the coverage gaps between two RSUs to extend the service range. The research goal is to maximize the sum offloading efficiency by jointly optimizing offloading decisions, computation resource allocation, and phase shift vectors of RISs. Since the original problem is a challenging mixed-integer non-linear problem (MINLP), we decompose it into a top-problem for optimizing offloading destinations and phase shift vectors, and a sub-problem for optimizing offloading ratios and computation resources. We propose a deep reinforcement learning (DRL)-based algorithm for quickly obtaining near-optimal offloading decisions and phase shift vectors, alongside a Dinkelbach-based method for obtaining optimal offloading ratios and computation resource allocation. Simulation results demonstrate that our proposed algorithm achieves near-optimal performance compared to other benchmarks and enables real-time decision-making.**

***Index Terms*—Vehicular edge computing (VEC), reconfigurable intelligent surface (RIS), deep reinforcement learning, offloading efficiency.**

I. Introduction

THE upcoming sixth-generation (6G) networks are expected to become more dynamic, heterogeneous, and with an ever-increasing growth of intelligent devices. Emerging applications, such as in-vehicle services like autonomous driving and traffic prediction, have put forward more stringent requirements for network performance, including low latency, high reliability, and energy efficiency [1], [2]. Since the traditional approaches are incapable of ensuring the quality of service (QoS) of connected devices, mobile edge computing (MEC) framework has been proposed, which further inspires vehicular edge computing (VEC) paradigm [3], [4]. The VEC framework enables intelligent vehicles to offload and execute latency-sensitive tasks on nearby edge servers via vehicle-to-infrastructure (V2I) and vehicle-to-vehicle (V2V) or other wireless links, not only effectively optimizing computing resources but also satisfying QoS demands [5], [6], [7].

Normally, to fully leverage the benefits of VEC, it is crucial to deploy a dedicated access point, known as the roadside unit (RSU), in close proximity to users' locations. The RSU is able to facilitate the exchange of information between vehicles and road infrastructures and enhance vehicle services by integrating edge servers to provide computing support [8], [9]. However, the limited radio frequency (RF) communication range necessitates dense deployment of RSUs to meet the service demands of growing number of smart vehicles [10]. This raises concerns regarding service costs, including infrastructure and operating expenses, which are major limitations of current wireless communication based VEC networks [11]. Therefore, it is a stringent need to guarantee the QoS of connected devices in vehicular networks with relatively low overheads.

Fortunately, recent progress in programmable meta-materials has led to the emergence of a revolutionary technique known as the reconfigurable intelligent surface (RIS), alternatively referred to as the intelligent reflecting surface [12]. The RIS is typically composed of a large array of two-dimensional structured passive reflecting elements, with full control over the phase shift and amplitude of each element. The primary function of RIS is to reconfigure the wireless channel. To be specific, it can establish a virtual line-of-sight (LoS) link between transceivers to enhance communication quality and extend signal coverage [13]. Since the reflecting elements of RIS are passive, low-cost and easy to deploy, they can efficiently enhance the energy and spectral efficiency of wireless networks with little additional cost [14]. These benefits have prompted researchers to incorporate RIS into wireless networks, facilitating the emergence of novel

Received 21 March 2024; revised 26 December 2024; accepted 10 February 2025. Date of publication 18 February 2025; date of current version 18 July 2025. This work was supported in part by Anhui Provincial Natural Science Foundation under Grant 2308085MF212 and in part by the Joint Fund Project of Natural Science Foundation of Anhui Province under Grant 2208085US05. The review of this article was coordinated by Dr. Tao Dusit Niyato. *(Corresponding author: Lei Shi.)*

The authors are with the School of Computer Science and Information Engineering, Anhui Provincial Key Laboratory of Industry Safety and Emergency Technology, Hefei University of Technology, Hefei 230601, China (e-mail: shilei@hfut.edu.cn).

Digital Object Identifier 10.1109/TVT.2025.3543664

high-performance systems. In latest works, RIS is anticipated to effectively enhance the communication and computation performance of edge networks [15].

As illustrated in [16], [17], [18], [19], the advantages of RIS-enabled edge networks are undoubtedly striking. However, current research overlooks the potential impact on network performance when vehicles are positioned outside RSU coverage areas in VEC networks. In such cases, if the RIS-assisted edge computing network can meet vehicle offloading requests and is highly efficient, it can positively reduce the original densely deployed RSU and facilitate cost reduction. In our prior research [20], we focused on successful offloading under RIS-assisted edge computing networks and demonstrated that the RIS could maintain network performance even with coverage gaps. In this article, we further study the offloading efficiency under RIS-assisted edge computing networks and consider the case of double RIS and partial offloading.

The main contributions of this paper can be summarized below.

- We propose a double roadside RIS-enabled VEC network model, where RISs are deployed between two RSUs but beyond the coverage of either. An optimization problem is formulated to maximize the sum offloading efficiency (SOE) by jointly optimizing the offloading decision, computation resource allocation, and the RIS phase shift vectors.
- To tackle the challenging problem, we design a (deep reinforcement learning) DRL-based algorithm to first decompose the original problem into a top-problem and a sub-problem. Subsequently, the offloading destinations and phase shift vectors of RISs are first optimized by the DRL, and the offloading ratio and computation resource allocation via solving the second subproblem by the Dinkelbach algorithm.
- Simulation results demonstrate that the proposed DRL algorithm achieves a near-optimal solution, effectively enhancing the SOE compared to other schemes, and maintains low execution time even with a large number of vehicles.

The rest of this paper is organized as follows: In Section II, we survey the related work. In Section III, we establish the mathematical model of the double roadside RIS-assisted VEC network and give the corresponding problem formation. In Section IV, we design a DRL-based algorithm to get the feasible solution. In Section V, we illustrate the simulation results and corresponding analysis. In the final section, we conclude this article.

II. Related Work

Several recent studies have explored offloading optimization in RIS-assisted edge networks, primarily focusing on the single-server scenario. However, in practical settings, RSUs are often densely deployed. Moreover, many of these studies rely on alternate algorithms to address the intricate joint optimization problem. Nevertheless, these algorithms share common limitations, including high operational speed requirements and achieving only sub-optimal performance. Since the DRL-based approach is proven to be effective in optimizing wireless networks [21], it also holds great promise in solving optimization problems in RIS-assisted edge networks. In the following, we will illustrate the relevant works specifically.

A. Task Offloading in RIS-Assisted Edge Networks

In general, computation offloading problems mainly take into consideration both communication and computation, making them non-convex and with complex coupled variables. In [16], authors consider a RIS-aided multiuser MEC scenario and investigate the total energy consumption minimization problem. To deal with the non-convexity, they decouple the problem and obtain the optimal phase shifts first, while adopting the Lagrange dual method and Karush-Kuhn-Tucker conditions to optimize the CPU frequency, transmit power, and offloading time allocation. When combining RIS and other communication techniques, authors in [17] analyze the sum energy consumption in a RIS-aided multi-user MEC system, where users using non-orothology multi-access (NOMA). Since the formulated problem is non-convex, the block coordinate descent (BCD) method is utilized to alternately optimize two separated subproblems. Numerical results demonstrate the proposed scheme can increase energy efficiency and achieve performance gains. In [18], a max-min computation efficiency problem is explored under the RIS-assisted secure MEC network framework. To address the non-convexity, the Dinkelbach method and BCD algorithm are utilized to transform the original problem into several subproblems, which are then solved iteratively. Authors in [19] also investigate the secure offloading problem in RIS-assisted MEC network and present a BCD algorithm to minimize the total latency.

These studies mainly rely on alternate optimization methods to address the radio and computation resource allocation subproblems and the RIS phase shift design subproblem separately. However, such approaches often result in high computational complexity and sub-optimal performance, hindering the practical deployment of RIS-assisted edge networks.

B. Deep Reinforcement Learning Empowered Optimization in Edge Networks

DRL can solve complicated and highly coupled networking problems by adopting adaptive modeling and intelligent learning capabilities, and have low computation complexity and execution time [22], [23]. Due to these advantages, it has found widespread adoption in common edge networks.

In [24], authors consider a wireless MEC network. They design a DRL-based online algorithm DROO to obtain task offloading decisions and wireless resource allocation, instead of solving combinatorial optimization problems. Numerical results show that the proposed algorithm can achieve near-optimal performance in a very short execution time. In [25], the computation latency minimization problem in an unmanned aerial vehicle-assisted MEC network is studied. To address the non-convex and continuous action space, they propose a deep deterministic policy gradient (DDPG) method, leading to significant processing delay reduction. Similarly, [26] introduces a multi-agent DDPG algorithm to jointly optimize resource allocation and task offloading under latency and resource constraints. In VEC networks, a task offloading problem considering both delay and energy consumption is explored in [27]. Authors use fuzzy logic to identify the data traffic of vehicle networks and employ a DRL-based algorithm to tackle the offload requests effectively. To account for secure offloading, [28] adopts an Asynchronous Advantage Actor-Critic learning algorithm to minimize system energy consumption. [29] aims to maximize the quality of experience by treating mobile vehicles as edge servers collaborating with fixed edge nodes. By designing appropriate rewards, the proposed DRL-based algorithm achieves a great convergence and stability.

However, few studies have explored DRL-facilitated optimization in RIS-assisted edge networks. In [30], authors propose a DRL algorithm to maximize the total utility of users in the RIS-assisted wireless-powered MEC network. In [31], a RIS-enabled edge heterogeneous network containing two types of base stations is presented. By introducing a two-timescale mechanism, the user association is optimized by a matching algorithm, while the offloading decision, computation resource and phase shifts are optimized through DRL. In [23], the authors address system efficiency under secrecy constraints in a cooperative RIS-enabled edge network with eavesdroppers by proposing a DDPG-based algorithm. This algorithm jointly optimizes transmit and jamming beamforming matrices and phase shifts, resulting in improved secrecy rates and energy efficiency compared to benchmarks.

From above literatures, it can be seen that DRL-based algorithms for RIS-assisted edge networks are largely unexplored. Meanwhile, existing alternative optimization methods have shown limitations in terms of both performance and convergence. Given the variable complexity and the need for real-time decision-making in our scenario, efficient algorithm design is essential.

III. System Model and Problem Formation

In this section, we present the system model and corresponding problem formation. We consider a double-RIS cooperatively assisted VEC scenario as shown in Fig. 1, in which RSUs are deployed at a certain distance apart on one side of the roads. Each RSU is equipped with an MEC server to provide computing services for connected vehicles. Given the limited range of RF communication, vehicles near the cell edge of RSUs often experience severe path loss and signal fading, resulting in weak signal transmission. Therefore, these vehicles can be considered outside the RSU's effective coverage range. A high density of RSUs is typically required to ensure reliable communication and computation service in these areas, which significantly increases equipment costs. To address this challenge, we propose deploying RISs in VEC networks. Each RIS is equipped with a uniform planar array (UPA) consisting of N passive reflecting elements, which is controlled by an RIS controller. These RISs could extend the RSU service range through single-reflection and

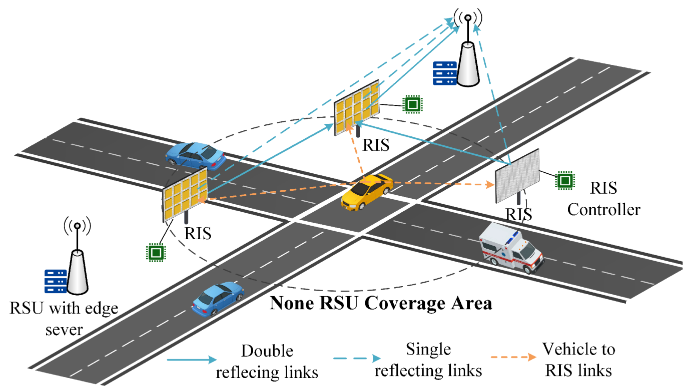


Fig. 1. Overview of the double RIS-assisted VEC network model.

double-reflection links, thereby improving service performance while reducing deployment expenses. Our main optimization objective is to maximize the sum offloading efficiency (SOE) of all vehicles by jointly optimizing offloading decisions, phase shift coefficients, and computation resources. In the following, we first illustrate the channel model and the offloading model, then give the overall formulated problem.

A. Channel Model

In this subsection, we give the illustration of related channel characteristics. First, we consider a period of time, and the time horizon can be divided into continuous time blocks, each time block has the same length. Let $\mathcal{M}$ be the set of vehicles in a time block. Then, denote $\mathcal{K} = \{k|k \in \{RSU_1, RSU_2, \ldots, RSU_K\}\}$ as the set of RSUs, and denote $\mathcal{S} = \{s|s \in \{RIS_1, RIS_2, \ldots, RIS_S\}\}$ as the set of RISs.

In the considered model, two types of links (i.e., single RIS-assisted links and double RIS-assisted links) will coexist during the task offloading process, and both are closely related to the offloading destination. For the single reflecting links, the signal from the vehicle will be reflected by either of the RISs and then directed towards the destination RSU. In the case of double RIS assisted links, the vehicle's signal will be first transmitted to a RIS situated away from the cell edge of the destination RSU, then delivered towards another RIS, and finally reflected to reach the offloading destination.

Based on the aforementioned links, the following channels are defined for each time block: (a) the channel from vehicle m to the RIS s denoted as $\mathbf{H}_{s,m} \in \mathbb{C}^{N\times 1}$, where $m \in \mathcal{M}$, and $s \in \mathcal{S}$. (b) the channel between RISs denoted as $\mathbf{D}_{s,s'} \in \mathbb{C}^{N\times N}$, where $s, s' \in \mathcal{S}$. (c) the channel from RIS s to RSU k denoted as $\mathbf{G}_{k,s} \in \mathbb{C}^{1\times N}$, where $s \in \mathcal{S}$, and $k \in \mathcal{K}$. It is assumed that the channel state information (CSI) of all these channels is perfectly known at the RSUs.

We consider the Rician fading for all three kinds of channels, and $\mathbf{H}_{s,m}$ can be modeled as

$$\mathbf{H}_{s,m} = \sqrt{\rho d_{r,m}^{-\lambda_{s,m}}} \left(\sqrt{\frac{\xi}{1+\xi}} \mathbf{H}_{s,m}^{\mathrm{LoS}} + \sqrt{\frac{1}{1+\xi}} \mathbf{H}_{s,m}^{\mathrm{NLoS}} \right), \tag{1}$$

where ρ is the path loss at the reference distance d_0, while $d_{s,m}$ and $\lambda_{s,m}$ respectively represent the distance and the corresponding path loss exponent from vehicle m to the RIS s. The vector $\mathbf{H}_{s,m}^{\mathrm{LoS}} \in \mathbb{C}^{N\times 1}$ and vector $\mathbf{H}_{s,m}^{\mathrm{NLoS}} \in \mathbb{C}^{N\times 1}$ are the line-of-sight (LoS) component and non-line-of-sight (NLoS) component, respectively. Specifically, $\mathbf{H}_{s,m}^{\mathrm{LoS}}$ consists of the array response, while $\mathbf{H}_{s,m}^{\mathrm{NLoS}}$ consists of independent components that follow $\mathcal{CN}(0,1)$. Besides, ξ is the Rician fading factor, which is the ratio of power between direct and scattered paths.

The channel from the RIS to RSU also obeys the Rician fading, thus it can be expressed as

$$\mathbf{G}_{k,s} = \sqrt{\rho d_{k,s}^{-\lambda_{k,s}}}\left(\sqrt{\frac{\xi}{1+\xi}}\mathbf{G}_{k,s}^{\mathrm{LoS}} + \sqrt{\frac{1}{1+\xi}}\mathbf{G}_{k,s}^{\mathrm{NLoS}}\right), \quad (2)$$

where $d_{k,s}$ and $\lambda_{k,s}$ respectively represent the distance and the corresponding path loss exponent from RIS s to the RSU k. The vector $\mathbf{G}_{k,s}^{\mathrm{LoS}} \in \mathbb{C}^{1\times N}$ and vector $\mathbf{G}_{k,s}^{\mathrm{NLoS}} \in \mathbb{C}^{1\times N}$ are the LoS component and NLoS component, respectively. Similarly, the channel between two RIS can be calculated as

$$\mathbf{D}_{s,s'} = \sqrt{\rho d_{s,s'}^{-\lambda_{s,s'}}}\left(\sqrt{\frac{\xi}{1+\xi}}\mathbf{D}_{s,s'}^{\mathrm{LoS}} + \sqrt{\frac{1}{1+\xi}}\mathbf{D}_{s,s'}^{\mathrm{NLoS}}\right), \quad (3)$$

All three kinds of channels $\mathbf{H}_{s,m}$, $\mathbf{G}_{k,s}$, $\mathbf{D}_{s,s'}$ are supposed to follow the block fading, i.e., channels would remain approximately constant within each time block but may change across different blocks due to the high mobility of vehicles.

Since all RISs have the same number of reflective elements N, the phase shift vector of each RIS in a time block is defined as $[\theta_{s,1}, \theta_{s,2}, \ldots, \theta_{s,N}]^T$, and the value range of each phase shift is $[0, 2\pi]$. Hence, the reflection matrix of the RIS s is given by

$$\mathbf{\Theta}_s = \mathrm{diag}(e^{j\theta_{s,1}}, e^{j\theta_{s,2}}, \ldots, e^{j\theta_{s,N}}) \in \mathbb{C}^{N\times N}, \quad (4)$$

where j denotes the imaginary unit. Note that, the amplitudes of all reflecting elements are set to 1. In practice, the phase shifts of the RIS are normally adjusted by the RIS controller using a finite number of discrete values. However, for simplicity of analysis, we assume that the phase shifts can be continuously varied, and we disregard any delays within the adjusting process.

B. Offloading Model

Since we consider partial offloading, for a vehicle's computing task, it can be executed on the RSU and locally at the same time. To indicate the offloading destination of each vehicle, we further define the binary variable α,

$$\alpha_{m,k} = \begin{cases} 1 & \text{if vehicles } m \text{ will offload its} \\ & \text{tasks to RSU } k; \\ 0 & \text{otherwise.} \end{cases} \quad (5)$$

Since we consider the double-RIS system, each vehicle is assumed to select a RIS as the second reflecting point for uplink transmission. To reduce path loss and obtain maximum beamforming gain, each vehicle is assumed to select the RIS physically closest to its ideal offloading RSU as the second reflecting point, and this selection is strongly related to offloading decisions. Meanwhile, all other available RISs can be utilized in the first reflection path. Based on the above statement, the equivalent channel from user m to the RSU k at a time block is given by

$$\begin{aligned}\bar{\mathbf{H}}_{m,k} = &\sum_{s\in\mathcal{S}} \varepsilon_{s,k}(\mathbf{G}_{s,k}\mathbf{\Theta}_s\mathbf{H}_{s,m} \\ &+ \sum_{\substack{s'\in\mathcal{S} \\ s\neq s'}}(\mathbf{G}_{s',k}\mathbf{\Theta}_{s'}\mathbf{H}_{s',m} + \mathbf{G}_{s,k}\mathbf{\Theta}_s\mathbf{D}_{s,s'}\mathbf{\Theta}_{s'}\mathbf{H}_{s',m})).\end{aligned} \quad (6)$$

The binary variable $\varepsilon_{s,k}$ indicates whether RIS s is the closest to RSU k. It equals 1 if RIS s is the closest and 0 otherwise. The first term represents the single-reflection channel via RIS s, while the second term accounts for the combined contributions of single-reflection channels through other RISs and double-reflection channels involving RIS s and other RISs.

We assume that multiple vehicles adopt the NOMA technique for simultaneous transmission, which means RSUs could utilize different frequency resources to alleviate the near-adjacent frequency interference. Therefore, the vehicles will only suffer interference from vehicles that have the same offloading destination, but will not affected by the vehicles whose offloading destination is the other RSU. Then, the signal-to-interference-plus-noise (SINR) can be calculated as:

$$\Gamma_{m,k} = \frac{p_m|\bar{\mathbf{H}}_{m,k}|^2\,\alpha_{m,k}}{\sum_{\substack{m'\in C \\ o_{m'}<o_m}} p_{m'}|\bar{\mathbf{H}}_{m',k}|^2\,\alpha_{m',k} + \sigma^2} \quad (7)$$

where p_m represents the transmit power of vehicle m, and σ^2 denotes the additive white Gaussian noise (AWGN) power. Besides, o_m denotes the successive interference cancellation (SIC) decoding order [32] of the signal of vehicle m, and $o_{m'} < o_m$ means the signal from vehicle m' will be decoded after the signal from vehicle m.

Since partial offloading is considered, we define the variable x_m as the ratio between the offloading task bits and the local execution task bits. Therefore, the transmission delay of vehicle m can be calculated as

$$T_m^{trans} = \frac{l_m x_m}{\sum_{k\in\mathcal{K}} Blog_2(1+\Gamma_{m,k})}, \quad (8)$$

where l_m represents the task bits from vehicle m to be executed within a time block and B denotes the uplink bandwidth. The computation delay at the RSU can be given by

$$T_m^{rsu} = \sum_{k\in\mathcal{K}} \frac{l_m x_m \gamma}{f_{m,k}}\alpha_{m,k}, \quad (9)$$

where γ is the number of CPU cycles needed by executing tasks per bit at RSU, and $f_{m,k}$ is the computation resource allocated for executing tasks of vehicle m. The local computing delay at vehicle m is expressed as

$$T_m^{loc} = \frac{l_m(1-x_m)\gamma'}{f'}, \quad (10)$$

where γ' represents the number of CPU cycles needed by executing per-bit tasks locally, while f' denotes the computation capacity of vehicle m, assuming the same computing capacity across all vehicles. It is important to note that local computing can perform in parallel to the uplink transmission and edge computing. Thus, the local computing latency and the total offloading latency including the uplink transmission time and the RSU computing time should be less than the task execution deadline D_m, i.e.,

$$T_m^{local} \leq D_m \tag{11}$$

$$T_m^{trans} + T_m^{rsu} \leq D_m \tag{12}$$

The energy consumption of the network mainly comes from two aspects: computing and transmission. The transmission energy overhead can be expressed as:

$$E_m^{trans} = p_m T_m^{trans}. \tag{13}$$

The computation energy cost includes both local execution and edge computing, and can be formulated as:

$$E_m^{com} = \rho l_m x_m \gamma + \rho' l_m (1 - x_m)\gamma', \tag{14}$$

where ρ and ρ' are coefficients that represent the circuit energy consumption per CPU cycle of the MEC server at the RSU and vehicle, respectively [33]. To this end, the total energy consumption of accomplished tasks can be calculated as

$$E^{total} = \sum_{m\in\mathcal{M}} (E_m^{trans} + E_m^{com}). \tag{15}$$

C. Problem Formation

The total offloaded task bits of the vehicle m during the current time block is the sum of tasks bits that executed on the RSU, i.e.,

$$L^{total} = \sum_{m\in\mathcal{M}} l_m x_m, \tag{16}$$

Therefore, the SOE that represents the ratio between sum offloading task bits and the overall network energy consumption can be expressed as

$$\mathbb{SOE} = \frac{L^{total}}{E^{total}} = \frac{\sum\limits_{m\in\mathcal{M}} l_m x_m}{\sum\limits_{m\in\mathcal{M}} (E_m^{trans} + E_m^{com})}. \tag{17}$$

By comparing (6) to (16), the $\mathbb{SOE}$ is actually a function related variable $\alpha_{m,k}$, $x_{m,k}$ and $\theta_{s,i}$.

The main objective of this work is to maximize the SOE of the entire network by jointly optimizing the set of offloading destination variables (i.e., $\boldsymbol{\alpha} = \{\alpha_{m,k} | \, m \in \mathcal{M}, k \in \mathcal{K}\}$), the set of offloading ratio (i.e., $\boldsymbol{x} = \{x_{m,k} | \, m \in \mathcal{M}, k \in \mathcal{K}\}$) RISs' phase shift vectors (i.e., $\boldsymbol{\theta} = \{\theta_{s,n} | \, s \in \mathcal{S}, n \in \{1, \ldots, N\}\}$), and the set of RSU computation resource (i.e., $\boldsymbol{f} = \{f_{m,k} | \, m \in \mathcal{M}, k \in \mathcal{K}\}$). Thus, we could establish the optimization problem as follows.

$$\mathcal{P}1: \max_{\boldsymbol{\alpha},\boldsymbol{x},\boldsymbol{\theta},\boldsymbol{f}} \mathbb{SOE}(\boldsymbol{\alpha}, \boldsymbol{x}, \boldsymbol{\theta}) \tag{18}$$

$$\text{s.t. } (1)\text{–}(10), (13)\text{–}(17) \tag{18a}$$

$$\sum_{k\in\mathcal{K}} \alpha_{m,k} \leq 1, \; \forall m \in \mathcal{M}, \tag{18b}$$

$$0 \leq x_m \leq 1, \; \forall m \in \mathcal{M}, \tag{18c}$$

$$T_m^{trans} + T_m^{rsu} \leq D_m, \; \forall m \in \mathcal{M}, \tag{18d}$$

$$T_m^{local} \leq D_m, \; \forall m \in \mathcal{M}, \tag{18e}$$

$$0 \leq \theta_{s,n} \leq 2\pi, \; \forall n \in \{1, \ldots, N\}, \forall s \in \mathcal{S}, \tag{18f}$$

$$\Gamma_{m,k} > \beta \alpha_{m,k}, \; \forall m \in \mathcal{M}, \; \forall k \in \mathcal{K}, \tag{18g}$$

$$\sum_{m\in\mathcal{M}} f_{m,k} \alpha_{m,k} \leq F_k, \; \forall k \in \mathcal{K}, \tag{18h}$$

$$0 \leq f_{m,k} \leq F_k, \; \forall k \in \mathcal{K}, \; \forall m \in \mathcal{M}. \tag{18i}$$

Among the constraints, constraint (18b) guarantees that each vehicle's task can only be transmitted to one RSU and (18c) restricts the offloading ratio. Constraints 18d) and (18e) restrict that the offload time and local computation time of each task should not exceed its latency requirement. Constraint (18f) regulates the value of each phase shifts, while constraint (18g) gives a threshold of SINR. Then, constraints (18h) and (18i) together ensure the allocation of computation resources on RSU. Specifically, constraint (18h) ensures that the overall allocated computation resource should not exceed RSU's computation capacity $F_k (k \in \mathcal{K})$, while constraint (18i) guarantees the allocated resource for each vehicle's task.

According to (7)−(17) and constraint (18d), it can be discovered that the binary variables $\boldsymbol{\alpha}$ and the continuous variables $\boldsymbol{x}, \boldsymbol{\theta}$ and $\boldsymbol{f}$ are deeply coupled with each other. So, the offloading efficiency maximization problem $\mathcal{P}1$ is a mixed-integer non-linear problem (MINLP), which is very challenging to obtain the global optimal solution in polynomial time. In the following, we will propose a DRL-based algorithm instead of alternate approaches or other optimization methods to address the formulated problem $\mathcal{P}1$.

IV. DRL-Based Joint Optimization Algorithm

In the previous section, we outline the challenges of solving problem $\mathcal{P}1$. Here, we present a feasible DRL-based solution.

We formulate the problem to maximize the SOE in any time block. Each decision within a time block only relies on the current vehicle tasks information (i.e., task bits and max delay) and the channel information related to vehicles, RSUs, and RISs, and is independent of the information and decisions from previous time blocks. By modeling these decisions as actions and representing the tasks and channel information as the state, we find that our problem has Markov properties, even future states are independent of the current action and state. It is a special case of a Markov Decision Process (MDP) and aligns with the characteristics of model-free reinforcement learning [34]. Therefore, we can model our problem as an MDP and use DRL to effectively optimize the SOE.

In our problem, the agent should be a powerful entity that communicates with and manages all vehicles and RISs. For simplicity, we assume that the agent is implemented by one of RSUs within the network. Since state transitions are independent of actions, the actor-critic framework is highly compatible. And we select the Proximal Policy Optimization (PPO) algorithm [35]

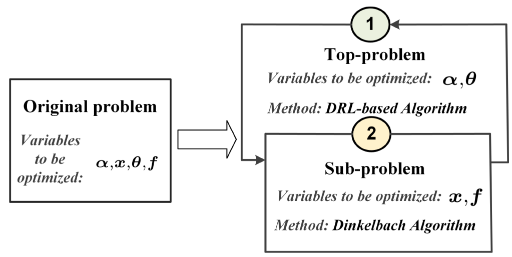


Fig. 2. The optimization structure of solving problem $\mathcal{P}1$.

to ensure stable policy updates under the hybrid action space. Using deep neural networks (DNN) to model both the actor (policy network) and the critic (value network), the agent will gradually learn informed policies.

However, due to numerous continuous and binary variables in $\mathcal{P}1$, optimizing them simultaneously by DRL would bring significantly large action and state spaces, leading to prolonged training time and uncertain training outcomes. We propose to use DRL to gradually learn the policy from exploration experience to obtain some near-optimal variables, and then optimize the remaining variables based on this. Below we give a specific description.

A. Algorithm Overview

In this subsection, we give the outline of the proposed phase shift vectors, offloading destination, offloading ratio as well as the resource allocation (PODORR) optimization algorithm. The non-linear and non-convexity of problem $\mathcal{P}1$ mainly comes from the coupled variables $\boldsymbol{\alpha}$ and $\boldsymbol{\theta}$ in (6) and (7) during the transmission session, and $\boldsymbol{\alpha}$, $\boldsymbol{x}$, and $\boldsymbol{f}$ in (8)–(11) during computation session. It can be seen that the offloading destination variable $\boldsymbol{\alpha}$ greatly influences both two sessions. Nevertheless, once the variable $\boldsymbol{\alpha}$ and $\boldsymbol{\theta}$ can be fixed, the original problem $\mathcal{P}1$ can be transformed to a sub-problem as

$$\mathcal{P}2 : \max_{\boldsymbol{x},\boldsymbol{f}} \ \mathbb{SOE}(\boldsymbol{x}) \tag{19}$$

$$\text{s.t. } (1)\text{–}(10), (13)\text{–}(17), \tag{19a}$$

$$(18c), (18d), (18e), (18h), (18i). \tag{19b}$$

According to (16), the denominator of the objective function $\mathbb{SOE}(\boldsymbol{x})$ in $\mathcal{P}2$ is an affine function about $\boldsymbol{x}$. Under given $\boldsymbol{\alpha}$ and $\boldsymbol{\theta}$, the numerator of $\mathbb{SOE}(\boldsymbol{x})$ is also an affine function about $\boldsymbol{x}$, which is

$$E^{total} = \sum_{m\in\mathcal{M}} \Bigg(p_m \frac{l_m x_m}{\sum_{k\in\mathcal{K}} Blog_2(1+\Gamma_{m,k})} + \rho l_m x_m \gamma + \rho' l_m (1-x_m)\gamma' \Bigg), \tag{20}$$

where $\Gamma_{m,k}$ is fixed. Thus, problem $\mathcal{P}2$ is a fraction optimization problem with convex objective function and can be solved optimally by exploiting Dinkelbach's iterative algorithm [36], [37] as long as $\boldsymbol{\alpha}$ and $\boldsymbol{\theta}$ are given.

Based on the above analysis, as shown in Fig. 2, we propose a layer-based structure for solving the offloading efficiency maximization problem:

- *Top-problem:* Optimize offloading destination $\boldsymbol{\alpha}$ and phase shift vector $\boldsymbol{\theta}$ through a DRL-based algorithm. In this part, a proximal policy optimization (PPO)-based DRL algorithm is proposed to autonomously learn the near-optimal value of $\boldsymbol{\alpha}$ and $\boldsymbol{\theta}$.
- *Sub-problem:* Optimize offloading ratio $\boldsymbol{x}$ and computation resource $\boldsymbol{f}$ through Dinkelbach algorithm. During each time step, when $\boldsymbol{\alpha}$ and $\boldsymbol{\theta}$ are fixed, the sub-problem will be solved to obtain the optimal variables.

The solution structure is presented in Fig. 2, and we will describe the details of this proposed solution in subSection IV-B and IV-C.

B. DRL-Based Algorithm for Phase Shift Vectors and Offloading Destinations Optimization

To tackle the top-problem, we propose a DRL-based algorithm that allows the agent to learn the offloading destination $\boldsymbol{\alpha}$ and phase shifts $\boldsymbol{\theta}$ without any prior knowledge. Cause the variable $\boldsymbol{\alpha}$ are discrete and variable $\boldsymbol{\theta}$ are continuous, the action space is hybrid. Therefore, we convert the action space into a uniformed continuous action space and employ the PPO framework to optimize both $\boldsymbol{\alpha}$ and $\boldsymbol{\theta}$.

As depicted in Fig. 3, PPO is a policy iteration reinforcement learning algorithm based on the actor and critic networks [38]. By interacting with the environment, the actor generates actions based on the state space. The critic evaluates the expected rewards of these actions and provides feedback through value estimates and advantage scores, enabling the actor to adjust its policy. Through iterative updates of both networks, the agent progressively learns good decisions. The following sections define the primary components of the DRL algorithm.

- *Observation state:* In each time step, the observation space will contain the channel gains and the task information including the amount of task bits and the restricted execution latency. Thus, the state vector consists of the three types of channel gain $\{|H_{s,m}|^2|s\in\mathcal{S}, m\in\mathcal{M}\}$, $\{|G_{k,s}|^2|k\in\mathcal{K}, s\in\mathcal{S}\}$, $\{|D_{s,s'}|^2|s,s'\in\mathcal{S}, s\neq s'\}$ and the task set $\{l_m|\, m\in\mathcal{M}\}$ and the latency requirement set $\{D_m|\, m\in\mathcal{M}\}$.
- *Action space:* The agent at each time step will select the appropriate offloading destination $\boldsymbol{\alpha}$ and the reflecting phase shift vectors $\boldsymbol{\theta}$. These two actions form a hybrid action space with both continuous and discrete actions. To convert it into a unified homogeneous action space, we conduct the following transformation for the binary variable $\boldsymbol{\alpha}$.

$$\alpha_{m,k} = \begin{cases} 1 & \text{if the ouput of DNN is greater than 0,} \\ 0 & \text{otherwise.} \end{cases} \tag{21}$$

- *Reward design:* In DRL, the design and setting of the reward function directly affect the agent's learning and decision-making process of behavior. Considering that the

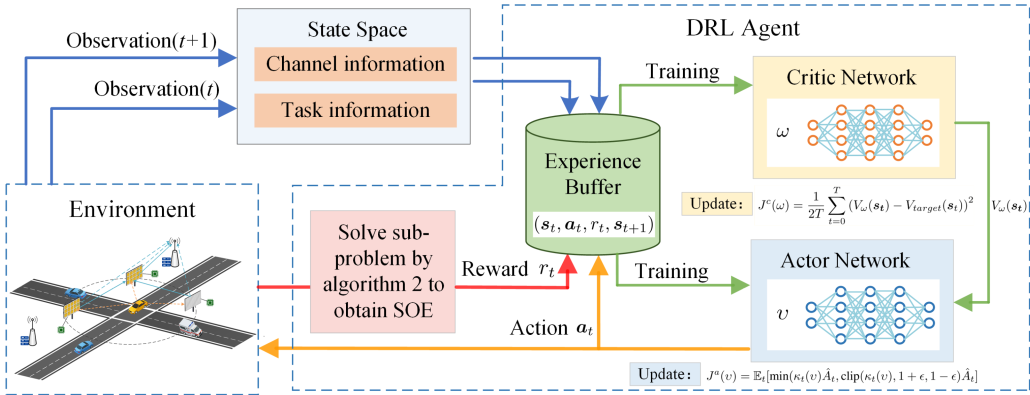


Fig. 3. The DRL framework of the proposed algorithm.

optimization objective is to maximize the SOE, with constraints (18b)–(18g), the design needs to comprehensively consider both aspects. Therefore, the instant reward function can be expressed as:

$$r = \frac{L^{total}}{E^{total}}, \tag{22}$$

where L^{total} and E^{total} denote the offloading bits and the total energy consumption at any time step respectively.

Exploration policy: We consider a hybrid action space comprising multi-dimensional continuous and discrete variables. By converting discrete variables to continuous ones by (21), the action space becomes fully continuous. In DRL, multi-dimensional Gaussian distributions are commonly used to determine continuous actions, with the actor network outputting the means and variances of these distributions. The corresponding actions are then obtained by sampling from these distributions. However, randomly initialized DNN parameters can lead to poor early-stage performance. To mitigate this, we propose a strategy to expand action candidates, thus enhancing the exploration ability. First, we add the stochastic noise on the output of DNN and executing multiple sample and could get the J candidate action values at time step t, i.e.,

$$\hat{\boldsymbol{a}}_t(J) = \{\hat{\boldsymbol{a}}_{t,1}, \ldots, \hat{\boldsymbol{a}}_{t,J}\} \tag{23}$$

When the variable value of each candidate action does not satisfy the constraint (18c) and (18d), the corresponding value will be replaced by a relative smaller or larger value. For each candidate, it will be selected to solve sub-problem (19) to get the SOE $\mathbb{SOE}(\boldsymbol{s}_t, \hat{\boldsymbol{\alpha}}_{t,j})$. Then, the best $\boldsymbol{\alpha}_t$ could obtained by

$$\boldsymbol{a}_t = \underset{j=\{1,\ldots,J\}}{\arg\max}\, \mathbb{SOE}(\boldsymbol{s}_t, \hat{\boldsymbol{a}}_{t,j}). \tag{24}$$

We select $\boldsymbol{\alpha}_t$ and store the corresponding trajectory $(\boldsymbol{s}_t, \boldsymbol{a}_t, r_t, \boldsymbol{s}_{t+1})$ in the experience replay buffer $\mathcal{D}$ to provide training samples. When the experience buffer reaches its storage limit, we follow the first input first output principle to replace the oldest trajectory stored in memory with a new trajectory.

Update policy: At each episode, we randomly select a batch of sample trajectories from the reply memory to train both critic and actor networks. Specifically, the network parameters of actor network can be adjusted along the direction of the gradient of policy loss $J^a(\upsilon)$, which is defined as

$$J^a(\upsilon) = \mathbb{E}_t[\min(\kappa_t(\upsilon)\hat{A}_t, \text{clip}(\kappa_t(\upsilon), 1+\epsilon, 1-\epsilon)\hat{A}_t], \tag{25}$$

where $\kappa_t(\upsilon) = \frac{\boldsymbol{\pi}_\upsilon(\boldsymbol{a}_t|\boldsymbol{s}_t)}{\boldsymbol{\pi}_{\upsilon'}(\boldsymbol{a}_t|\boldsymbol{s}_t)}$ denotes differences between new policy $\boldsymbol{\pi}_\upsilon(\boldsymbol{a}_t|\boldsymbol{s}_t)$ and old policy $\boldsymbol{\pi}_{\upsilon'}(\boldsymbol{a}_t|\boldsymbol{s}_t)$ during the update process, and clip$(*)$ is a clip function that restricts the value of $\kappa(\upsilon)$. $\hat{A}_t$ is the advantage function indicating the difference between the value of the action-value under current state $\boldsymbol{s}_t$ and the average value, which can be estimated by

$$\hat{A}_t = \hat{Q}^{\boldsymbol{\pi}_{\upsilon'}}(\boldsymbol{s}_t, \boldsymbol{a}_t) - V_\omega(\boldsymbol{s_t}). \tag{26}$$

The item $\hat{Q}^{\boldsymbol{\pi}_{\upsilon'}}(\boldsymbol{s}_t, \boldsymbol{a}_t)$ represents the action-value function and could be estimated by Monte Carlo methods. $V_\omega(\boldsymbol{s_t})$ is the estimated average value under state $\boldsymbol{s}_t$ which is the output of critic network.

After collecting a batch of samples, the critic network utilizes the mean square error (MSE) loss for training and updating, which can be defined as

$$J^c(\omega) = \frac{1}{2T}\sum_{t=0}^{T}\left(V_\omega(\boldsymbol{s_t}) - V_{target}(\boldsymbol{s}_t)\right)^2, \tag{27}$$

where $V_\omega(\boldsymbol{s_t})$ is the value function of state $\boldsymbol{s}_t$, computing by the value network, while $V_{target}(\boldsymbol{s}_t)$ is the target value function that stands for the returns. Furthermore, the target value function that is computed by current policy and value function which can be given as

$$V_{target}(\boldsymbol{s}_t) = \sum_{q=t}^{T-1}\lambda^{q-t}r_q + \lambda^{T-t}V_\omega(\boldsymbol{s}_T), \tag{28}$$

where T is the overall time step length in each episode, t is the current time step. Besides, r_q represents the instant reward at time step $q \in (t, T-1)$, and λ is the discount factor.

Finally, the DRL-based PODORR algorithm for phase shift vectors and offloading destinations optimization is summarized in Algorithm 1.

Algorithm 1: The DRL-based PODORR Algorithm.

Require: $\mathcal{M}$: The set of vehicles in a time block, $\mathcal{K}$: The set of RSUs, $\mathcal{S}$: The set of RISs, l_m: The set of task bits, D_m: The set of offloading deadlines.
Ensure: Optimal phase shift vectors $\boldsymbol{\theta}^*$, offloading ratio $\boldsymbol{x}^*$, offloading destination $\boldsymbol{\alpha}^*$ and computation resource $\boldsymbol{f}^*$.
Set the maximum training step, the memory size, the batch size T, and the training interval δ.
Initialize the actor network and the critic network.
Initialize state $\boldsymbol{s_1}$ and the exploration times n.
for $step = 1, \ldots, MAXSTEP$ **do**
  Sample a set of candidate actions $\hat{\boldsymbol{a}}_t$ according to policy $\boldsymbol{\pi}_\upsilon$.
  Solve sub-problem $\mathcal{P}2$ according to **Algorithm 2** for each candidate and obtain the optimal $\boldsymbol{x}$ and $\boldsymbol{f}$.
  Select the best $\boldsymbol{a}_t = \arg\max_{j=\{1,\ldots,J\}} \mathbb{SOE}(\boldsymbol{s}_t, \hat{\boldsymbol{a}}_{t,j})$.
  Execute $\boldsymbol{a}_t$ and obtain new state $\boldsymbol{s}_{t+1}$ as well as reward r_t.
  Update the replay buffer $\mathcal{D}$ by adding the sample $(\boldsymbol{s}_t, \boldsymbol{a}_t, r_t, \boldsymbol{s}_{t+1})$.
  Update $\boldsymbol{s}_t \leftarrow \boldsymbol{s}_{t+1}$.
  if $step$ mod $\delta = 0$ **then**
    Randomly sample $((\boldsymbol{s}_i, \boldsymbol{a}_i, r_i, \boldsymbol{s}_{i+1}))_{i=0:T-1}$ from $\mathcal{D}$.
    Update υ by computing $\nabla_\upsilon J^a(\upsilon)$ according to (25).
    Update ω by computing $\nabla_\omega J^c(\omega)$ according to (27).
  end if
end for

C. Dinkelbach-Based Algorithm for Offloading Ratio Optimization and Computation Resource Allocation

As long as the actions are sampled during each time step, the offloading destination $\boldsymbol{\alpha}$, and the phase shift vector $\boldsymbol{\theta}$ is given, so the original problem $\mathcal{P}1$ can be transformed into the offloading ratio and resource allocation problem $\mathcal{P}2$. However, it is noted that due to the non-linear constraint (18 d), problem $\mathcal{P}2$ is still non-convex. To tackle this non-convexity, we first introduce a slack variable $\boldsymbol{\tau} = \{\tau_m | \, m \in \mathcal{M}\}$ to decouple constraint (18 d). Therefore, the sub-problem $\mathcal{P}2$ can be rewritten as

$$\mathcal{P}3: \max_{\boldsymbol{x},\boldsymbol{f},\boldsymbol{\tau}} \ \mathbb{SOE}(\boldsymbol{x}) \tag{29}$$

$$\text{s.t. } (1)\text{–}(10), (13)\text{–}(17), \tag{29a}$$

$$(18c), (18e), (18h), (18i), \tag{29b}$$

$$\frac{l_m x_m}{\sum\limits_{k\in\mathcal{K}} Blog_2(1+\Gamma_{m,k})} \leq \tau_m, \forall m \in \mathcal{M}, \tag{29c}$$

$$\frac{l_m x_m}{f_m} \leq D_m - \tau_m, \forall m \in \mathcal{M}, \tag{29d}$$

$$0 \leq \tau_m \leq D_m, \forall m \in \mathcal{M}. \tag{29e}$$

Algorithm 2: The Dinkelbach-based ORR Algorithm.

Require: $\mathcal{M}$: The set of vehicles in a time block, $\mathcal{K}$: The set of RSUs, l_m: The set of tasks, D_m: The set of offloading deadlines, $\boldsymbol{\alpha}$: The set of offloading destinations, $\boldsymbol{\theta}$: The set of phase shift vectors.
Ensure: Optimal offloading ratio $\boldsymbol{x}^*$ and computation resource $\boldsymbol{f}^*$.
Initialize max iteration times, error tolerance $e \in (0, 1)$, and $\phi^{(0)} = 0$.
while $\psi^{(i)} > e$ **do**
  Calculate $\psi^{(i)} = \sum_{m\in\mathcal{M}} l_m x_m^{(i)} - \phi^{(i)} \sum_{m\in\mathcal{M}} (E_m^{trans\,(i)} + E_m^{com\,(i)})$.
  Let $(\boldsymbol{x}^*, \boldsymbol{f}^*) = (\boldsymbol{x}^{(i)}, \boldsymbol{f}^{(i)})$.
  Update $\phi^{(i)} = \frac{\sum_{m\in\mathcal{M}} l_m x_m^{(i)}}{\sum_{m\in\mathcal{M}} (E_m^{trans\,(i)} + E_m^{com\,(i)})}$.
  Update $i = i + 1$.
end while
Obtain optimal $\boldsymbol{x}^*$ and $\boldsymbol{f}^*$.

Note that the constraint (29d) is still non-convex due to the coupled variable $\boldsymbol{x}$, $\boldsymbol{f}$ and $\boldsymbol{\tau}$. We further introduce a slack variable $\boldsymbol{u} = \{u_m | m \in \mathcal{M}\}$ to decouple (29d), which is

$$u_m = \tau_m f_m, \forall m \in \mathcal{M}. \tag{30}$$

Note that, the value range of u_m is $[0, \sum_{k\in\mathcal{K}} F_k \alpha_{m,k}]$. Moreover, by replacing the objective function with the auxiliary function $\phi(\phi \geq 0)$, the problem $\mathcal{P}2$ can be finally reformulated as

$$\mathcal{P}4: \max_{\boldsymbol{x},\boldsymbol{f},\boldsymbol{\tau},\boldsymbol{u}} \sum_{m\in\mathcal{M}} l_m x_m - \phi \sum_{m\in\mathcal{M}} (E_m^{trans} + E_m^{com}) \tag{31}$$

$$\text{s.t. } (1)\text{–}(10), (13)\text{–}(17), (30), \tag{31a}$$

$$(18c), (18e), (18h), (18i), (29c), (29e) \tag{31b}$$

$$l_m x_m \leq D_m f_m - u_m, \forall m \in \mathcal{M}, \tag{31c}$$

$$0 \leq u_m \leq F_k \alpha_{m,k}, \forall m \in \mathcal{M}, k \in \mathcal{K}. \tag{31d}$$

To this end, the objective function of problem $\mathcal{P}4$ is convex and all constraints are verified to be convex as well. In turn, the Dinkelbach algorithm can be utilized to solve P4 iteratively until ϕ achieves convergence. The detailed process is outlined in Algorithm 2.

D. Computational Complexity Analysis

In this section, we analyze the complexity of our proposed algorithm. The variable $\boldsymbol{\alpha}$ and $\boldsymbol{\theta}$ are obtained by the actor network (i.e., the DNN), which operates as a single inference process with a complexity $\mathcal{O}(|\mathcal{M}| + |\mathcal{S}| \cdot N)$. While given $\boldsymbol{\alpha}$ and $\boldsymbol{\theta}$, the optimal $\boldsymbol{x}$ and $\boldsymbol{f}$ are then computed through Dinkelbach-based Algorithm 2. Since the Dinkelbach algorithm leverages Newton's method, each iteration has a complexity of $\mathcal{O}((|\mathcal{M}| \cdot (|\mathcal{K}| + 1))^3$. The number of iteration times depends on the accuracy, represented by the error tolerance e, which scales as $log(\frac{1}{e})$. Therefore, the corresponding complexity of Algorithm 2 is give by $\mathcal{O}((|\mathcal{M}| \cdot (|\mathcal{K}| + 1))^3 \cdot log(\frac{1}{e}))$. The overall

TABLE I
ENVIRONMENT SIMULATION PARAMETERS

Symbol	Network Parameter	Value
l_m	Task data amount for each vehicle	[0.5, 0.8] Mbits
D_m	Delay constraint of each task	[0.8, 1] s
F_k	The computing capability of two edge servers	10 GHz, 8 GHz
f'	The computing capability per vehicle	0.5 GHz
C	Number of CPU-cycles needed by executing tasks per bit	1000 cycles/bit
ρ	Consumed energy per CPU cycle of servers	10^{-11} J/cycle
ρ'	Consumed energy per CPU cycle of vehicles	10^{-10} J/cycle
p_m	The transmission power per vehicle	200 mW
B	Uplink bandwidth per vehicle	10 MHz
β	The SINR threshold	0.1
σ^2	The Gaussian white noise power	-70 dBm

TABLE II
THE DRL TRAINING RELATED PARAMETERS

Parameter	Value	Parameter	Value
Parameter of clip function	0.2	Discount factor	0.95
Maximum training step	2e4	Learning rate of actor network	1e-3
Training times per episode	10	Learning rate of critic network	5e-3

complexity of the proposed algorithm is $\mathcal{O}((|\mathcal{M}| \cdot (|\mathcal{K}|+1))^3 \cdot log(\frac{1}{e}) + |\mathcal{M}| + |\mathcal{S}| \cdot N) \approx \mathcal{O}((|\mathcal{M}| \cdot (|\mathcal{K}|+1))^3 \cdot log(\frac{1}{e}))$.

V. SIMULATION RESULTS

In this section, simulation results are presented to verify the convergence performance and effectiveness of the proposed algorithm. We consider a VEC network scenario with a two-way road, where two RSUs and RISs are placed on opposite sides. The road consists of four lanes, with two lanes in each direction, each 4.5 meters wide. Vehicles follow a spatial Poisson process, with vehicle density depending on speed. The average distance between vehicles is 2.5v, where v is the speed in m/s [32]. Specifically, under a three-dimensional Cartesian coordinate system, the two RSUs are positioned at (20, -300, 15) and (20, 300, 15), and the two RISs are located at (0, -10, 15) and (0, 10, 15). Each RSU has a coverage radius of 200 m, leading to a non-coverage area with 200 m between them. Vehicles are distributed within this area, with x-axis coordinates from 0 to 18, and fixed z-axis coordinates of 0. Besides, the path loss at the reference distance $d_0 = 1$m is configured as -20 dB, and the Rician factor is configured as 5. The path loss exponent for the RIS-RSU link is set to 3, while it is set to 2.2 for the RIS-RIS link and the vehicle-RIS link [39]. Other simulation parameters are shown in Table I.

Our algorithm employs two fully connected DNNs, each with an input layer, two hidden layers of 128 neurons, and an output layer. Table II summarizes the related key hyperparameters.

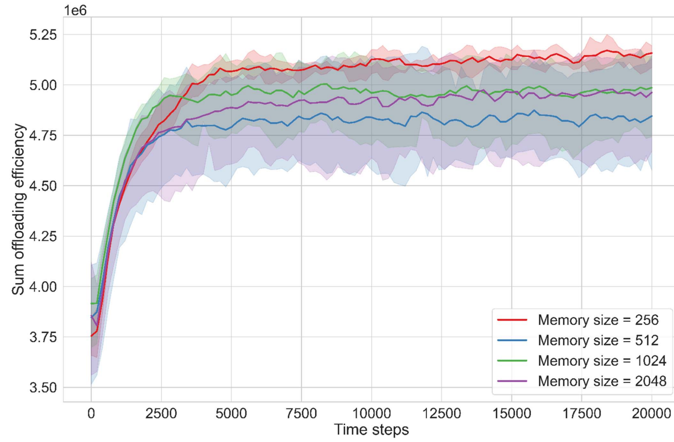


Fig. 4. The convergence performance under different memory size.

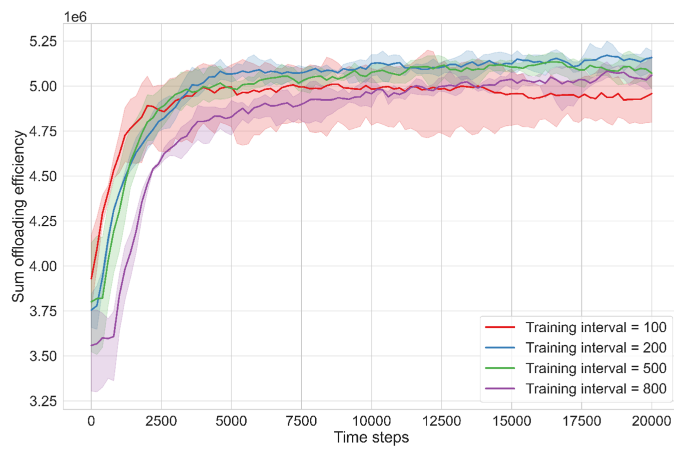


Fig. 5. The convergence performance under different training interval.

A. Convergence Performance

In this section, we evaluate the impact of various training parameters on PODORA's convergence performance, including different memory sizes, training intervals, and batch sizes. We first consider the system with 20 connected vehicles with an unified speed of 60 km/h, the number of RSU and RIS are both 2. The non-coverage area is set to 200 m, while the number of reflecting elements per RIS is 50. The reward is recorded every 200 time steps during training. To show convergence more clearly, we smooth the reward using a weighted average, with 0.75 weight for historical data and 0.25 for current data. The standard deviation from three training runs is shown as the shaded area in Figs. 4 –6, illustrating the stability of the results.

As depicted in Fig. 4, a small memory size (e.g., 256) leads to a faster convergence and achieves the highest offloading efficiency result among other memory size settings. Conversely, with a larger memory size (e.g., 512 to 2048), it takes longer for old samples to be replaced, resulting in many outdated samples in memory. Before the convergence, these older samples may not be as effective as recently obtained samples. Therefore, the algorithm will produce relatively lower and very fluctuating results in terms of SOE. Below we set the reply memory size to 256.

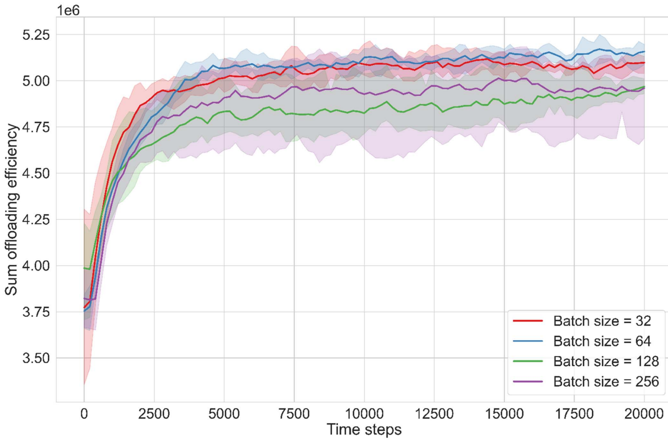


Fig. 6. The convergence performance under different batch size.

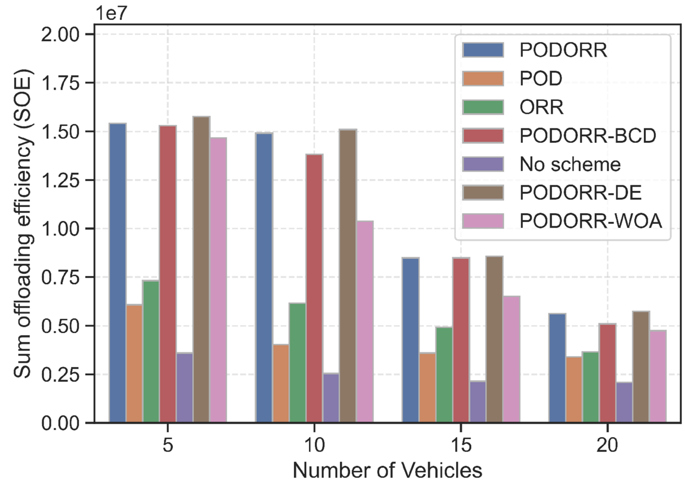


Fig. 7. The SOE versus different number of vehicles.

Fig. 5 illustrates the convergence performance for different training intervals, each interval comprising a fixed number of time blocks. Clearly, smaller training intervals (e.g., 100 or 200) yield more frequent updates to the actor and critic networks, accelerating convergence. As the training interval increases, the update frequency of the DNNs decreases, resulting in a relatively slower convergence speed. Moreover, when the training interval is set to 800, not only the convergence speed is slow, but also leads to low algorithm performance in terms of offloading efficiency. In contrast, we can see that the smallest training interval also leads to slightly low offloading efficiency performance and a certain level of fluctuation. Therefore, a medium training interval of 200 achieves a good balance between algorithm performance and convergence speed.

Fig. 6 shows the convergences under different batch sizes, with a training interval of 200 and a memory size of 256. For small batch size (e.g., 32), only a small batch of samples in memory is used for training, resulting in a relatively poor algorithm performance of offloading efficiency. But for a large batch size (e.g., 128 and 256), many old samples are used for training, leading to a certain level of fluctuation and causing slow convergence. Conversely, a medium batch size (e.g., 64) strikes a better balance on the training effectiveness and stability. Therefore, we set the batch size to 64.

B. Evaluation of Sum Offloading Efficiency

Next, we study the optimization performance of our PODORR algorithm with other benchmarks and under different network scenario settings. Specifically, the following six methods are regarded as the benchmarks.

- *PODORR-BCD:* This algorithm leverages the BCD method [18] to iteratively optimize all variables. The original problem (18) is divided into two subproblems: one for optimizing $\boldsymbol{\alpha}$, $\boldsymbol{x}$, and $\boldsymbol{f}$, and another for optimizing $\boldsymbol{\theta}$. A many-to-one matching method [30] is used to optimize $\boldsymbol{\alpha}$, while $\boldsymbol{x}$ and $\boldsymbol{f}$ are refined through Algorithm 2 during the matching process. Subsequently, an iterative algorithm [40] optimizes $\boldsymbol{\theta}$ based on the updated variables. This process repeats until a stopping criterion is met.
- *PODORR-DE:* This method utilizes the famous differential evolution (DE) algorithm [41] to optimize all variables together.
- *PODORR-WOA:* This approach utilizes one of the latest intelligent optimization methods, the whale optimization algorithm (WOA) [42] to optimize all variables.
- *POD:* This algorithm is a DRL-based algorithm that is similar to the PODORR algorithm when optimizing $\boldsymbol{\alpha}$ and $\boldsymbol{\theta}$, but without adopting Algorithm 2.
- *ORR:* This method only optimizes variables $\boldsymbol{x}$ and $\boldsymbol{f}$ according to Algorithm 2, but let offloading destinations and phase shift vectors be randomly selected.
- *No scheme:* This scheme means all variables are randomly selected without adopting any of optimization method.

Next, we evaluate the SOE performance of proposed algorithm with the mentioned benchmarks. Fig. 7 compares the SOE achieved by various algorithms with different number of vehicles at 60 km/h. As the number of vehicles increases, the SOE decreases due to higher energy costs. The PODORR algorithm consistently delivers near-optimal performance, matching PODORR-DE and PODORR-BCD, and outperforming the WOA method, which struggles to find optimal solutions due to its simplified search performance. In contrast, the remaining three methods lag far behind these algorithms.

Fig. 8 explores the impact of vehicle speed on SOE with 10 vehicles. In this simulation, the speed varies from 60 to 120 km/h, while two RISs with 80 reflecting elements are deployed in a 200 m non-coverage area. The results show that the offloading efficiency generally decreases with increasing speed. Notably, our algorithm experiences a 16.01% performance reduction at 120 km/h compared to 60 km/h. This occurs because higher speeds increase the spacing and affect the distribution of vehicles, thereby raising the path loss [32] of the vehicle-RIS link.

In Fig. 9, we compare the SOE achieved by different algorithms for different sizes of non-coverage areas, by keeping 10 vehicles at 60 km/h under a two-RIS system with 80 reflection elements. The non-coverage area size is adjusted by modifying the y-axis coordinates of the two RSUs, setting them to (−225,

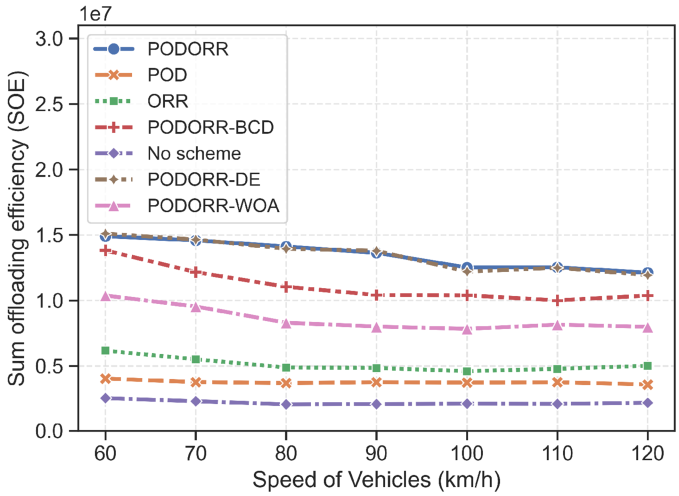


Fig. 8. The SOE versus different vehicle speed.

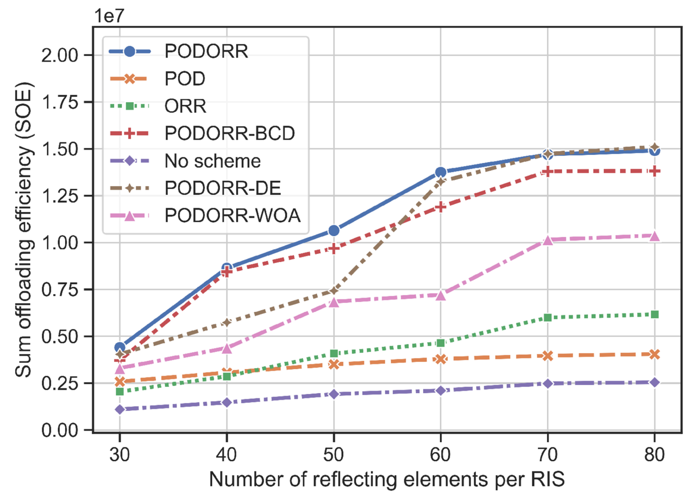


Fig. 10. The SOE versus different number of reflecting elements per RIS.

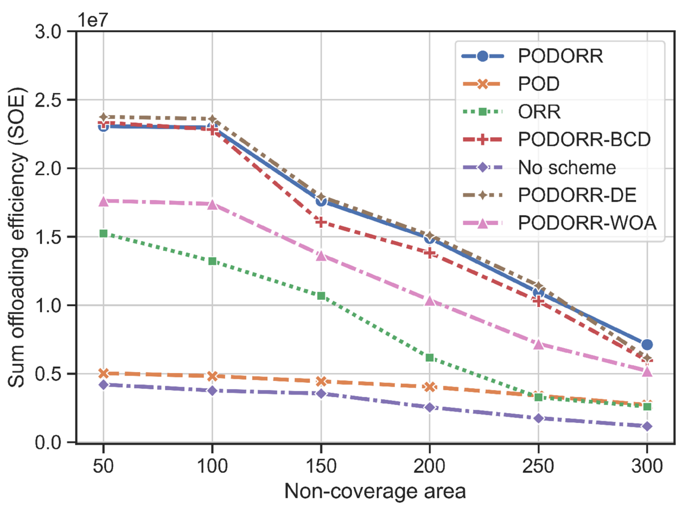


Fig. 9. The SOE versus different size of Non-coverage area.

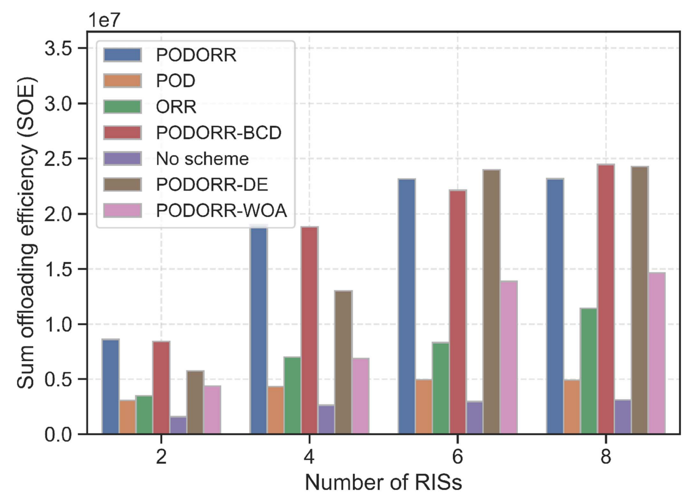


Fig. 11. The SOE versus different number of RISs.

−350) and (225, 350), respectively. Initially, the PODORR-BCD, PODORR-DE, and PODORR-BCD algorithms exhibit similar performance and can maintain the maximum SOE under a 100 m uncovered area. However, when the distance exceeds 100 m, the SOE of all algorithms tends to decrease.

Fig. 10 studies the SOE of different algorithms under different numbers of reflecting elements in a two RIS system, where 10 vehicles travel at a speed of 60 km/h in a 200 m non-coverage area. As the number of reflecting elements increases, all algorithms show significant initial performance gains, which reach a stable level after 60 elements. Notably, PODORR outperforms PODORR-DE when the element number is low (30 to 50), which may be due to the limited solution space under harsh channel conditions, where our algorithms show better capabilities. However, PODORR-DE approaches PODORR's performance when the element number is over 60, benefiting from expanded solution spaces with a large number of RIS elements. The BCD algorithm lags behind in such conditions, achieving nearly 10% lower performance than PODORR. Moreover, the POD algorithm consistently surpasses the ORR method, highlighting the importance of optimizing phase shifts and offloading destinations.

In Fig. 11, we compare the SOE performance of each algorithm with different numbers of RIS. The experiment uses 10 vehicles at 60 km/h, a 200 m non-coverage area, and a set of 2 to 8 RISs, each with 40 reflectors, deployed at 10-meter intervals. As the figure shows, SOE improves as the number of RISs increases but stabilizes at 6 RISs. Notably, deploying 6 RISs provides a 168% performance gain compared to 2 and 4 RISs in our algorithm.

C. Evaluation of Execution Latency

In Table III, we evaluate the execution latency of the PODORR algorithm compared to five other methods under different numbers of vehicles and RISs. For experiments varying the number of vehicles, we use two RISs with 80 reflection elements each. For experiments varying the number of RISs, we use 10 vehicles and 40 elements. The executions are performed on an AMD R7-5800H CPU. As seen in the table, compared with the DE,

TABLE III
EXECUTION LATENCY UNDER DIFFERENT ALGORITHMS

Number of vehicles	PODORR	PODORR-BCD	PODORR-DE	PODORR-WOA	POD	ORR
5	7.98e-2s	225.80s	30.77s	18.77s	7.81e-2s	7.45e-2s
10	8.04e-2s	280.41s	57.31s	24.28s	7.68e-2s	7.76e-2s
15	8.66e-2s	350.01s	80.46s	43.25s	8.60e-2s	8.01e-2s
20	9.02e-2s	433.20s	91.68s	56.89s	8.66e-2s	7.97e-2s
Number of RISs	**PODORR**	**PODORR-BCD**	**PODORR-DE**	**PODORR-WOA**	**POD**	**ORR**
2	7.83e-2s	100.44s	30.53s	8.85s	7.21e-2s	6.08e-2s
4	8.21e-2s	141.45s	45.62s	18.05s	7.91e-2s	6.63e-2s
6	9.04e-2s	183.83s	173.97s	30.33s	8.01e-2s	6.83e-2s
8	9.53e-2s	239.77s	261.84s	42.49s	8.66e-2s	6.94e-2s

WOA and BCD, our algorithm reduces the execution latency by up to 4 orders of magnitude. Thus, the PODORR algorithm proposed in this paper can not only effectively improve the SOE but also has low execution time.

VI. Conclusion

In this paper, we have proposed a deep reinforcement learning-based algorithm PODORR to maximize the SOE in double RIS-assisted VEC networks with partial computation offloading. The algorithm learns from past experiences to improve its offloading destinations and phase shift vectors via the proximal policy optimization framework. Subsequently, the Dinkelbach algorithm is utilized to optimize the offloading ratio and computation resources. Compared to other benchmarks, the proposed PODORR algorithm achieves near-optimal performance and reduces the CPU execution latency by up to 4 times of magnitude. We believe this work can help realize real-time and efficient offload optimization for RIS-assisted VEC networks, thereby having a positive impact on the reduction of densely deployed RF communication infrastructures and advancing the implementation of future VEC networks.

References

[1] S. Munawar, Z. Ali, M. Waqas, S. Tu, S. A. Hassan, and G. Abbas, "Cooperative computational offloading in mobile edge computing for vehicles: A model-based DNN approach," *IEEE Trans. Veh. Technol.*, vol. 72, no. 3, pp. 3376–3391, Mar. 2023.

[2] H. M. F. Noman et al., "Machine learning empowered emerging wireless networks in 6G: Recent advancements, challenges and future trends," *IEEE Access*, vol. 11, pp. 83017–83051, 2023.

[3] R. Meneguette, R. De Grande, J. Ueyama, G. P. R. Filho, and E. Madeira, "Vehicular edge computing: Architecture, resource management, security, and challenges," *ACM Comput. Surv.*, vol. 55, no. 1, pp. 1–46, 2021.

[4] L. Liu, C. Chen, Q. Pei, S. Maharjan, and Y. Zhang, "Vehicular edge computing and networking: A survey," *Mobile Networks Appl.*, vol. 26, pp. 1145–1168, 2021.

[5] J. Zhang, H. Guo, J. Liu, and Y. Zhang, "Task offloading in vehicular edge computing networks: A load-balancing solution," *IEEE Trans. Veh. Technol.*, vol. 69, no. 2, pp. 2092–2104, Feb. 2020.

[6] H. Lin, S. Zeadally, Z. Chen, H. Labiod, and L. Wang, "A survey on computation offloading modeling for edge computing," *J. Netw. Comput. Appl.*, vol. 169, 2020, Art. no. 102781.

[7] N. Jaiswal and N. Purohit, "Performance analysis of NOMA-enabled vehicular communication systems with transmit antenna selection over double Nakagami-m fading," *IEEE Trans. Veh. Technol.*, vol. 70, no. 12, pp. 12725–12741, Dec. 2021.

[8] L. Geng, H. Zhao, J. Wang, A. Kaushik, S. Yuan, and W. Feng, "Deep-reinforcement-learning-based distributed computation offloading in vehicular edge computing networks," *IEEE Internet Things J.*, vol. 10, no. 14, pp. 12416–12433, Jul. 2023.

[9] W. Yao, J. Liu, C. Wang, and Q. Yang, "Learning-based RSU placement for C-V2X with uncertain traffic density and task demand," in *Proc. 2023 IEEE Wireless Commun. Netw. Conf.*, 2023, pp. 1–6.

[10] H. Yu, R. Liu, Z. Li, Y. Ren, and H. Jiang, "An RSU deployment strategy based on traffic demand in vehicular ad hoc networks (VANETs)," *IEEE Internet Things J.*, vol. 9, no. 9, pp. 6496–6505, May 2022.

[11] X. Wu, M. D. Soltani, L. Zhou, M. Safari, and H. Haas, "Hybrid LiFi and WiFi networks: A survey," *IEEE Commun. Surveys Tuts.*, vol. 23, no. 2, pp. 1398–1420, Secondquarter 2021.

[12] R. Chen, M. Liu, Y. Hui, N. Cheng, and J. Li, "Reconfigurable intelligent surfaces for 6G IoT wireless positioning: A contemporary survey," *IEEE Internet Things J.*, vol. 9, no. 23, pp. 23570–23582, Dec. 2022.

[13] Q. Wu, S. Zhang, B. Zheng, C. You, and R. Zhang, "Intelligent reflecting surface-aided wireless communications: A tutorial," *IEEE Trans. Commun.*, vol. 69, no. 5, pp. 3313–3351, May 2021.

[14] W. Mei, B. Zheng, C. You, and R. Zhang, "Intelligent reflecting surface-aided wireless networks: From single-reflection to multireflection design and optimization," in *Proc. IEEE*, vol. 110, no. 9, pp. 1380–1400, Sep. 2022.

[15] N. Chen, C. Liu, H. Jia, and M. Okada, "Intelligent reflecting surface aided network under interference toward 6G applications," *IEEE Netw.*, vol. 36, no. 4, pp. 18–27, Jul./Aug., 2022.

[16] Z. Chu, P. Xiao, M. Shojafar, D. Mi, J. Mao, and W. Hao, "Intelligent reflecting surface assisted mobile edge computing for Internet of Things," *IEEE Wireless Commun. Lett.*, vol. 10, no. 3, pp. 619–623, Mar. 2021.

[17] Z. Li et al., "Energy efficient reconfigurable intelligent surface enabled mobile edge computing networks with NOMA," *IEEE Trans. Cogn. Commun. Netw.*, vol. 7, no. 2, pp. 427–440, Jun. 2021.

[18] S. Mao et al., "Reconfigurable intelligent surface-assisted secure mobile edge computing networks," *IEEE Trans. Veh. Technol.*, vol. 71, no. 6, pp. 6647–6660, Jun. 2022.

[19] X. Chen, H. Xu, G. Zhang, Y. Chen, and R. Li, "Secure computation offloading assisted by intelligent reflection surface for mobile edge computing network," *Phys. Commun.*, vol. 57, 2023, Art. no. 102003.

[20] Y. Xie, L. Shi, Z. Li, X. Ding, and F. Liu, "Roadside IRS assisted task offloading in vehicular edge computing network," in *Proc. 19th EAI Int. Conf. Collaborative Comput., Netw., Appl. Worksharing*, 2024, pp. 365–384.

[21] A. Feriani and E. Hossain, "Single and multi-agent deep reinforcement learning for AI-enabled wireless networks: A tutorial," *IEEE Commun. Surveys Tuts.*, vol. 23, no. 2, pp. 1226–1252, Secondquarter 2021.

[22] Y. Bai, H. Zhao, X. Zhang, Z. Chang, R. Jäntti, and K. Yang, "Toward autonomous multi-UAV wireless network: A survey of reinforcement learning-based approaches," *IEEE Commun. Surveys Tut.*, vol. 25, no. 4, pp. 3038–3067, Fourthquarter 2023.

[23] T. Zhang, H. Wen, Y. Jiang, and J. Tang, "Deep-reinforcement-learning-based IRS for cooperative jamming networks under edge computing," *IEEE Internet Things J.*, vol. 10, no. 10, pp. 8996–9006, May 2023.

[24] L. Huang, S. Bi, and Y.-J. A. Zhang, "Deep reinforcement learning for online computation offloading in wireless powered mobile-edge computing networks," *IEEE Trans. Mobile Comput.*, vol. 19, no. 11, pp. 2581–2593, Nov. 2020.

[25] Y. Wang, W. Fang, Y. Ding, and N. N. Xiong, "Computation offloading optimization for UAV-assisted mobile edge computing: A deep deterministic policy gradient approach," *Wireless Networks*, vol. 27, no. 4, pp. 2991–3006, 2021.

[26] X. Chen and G. Liu, "Energy-efficient task offloading and resource allocation via deep reinforcement learning for augmented reality in mobile edge networks," *IEEE Internet Things J.*, vol. 8, no. 13, pp. 10843–10856, Jul. 2021.

[27] S. Cao et al., "Reinforcement learning based tasks offloading in vehicular edge computing networks," *Comput. Networks*, vol. 234, 2023, Art. no. 109894.

[28] Y. Ju, Z. Cao, Y. Chen, L. Liu, Q. Pei, and S. Mumtaz, "Energy efficient secure offloading in NOMA-aided vehicular networks using A3C learning," in *Proc. IEEE Int. Conf. Commun.*, 2023, pp. 6114–6119.

[29] J. Lin, S. Huang, H. Zhang, X. Yang, and P. Zhao, "A deep-reinforcement-learning-based computation offloading with mobile vehicles in vehicular edge computing," *IEEE Internet Things J.*, vol. 10, no. 17, pp. 15501–15514, Sep. 2023.

[30] X. Zhang, Y. Shen, B. Yang, W. Zang, and S. Wang, "DRL based data offloading for intelligent reflecting surface aided mobile edge computing," in *Proc. 2021 IEEE Wireless Commun. Netw. Conf.*, 2021, pp. 1–7.

[31] Z. Wang, Y. Wei, Z. Feng, F. R. Yu, and Z. Han, "Resource management and reflection optimization for intelligent reflecting surface assisted multi-access edge computing using deep reinforcement learning," *IEEE Trans. Wireless Commun.*, vol. 22, no. 2, pp. 1175–1186, Feb. 2023.

[32] Y. Chen, Y. Wang, J. Zhang, and Z. Li, "Resource allocation for intelligent reflecting surface aided vehicular communications," *IEEE Trans. Veh. Technol.*, vol. 69, no. 10, pp. 12321–12326, Oct. 2020.

[33] M. W. Baidas, "Resource allocation for offloading-efficiency maximization in clustered NOMA-enabled mobile edge computing networks," *Comput. Networks*, vol. 189, 2021, Art. no. 107919.

[34] H. - n. Wang et al., "Deep reinforcement learning: A survey," *Front. Inf. Technol. Electron. Eng.*, vol. 21, no. 12, pp. 1726–1744, 2020.

[35] L. Li, W. Li, J. Wang, X. Chen, Q. Peng, and W. Huang, "UAV trajectory optimization for spectrum cartography: A PPO approach," *IEEE Commun. Lett.*, vol. 27, no. 6, pp. 1575–1579, Jun. 2023.

[36] R. G. Ródenas, M. L. López, and D. Verastegui, "Extensions of Dinkelbach's algorithm for solving non-linear fractional programming problems," *Top*, vol. 7, pp. 33–70, 1999.

[37] S. Schaible, "Fractional programming. II, on Dinkelbach's algorithm," *Manage. Sci.*, vol. 22, no. 8, pp. 868–873, Apr. 1976.

[38] R. Zhang, K. Xiong, Y. Lu, P. Fan, D. W. K. Ng, and K. B. Letaief, "Energy efficiency maximization in RIS-assisted SWIPT networks with RSMA: A PPO-based approach," *IEEE J. Sel. Areas Commun.*, vol. 41, no. 5, pp. 1413–1430, May 2023.

[39] B. Zheng, C. You, and R. Zhang, "Double-IRS assisted multi-user MIMO: Cooperative passive beamforming design," *IEEE Trans. Wireless Commun.*, vol. 20, no. 7, pp. 4513–4526, Jul. 2021.

[40] J. Zuo, Y. Liu, Z. Qin, and N. Al-Dhahir, "Resource allocation in intelligent reflecting surface assisted NOMA systems," *IEEE Trans. Commun.*, vol. 68, no. 11, pp. 7170–7183, Nov. 2020.

[41] R. Storn, "On the usage of differential evolution for function optimization," in *Proc. North Amer. Fuzzy Inf. Process.*, 1996, pp. 519–523.

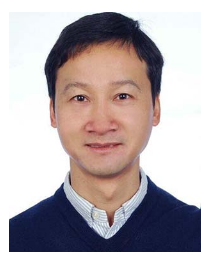
[42] S. Mirjalili and A. Lewis, "The Whale optimization algorithm," *Adv. Eng. Softw.*, vol. 95, pp. 51–67, 2016.

Yibin Xie received the B.S. degree from Anhui Normal University, Wuhu, China, in 2021, and the M.S. degree from the Hefei University of Technology, Hefei, China, in 2024. He is currently working toward the Ph.D. degree with Trinity College Dublin, Dublin, Ireland. His research interests include wireless communication and networking.

Lei Shi (Member, IEEE) received the B.S., M.S., and Ph.D. degrees from the Hefei University of Technology, Hefei, China, in 2002, 2005, and 2012, respectively. He is currently an Associate Professor with the School of Computer Science and Information Engineering, Hefei University of Technology. His research interests include federated learning, edge computing and wireless network optimization.

Zhehao Li received the B.S. degree from Anhui Normal University, Wuhu, China, in 2017, and the M.S. degree in 2021 from the Hefei University of Technology, Hefei, China, where he is currently working toward the Ph.D. degree with the School of Computer Science and Information Engineering. His research interests include federated learning and edge computing.

Xu Ding (Member, IEEE) was born in 1984. He received the Ph.D. degree from the Hefei University of Technology, Hefei, China, in 2015. He is currently an Associate Research Fellow with the Anhui Province Key Lab of Aerospace Structural Parts Forming Technology and the School of Mechanical Engineering, Hefei University of Technology. His research interests include causal inference in machine learning and intelligent fault diagnosis framework.



Yuqi Fan received the B.S. and M.S. degrees in computer science and engineering from the Hefei University of Technology, Hefei, China, in 1999 and 2003, respectively, and the Ph.D. degree in computer Science and Engineering from Wright State University, Dayton, OH, USA, in 2009. He is currently an Associate Professor with the School of Computer Science and Information Engineering, Hefei University of Technology, China. His research interests include blockchain, cloud computing, social networks, computer networks, SDN, and IoT.