# Deep Reinforcement Learning-Based Collaborative Computation Offloading for Distributed Vehicular Edge Computing

Chaogang Tang, *Member, IEEE*, Zhao Li, Huaming Wu, *Senior Member, IEEE*, Shuo Xiao, and Ruidong Li, *Senior Member, IEEE*

***Abstract*—In Vehicular Edge Computing (VEC), apart from the Road Side Units (RSUs) that can undertake the computation, smart vehicles that incorporate high-end multi-core processors into On-Board Units (OBU) can also contribute their computing resources for vehicular tasks in a pay-as-you-go fashion. Designing an appropriate pricing strategy for vehicles with abundant computing resources is essential yet challenging, as it requires balancing profit-seeking objectives with the needs of service requestors. On the other hand, considering the perspective of vehicles with offloading requests, task offloading should strike a balance between achieving ultra-low task latency and minimizing the associated offloading costs. To tackle these issues, we propose a Collaborative Computation Offloading Scheme (CCOS) for the VEC system. In particular, we take into account the fluctuation of service pricing, to cater to the monetary constraints of service requesters. A Mixed-Integer Nonlinear Programming (MINLP) problem is formulated to minimize the weighted sum of task completion latency and the offloading costs. The optimization problem is decomposed into two subproblems, i.e., the task offloading problem and the computing resource allocation problem, respectively. The task offloading problem is essentially a combinatorial optimization problem that necessitates exponential time complexity for determining the optimal solution. Hence, a Deep Reinforcement Learning (DRL)-based algorithm is put forward to solve this subproblem. The resource allocation problem, however, has been proven to be a convex optimization problem, and the scheduling and allocation of computing resources can be performed in parallel, since each edge node is aware of its own task offloading requests. Simulation results demonstrate that our strategy outperforms other approaches in terms of the convergence rate, task completion rate, and optimal values.**

***Index Terms*—Vehicular edge computing, collaborative computation offloading, combinatorial optimization, resource allocation, deep reinforcement learning, service pricing.**

Received 21 March 2025; revised 18 August 2025; accepted 16 September 2025. This work was supported in part by the National Natural Science Foundation of China under Grant 62271486, Grant 62476276, and Grant 62071327; in part by the Emerging Frontiers Cultivation Program of Tianjin University Interdisciplinary Center; and in part by Tianjin Science and Technology Planning Project under Grant 22ZYYYJC00020. The Associate Editor for this article was S. Wan. *(Corresponding author: Huaming Wu.)*

Chaogang Tang, Zhao Li, and Shuo Xiao are with the School of Computer Science and Technology and the School of Artificial Intelligence, China University of Mining and Technology, Xuzhou 221116, China (e-mail: cgtang@cumt.edu.cn; lizhaolzl@163.com; sxiao@cumt.edu.cn).

Huaming Wu is with the Center for Applied Mathematics, Tianjin University, Tianjin 300072, China (e-mail: whming@tju.edu.cn).

Ruidong Li is with the Institute of Science and Engineering, Kanazawa University, Kanazawa 920-1192, Japan (e-mail: liruidong@ieee.org).

Digital Object Identifier 10.1109/TITS.2025.3611955

## I. Introduction

THE cult 1980s television show "Knight Rider" satisfies the fantasy about future vehicles, e.g., in regards to appearance, functionality and human-vehicle interaction. Four decades later, familiar scenes from the sci-fi TV series are being replicated in reality, thanks to dramatic breakthroughs in Information and Communication Technologies (ICTs), the Internet of Things (IoT), and Artificial Intelligence (AI) technologies [1]. For instance, Vehicular Edge Computing (VEC), as one of the key enablers of intelligent transportation, has garnered extensive attention from both industry and academia over the past few years. This computing paradigm enables vehicular tasks to be processed in proximity to data sources, thus satisfying the ultra-low task latency requirement in time-critical application scenarios such as autonomous driving, and human-vehicle interaction. Apart from the Road Side Unit (RSU) that can undertake the computation, various smart vehicles, e.g., Tesla automobiles equipped with powerful AMD Ryzen chips, can also contribute their computing resources for task processing in VEC systems. Such vehicles are also referred to as service vehicles, which typically provide and monetize their idle computing resources. In contrast, vehicles generating offloading requests are also known as task vehicles.

Offloading tasks to service vehicles instead of RSU can not only serve as an alternative offloading approach, but also enhance the robustness of VEC systems, particularly in scenarios where the edge server is overloaded. This multi-destination computation offloading is also termed collaborative computation offloading in VEC [2], [3], [4]. In practice, however, few studies have focused on designing efficient incentive mechanisms to effectively encourage service vehicles to contribute their resources. Meanwhile, designing an appropriate pricing strategy for service vehicles is essential, yet challenging, as it requires balancing profit-seeking objectives with the needs of service requestors, such as their monetary constraints. However, considering the perspective of task vehicles, task offloading should strike a balance between achieving ultra-low task latency and minimizing the associated offloading costs.

On the other hand, service vehicles that integrate high-end multi-core processors facilitate the widespread implementation of AI models and algorithms [5], [6], [7], and general AI approaches are transitioning toward edge-based training and

end (vehicle)-based inference in complex VEC environments [8], [9], [10]. In this paper, we introduce a collaborative computation offloading scheme (CCOS) based on Deep Reinforcement Learning (DRL), leveraging the strengths of AI methodologies. Specifically, we consider the variability of service pricing, focusing on vehicles providing computing services, to accommodate the financial limitations of task vehicles. The objective of this paper is to optimize the weighted sum of task completion latency and the offloading costs, by jointly optimizing the task offloading and resource allocation decisions in VEC. The major contributions of this paper are summarized as follows.

- Vehicles are classified into task vehicles and service vehicles, respectively, according to their practical demands. A collaborative computation offloading strategy is put forward, where vehicular tasks can be offloaded and executed in either the RSU or other vehicles. By doing so, the robustness of VEC systems can be enhanced, especially considering that the edge server (RSU) can occasionally be overloaded and overwhelmed by large amounts of offloading requests.
- This paper formulates a joint optimization problem for task offloading and resource allocation in VEC. The objective is to minimize both task completion latency and offloading costs during the optimization period by simultaneously optimizing task offloading and resource allocation decisions in VEC. Specifically, the optimization problem is modeled as a Mixed-Integer Nonlinear Programming (MINLP) problem.
- To lower the difficulty in solving the optimization problem, we decompose it into two subproblems, i.e., the computing resource allocation problem and the task offloading problem. The former can be proven to be a convex optimization problem and the latter is actually a combinatorial optimization problem. Hence, we put forward a DRL-based strategy to solve the task offloading problem.
- Comprehensive numerical evaluation is carried out from multiple perspectives, while several existing strategies are selected as the benchmarks in the simulation. Experimental results demonstrate that our strategies and algorithms outperform other approaches in terms of the convergence rate, task completion rate, and optimal values.

The subsequent sections of this paper are structured as follows: A comprehensive review of the state-of-the-art literature on this topic is provided in Section II. Section III presents the system model and then formulates the optimization problem. The algorithm design is presented in Section IV, followed by the numerical evaluation in Section V. Finally, the conclusion comes in Section VI.

## II. Related Work

Extensive literature has delved into task offloading in edge computing environments [11], [12]. In this section, the state-of-the-art literature on this topic is reviewed.

Wang et al. [2] stated that legacy vehicles will coexist with smart vehicles for a long time. For the tasks from the legacy vehicles, smart vehicles can serve as the offloading destinations. Hence, they model the collaborative task offloading problem as a Markov decision process (MDP), in the hope of optimizing the average response latency for the tasks. A heuristic strategy is then adopted to solve the problem, and the simulation results prove its advantages over other approaches.

Owing to the limited computing capabilities of vehicles, it is very difficult for vehicular networks to handle latency-sensitive and compute-intensive tasks. Hence, fog computing is put forward in [3] as a supplement to assist computation in time-critical scenarios. Specifically, they propose an efficient solution that combines federated learning with deep Q-learning techniques in a collaborative computing paradigm. They also take into account multiple computing and communication constraints in the work. The simulation results show that their strategy is much better than others, e.g., with a minimal and maximal improvement of 8.21% and 49%, respectively.

As computation-intensive tasks in intelligent transportation systems continue to increase, substantial computing resources are required from vehicles, edge servers, and remote cloud centers. Vehicular Cloud Computing (VCC) can provide a plethora of computing and storage resources, however, it is constrained by lengthy data transmission across the core network. On the contrary, while VEC can provide latency-sensitive computing services, its computational capabilities are constrained. Hence, Mittal et al. [13] proposed a distributed task orchestration framework to realize the efficient collaboration between edge computing and cloud computing. Meanwhile, Liu et al. [14] explored collaborative task offloading and resource allocation. Their framework facilitates efficient collaboration among vehicles, edge servers, and the cloud while enabling intelligent management of heterogeneous resources. Specifically, they proposed a joint optimization problem aiming to maximize system utility. They employed an asynchronous DRL algorithm to seek the optimal solution to this problem.

Recently, lightweight Deep Neural Networks (DNNs) have been increasingly deployed in Intelligent IoT systems. Realizing energy-efficient offloading for tasks with strict latency requirements in DNN-based IoT systems is still challenging. Hence, Chen et al. [15] constructed an energy consumption model and used a hybrid offloading method combining PSO with GA algorithms to realize task offloading. The simulation demonstrated the advantages over other approaches. In contrast to cloud computing, Mobile Edge Computing (MEC) can effectively reduce long-distance transmission over the core network, thereby enhancing the quality of service (QoS) for Internet of Vehicles (IoV). However, both system congestion and waiting delay are often overlooked in current works when dealing with large volumes of computational data. Hence, Sun et al. [16] put forward a joint on-board task offloading and job scheduling strategy in the context of collaborative cloud-edge computing paradigms. An ant colony optimization algorithm is employed to achieve multi-objective optimization. Experimental results substantiate the effectiveness of the approach.

Li et al. [17] designed a collaborative task offloading and service caching replacement scheme. It primarily focused on the cooperation in task processing between adjacent RSUs.

TABLE I
COMPARISON BETWEEN THIS WORK AND EXISTING STUDIES

| Reference | Offloading Strategy | Pricing Strategy | Algorithm | Computing Servers | Optimization Objective |
|---|---|---|---|---|---|
| [2] | Binary | - | Heuristic algorithm | Multiple vehicles | Processing latency |
| [3] | Binary | Static | Federated learning and DQN | RSU and cloud | Latency and energy |
| [13] | Binary | - | Hashing-based algorithm | Vehicle, RSU and cloud | Processing latency |
| [14] | Binary | - | DRL-based algorithm | Vehicle, RSU and cloud | System utility |
| [15] | Binary | - | PSO-GA algorithm | End,edge and cloud | Energy consumption |
| [16] | Binary | - | ACO algorithm | RSU and cloud | System utility |
| [17] | Partial | Static | DRL-based algorithm | RSU | System cost and delay |
| [18] | Binary | - | DRL-based algorithm | RSU | Offloading utility & completion ratio |
| [19] | Binary | - | DDPG algorithm | RSU | Average execution delay |
| [20] | Binary | - | DQN-based algorithm | Edge and cloud | System latency |
| [21] | Binary | - | Improved CMA-ES algorithm | Edge | System utility with five metrics |
| [22] | Binary | Static | Heuristic algorithm | Edge | System costs |
| [23] | Binary | - | Auction-bid scheme | RSU | Processing latency |
| [24] | Partial | - | Heuristic algorithm | Edge | Average completion time |
| [25] | Binary | - | Semidefinite relaxation approach | RSU | Latency and computation cost |
| Our work | Binary | Dynamic | DRL-based algorithm | Vehicle and RSU | System cost and processing latency |

A proactive strategy was proposed in [18] to reduce the performance instability events. Particularly, an adaptive task offloading scheme with the proactive adjustment capability is designed, and then a DRL-based task offloading algorithm is introduced.

Edge caching plays an important role in task offloading in VEC. However, it is difficult to design an efficient offloading and caching scheme in VEC, since there is no global controller knowing all the information of entities in VEC. Therefore, Liu and Chen [19] aimed to solve a multiuser task offloading, computation caching and resource allocation problem in the VEC system. A deep deterministic policy gradient (DDPG) algorithm is introduced for the formulated problem. Simulation results have proven its advantages in several aspects.

A NOMA-based VEC model is put forward in [22], aiming to minimize the total system cost while meeting the severe latency requirement of tasks. The decision variables encompass multiple decision policies, such as task offloading, clustering, subchannel and computation resource allocation, as well as transmission power control. The formulated MINLP problem is decomposed into two subproblems to alleviate the computational complexity, and two heuristic algorithms are proposed for their respective solutions. Other outstanding works have also investigated issues of task offloading, service caching and computing resource allocation from different perspectives, such as [23], [24], [25], and [26]. Particularly, the distinctions between our work and the existing literature are systematically summarized in Table I. Although the aforementioned works focus on enhancing the performance of edge computing systems through minimizing task offloading latency and optimizing resource allocation, they often overlook the pay-as-you-go model for resource provisioning in these systems. A reasonable pricing strategy not only needs to balance profit-maximizing objectives with the requirements of service requesters but also aims to strike a balance between achieving ultra-low task latency and reducing associated offloading costs. As a result, in this paper, we propose a collaborative computation offloading mechanism for the VEC system that accounts for service pricing fluctuations, thereby accommodating the financial constraints of service requesters.

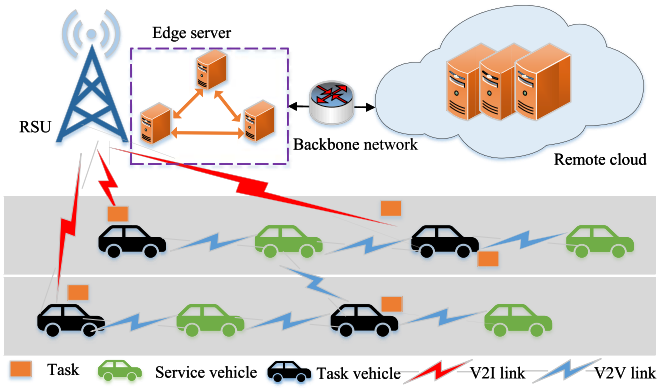



Fig. 1. The considered application scenario in this paper.

## III. SYSTEM MODEL AND PROBLEM FORMULATION

### A. System Model

Fig. 1 illustrates the system model proposed in this paper, consisting of multiple vehicles, an RSU, and a cloud center. The vehicles are classified into two types: task vehicles and service vehicles. The RSU, integrated with edge servers, offers robust computing and networking resources in close proximity to the vehicles. Meanwhile, the cloud center, rich in diverse resources, is located at a greater distance from the VEC system.

Generally, both RSU and service vehicles can serve as potential offloading destinations for tasks generated by task vehicles. In this paper, we collectively refer to both the RSU and service vehicles as edge nodes (ENs), denoted by the set $\mathcal{M} = \{0, \ldots, M\}$, where $m = 0$ corresponds to the RSU $R$, and $1 \leq m \leq M$ represents the service vehicle $m$. The key difference between RSU and other ENs lies in the computing capabilities and service prices. In addition, we denote the set of task vehicles by $\mathcal{N} = \{1, \ldots, N\}$, where $M$ and $N$ represent the total number of service vehicles and task vehicles, respectively.

We assume that each task vehicle $n$ generates one task in the optimization period. Each task is characterized by a tuple of six parameters: $(l_n, vel_n, c_n, s_n, b_n, d_n)$, where $l_n$ indicates the

location of vehicle $n$, $vel_n$ is the velocity of vehicle $n$, $c_n$ stands for the task-input data size, $s_n$ denotes the required computing resources (e.g., CPU cycles) for the task, $b_n$ represents the monetary budget for the task, and $d_n$ is the latency constraint for the task. EN $m$ can be described by a tuple of four parameters $(l_m, \kappa_m, F_m, p_m)$, where $l_m$ indicates the location of EN $m$, $\kappa_m$ denotes the total communication resources of $m$ (e.g., in the form of the total bandwidth or total transmission rate), $F_m$ denotes the total computing resources of $m$ (e.g., in CPU cycles/s), and $p_m$ is the service price per computing resource allocated for task execution.

We assume that the generated tasks are indivisible, and can only be offloaded and executed externally. A binary variable $\alpha_{n,m}$ is defined to indicate whether the task from vehicle $n$ is offloaded to EN $m$. Specifically, if the task is offloaded to $m$ for execution, then $\alpha_{n,m} = 1$, and $\alpha_{n,m} = 0$, otherwise. If the vehicle $n$ offloads its task to EN $m$, the response delay (i.e., task completion latency) typically consists of three components: i) the offloading delay, representing the time required to transmit the task; ii) the computational latency, representing the time required by $m$ to execute the task; iii) the feedback delay, which accounts for the time taken by $m$ to send the computation result back to $n$. Since the size of the computation result is much smaller than that of the task input data, we omit the feedback delay in this paper. The calculation methods for these delays will be presented in the following subsections.

### B. Task Offloading Model

Let $r_{n,m}$ denote the transmission rate for offloading the task from $n$ to $m$. $r_{n,m}$ actually depends upon the quantity of communication resources allocated to the task by $m$. For simplicity, we assume that all the tasks offloaded to $m$ share the communication resources equally. Hence, $r_{n,m}$ can be represented as [27]:

$$r_{n,m} = \frac{\kappa_m}{\sum_{i \in \mathcal{N}} \alpha_{i,m}}. \tag{1}$$

The offloading delay, denoted by $T^{off}_{n,m}$, can be calculated as:

$$T^{off}_{n,m} = \frac{c_n}{r_{n,m}}. \tag{2}$$

### C. Computation Model

If the task from vehicle $n$ is offloaded to EN $m$ for execution, $m$ can execute the task immediately once the offloading process is accomplished. The computational latency, denoted by $T^{com}_{n,m}$, is given as:

$$T^{com}_{n,m} = \frac{s_n}{f_{m,n}}, \tag{3}$$

where $f_{m,n}$ denotes the total computing resources allocated by $m$ to the task from $n$. The total response delay for the task from $n$, denoted by $T_n$, can be represented as:

$$T_n = \sum_{m \in \mathcal{M}} \alpha_{n,m} (T^{off}_{n,m} + T^{com}_{n,m}). \tag{4}$$

### D. Resource Pricing Model

It is generally understood that the computing resources at ENs are provisioned in a pay-as-you-go fashion. Namely, task vehicles need to pay ENs for using their computing services, which is considered as an efficient incentive approach to motivate ENs to contribute their computing resources. Driven by the profit-seeking instinct, ENs usually price their computing resources towards profit maximization. In addition, the service price of service vehicles is more fluctuant than that of RSU, in the sense that service vehicles are usually selfish individuals that are easily affected by the number of offloading requests, while RSU is usually deployed by non-commercial entities (e.g., local government), aiming to provide cost-effective and sustainable computing services.

To embody the above characteristics, we present the resource pricing models for service vehicles and RSU, respectively. Particularly, the service price $p_m$ can be defined as follows:

$$p_m = \begin{cases} \widetilde{p}_m \cdot e^{-\frac{\sum_{n \in \mathcal{N}} \alpha_{n,m}}{N}}, 1 \le m \le M, \\ \widetilde{p}_{m,R}, \qquad\qquad m = 0, \end{cases} \tag{5}$$

where $\widetilde{p}_m$ and $\widetilde{p}_{m,R}$ denote the initial service prices for the service vehicle $m$ and RSU $R$, respectively. The price of the service vehicle $m$ has the following properties. $e^{-\sum_{n \in \mathcal{N}} \alpha_{n,m}/N}$ can be viewed as a discount coefficient that depends on the number of offloading requests. As the number of tasks offloaded to $m$ increases, the discount also increases. Thus, the service price $p_m$ decreases with the increasing number of offloaded tasks. Similarly, the service price $p_m$ increases with the decreasing number of offloaded tasks. Actually, the task vehicles need to strike a balance between response latency and offloading cost for their tasks. For instance, in the case of the service price of RSU being higher than that of service vehicles, task vehicles can choose the service vehicle with a lower service price as the offloading destination to achieve cost-efficient computation offloading. Such task offloading, however, could incur long task completion latency. In contrast, task vehicles can also choose RSU as the offloading destination at a higher offloading cost.

If the vehicle $n$ offloads its task to EN $m$ for execution, the cost that $n$ should pay to $m$ can be expressed as:

$$C_{n,m} = p_m f_{m,n}. \tag{6}$$

Thus, the generalized cost for task vehicle $n$ is:

$$C_n = \sum_{m \in \mathcal{M}} \alpha_{n,m} p_m f_{m,n}. \tag{7}$$

### E. Problem Formulation

We formulate a vehicle-RSU collaborative computation offloading optimization problem for VEC, with one goal to minimize the overall response latency for all the tasks from the angle of QoE, and the other goal to minimize the total costs for all the vehicles from the cost-effective angle, by jointly optimizing the offloading policy and the computing resource allocation policy. However, as described earlier, the two optimization objectives contradict each other, since lower response latency requires more computing resources, incurring

more monetary costs, and less monetary costs mean that more tasks share the same computing resources, leading to higher response latency. Accordingly, we compromise the optimization with a weighted additive technology to balance the response latency and the cost.

Let $\alpha_n = \{\alpha_{n,m} | \forall m \in \mathcal{M}\}$ denote the offloading decision of task vehicle $n$ and thus $\boldsymbol{\alpha} = \{\alpha_n | \forall n \in \mathcal{N}\}$ denotes the offloading policy. Let $f_m = \{f_{m,n} | \forall n \in \mathcal{N}\}$ denote the computing resource allocation of EN $m$ and thus $\boldsymbol{f} = \{f_m | \forall m \in \mathcal{M}\}$ denotes the computing resource allocation policy. The objective function regarding $\boldsymbol{\alpha}$ and $\boldsymbol{f}$ can be given as:

$$J(\boldsymbol{\alpha}, \boldsymbol{f}) = \sum_{n=1}^{N} \beta_1 T_n + \beta_2 C_n, \tag{8}$$

where $\beta_i (i \in \{1, 2\})$ are the weights to indicate the pReferences towards the response latency and the system costs, respectively. A larger $\beta_1$ indicates that the response latency as the performance metric is paid more attention by vehicle users. Contrarily, a larger $\beta_2$ means that the cost for task execution is more sensitive to the vehicle users. Then, our optimization problem in this paper is formulated as follows:

$$\begin{aligned}
\boldsymbol{\mathcal{P}_0}: \min_{\boldsymbol{\alpha}, \boldsymbol{f}} \quad & J(\boldsymbol{\alpha}, \boldsymbol{f}), \\
\text{s.t.} \quad & \sum_{m=1}^{M} \alpha_{n,m} = 1, \quad \forall n \in \mathcal{N}, && (9) \\
& T_n \le d_n, \quad \forall n \in \mathcal{N}, && (10) \\
& C_n \le b_n, \quad \forall n \in \mathcal{N}, && (11) \\
& \sum_{n \in \mathcal{N}} \alpha_{n,m} f_{m,n} \le F_m, \ \forall m \in \mathcal{M}, && (12) \\
& f_{m,n} \ge 0, \quad \forall n \in \mathcal{N}, \ \forall m \in \mathcal{M}, && (13) \\
& \alpha_{n,m} \in \{0, 1\}, \quad \forall n \in \mathcal{N}, \ \forall m \in \mathcal{M}, && (14)
\end{aligned}$$

where the constraint (9) denotes that one task can be offloaded to only one EN. Constraint (10) means that the task should be accomplished before its deadline. Constraint (11) means that the cost paid to the EN by a task vehicle should not exceed the given monetary budget. Constraint (12) indicates that the computing resources allocated to the offloaded tasks should not exceed the maximal computing capability of the EN.

The problem $\boldsymbol{\mathcal{P}_0}$ is essentially a MINLP problem, making it highly challenging to solve directly. Given the stringent latency requirements for vehicular tasks in large-scale VEC networks, algorithms with exponential time complexity for finding the optimal solution are practically infeasible. As a result, we aim to apply a suboptimal algorithm with lower time complexity to address this problem.

### F. Problem Decomposition

The expression of the objective function $J(\boldsymbol{\alpha}, \boldsymbol{f})$ can be expanded as:

$$\begin{aligned}
J(\alpha, f) &= \sum_{n=1}^{N} \beta_1 T_n + \beta_2 C_n \\
&= \sum_{n=1}^{N} \sum_{m=1}^{M} \left\{ \alpha_{n,m} \beta_1 \left( \frac{c_n \sum_{n \in \mathcal{N}} \alpha_{n,m}}{\kappa_m} + \frac{s_n}{f_{m,n}} \right) \right. \\
&\quad \left. + \alpha_{n,m} \beta_2 p_m f_{m,n} \right\} \qquad (15)
\end{aligned}$$

It is easily observed that the decision variables $\boldsymbol{\alpha}$ and $\boldsymbol{f}$ are tightly coupled in the objective function as well as in the constraints. To simplify the problem **P0**, we decouple the two variables from each other, and thus the original problem can be decomposed into two subproblems, i.e., the task offloading problem $\boldsymbol{\mathcal{P}_1}$ and the computing resource allocation problem $\boldsymbol{\mathcal{P}_2}$, respectively. In particular, by temporarily fixing the offloading variable $\boldsymbol{\alpha}$, the problem $\boldsymbol{\mathcal{P}_0}$ can be rewritten as:

$$\begin{aligned}
& \min_{\boldsymbol{\alpha}} (\min_{\boldsymbol{f}} \ J(\boldsymbol{\alpha}, \boldsymbol{f})), \\
& \text{s.t.} \quad (9) - (14). && (16)
\end{aligned}$$

Solving the above problem is equivalent to solving the following task offloading problem $\boldsymbol{\mathcal{P}_1}$,

$$\begin{aligned}
\boldsymbol{\mathcal{P}_1}: \quad & \min_{\boldsymbol{\alpha}} \ J^*(\boldsymbol{\alpha}), \\
& \text{s.t.} \ (9), (14), && (17)
\end{aligned}$$

where $J^*(\boldsymbol{\alpha})$ is the function with optimal value regarding the resource allocation problem $\boldsymbol{\mathcal{P}_2}$, i.e.,

$$\begin{aligned}
\boldsymbol{\mathcal{P}_2}: J^*(\boldsymbol{\alpha}) = & \min_{\boldsymbol{f}} \ J^*(\boldsymbol{\alpha}, \boldsymbol{f}), \\
& \text{s.t.} \quad (10), (11), (12), (13). && (18)
\end{aligned}$$

Note that temporarily fixing the offloading variable $\boldsymbol{\alpha}$ also decouples the constraints on task offloading (9), and (14) from the constraints on the resource allocation (10), (11), (12), and (13). More importantly, the transformation from $\boldsymbol{\mathcal{P}_0}$ to subproblems $\boldsymbol{\mathcal{P}_1}$ and $\boldsymbol{\mathcal{P}_2}$ does not alter the optimality of the solution [28]. Therefore, we shift our attention from solving $\boldsymbol{\mathcal{P}_0}$ to solving the subproblems $\boldsymbol{\mathcal{P}_1}$ and $\boldsymbol{\mathcal{P}_2}$, respectively.

## IV. The Proposed CCOS

### A. Computing Resource Allocation

Given the feasible offloading decision, the subproblem $\boldsymbol{\mathcal{P}_2}$ strives to acquire $J^*(\boldsymbol{\alpha})$ by optimizing the computing resource allocation policy. From the expanded expression of the objective function $J(\boldsymbol{\alpha}, \boldsymbol{f})$, we have the following lemma.

*Lemma 1:* Given the task offloading decision $\boldsymbol{\alpha}$, the computing resource optimization problem $\boldsymbol{\mathcal{P}_2}$ regarding $\boldsymbol{f}$ is convex.

*Proof:* For simplicity, we define $g_n$ as the weighted sum of the response latency and monetary cost for vehicle $n$,

$$g_n \triangleq \sum_{m=1}^{M} \alpha_{n,m} \beta_1 \left( \frac{c_n \sum_{n \in \mathcal{N}} \alpha_{n,m}}{\kappa_m} + \frac{s_n}{f_{m,n}} \right) + \alpha_{n,m} \beta_2 p_m f_{m,n}.$$

We then prove that $g_n$ is convex w.r.t. $\boldsymbol{f}$ given $\boldsymbol{\alpha}$ as follows. The partial derivatives by differentiating $g_n$ w.r.t. $f_m$ can be calculated as:

$$\frac{\partial g_n}{\partial f_{m,i}} = \begin{cases} -\frac{\beta_1 s_n}{f_{m,n}^2} + \beta_2 p_m, & i = n, \\ 0, & \text{otherwise.} \end{cases}$$

Then, the second partial derivatives are:

$$\frac{\partial^2 g_n}{\partial f_{m,i} \partial f_{m,j}} = \begin{cases} \frac{2\beta_1 s_n}{f_{m,n}^3}, & i = n \text{ and } j = n, \\ 0, & \text{otherwise.} \end{cases}$$

The Hessian matrix of $g_n$ is constructed as:

$$\mathcal{M}(g_n) = \begin{bmatrix} 0 & \cdots & \cdots & \cdots & 0 \\ 0 & \cdots & \cdots & \cdots & 0 \\ 0 & \cdots & \frac{2\beta_1 s_n}{f_{m,n}^3} & \cdots & 0 \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ 0 & \cdots & \cdots & \cdots & 0 \end{bmatrix}.$$

Then, we have $Y^T \mathcal{M} Y \geq 0$ always holds for an arbitrary nonzero vector $Y$, so $\mathcal{M}(g_n)$ is positive semi-definite. Hence $g_n$ is convex w.r.t. $f_m$. Owing to the additivity attribute of convex function, $J(\boldsymbol{\alpha}, \boldsymbol{f}) = \sum_{n \in \mathcal{N}} g_n$ is also convex w.r.t. $f_m$ ($\forall m \in \mathcal{M}$). Thus, the objective function is convex regarding the computing resource allocation policy.

The left-hand side of the constraint (10) is easily proven to be convex. The second partial derivatives by differentiating $T_n$ w.r.t. $f_m$ are,

$$\frac{\partial^2 T_n}{\partial f_{m,i} \partial f_{m,j}} = \begin{cases} \frac{2 s_n}{f_{m,n}^3}, & i = n \text{ and } j = n, \\ 0. & \text{otherwise.} \end{cases}$$

The resulting Hessian matrix of $T_n$, denoted by $\mathcal{M}(T_n)$, is positive semi-definite, since it satisfies that $Y^T \mathcal{M}(T_n) Y \geq 0$ always holds for an arbitrary nonzero vector $Y$. The constraints (11), (12) and (13) are also convex. As a result, the problem $\boldsymbol{\mathcal{P}_2}$ is convex, and the proof is completed.■

The computing resource allocation policy can be determined via Karush-Kuhn-Tucker (KKT) conditions. Particularly, the Lagrangian function for the problem $\boldsymbol{\mathcal{P}_2}$ is constructed as shown in Eq. (19), as shown at the bottom of the page. $v = \{v_n | v_n \geq 0, \forall n \in \mathcal{N}\}$, $\mu = \{\mu_n | \mu_n \geq 0, \forall n \in \mathcal{N}\}$ and $\lambda = \{\lambda_m | \lambda_m \geq 0, \forall m \in \mathcal{M}\}$ are the Lagrangian multipliers. According to the KKT conditions, let the first partial derivative of $L(\boldsymbol{\alpha}, \boldsymbol{f}, v, \mu, \lambda)$ regarding $\mathcal{F}$ equal zero, i.e., $\partial L(\boldsymbol{\alpha}, \boldsymbol{f}, v, \mu, \lambda)/\partial \boldsymbol{f} = 0$, and we have

$$\frac{\partial L(\boldsymbol{\alpha}, \boldsymbol{f}, v, \mu, \lambda)}{\partial f_{m,n}} = p_m(\mu_n^* + \beta_2) - \frac{s_n(\beta_1 + v_n^*)}{f_{m,n}^{*2}} + \lambda_m^* = 0,$$

where $\alpha_{n,m} = 1, \forall n \in \mathcal{N}, \forall m \in \mathcal{M}$.

The optimal solution $f_{m,n}^*$ can be obtained as:

$$f_{m,n}^* = \sqrt{\frac{s_n(\beta_1 + v_n^*)}{p_m(\mu_n^* + \beta_2) + \lambda_m^*}}, \quad \forall n \in \mathcal{N}, \ \forall m \in \mathcal{M}. \tag{20}$$

Apart from the optimal resource allocation policy $f_{m,n}^*$ ($\alpha_{n,m} = 1, \forall n \in \mathcal{N}, \forall m \in \mathcal{M}$), the optimal values for the Lagrangian multipliers $v_n^*$, $\mu_n^*$ and $\lambda_m^*$ ($\forall n \in \mathcal{N}, \forall m \in \mathcal{M}$) can also satisfy the KKT conditions. We notice that the resource allocation policy can actually be implemented in parallel, because temporarily fixing the offloading variable $\boldsymbol{\alpha}$ makes each EN $m$ aware of their own offloading requests from nearby vehicles. For an arbitrary EN $m$ ($\forall m \in \mathcal{M}$), the KKT conditions include

$$\begin{cases} f_{m,n}^* = \sqrt{\frac{s_n(\beta_1 + v_n^*)}{p_m(\mu_n^* + \beta_2) + \lambda_m^*}}, \\ v_n^*(T_n - d_n) = 0, \\ \mu_n^*(C_n - b_n) = 0, \\ \lambda_m^* \left( \sum_{n \in \mathcal{N}} f_{m,n}^* - F_m \right) = 0, \\ T_n \leq d_n, \\ C_n \leq b_n, \\ \sum_{n \in \mathcal{N}} f_{m,n}^* \leq F_m, \\ \forall n \in \mathcal{N}, \alpha_{n,m} = 1. \end{cases} \tag{21}$$

For the task from vehicle $n$, there are eight candidate solutions for the computing resource allocation $f_{m,n}^*$, each of which needs to be discussed separately. For example, if the solution $f_{m,n}^*$ strictly satisfies the inequalities (10), (11), and (12), it means that $f_{m,n}^*$ lies within the interior of the feasible regions defined by these three constraints. In this case, we have $v_n^* = 0$, $\mu_n^* = 0$, and $\lambda_m^* = 0$, and the optimal resource allocation is given by:

$$f_{m,n}^* = \sqrt{\frac{s_n \beta_1}{p_m \beta_2}}. \tag{22}$$

If the solution $f_{m,n}^*$ lies on the boundary of the feasible region for only one constraint, say constraint (10), and we have $T_n - d_n = 0$, then $\mu_n^* = 0$ and $\lambda_m^* = 0$. By combining the optimal resource allocation solution shown in Eq. (20), we can obtain the solution:

$$f_{m,n}^* = \frac{s_n}{d_n - T_{n,m}^{off}}, \quad v_n^* = \frac{p_m s_n \beta_2}{(d_n - T_{n,m}^{off})^2}, \ \mu_n^* = 0, \lambda_m^* = 0. \tag{23}$$

The remaining candidate solutions can be enumerated in a similar manner. Finally, we need to check whether these candidates satisfy all the constraints in order to determine the optimal solution for the task from vehicle $n$ offloaded to computing resource $m$.

The corresponding algorithm is outlined in Alg. 1. The input to the algorithm involves parameters such as $\boldsymbol{\alpha}$, $\mathcal{M}$, $\mathcal{N}$, $\beta_1$, $\beta_2$, $\widetilde{p}_m$, and $\widetilde{p}_{m,R}$. Given the task offloading policy $\boldsymbol{\alpha}$, the resource price $p_m(\forall m \in \mathcal{M})$ can be determined via Eq. (5) (line 1). Then the Lagrange function $L(\boldsymbol{\alpha}, \boldsymbol{f}, v, \mu, \lambda)$ can be constructed via Eq. (19). Providing $\boldsymbol{\alpha}$ means each EN $m$ acquires the information of tasks offloaded to them. Thus, the optimal computing resource allocation policy for each EN $f_m^*$ can be determined in parallel. The optimal computing resource allocation policy is determined, and then each EN can allocate

$$L(\boldsymbol{\alpha}, \boldsymbol{f}, v, \mu, \lambda) = \sum_{n=1}^{N} \sum_{m=1}^{M} \left\{ \alpha_{n,m} \beta_1 \left( \frac{c_n \sum_{n \in \mathcal{N}} \alpha_{n,m}}{\kappa_m} + \frac{s_n}{f_{m,n}} \right) + \alpha_{n,m} \beta_2 p_m f_{m,n} \right\} + \sum_{n=1}^{N} v_n (T_n - d_n) + \sum_{n=1}^{N} \mu_n (C_n - b_n) + \sum_{m=1}^{M} \lambda_m \left( \sum_{n \in \mathcal{N}} \alpha_{n,m} f_{m,n} - F_m \right). \tag{19}$$

the computing resources to the offloaded tasks (lines 4-13). Meanwhile, the local value for the objective function $G_m$ is recorded (line 14). At last, the optimal objective value $J^*(\boldsymbol{\alpha})$ is calculated and returned together with the optimal resource allocation policy $\{f^*_{m,n}|\forall m \in \mathcal{M}, \forall n \in \mathcal{N}\}$.

**Algorithm 1** Distributed Lagrange-Based Approach for Resource Allocation

**Input:** $\boldsymbol{\alpha}$, $\mathcal{M}$, $\mathcal{N}$, $\beta_1$, $\beta_2$, $\widetilde{p}_m$, $\widetilde{p}_{m,R}$
**Output:** $\{f^*_{m,n}|\forall m \in \mathcal{M}, \forall n \in \mathcal{N}\}$, $J^*(\boldsymbol{\alpha})$
1 Calculate $p_m$ according to Eq. (5) for $\forall m \in \mathcal{M}$;
2 Construct the Lagrange function via Eq. (19);
3 **for** each EN $m \in \mathcal{M}$ **do**
  // Implement the policy in a distributed way
4  **for** each vehicle $n$ with $\alpha_{n,m} = 1$ **do**
5  Calculate $f^*_{m,n} = \sqrt{\frac{s_n(\beta_1+v^*_n)}{p_m(\mu^*_n+\beta_2)+\lambda^*_m}}$;
6  Enumerate solution candidates $\mathcal{S}$ via Eq. (21);
7  **for** each $s = (f^*_{m,n}, v^*_n, \mu^*_n, \lambda^*_m)$ in $\mathcal{S}$ **do**
8  **if** $s$ satisfy constraints (10)-(13) **then**
9  Record $s$;
10  $g_n = \beta_1(\frac{c_n \sum_{n\in\mathcal{N}} \alpha_{n,m}}{\kappa_m} + \frac{s_n}{f^*_{m,n}}) + \beta_2 p_m f^*_{m,n}$;
11  **end**
12  **end**
13  **end**
14  $G_m = \sum_{\forall n\in\mathcal{N}\wedge\alpha_{n,m}=1} g_n$;
15 **end**
16 $J^*(\boldsymbol{\alpha}) = \sum_{\forall m\in\mathcal{M}} G_m$;
17 Return $\{f^*_{m,n}|\forall m \in \mathcal{M}, \forall n \in \mathcal{N}\}$, $J^*(\boldsymbol{\alpha})$;

### B. Task Offloading Decisions

The task offloading decision $\alpha_{n,m}$ is a discrete variable, and the task from $n$ can be offloaded to one of the $M+1$ possible ENs for execution. Finding the optimal solution regarding task offloading requires traversing the solution space of $O((M+1)^N)$, which is practically prohibitive in large-scale VEC systems. Iteration-based evolutionary algorithms, such as Genetic Algorithm (GA), are time-consuming and also tend to be trapped in local optima. Accordingly, we propose a DRL-based approach to solve the problem $\boldsymbol{\mathcal{P}_1}$.

DRL-based approaches have been widely adopted to solve complicated decision-making optimization problems. In time step $t$, an agent is trained to acquire the current state $s_t(\in \mathcal{S})$ that is usually an abstract description of knowledge and experience by interacting with its environment. Then, the agent takes the currently "best" action $a_t(\in \mathcal{A})$ to explore the environment and obtain a reward $r_t$. In the meanwhile, the state $s_t$ of the environment, affected by the adopted action $a_t$, turns into $s_{t+1}$. The agent aims to find the optimal policy $\pi : \mathcal{S} \to \mathcal{A}$ (mapping from the state space $\mathcal{S}$ to the action space $\mathcal{A}$) that can maximize the expected total discounted reward. Specifically, the state space $\mathcal{S}$, action space $\mathcal{A}$ and the reward function can be given below.

- *State Space*: $\mathcal{S}$ is a collection of all the environment states in the VEC system, which can be defined as $\mathcal{S} \triangleq \{s_t = (S_\mathcal{N}, S_\mathcal{M})\}$, $t \in \{0, 1, 2, \ldots\}$. $S_\mathcal{N}$ denotes the states of the vehicular tasks and $S_\mathcal{N} = \{(l_n, vel_n, c_n, s_n, b_n, d_n)\}_{n\in\mathcal{N}}$. $S_\mathcal{M}$ denotes the states of the ENs in the VEC system and $S_\mathcal{M} = \{(l_m, \kappa_m, F_m, p_m)\}_{m\in\mathcal{M}}$.
- *Action Space*: $\mathcal{A}$ is a collection of all the actions in the VEC system that the agent can take according to the observed state. In this paper, the agent should decide where to offload the task based on the current environment state, and thus $\mathcal{A}$ can be defined as $\mathcal{A} = \{\boldsymbol{\alpha}_t\}$, $t \in \{0, 1, 2, \ldots\}$. $\boldsymbol{\alpha}_t$ is the task offloading decision the agent makes in time step $t$, and $\boldsymbol{\alpha}_t = \{\alpha_n|\forall n \in \mathcal{N}\}$ and $\alpha_n = \{\alpha_{n,m}|\forall m \in \mathcal{M}\}$.
- *Reward Function*: The agent will get an immediate reward $r_t$ after it takes the action at current state $s_t$. $r_t$ can weigh up the pros and cons of the adopted action in the current state. A larger $r_t$ means that it is more desirable to take this action than others. The immediate reward $r_t$ can be calculated by the reward function. Our problem $\boldsymbol{\mathcal{P}_1}$ is to minimize the objective function. Hence, we need to

**Algorithm 2** DRL-Based Collaborative Computation Offloading Algorithm

**Input:** $\mathcal{M}$, $\mathcal{N}$, $K$, $T$
**Output:** Task offloading decision $\boldsymbol{\alpha}$
1 Initialize the DNN parameter $\boldsymbol{\theta}$;
2 **do**
3  **for** $k = 1$ to $K$ **do**
  // Collect $K$ complete episodes
4  $t = 0$;
5  **while** $t < T$ **do**
6  Input $s_t$ to DNN;
7  Get the output vector $\boldsymbol{o}$ from DNN;
8  Transform $\boldsymbol{o}$ into the probability matrix $\mathcal{M}$;
9  Sample an action $a_t$ to generate decision $\boldsymbol{\alpha}$;
10  Obtain resource allocation policy $\boldsymbol{f}$ by Alg.1;
11  Calculate reward $r_{t+1}$ by Eq.(24);
12  Obtain the next state $s_{t+1}$;
13  $s_t = s_{t+1}$ and $t = t + 1$;
14  **end**
15  Store all the data $(s_t, a_t, r_{t+1})_{t=0}^{T-1}$ for episode $k$;
16  **end**
  // Train the network
17  **for** each episode $k$ **do**
18  **for** each entry $(s_t, a_t, r_{t+1})$ in the episode **do**
19  Input $s_t$ and get matrix $\mathcal{M}$;
20  Select corresponding values $\pi(a_t|s_t;\boldsymbol{\theta})$ from $\mathcal{M}$ given $a_t$;
21  $G_t = \sum_{k=t+1}^{T} \gamma^{k-t-1} r_k$;
22  $B_t = \frac{1}{K}\sum_{i=1}^{K} G^i_t$;
23  Update policy network by: $\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t + \eta\gamma^t(G_t - B_t)\nabla_{\boldsymbol{\theta}} \ln \pi(a_t|s_t;\boldsymbol{\theta})$;
24  **end**
25  **end**
26 **while** *Policy not converged*;

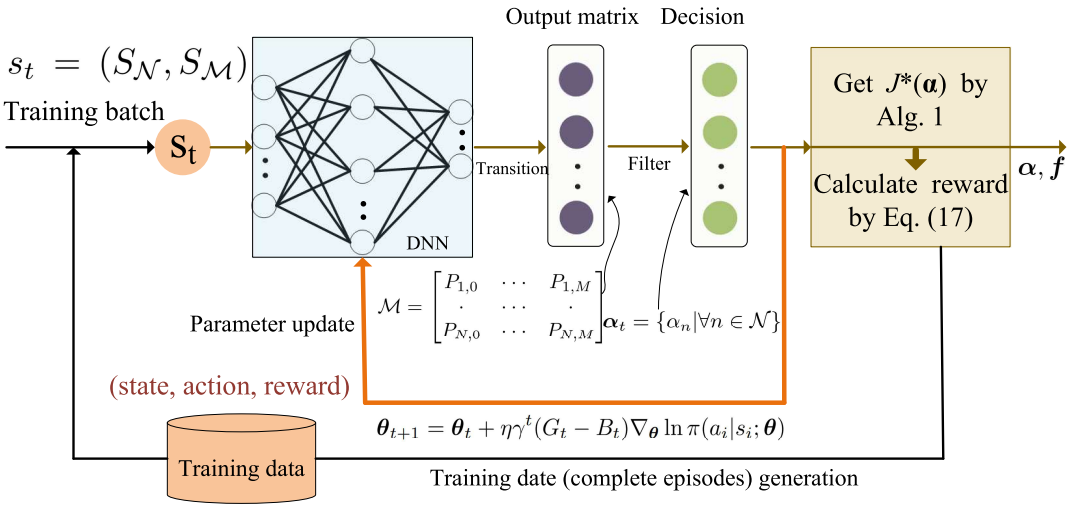



Fig. 2. The complete procedure for DRL-based collaborative computation offloading.

convert our optimization objective to a reward function. The immediate reward $r_t$ can be defined as

$$r_t = \sum_{n=1}^{N} \beta_1 d_n + \beta_2 b_n - J^*(\boldsymbol{\alpha}). \tag{24}$$

The action space $\mathcal{A}$ in this paper is of size $(M+1)^N$. Deep Q-Network (DQN) often suffers from Q-value overestimation and high-variance updates when applied to combinatorial action spaces. To address these challenges, we propose a REINFORCE-based approach in this paper, which offers advantages in policy representation efficiency and allows for the feasible integration of constraints. The whole framework of our solution is depicted in Fig. 2. A deep neural network (DNN) is adopted to generate the offloading policy $\pi$ with embedded parameters $\boldsymbol{\theta}$. The DNN takes the state $s_t$ as the input and outputs a probability distribution of actions that can be taken from $s_t$. Then, the output vector can be transformed into one $N \times (M+1)$ probability matrix $\mathcal{M}$ with row $i$ denoting the probabilities for all the actions to be taken. For example, the element $P_{i,j}$, as shown in this figure, denotes the probability of offloading the task from vehicle $i$ to the EN $j$.

The task offloading decision $\boldsymbol{\alpha}$ is constructed based on the actual actions from the training samples. Given the task offloading policy $\boldsymbol{\alpha}$, the resource allocation problem **P2** can be solved by calling the Alg. 1. Then, the immediate reward can be calculated via Eq. (24). The DNN parameters $\boldsymbol{\theta}$ can be updated towards maximizing the expected total discounted reward via the policy gradient approach. The parameter update can be formulated as:

$$\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t + \eta\gamma^t(G_t - B_t)\nabla_{\boldsymbol{\theta}} \ln \pi(a_i|s_i;\boldsymbol{\theta}), \tag{25}$$

where $\eta$ and $\gamma$ represent the learning rate and the discount factor, respectively. The total discounted reward at the $t$-th time step, denoted as $G_t$, can be computed using a Monte Carlo approach:

$$G_t = \sum_{k=t+1}^{T} \gamma^{k-t-1} r_k. \tag{26}$$

To reduce variance during neural network training, a baseline utility $B_t$ is used. It is defined as the average of the total discounted rewards at the $t$-th time step over the most recent $K$ episodes:

$$B_t = \frac{1}{K} \sum_{i=1}^{K} G_t^i, \tag{27}$$

where $G_t^i$ represents the total discounted reward at the $t$-th time step for the $i$-th episode.

Additionally, the corresponding algorithm is depicted in Alg. 2, which consists of two parts, i.e., the training data generation process (lines 3-16) and the DNN training process (lines 17-25), respectively. The training data is actually a series of complete episodes and we generate $K$ complete episodes at the data collection phase by the initial DNN. Then, the DNN training process starts by inputting these episodes. We assume each episode has $T$ time steps, and the parameter $\boldsymbol{\theta}$ can be updated within each time step. This process ceases until the policy network converges.

## V. Performance Evaluation

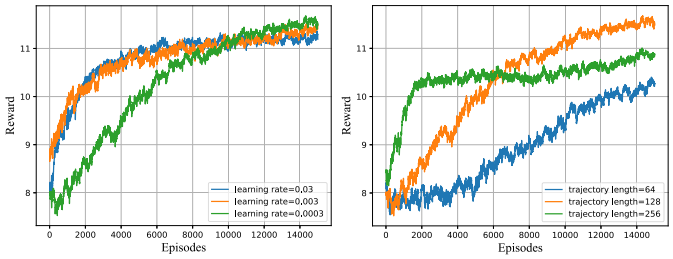
### A. Simulations Settings

The proposed CCOS optimization in VEC is extensively evaluated through simulations in this section. First, the following three baseline approaches are adopted to compare to our strategy.

- **Deep Q-network (DQN)** [29]: DQN algorithm employs neural networks to approximate the Q-function, where the current state serves as the input and the output consists of the Q values associated with all feasible actions that can be taken from this state. The state space, action space, and reward function in DQN adhere to the aforementioned definitions without further elaboration.
- **All-in-RSU approach** [30]: Since the computing capability of RSU is much more powerful than the service vehicles, all the tasks can be offloaded to RSU for execution.

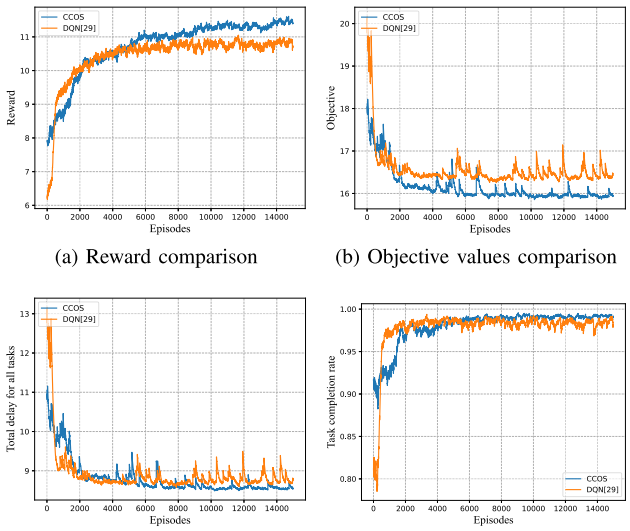
TABLE II
PARAMETER SETTINGS

| Parameter | Value | Parameter | Value |
|---|---|---|---|
| $N$ | [10, 30] | $M$ | 4 |
| $vel_n$ | $5 \sim 12$ m/s | $c_n$ | $50 \sim 100$ KB |
| $s_n$ | $200 \sim 400$ MHz | $b_n$ | [1, 2] |
| $d_n$ | $0.5 \sim 1.0$s | $\beta_1$ | 1 |
| $\beta_2$ | 0.5 | $\gamma$ | 0.9 |
| $\widetilde{p}_m$ | $0.001 \sim 0.0015$/MHz | $\widetilde{p}_{m,R}$ | 0.001/MHz |
| $F_0$ | 20 GHz | $F_m(m > 0)$ | 4 GHz |




(a) Different learning rates (b) Different trajectory length

Fig. 3. Training process under different learning rates and trajectory length.




(a) Reward comparison (b) Objective values comparison

(c) Task completion latency compar-(d) Task completion rate comparison ison

Fig. 4. Performance comparison between DRL-based approaches with different numbers of episodes.

- **Random approach**: This approach is the simplest strategy for collaborative computation offloading. The generated tasks by vehicles are offloaded to the edge nodes in $\mathcal{M}$ for execution in a random way.

In addition, parameters in the simulation are set as follows. The RSU and vehicles are randomly generated in the simulation. The communication coverage of the RSU and the vehicle is set to 100m and 40m, respectively. The number of task vehicles ranges between 10 and 30, while the default value for the number of service vehicles is 4. Other key parameters involved in the simulation are summarized in Table II. Note that the parameter settings are rigorously grounded in empirical studies and several refer to previous works [31], [32].

## B. Simulation Results and Analysis

*1) Hyperparameters Analysis:* The initial set of experiments has been conducted to assess the impact of learning rates and trajectory length on the training process. The simulation results are shown in Fig. 3, with Fig. 3(a) denoting the impact of learning rates and Fig. 3(b) denoting the impact of trajectory length, respectively. The simulation is configured with a total of 20 task vehicles and 4 service vehicles.

In Fig. 3(a), the trajectory length is set to 128, and three different values for the learning rates are investigated. Generally, the convergence of our strategy is relatively stable under different learning rates. From this figure, we can observe that the performance is the best when the learning rate is set to 0.0003 in the simulation. In Fig. 3(b), the learning rate is set to 0.0003, and three different values for the sampling trajectory length are investigated in the simulation. In comparison to the impact of learning rates, it appears that the training process is more significantly influenced by the length of the sampling trajectory. For instance, the reward increases rapidly with a trajectory length equal to 256, while it increases very slowly with a trajectory length equal to 64. Obviously, the strategy can achieve the best performance, when the trajectory length is equal to 128. Unless otherwise stated, the subsequent simulation will be conducted using a learning rate of 0.0003 and a trajectory length of 128 as the default values.

*2) Convergence Performance:* In the next, we compare our CCOS strategy with the DQN approach as the number of episodes increases. Four evaluation metrics are selected, i.e., the total reward, the value of the objective function, the task completion latency, and the task completion rate. Note that, the completion rate serves as an indicator of the VEC system's reliability, which is defined as the ratio between the number of offloaded tasks without any constraint violation and the total number of offloaded tasks. The simulation results are shown in Fig. 4, where Fig. 4(a), Fig. 4(b), Fig. 4(c), Fig. 4(d) present the comparison between the CCOS and DQN in terms of reward, objective function, task completion latency, and task completion rate, respectively. The following observations can be made based on the findings from the figure. The proposed approach in this paper is better than DQN in terms of the evaluation metrics. In particular, our approach regarding the capability to obtain the optimal objective value is much better than the DQN approach, as shown in Fig. 4(b). Generally, the decision-making ability of our approach in a dynamic environment outperforms that of DQN.

*3) Impact of Task Completion Latency:* The performance comparison regarding the average task completion latency is conducted and the simulation results are presented in Fig. 5, where the x-coordinate denotes the number of tasks and the y-coordinate denotes the average completion latency for all the tasks in the optimization period. As shown in the figure, as the number of tasks increases, the average completion latency for all the approaches increases, since the amount of computing resources allocated to individual tasks reduces. The DQN approach is better than CCOS when the number

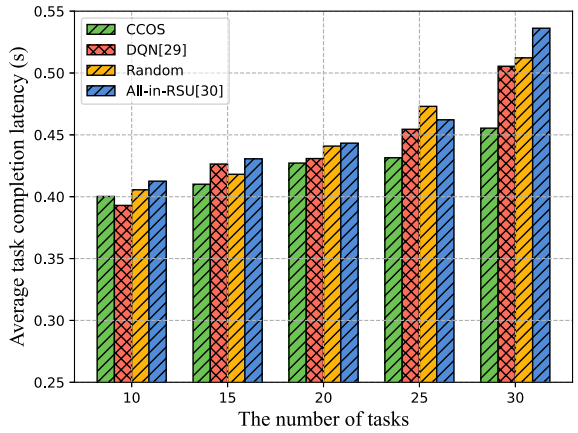



Fig. 5. Performance comparison regarding average task completion latency.

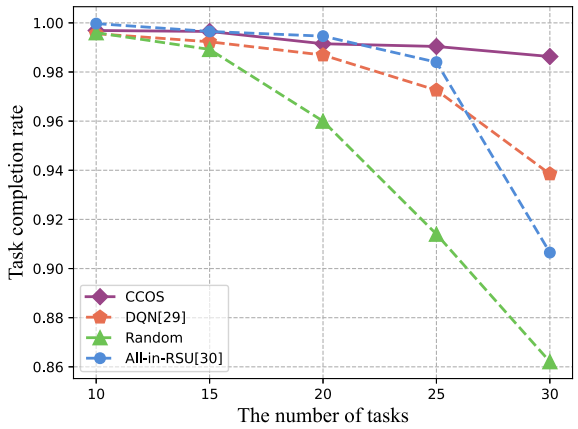



Fig. 6. Performance comparison regarding average task completion rate.

of tasks is equal to 10. When the number of vehicular tasks increases, CCOS is better than it. The policy gradient-based optimization in our approach can exhibit superior decision-making capabilities compared to the DQN approach, enabling it to effectively adapt to dynamic environments and make optimal offloading decisions with respect to the completion latency. On the other hand, the random approach is highly dynamic. When all the tasks are offloaded to the RSU for execution, the performance is the worst all the time, regardless of variations in the number of tasks. This observation can be predictable, since the all-in-RSU approach only leverages the computing capability of the RSU, ignoring the computing resources of other service vehicles.

*4) Impact of Task Completion Rate:* Then, we present the performance comparison regarding the task completion rate with different numbers of tasks and the simulation results are shown in Fig. 6. When the number of tasks is small, both the service vehicle and the RSU can meet the requirements of the tasks. This conclusion can be found in this figure, with the number of tasks ranging between 10 and 15. However, when the number of tasks increases, the random approach becomes difficult to adapt to the dynamic environment, and thus its performance gradually deteriorates. Similarly, within the computing capacity of RSU, the all-in-RSU approach can meet the requirements of the vast majority of tasks. However,

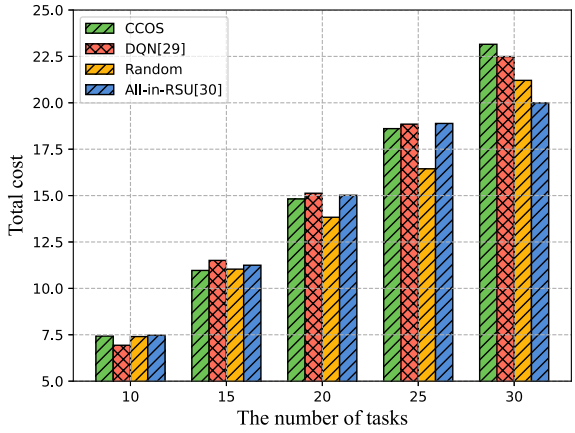



Fig. 7. Performance comparison regarding the offloading cost.

when RSU is overloaded or overwhelmed by the offloading requests, the amount of computing resources allocated to individual tasks is not enough to support the strict latency requirement of tasks, which results in a significant increase in task completion latency. Such an increment gives rise to the constraint violation, consequently leading to a decrease in the rate of task completion. CCOS exhibits a higher task completion rate compared to other approaches, primarily due to its superior adaptability in highly dynamic environments. Notably, the random approach demonstrates the lowest task completion rate.

*5) Impact of Offloading Cost:* The simulation results for performance comparison regarding the offloading cost are presented in Fig. 7. In general, the more the allocation of computing resources to a task, the lower its completion latency and the higher its offloading costs. Although service vehicles offer a discounted service price based on the number of received tasks, the initial unit price is higher than that of RSU. In addition, due to the limitation of computing resources for service vehicles, it is impossible to offload all the tasks to the same service vehicle. For the weights of response latency and offloading costs in the objective function, we set $\beta_1$ and $\beta_2$ to 1 and 0.5, respectively, which means that we pay more attention to the response delay (i.e., the task completion latency). Accordingly, more computing resources are allocated among these tasks to satisfy their latency requirements, thus bringing about more computing costs. Therefore, we can observe from the figure that the offloading costs of CCOS and the DQN approach are much higher than the other two approaches. There is no consistent pattern between our approach and the DQN approach. It takes less cost for the random approach and the all-in-RSU approach.

*6) Impact of Objective Values:* The simulation results for performance comparison regarding the objective values are presented in Fig. 8. While the advantages of DRL-based approaches may not be apparent when dealing with a small number of tasks, CCOS demonstrates increasingly prominent benefits in terms of the objective function and exhibits a higher task completion rate as discussed earlier, when the number of tasks increases. It shall be noted that the random approach seems to have the best optimal values as shown

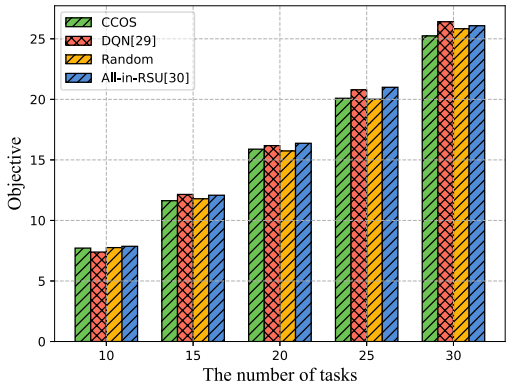
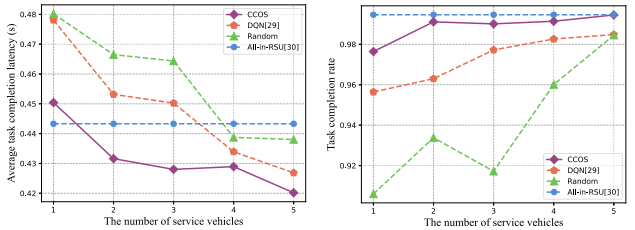
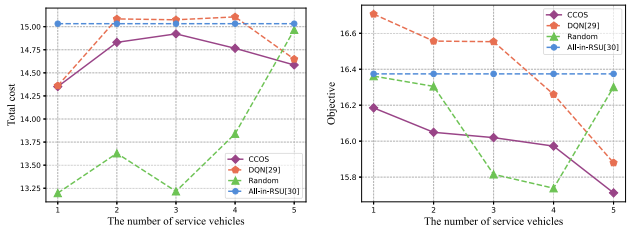



Fig. 8. Performance comparison regarding the objective values.




(a) Average task completion latency comparison (b) Task completion rate comparison




(c) The offloading costs comparison (d) Objective values comparison

Fig. 9. Performance comparison with different numbers of service vehicles.

in the simulation results. However, this observation does not align with objective facts for the following reasons. As shown earlier in Fig. 6, the random approach has the worst task completion rate, which means that the tasks in this approach may take less offloading costs to accomplish the task, as shown in Fig. 7. The reduction in offloading costs may lead to the reduction of objective values, as the offloading cost as the second component constitutes the objective function.

*7) Impact of the Number of Service Vehicles:* The performance comparison for different approaches regarding the number of service vehicles is conducted and the simulation results are presented in Fig. 9. Similar to the earlier simulation configuration, we also select the task completion latency, the task completion rate, the objective values, and the offloading costs as the evaluation metrics for different approaches.

The average task completion latency and completion rate of the four approaches with different numbers of service vehicles are presented in Fig. 9(a) and Fig. 9(b), respectively. The simulation is conducted with a fixed number of 20 task vehicles. First of all, the average task completion latency and task completion rate in the all-in-RSU approach remain unchanged, since this approach does not offload vehicular tasks to any service vehicles. Also, we can observe that the average task completion latency for other approaches gradually decreases, with the increasing number of service vehicles. This can be attributed to the increased contribution of computing resources from newly joined service vehicles. On the other hand, the growth in computing resources also mitigates the pressure of RSU and existing service vehicles, which directly improves the reliability of the VEC system, e.g., the task completion rate for the approaches gradually increases as the number of service tasks increases.

Fig. 9(c) presents the comparison of offloading costs with the increasing number of service vehicles. Both the CCOS and DQN approaches increase first and then begin to decrease. The increment of the offloading costs at the beginning can be attributed to the fact that more service vehicles bring more computing resources that can be allocated to the task execution. Hence, the offloading costs increase. The latter reduction is due to the fact that more service vehicles will effectively alleviate the burden on task-input data transmission and consequently reduce transmission delay. As a result, the overall response latency can be reduced, thereby mitigating the urgent demand for computing resources to some extent.

Fig. 9(d) presents the comparison of the objective values with the increasing number of service vehicles. Firstly, both our approach and the DQN approach exhibit a gradual decrease in objective values, indicating their ability to adapt to environmental changes. However, our approach demonstrates superior decision-making capabilities compared to the DQN approach. In contrast, the random approach exhibits significant fluctuations. Generally, increasing the number of service vehicles can enhance resource allocation, improve system efficiency, reduce task completion latency, and increase task completion rates. Nevertheless, employing the random approach leads to substantial fluctuations and cannot make optimal offloading decisions.

### C. Engineering Applications

The Internet of Vehicles (IoV), as the most pivotal application of 5G+ infrastructure, has recently received strong support from national policies in China. The country is actively promoting the coordinated development of "5G+ vehicle networking", integrating it into the national new information infrastructure construction project, and further encouraging the widespread implementation of LTE-V2X. Furthermore, the collaborative advanced engineering team established by renowned communication technology corporation Verizon and automaker Nissan has successfully demonstrated the processing of data collected from vehicles and RSUs at the edge of Verizon's wireless network, enabling near real-time delivery of results back to the vehicles. Such advancements in the engineering field accelerate the practical application of theoretical results revolving around task offloading and resource allocation in VEC systems.

## VI. Conclusion and Future Work

The integration of high-end multi-core processors into service vehicles not only enables local computation but also facilitates resource provisioning in a pay-as-you-go manner. Hence, in addition to RSU as the offloading destination,

vehicles with abundant computing resources can also serve as ENs for profit-seeking purposes. In this paper, we consider collaborative computation offloading in the VEC system, in which the service pricing fluctuates according to the dynamic environments such as the offloading requests. We strive to minimize the weighted sum of the task completion latency and the offloading costs for all the tasks in the optimization period. Specifically, a DRL-based algorithm is adopted to solve the task offloading problem. The simulation results demonstrate significant advantages over the comparison baselines in terms of optimal values, task completion rate, and other evaluation metrics.

For future work, we plan to integrate a more complex VEC environment where multiple RSUs and many vehicles with high mobility can serve as ENs to contribute their idle computing resources.

## REFERENCES


[1] C. Tang, H. Wu, and S. Xiao, "Lightweight reputation management for multi-role Internet of Vehicles," *IEEE Internet Things Mag.*, vol. 6, no. 2, pp. 38–42, Jun. 2023.

[2] Z. Wang, D. Zhao, M. Ni, L. Li, and C. Li, "Collaborative mobile computation offloading to vehicle-based cloudlets," *IEEE Trans. Veh. Technol.*, vol. 70, no. 1, pp. 768–781, Jan. 2021.

[3] K. Mishra, G. N. V. Rajareddy, U. Ghugar, G. S. Chhabra, and A. H. Gandomi, "A collaborative computation and offloading for compute-intensive and latency-sensitive dependency-aware tasks in dew-enabled vehicular fog computing: A federated deep Q-learning approach," *IEEE Trans. Netw. Service Manage.*, vol. 20, no. 4, pp. 4600–4614, Dec. 2023.

[4] H. Wu, L. Tian, H. Tang, R. Li, and P. Jiao, "Graph convolutional reinforcement learning-guided joint trajectory optimization and task offloading for aerial edge computing," *IEEE Trans. Intell. Transp. Syst.*, pp. 1–12, 2024.

[5] T. H. Binh, D. B. Son, H. Vo, B. M. Nguyen, and H. T. T. Binh, "Reinforcement learning for optimizing delay-sensitive task offloading in vehicular edge–cloud computing," *IEEE Internet Things J.*, vol. 11, no. 2, pp. 2058–2069, Jan. 2024.

[6] P. Li, Z. Xiao, X. Wang, K. Huang, Y. Huang, and H. Gao, "EPtask: Deep reinforcement learning based energy-efficient and priority-aware task scheduling for dynamic vehicular edge computing," *IEEE Trans. Intell. Vehicles*, vol. 9, no. 1, pp. 1830–1846, Jan. 2024.

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

[8] W. Y. B. Lim et al., "Decentralized edge intelligence: A dynamic resource allocation framework for hierarchical federated learning," *IEEE Trans. Parallel Distrib. Syst.*, vol. 33, no. 3, pp. 536–550, Mar. 2022.

[9] H. Shen, Y. Tian, T. Wang, and G. Bai, "Slicing-based task offloading in space-air-ground integrated vehicular networks," *IEEE Trans. Mobile Comput.*, vol. 23, no. 5, pp. 4009–4024, May 2024.

[10] H. Tang, M. Du, H. Wu, P. Jiao, and R. Li, "TLCO: Topological link-aware task co-offloading method for joint V2V and V2I system," *IEEE Trans. Intell. Transp. Syst.*, pp. 1–13, 2025.

[11] J. Du, H. Wu, M. Xu, and R. Buyya, "Computation energy efficiency maximization for NOMA-based and wireless-powered mobile edge computing with backscatter communication," *IEEE Trans. Mobile Comput.*, vol. 23, no. 6, pp. 6954–6970, Jun. 2024.

[12] M. Goudarzi, H. Wu, M. Palaniswami, and R. Buyya, "An application placement technique for concurrent IoT applications in edge and fog computing environments," *IEEE Trans. Mobile Comput.*, vol. 20, no. 4, pp. 1298–1311, Apr. 2021.

[13] S. Mittal, R. K. Dudeja, R. S. Bali, and G. S. Aujla, "A distributed task orchestration scheme in collaborative vehicular cloud edge networks," *Computing*, vol. 106, no. 4, pp. 1151–1175, Apr. 2024.

[14] L. Liu, J. Feng, X. Mu, Q. Pei, D. Lan, and M. Xiao, "Asynchronous deep reinforcement learning for collaborative task computing and on-demand resource allocation in vehicular edge computing," *IEEE Trans. Intell. Transp. Syst.*, vol. 24, no. 12, pp. 15513–15526, Dec. 2023.

[15] X. Chen, J. Zhang, B. Lin, Z. Chen, K. Wolter, and G. Min, "Energy-efficient offloading for DNN-based smart IoT systems in cloud-edge environments," *IEEE Trans. Parallel Distrib. Syst.*, vol. 33, no. 3, pp. 683–697, Mar. 2022.

[16] Y. Sun, Z. Wu, K. Meng, and Y. Zheng, "Vehicular task offloading and job scheduling method based on cloud-edge computing," *IEEE Trans. Intell. Transp. Syst.*, vol. 24, no. 12, pp. 14651–14662, Dec. 2023.

[17] Z. Li, C. Yang, X. Huang, W. Zeng, and S. Xie, "CoOR: Collaborative task offloading and service caching replacement for vehicular edge computing networks," *IEEE Trans. Veh. Technol.*, vol. 72, no. 7, pp. 9676–9681, Jul. 2023.

[18] J. Liu, N. Liu, L. Liu, S. Li, H. Zhu, and P. Zhang, "A proactive stable scheme for vehicular collaborative edge computing," *IEEE Trans. Veh. Technol.*, vol. 72, no. 8, pp. 10724–10736, Aug. 2023.

[19] L. Liu and Z. Chen, "Joint optimization of multiuser computation offloading and wireless-caching resource allocation with linearly related requests in vehicular edge computing system," *IEEE Internet Things J.*, vol. 11, no. 1, pp. 1534–1547, Jan. 2024.

[20] X. Chen, S. Hu, C. Yu, Z. Chen, and G. Min, "Real-time offloading for dependent and parallel tasks in cloud-edge environments using deep reinforcement learning," *IEEE Trans. Parallel Distrib. Syst.*, vol. 35, no. 3, pp. 391–404, Mar. 2024.

[21] B. Cao, Z. Li, X. Liu, Z. Lv, and H. He, "Mobility-aware multiobjective task offloading for vehicular edge computing in digital twin environment," *IEEE J. Sel. Areas Commun.*, vol. 41, no. 10, pp. 3046–3055, Oct. 2023.

[22] J. Du, Y. Sun, N. Zhang, Z. Xiong, A. Sun, and Z. Ding, "Cost-effective task offloading in NOMA-enabled vehicular mobile edge computing," *IEEE Syst. J.*, vol. 17, no. 1, pp. 928–939, Mar. 2023.

[23] B. Hu, Y. Shi, and Z. Cao, "Adaptive energy-minimized scheduling of real-time applications in vehicular edge computing," *IEEE Trans. Ind. Informat.*, vol. 19, no. 5, pp. 6895–6906, May 2023.

[24] Y. Sahni, J. Cao, L. Yang, and Y. Ji, "Multi-hop multi-task partial computation offloading in collaborative edge computing," *IEEE Trans. Parallel Distrib. Syst.*, vol. 32, no. 5, pp. 1133–1145, May 2021.

[25] L. Liu, M. Zhao, M. Yu, M. A. Jan, D. Lan, and A. Taherkordi, "Mobility-aware multi-hop task offloading for autonomous driving in vehicular edge computing and networks," *IEEE Trans. Intell. Transp. Syst.*, vol. 24, no. 2, pp. 2169–2182, Feb. 2023.

[26] W. Miao, G. Min, X. Zhang, Z. Zhao, and J. Hu, "Performance modelling and quantitative analysis of vehicular edge computing with bursty task arrivals," *IEEE Trans. Mobile Comput.*, vol. 22, no. 2, pp. 1129–1142, Feb. 2023.

[27] J. Zhou and X. Zhang, "Fairness-aware task offloading and resource allocation in cooperative mobile-edge computing," *IEEE Internet Things J.*, vol. 9, no. 5, pp. 3812–3824, Mar. 2022.

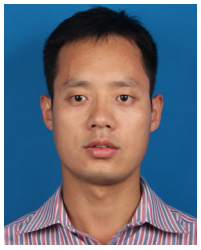
[28] T. X. Tran and D. Pompili, "Joint task offloading and resource allocation for multi-server mobile-edge computing networks," *IEEE Trans. Veh. Technol.*, vol. 68, no. 1, pp. 856–868, Jan. 2019.

[29] L. Zhang et al., "DQN-based mobile edge computing for smart Internet of Vehicle," *EURASIP J. Adv. Signal Process.*, vol. 2022, no. 1, p. 45, Dec. 2022.

[30] W. Zhou et al., "Profit maximization for cache-enabled vehicular mobile edge computing networks," *IEEE Trans. Veh. Technol.*, vol. 72, no. 10, pp. 13793–13798, Oct. 2023.

[31] C. Tang, Y. Ding, S. Xiao, Z. Huang, and H. Wu, "Collaborative service caching, task offloading, and resource allocation in caching-assisted mobile edge computing," *IEEE Trans. Services Comput.*, vol. 18, no. 4, pp. 1966–1981, Jul. 2025.

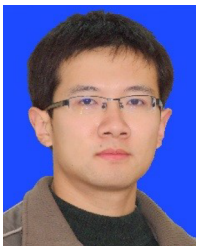
[32] C. Tang, Z. Li, S. Xiao, H. Wu, and W. Chen, "A bandwidth-fair migration-enabled task offloading for vehicular edge computing: A deep reinforcement learning approach," *CCF Trans. Pervas. Comput. Interact.*, vol. 6, no. 3, pp. 255–270, Sep. 2024.

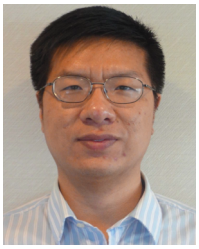
**Chaogang Tang** (Member, IEEE) received the B.S. degree from Nanjing University of Aeronautics and Astronautics, Nanjing, China, and the joint Ph.D. degree from the School of Information Science and Technology, University of Science and Technology of China, Hefei, China, and the Department of Computer Science, City University of Hong Kong, in 2012. He is currently with China University of Mining and Technology. His research interests include mobile cloud computing, fog computing, the Internet of Things, and big data.

**Zhao Li** received the B.S. degree in computer science and technology from Henan University of Science and Technology, China, in 2022. He is currently pursuing the M.S. degree with the School of Computer Science and Technology, China University of Mining and Technology. His current research interests include vehicular edge computing and deep reinforcement learning.

**Huaming Wu** (Senior Member, IEEE) received the B.E. and M.S. degrees in electrical engineering from Harbin Institute of Technology, China, in 2009 and 2011, respectively, and the Ph.D. degree (Hons.) in computer science from Freie Universität Berlin, Germany, in 2015. He is currently a Professor at the Center for Applied Mathematics, Tianjin University, China. His research interests include mobile cloud computing, edge computing, the Internet of Things, deep learning, complex networks, and DNA storage.



**Shuo Xiao** received the Ph.D. degree in traffic information engineering and control from Beijing Jiaotong University in 2010. He has been with China University of Mining and Technology since 2010, where he is currently a Professor. His research interests include the Internet of Things and measurement systems.



**Ruidong Li** (Senior Member, IEEE) received the M.Sc. and Ph.D. degrees in computer science from the University of Tsukuba in 2005 and 2008, respectively. He is currently an Associate Professor at Kanazawa University, Japan. Before joining this university, he was a Senior Researcher at the National Institute of Information and Communications Technology (NICT), Japan. His research interests include future networks, big data, intelligent internet edge, the Internet of Things, network security, information-centric network, artificial intelligence, quantum internet, cyber-physical systems, and wireless networks. He is a member of IEICE. He serves as the Secretary for the IEEE ComSoc Internet Technical Committee (ITC) and the Founder and Chair for the IEEE SIG on Big Data Intelligent Networking and the IEEE SIG on Intelligent Internet Edge. He also served as the chairs for several conferences and workshops, such as the General Co-Chair for IEEE MSN 2021, AIVR2019, and IEEE INFOCOM 2019/2020/2021 ICCN Workshop, and the TPC Co-Chair for IWQoS 2021, IEEE MSN 2020, BRAINS 2020, IEEE ICDCS 2019/2020 NMIC Workshop, and ICCSSE 2019. He is an Associate Editor of IEEE INTERNET OF THINGS JOURNAL and also served as the Guest Editor for a set of prestigious magazines, transactions, and journals, such as *IEEE Communications Magazine*, *IEEE Network*, IEEE TRANSACTIONS ON NETWORK SCIENCE AND ENGINEERING.