Hindawi
Mobile Information Systems
Volume 2022, Article ID 9230521, 11 pages
https://doi.org/10.1155/2022/9230521

Research Article

Joint Optimization for MEC Computation Offloading and Resource Allocation in IoV Based on Deep Reinforcement Learning

Jian Wang,[1] **Yancong Wang**,[1] **and Hongchang Ke**[2]

[1]*College of Computer Science and Technology, Jilin University, Changchun 130012, China*
[2]*School of Computer Technology and Engineering, Changchun Institute of Technology, Changchun 130012, China*

Correspondence should be addressed to Hongchang Ke; kehongchang1981@163.com

Received 18 June 2022; Accepted 13 July 2022; Published 13 August 2022

Academic Editor: Konglin Zhu

In the Internet of Vehicle (IoV), the limited computing capacity of vehicles hardly processes the intensive computation tasks locally. The computation tasks can be offloaded to multiaccess edge computing (MEC) servers for processing, where MEC provides the required computing capacity to the nearby vehicles. In this paper, we consider a scenario where there are cooperation and competition between vehicles, the offloading decision of any vehicle will affect the decisions of the others, and the computing resource allocation strategies by MEC will dynamically change. Therefore, we propose a joint optimization scheme for computation offloading decisions and computing resource allocation based on decentralized multiagent deep reinforcement learning. The proposed scheme learns the optimal actions to minimize the total weighted cost which is designed as the vehicles' satisfaction based on the type of stochastic arrival tasks and dynamic interaction between MEC server and vehicles within different RSUs coverages. The numerical results show that the proposed algorithms based on decentralized multiagent deep deterministic policy gradient (DDPG) which is named De-DDPG can autonomously learn the optimal computation offloading and resource allocation policy without a priori knowledge and outperform the other three baseline algorithms in terms of the rewards.

1. Introduction

With the development of wireless communication technology and the rapid growth of vehicles, Internet of Vehicle (IoV) has become one of the most important applications of the Internet of Things (IoT) [1, 2]. However, due to the limitation of the computing resource of the vehicles, several tasks cannot be executed locally within the required delay [3]. To solve this problem, offloading IoV tasks to mobile edge computing (MEC) server is proposed as a feasible solution [4]. MEC is close proximity to the mobile vehicles, supplying more sufficient computation resource to the offloaded tasks [5, 6].

In recent years, many researches regarding MEC computation offloading in IoV have been studied [7, 8]. Some researchers have conducted to develop the optimization scheme in computation offloading under certain constraints, such as reducing the delay, computation resource overhead, and energy consumption [9–11]. Moreover, computation congestion that affects the performance of MEC server and the load balance of computation resource among MEC server have been considered into the computation offloading problem [12]. Although MEC server provides vehicles with the resource far beyond theirs, the resource of the MEC server may also be insufficient when massive vehicles access to MEC server simultaneously. Therefore, the rational resource allocation to optimize the performance of various objectives is also a significant issue of edge computing offloading [13–15]. Because of the high-speed mobility of vehicles and the randomness of tasks, as well as the cooperation and competition between vehicles in IoV, the computation resource allocation policy of MEC server based on different offloading decisions of vehicles has been discussed by many researchers. By jointly optimizing resource allocation and offloading strategy in IoV, the overall cost of computation resource, energy, and the delay is minimized in [16–18]. However, these methods require a large number of iterations to obtain a satisfied local optimum, which is not suitable for application

scenarios where the environment changes rapidly and decisions need to be made in real time. Meanwhile, solving this type of optimization problem is usually nonconvex and NP-hard.

Deep reinforcement learning (DRL) which is the combination of deep learning (DL) and reinforcement learning (RL) can tackle the nonconvex optimization problem and has been widely used as an effective approach to optimize different issues including offloading decision-making and resource allocation strategy [14, 19–23]. The previous works make many efforts to optimize task offloading in IoV. For example, deep Q-network (DQN) is adopted in multiple vehicles offloading system to obtain the optimized offloading decisions which maximize the QoS of digital twinning-empowered IoV system [23]. Similar work proposes multiagent DQN-based computation offloading scheme, in which the uncertainty environment is considered so that the vehicles can make offloading decisions to achieve an optimal long-term reward [24]. A dynamic task offloading scheme based on Q-learning is implemented to minimize the delay, energy consumption, and total overhead in IoV system [25]. URLLC-aware task offloading algorithm based on deep Q-learning is studied to maximize the throughput of vehicles with satisfied constraints in [26]. Jointly considering the task priority, vehicles' service availability, and computation resource sharing incentive, an optimal offloading policy based on soft actor-critic (SAC) maximizes both expected reward and the policy entropy of the offloading tasks in the dynamic vehicular environment [27]. Moreover, DQN-based joint computation offloading and task migration optimization are applied to minimizing the total system cost in a 5G vehicle-aware MEC network [28]. The two-stage scheme is designed to joint optimization, where DQN is used in the first step to obtain the offloading strategy and deep deterministic policy gradient (DDPG) is utilized to generate the transmit power determination strategy of the vehicles [29]. None of the above researches consider the joint optimization of offloading strategy and computation resource allocation when multiple agents interact in a dynamic IoV environment.

Different from the existing works, we propose a decentralized multiagent deep reinforcement learning-based method to solve the joint optimization of computation offloading decision and resource allocation for MEC server in IoV. The objective of our work is to minimize the weighted cost of multiagent. In summary, our main contributions are as follows:

(1) We propose a IoV scenario supported by MEC server for dynamic task offloading decision and computation resource allocation in the environment with multiple RSUs cover multiple vehicles. In this cooperative scenario, because of the mobility of multivehicle and the stochastic arrival tasks, the computation offloading decision and resource allocated to multiple RSUs and multiple vehicles change in different time slots.

(2) Based on the proposed model, we consider both offloading decision-making and computation resource allocation to gain the minimum weighted cost, which is related to the end-to-end delay and computation resource cost. Moreover, we formulate the problem as a Markov decision process (MDP) and design the state, action, and reward functions.

(3) In order to effectively solve the abovementioned problem with continuous variables and meet the requirement of convergence, a joint optimization scheme based on decentralized multiagent DDPG (De-DDPG) is proposed. The simulation results show that the convergence of our proposed algorithm is verified and our proposed algorithm has better performance than other three baseline algorithms.

The remaining of this paper is organized as follows: In Section 2, an MEC framework with multiple RSUs and vehicles is introduced, and we construct the network model, communication model, and computation model. Section 3 describes the problem statement of the joint optimization. The solution based on decentralized multiagent DDPG (De-DDPG) is proposed in Section 4. In Section 5, the simulation results and analysis are presented. Finally, we conclude this paper in Section 6.

2. System Model

2.1. Network Model. A three-layer Internet of Vehicle (IoV) is considered in this paper (see Figure 1), which consists of an MEC server, M roadside units (RSUs), and N vehicles on a multilane road of length L.

The MEC server is connected to M RSUs via the fiber-optic link for receiving and transmitting the computation tasks. We assume that the total computing resource of MEC server is denoted as F. The RSUs denoted by $\mathcal{M} = \{1, 2, \ldots, M\}$ locate along the road with the same coverage range l. Therefore, we divide the road into M segments, and all vehicles are randomly and independently distributed in the segments with arrival rate λ. The RSU is responsible for forwarding messages between the MEC server and the vehicles. A set of vehicles periodically send messages to RSU within its communication range, which is denoted as $= \{1, 2, \ldots, N\}$. Vehicles have the same local computing capacity which is determined by the onboard unit (OBU) [30]. For each vehicle-i, it sends not only task messages but also its driving characteristics $\{p_i, v_i\}$, where p_i and v_i represent its 1-D position and speed, respectively. Here, we assume that the distances between vehicles follow the exponential distribution and the speeds of the vehicles are truncated Gaussian distributed, which is more appropriate for the actual situation of the road [31, 32]. In addition, we assume that each vehicle only processes one computation task within the current time period. The computation task of each vehicle is denoted as $T_i = \{C_i, D_i^{\text{in}}, D_i^{\text{out}}, t_i^{\max}\}$, where C_i is the required computation capacity to complete the task, D_i^{in} and D_i^{out} are the data size of the input and output for computing, respectively, and $t_i^{\max}$ is the maximum tolerable delay for the task completion. Vehicle needs to execute a computation task within a tolerable time period, and the task can be either processed locally or offloaded to the MEC server. We define the binary offloading strategy of vehicles as $\mathcal{X} = \{x_i | x_i \in \{0, 1\}, i \in N\}$, where $x_i = 0$ and $x_i = 1$ means

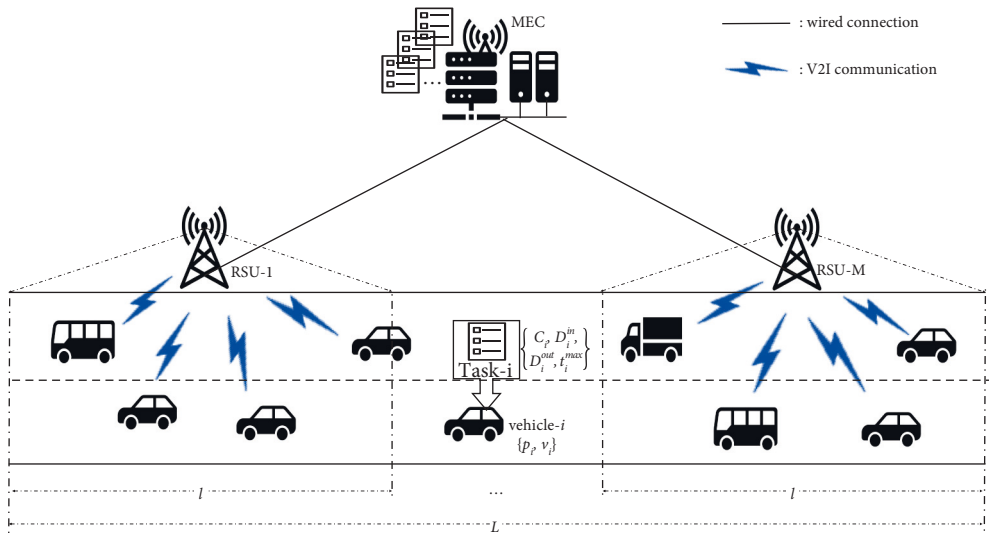

Figure 1: The system model of the MEC server and multivehicles.

that the vehicle-i decides to execute the computation tasks locally or offloads the task to the MEC server, respectively.

Moreover, when the vehicle leaves the coverage of the RSU, the vehicle will be disconnected from the RSU and can no longer transmit data to the MEC server through the RSU. The time available by the vehicle before leaving the communication range of RSU-j, $j = \lceil p_i/l \rceil, j \in M$, i.e., sojourn time, can be given as

$$t_{ij}^{soj} = \frac{l\lceil p_i/l \rceil - p_i}{v}, \qquad v = \frac{L\lambda}{N}, \tag{1}$$

where v represents the vehicle's equivalent speed and $\min\{v_i\} \le v \le \max\{v_i\}, i \in N$.

2.2. Communication Model. When the vehicles decide to offload the task to MEC server, the vehicles will transmit data to MEC server through the RSUs. Generally, the propagation time of the fiber-optic transmission between RSUs and the MEC server can be ignored [12]. We consider the V2I communication between the vehicle and the RSU is based on IEEE 802.11p in this work [33]. According to [34], the uplink and downlink transmitting rate $(r_{ij}^{UL}, r_{ij}^{DL})$ of the wireless communication between vehicle-i and its belonged RSU-j is expressed as

$$r_{ij}^{UL/DL} = \frac{D_i^{\text{in}}/D_i^{\text{out}} N_j \tau_{ij}\left(1-\tau_{ij}\right)^{N_j-1}}{\left(1-\tau_{ij}\right)^{N_j}\sigma + T_{ij}^{\text{success}} N_j \tau_{ij}\left(1-\tau_{ij}\right)^{N_j-1} + 1 - \left(1-\tau_{ij}\right)^{N_j} - N_j\tau_{ij}\left(1-\tau_{ij}\right)^{N_j-1}(\text{RTS} + \text{AIFS} + \delta)}. \tag{2}$$

where N_j is the number of vehicles which decide to offload task to MEC server via RSU-j. τ_{ij} represents the probability that vehicle-i connects to the RSU-j in a random time slot. σ is the duration of a time slot. RTS stands for request to send interval, AIFS denotes the arbitration inter-frame spacing interval, and δ expresses the propagation delay. T_{ij}^{success} is defined as the success transmission period between vehicle-i and RSU-j, which is written as

$$T_{ij}^{\text{success}} = \Phi + \frac{D_i^{\text{in}}/D_i^{\text{out}}}{\omega_j \log\left(1 + P_i h_{ij}\right)}, \tag{3}$$

where Φ is specific to the MAC protocol, and it equals $H + \text{SIFS} + \delta$ $+\text{ACK} + \text{AIFS} + \delta + \text{RTS} + \text{SIFS} + \delta + \text{CTS} + \text{SIFS} + \delta$. $H = \text{PHY}_{\text{head}} + \text{MAC}_{\text{head}}$ represents the packet header's overhead. SIFS, ACK, and CTS stand for short interface space interval, acknowledgment interval, and CTS interval, respectively. ω_j denotes the bandwidth of RSU-j, P_i is the transmission power of vehicle-i, and h_{ij} stands for the channel gain between vehicle-i and RSU-j.

The uplink/downlink transmitting time under this situation is calculated as

$$t_{\text{UL/DL}}^{\text{mec}} = \frac{D_i^{in}/D_i^{\text{out}}}{r_{ij}^{\text{UL/DL}}}. \tag{4}$$

And the two-way transmission time between vehicle and RSU is given by

$$t_{\text{trans}}^{\text{mec}} = t_{\text{UL}}^{\text{mec}} + t_{\text{DL}}^{\text{mec}}. \tag{5}$$

2.3. Computation Model. The processing time is considered under two situations: the task is processed locally, and the task is offloaded to the MEC server for computing.

2.3.1. Local Processing Model. When vehicle-i processes its computation task locally ($x_i = 0$), the processing time of vehicle-i t_i^{loc} is only dependent on its own computing capacity. The local execution time $t_{\text{exe}}^{\text{loc}}$ is formulated as

$$t_i^{\text{loc}} = t_{\text{exe}}^{\text{loc}} = \frac{C_i}{f_{\text{loc}}}. \tag{6}$$

Here, f_{loc} is denoted as the vehicle's computation capacity, which is related to the vehicle's CPU cycle frequency.

2.3.2. MEC Processing Model. When the task is offloaded to the MEC server ($x_i = 1$), the end-to-end delay of vehicle-i includes the task execution time and the transmitting time. The execution time of vehicle-i offloading the task to MEC server is given as

$$t_{\text{exe}}^{\text{mec}} = \frac{C_i}{f_j^{\text{mec}}}, \tag{7}$$

where f_j^{mec} denotes the computation capacity assigned to RSU-j which connects to vehicle-i by the MEC server, and f_j^{mec} denotes the allocated CPU cycle frequency of RSU-j by the MEC server. The end-to-end delay between vehicle-i and the MEC server is obtained by

$$t_i^{\text{mec}} = t_{\text{exe}}^{\text{mec}} + t_{\text{trans}}^{\text{mec}}. \tag{8}$$

The main notations and descriptions are described in Table 1.

3. Problem Statement

In this section, the optimization problem is formulated by jointly considering the offloading decision and resource allocation with the aim of load balance and system cost minimization. First of all, we define the cost function as follows.

Cost function is considered to quantify the satisfaction level of the vehicle's offloading decision, which is inversely related to the satisfaction and identified by the delay sensitivity and the cost of computation resource. The logarithmic function is known as proportional fairness in many researches [35], which can achieve load balance, and a logarithm function is used to represent the cost function in this paper. The processing delay of a task is generally considered to be inversely proportional to the satisfaction; that is, the shorter the task processing delay, the higher the satisfaction. In addition, if the task is completed within the maximum tolerable delay, the satisfaction of the vehicle should be non-negative. But once the completion processing time of the task exceeds its maximum tolerable delay, the processing result of the task will lose its value because the tasks in IoV are extremely tolerant of delays. Here, the penalty mechanism is brought into consideration. Another metric in the cost function is the computation resource cost. It is necessary to pay for the vehicle's computation resource when the vehicle processes the task locally. Furthermore, when the task is offloaded to the MEC server, it takes the vehicle's corresponding cost for computation resources allocated by the MEC server, which will also reduce the satisfaction of the vehicle. Therefore, the cost function for vehicle-i to process the task locally is given by

$$U_i^l = \begin{cases} \beta \log\left(1 + \left(t_i^{\text{actual}} - t_i^{\text{loc}}\right)^+\right) + (1-\beta)\rho f_{\text{loc}}, & t_i^{\text{loc}} \le t_i^{\max}, \\ P, & t_i^{loc} > t_i^{\max}, \end{cases} \tag{9}$$

where $\beta \in (0, 1)$ and $1 - \beta$ represent the weights of delay and computation resource cost, respectively. The weighted function provides a flexible scheme for different applications' specific requirements by adjusting the weight parameters. $(z)^+ = \max(z, 0)$ ensures that U_i^l is non-negative. ρ is the unit cost of the computing resource. And, $P > 0$ represents the penalty for the task that is not completed within its maximum tolerable delay.

Similarly, the cost function of vehicle-i offloaded the task to the MEC server for processing and can be expressed as

$$U_i^{\text{mec}} = \begin{cases} \beta \log\left(1 + \left(t_i^{\text{actual}} - t_i^{\text{mec}}\right)^+\right) + (1-\beta)\rho f_i^{\text{mec}}, & t_i^{\text{mec}} \le t_i^{\text{actual}}, \\ P, & t_i^{\text{mec}} > t_i^{\text{actual}}. \end{cases} \tag{10}$$

Because when the vehicle leaves the coverage of the RSU, the vehicle will disconnect to the MEC server through the RSU regardless of whether the task is processed or not. $t_i^{\text{actual}} = \min\left\{t_{ij}^{soj}, t_i^{\max}\right\}$ is used to depict the task's actual tolerant delay.

Combining equations (9) and (10), the cost function of vehicle-i can be expressed as

$$U_i = \begin{cases} U_i^{\text{loc}}, & \text{if } x_i = 0, \\ U_i^{\text{mec}}, & \text{if } x_i = 1. \end{cases} \tag{11}$$

This work aims to minimize the system cost by jointly determining the offloading decisions of vehicles and the computation resource allocation of the MEC server. The optimization problem is formulated as

$$\begin{aligned} &\min_{\mathcal{X}, \mathcal{F}} \quad \sum_{i=1}^{N} U_i, \\ &s.t. C1: \ 0 \le f_{\text{loc}} < F, \\ &C2: \ 0 \le f_i^{\text{mec}} \le x_i F, \forall i \in \mathcal{N}, \forall j \in \mathcal{M}, \\ &C3: \ \sum_{j=1}^{M} f_i^{\text{mec}} \le F, j \in \mathcal{M}, \\ &C4: \ x_i = \{0, 1\}, i \in N. \end{aligned} \tag{12}$$

Constraint C1 ensures that the available local computation resource is non-negative and less than the MEC server. C2 is the constraint of the available computation resource assigned for each vehicle-i within the coverage of RSU-j by the MEC server. The sum of the computation resource allocated to all the offloading tasks through RSU-j does not exceed the total computation resource of the MEC

Table 1: Notation description.

Notation	Description
L	The length of the selected road
l	The coverage range of the RSU
$\mathscr{M}, M$	Set/number of RSUs
$\mathscr{N}, N$	Set/number of vehicles
i,j	The vehicle index $i \in \mathscr{N}$/the RSU index $j \in \mathscr{M}$
λ	The arrival rate of vehicles
p_i, v_i	Vehicle's position/speed
$C_i, D_i^{\text{in}}, D_i^{\text{out}}, t_i^{\max}$	Required computation resource/input data size/output data size/maximum tolerable delay of the computation task T_i
t_{ij}^{soj}	The time available by the vehicle before leaving the communication range of RSU--j
v	The equivalent speed of vehicles
r_{ij}^{UL}, r_{ij}^{DL}	The available uplink/downlink transmission rate of vehicle-i
N_j	The number of vehicles offloads task to MEC server via RSU-j
τ_{ij}	The probability of vehicle-i connects to the RSU-j in a random time slot
σ	The duration of a time slot
δ	The propagation delay
T_{ij}^{success}	The success transmission period between vehicle-i and RSU-j
ω_j	The bandwidth of RSU-j
P_i	The transmission power of vehicle-i
h_{ij}	The channel gain between vehicle-i and RSU-j
$t_{\text{UL}}^{\text{mec}}, t_{\text{DL}}^{\text{mec}}$	The uplink/downlink transmitting time
$t_{\text{trans}}^{\text{mec}}$	The two-way transmission time between vehicle and RSU
x_i, $\mathscr{X}$	The binary offloading strategy of vehicle-i/vehicles
$t_{\text{exe}}^{\text{loc}}$, $t_{\text{exe}}^{\text{mec}}$	The task execution time locally/in the MEC server
t_i^{loc}, t_i^{mec}	Total time for processing task T_i locally/in the MEC server
f_{loc}, f_j^{mec}, F	Computing resource of the vehicle/allocated to MEC-j/the MEC server
$U_i^l, U_i^{\text{mec}}, U_i$	The cost of vehicle-i locally/in MEC processing/vehicle-i under different task offloading decisions
P	The penalty for offloading failure
β	The weighted parameters of delay and computation resource cost
t_i^{actual}	The task's actual tolerant delay
ρ	The unit cost of the computing resource of the MEC server

server in the constraint C3. C4 shows the binary offloading decision constraint for vehicle's task.

Since the cost function in the above problem involves the end-to-end delay, which is related to the indicators of the stochastic arrival tasks $C_i, D_i^{\text{in}}, D_i^{\text{out}}$, the computing resource is allocated to RSU-j f_j^{mec} and the relative position of vehicle to RSU based on vehicle's driving characteristics p_i, v_i. Therefore, the computation complexity is an additive change on all of tasks and the vehicle characteristics. In addition, the computation complexity also depends on the number of the generated tasks. In this optimal problem, the offloading decisions $\mathscr{X}$ and the allocated computation resource $\mathscr{F}$ are two main challenges which make the problem into a mixed-integer nonlinear programming problem that is generally nonconvex and NP-hard [36]. We adopt a multiagent deep reinforcement learning approach to feasibly solve the problem of jointly optimizing the computation offloading decision and computation resource allocation.

4. DRL for Computation Offloading and Resource Allocation

4.1. Scheme Design. We assume that the state is determined by the arrival tasks and the vehicle's characteristics which are updated in each step. The state of the next time slot is related to the state of the current time slot. Therefore, the formulated problem can be modeled as a Markov decision process (MDP). MDP is the iterative process in which agents observe the states in state space from the environment, select an action from action space, obtain an immediate reward sequentially, and then transit to another state, which can be represented as a tuple $<S, A, P_{s,a}, R, \gamma>$, where S is state space, A is action space, $P_{s,a}$ is transition probability space, R is reward space, and γ is discount factor. MDP policy is completely dependent on the current state. The state space is designed to accommodate the proposed IoV environment. Each vehicle acts as the agent. At first, we define the state space, action space, and reward space as follows.

4.1.1. State Space. The state at time slot t is corresponding to the required computation capacity to complete the task C_i, the input data size of the task D_i^{in}, the output data size of the task D_i^{out}, the position of vehicle p_i, the speed of vehicle v_i, and the computing resource allocated to RSU-j. Thus, the state $s_i(t) \epsilon S$ can be described as

$$s_i(t) = \left\{C_i(t), D_i^{\text{in}}(t), D_i^{\text{out}}(t), p_i(t), v_i(t), f_j^{\text{mec}}(t)\right\}_{\forall i \epsilon \mathscr{N}, \forall j \epsilon \mathscr{M}}. \tag{13}$$

4.1.2. Action Space. The action is the joint decision-making for the computation offloading and the resource allocation. The vehicle needs to decide to process the task locally or

offload to the MEC server. If the task is offloaded to the MEC server, the computation resource is allocated to the vehicle by the MEC server via the linked RSU. Therefore, the action $a_i(t) \epsilon A$ is composed of the binary offloading decision and the computation resource allocated to vehicle-i, depicted as

$$a_i(t) = \left\{\left\{x_i(t), f_i^{\text{mec}}(t)\right\}_{\forall i \in \mathcal{N}}\right\}. \tag{14}$$

4.1.3. Reward Space. We assume that all vehicles with the same functionality can share the same reward function. Each agent selects its action based on the reward to obtain the maximum global reward. The long-term weighted sum of the cost function of all the tasks is considered as the objective, and we define the below function to maximize the reward function (minimize the cost function) during the whole time period T.

$$R_i(t) = -\lim_{T \longrightarrow \infty} \frac{1}{T} \sum_{t=1}^{T} U_i(t) | s_i(t). \tag{15}$$

The average rewards of all agents in time slot t can be calculated as

$$R(t) = \frac{1}{N} \sum_{i=1}^{N} R_i(t), \forall i \in \mathcal{N}. \tag{16}$$

Minimizing the weighted cost function of the proposed model amounts to maximizing the average cumulative reward. The expectations of future rewards can be used to measure whether the selected action is appropriate or not. The reward is the return of the selected action based on the state in time slot t. Therefore, the cumulative reward which is generally indicated as the weighted expectation is maximized to select the optimal actions, formulated as

$$Q^{\pi}(s(t), \pi(t)) = \mathbb{E}\left[R(s(t), a(t)) + \gamma Q^{\pi}(s(t+1), \pi(t+1))\right], \tag{17}$$

where $\gamma \in [0, 1]$ is the discount factor, π is the policy, and π^* is the optimal policy. $Q^*(s, a) = \max_{\pi^*} Q^{\pi}(s, \pi)$ corresponds to the agents' optimal policy $\pi^* = \{\pi_1^*, \pi_2^*, \ldots, \pi_N^*\}$.

4.2. Optimal Scheme Based on Decentralized DDPG. After formulating the MDP, we propose the optimization strategy based on the decentralized multiagent DDPG (De-DDPG) in this subsection, in which each agent is initialized with four deep neural networks (DNNs): the critic network, the actor network, and two copies of the actor and critic networks as target networks, respectively (see Figure 2) [37]. Each agent's state, action, and reward are obtained and used to train the DNNs during the training procedure. After training, each agent can select the next step strategy by its own actor network according to the local observation from the environment.

As shown in Algorithm 1, the process of De-DDPG algorithm can be divided into three parts: initialization, interaction, and update. At the beginning of the algorithm, four networks of each agent and the replay buffer B are initialed, where the critic network is $Q(s, a|\theta_i^Q)$, the actor network is $\mu(s|\theta_i^{\mu})$, the target critic network is $Q'(s, a|\theta_i^{Q'})$, and the target actor network is $\mu'(s|\theta_i^{\mu'})$, respectively. In addition, the replay buffer can be large because the proposed De-DDPG is an off-policy algorithm, which allows the algorithm to benefit from learning across a set of uncorrelated transitions [38]. In the interaction procedure, for each episode, a sampled noise from the random noise process $\mathcal{N}_t$ is added to an exploration policy μ' into the actor policy. The reason for introducing random noise is to solve the problem of insufficient exploration of the environment by the output actions in deterministic policy algorithms. The Ornstein–Uhlenbeck process is used to generate temporally correlated exploration for exploration efficiency. Then, the actions interact with the environment and obtain the corresponding rewards and the next step states. According to the observation, the transitions $(s_i(t), a_i(t), R_i(t), s_i(t+1))$ store in the replay buffer B. When updating, a random mini-batch of Z transitions is sampled from the replay buffer. Then, update each agent's critic network, actor network, and two target networks in turn. Loop through each episode until the algorithm ends. In the update process, the critic network $Q(s, a|\theta_i^Q)$ is updated by minimizing the loss $L(\theta^Q)$ in Algorithm 1 which is the approximation function of other agent policy by each agent. Here, y_i is the predication of the next action in target actor network in formula (17). The actor network $\mu(s|\theta_i^{\mu})$ is updated by using the sampled policy gradient $\nabla_{\theta^{\mu}} J$ in Algorithm 1 which is the unbiased estimation of the policy gradient expectation calculated by the mini-batch transitions according to Monte Carlo method. After training the mini-batch transitions and updating the weights of critic network and actor network (θ_i^Q and θ_i^{μ}), the weights of two target networks for each agent ($\theta_i^{Q'}$ and $\theta_i^{\mu'}$) can be soft updated as a running average algorithm, which is shown in Algorithm 1, respectively.

5. Numerical Results

This section describes the comprehensive numerical simulation analysis from simulation setup, simulation comparison, and simulation results.

5.1. Simulation Setup. Firstly, we evaluate and verify our proposed De-DDPG by TensorFlow 1.13.1. A personal computer with a RTX2070 GPU and 8GB video memory is used to train and test De-DDPG.

There are 20 vehicles driving on the road, 4 RSUs are located at the stationary region on the roadside, and an MEC server directly connects with RSUs. That is to say, 2 groups of RSUs are set to serve all vehicles. One RSU group includes the main RSU and a secondary RSU, and the purpose is to prevent the main RSU from being abnormal due to accidents (such as power failure and communication blockage). Therefore, $N = 20$, $M = 4$. The required computation resource is set as $C_i = 0.54$ (G cycles/Mbits). The size of arrival tasks follows uniform distribution $D_i^{\text{in}} \sim U(1.6, 4.6)$ MetaBits. The arrival probability of tasks is 0.45. The maximum tolerable delay of the computation task is set as $t_i^{\max} = 4$ time

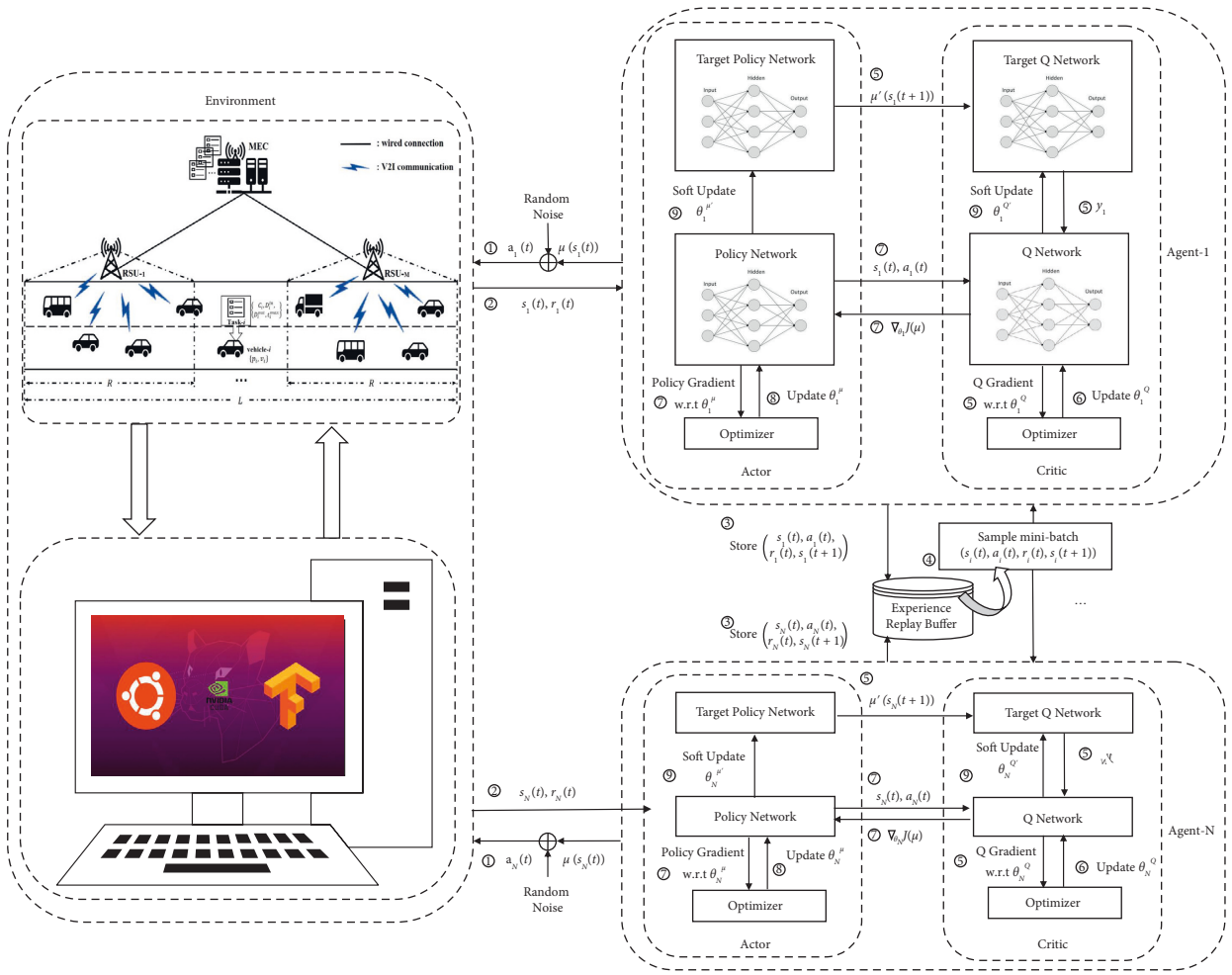

Figure 2: Structure of the proposed multiagent DDPG (De-DDPG) scheme.

slots. One time slot $t \in T$ is set to $t = 1$ ms. The uplink transmission rate of vehicle-i r_{ij}^{UL} is between 0.85 and 0.95 Mbits per time slot. The bandwidth of RSU $\omega_j = 10$ MHz. The transmission power for vehicle-i is $P_i = 1W$. The max computation resource allocated to RSU-j is 3.8 gigacycles. The computation resource of each vehicle is 0.5 gigacycles.

In terms of each vehicle-i ($i \in N$), the neural networks (two actor networks and two critic networks) for De-DDPG are composed of four layers, i.e., an input layer and two fully connected layers. Table 2 illustrates the parameters and values.

5.2. Simulation Comparison. For verifying the performance of the proposed DE-DDPG, three benchmark algorithms are set: centralized deep deterministic policy gradient (Ce-DDPG), all tasks offloaded to RSU(A-RSU), and all tasks executed by local processor (A-LP), which are described as follows:

(1) Ce-DDPG: on the MEC side, a centralized controller captures global information such as tasks generated from vehicles, computation, and communication resources of RSUs. That is to say, there is only one agent which interacts with the MEC environment. In order to improve the convergence of Ce-DDPG, the allocated computation resources by all RSUs are the same. The structure of the neural network is the same as each sub-network of De-DDPG.

(2) A-RSU: all tasks from vehicles are offloaded to RSU in the corresponding coverage region. Furthermore, the computation resources allocated for each vehicle are the same.

(3) A-LP: all tasks are executed by the local processor of the vehicle.

5.3. Simulation Results. In this section, we analyze the simulation results in detail from the two aspects: the convergence of proposed De-DDPG and the advantages of De-DDPG compared to three baseline algorithms.

In terms of De-DDPG's convergence, the average cumulative reward of De-DDPG in one period (the whole time slots) for each episode is formulated.

Figure 3 shows the convergence of proposed De-DDPG with different critics' learning rate α. The choice of learning rates α can obviously affect the convergence effect and speed of De-DDPG. From Figure 3, it can be observed that De-DDPG cannot be convergent when $\alpha = 0.01$. However, from

Randomly initialize critic network $Q(s, a|\theta_i^Q)$ and actor $\mu(s|\theta_i^Q)$ with weights θ_i^Q and θ_i^Q
Initialize target network Q' and μ' with weights $\theta_i^{Q'} \leftarrow \theta_i^Q$, $\theta_i^{\mu} \leftarrow \theta_i^{\mu}$
Initialize replay buffer B
for for episode $k = 1, 2, \ldots, K$ **do**
  Initialize a random process N_s foe action exploration
  Receive initial observation state $\mathbf{S} = \{s_i(1), s_i(2), \ldots s_i(N)\}$
  for $i = 1, 2, \ldots, N$**do**
    for $t = 1, 2, \ldots, T$**do**
      Select action $a_i(t)\mu(s_i^t|\theta_i^{\mu}) + N_s$ according to the current policy and exploration noise N_s
      Execute action $a_i(t)$ and observe reward R_i and observe the next state $s_i(t+1)$
      Store all transitions $(s_i(t), \mathbf{a}_i(t)R_i(t), s_i(t+1))$ in B
      Sample a random mini-batch of Z transitions $\{s_i(t), \mathbf{a}_i(t)R_i(t), s_i(t+1)\}$ from B
      Set
        $y_i = R_i(t) + \gamma Q_{\pi}(s_i(t), \mathbf{a}_i(t)|\theta^Q)|_{a_i^{k+1'}=\mu'(s_i^{k=t})}$
      Update critic network $Q(s, \mathbf{a}|\theta^Q)$ by minimizing the loss
        $L(\theta^Q) = 1/Z\sum_I (y_i - Q_{\mu}(s_i, \mathbf{a}_i(t)|\theta_i^Q))^2$
      Update the actor policy by using the sampled policy gradient
        $\nabla_{\theta_i^{\mu}} J \approx 1/Z\sum_i \nabla_a Q(s_i(t), \mathbf{a}_i(t)|\theta_i^Q)|_{a_i=\mu(s_i)} \nabla_{\theta_i^{\mu}} \mu(s_i(t)|\theta_i^{\mu})$
      Update the target networks for each agent i:
        $\theta_i^{Q'} \leftarrow \eta\theta_i^Q + (1-\eta)\theta_i^{Q'}$,
        $\theta_i^{\mu'} \leftarrow \eta\theta_i^{\mu} + (1-\eta)\theta_i^{\mu'}$.
    end for
  end for
end for

Algorithm 1: Decentralized multiagent DDPG optimization method.

Table 2: Main hyperparameters of the De-DDPG.

Parameters	Value
Size of the first hidden layer for actor and critic	300
Size of the second hidden layer for actor and critic	300
Learning rate of actor and critic α'/α	0.0001/0.001
Size of experienced memory B	20000
Parameters for OU noise θ, μ, σ	0.15, 0.15, 0.10
Discount factor γ	0.95
Penalty for failed task execution P	8
Total number of all episodes K	1000
Total time periods of one episode T	110

the blue curve when $\alpha = 0.0001$, although De-DDPG can be eventually convergent, the convergence speed is too slow to influence the performance of De-DDPG. Therefore, we set $\alpha = 0.001$ because De-DDPG is more stable. In terms of actor's learning rate, we set $\alpha' = 0.0001$.

f_j^{mec} is the total computation resources preallocated to RSU-j from the MEC server. To ensure the fairness of computation resources allocation and the robustness of the proposed De-DDPG, we periodically update the allocated computation resources of each RSU by adding the fluctuation volatility rate. As shown in Figure 4, when the volatility rate is set to 1, 3, 5, the convergence and performance of De-DDPG are different. When $\lambda_j = 1$, the performance of De-DDPG is better than the other two curves. However, the stability of De-DDPG is slightly worse. As the volatility rate increases, the performance of De-DDPG decreases, but we can see that when the volatility rate is set to 3, the stability and convergence of the De-DDPG are optimal.

Figure 5 reveals the convergence of De-DDPG with different control parameters ρ. From formula (10), the greater the ρ is, the greater the cost is (the less the reward is). Although the average cumulative rewards of De-DDPG are worst when $\rho = 0.08$, the stability is much better than that of the curves when $\rho = 0.04$ and $\rho = 0.06$. From the curves shown in Figure 5, as the control parameter ρ increases, the performance of De-DDPG declines less and less. However, when $\rho = 0.08$, the convergence effect and training speed of De-DDPG are obviously improved. Therefore, we set the control parameter for the cost of computation resource $\rho = 0.08$ in this paper.

Furthermore, Figures 6–8 will verify the performance and advantages of the proposed De-DDPG compared to other baselines Ce-DDPG, A-RSU, and A-LP. The average cumulative reward of De-DDPG for all episodes (1000 episodes) is introduced to show the performance of four algorithms.

Figure 6 illustrates the comparison of four algorithms with the different numbers of arrival vehicles. When the number of vehicles N within the coverage region of each RSU is 6 or 8, because the computation resources of each RSU are sufficient to meet the computational demands of the vehicles for all generated tasks, the performance of De-DDPG, Ce-DDPG, and A-RSU is not significantly different. In terms of A-LP, regardless of the number of vehicles, its local processing ability remains unchanged, and the computation resources of RSUs have no effect on its performance. On the contrary, because the computation ability of the local processor for vehicles is insufficient for arrival tasks, a large number of penalties P in each episode will be incurred due to the incomplete tasks, thus affecting the average

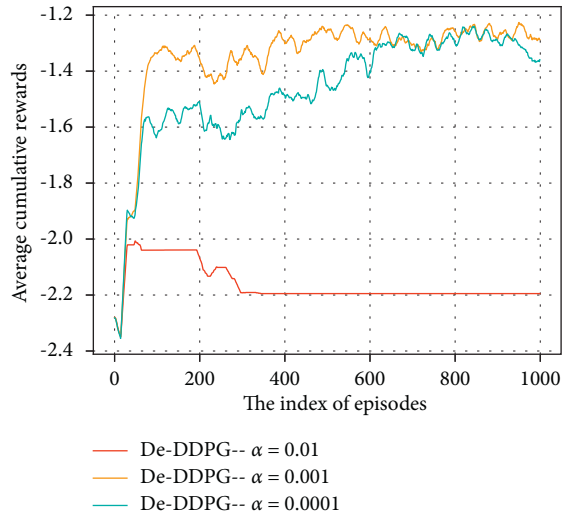

Figure 3: Convergence of proposed De-DDPG with different learning rates α.

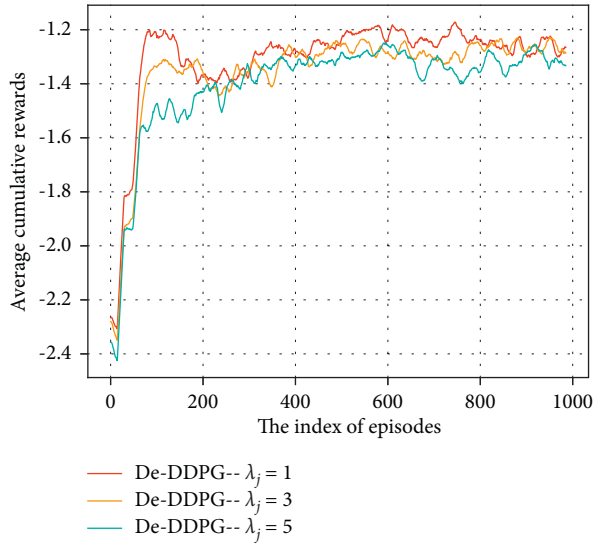

Figure 4: Convergence of proposed De-DDPG with different volatility rates λ_j.

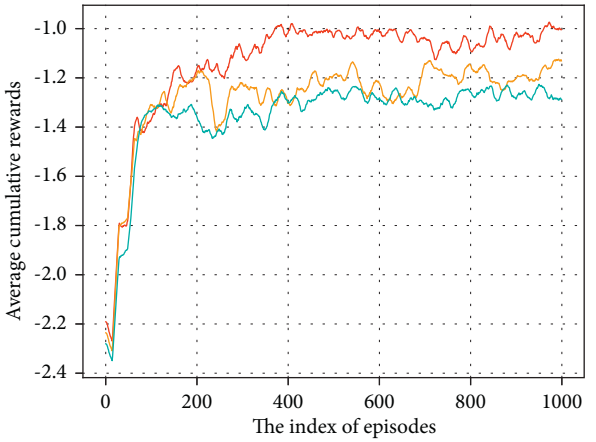

Figure 5: Convergence of proposed De-DDPG with different control parameters ρ.

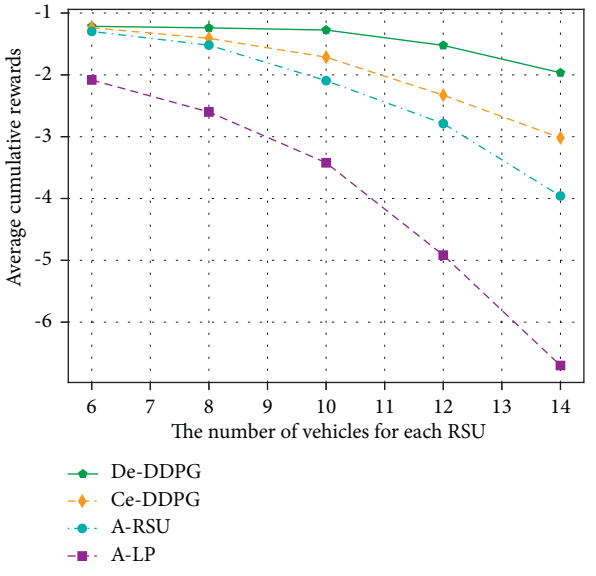

Figure 6: Comparison on all algorithms with the number of vehicles N.

cumulative reward of A-LP. However, when the number of vehicles N within the coverage region of each RSU is greater than 10, the computation resources of RSUs are insufficient for completing all tasks generated by all vehicles, and the performance of four algorithms degrades significantly. From the curves, the performance of the proposed De-DDPG is better than that of the other three algorithms.

The uplink transmission rate of the wireless communication between vehicle-i and its belonged RSU-j r_{ij}^{UL} can influence the performance of four algorithms. Figure 7 shows the performance of four algorithms with different uplink transmission rates r_{ij}^{UL}. We can see that A-LP is not influenced by the transmission rate r_{ij}^{UL} because A-LP executes all tasks by its own local processor. For De-DDPG, Ce-DDPG, and A-RSU, different uplink transmission rates r_{ij}^{UL} mean different transmission delays $t_{\mathrm{UL}}^{\mathrm{mec}}$ which affairs the average cumulative rewards of all three algorithms. As shown in Figure 7, the performance of De-DDPG is obviously better than that of Ce-DDPG and A-RSU due to the number of agents participating in training, offloading decisions, and the ratio of resource allocation.

In Figure 8, we describe the comparison of four algorithms with different trade-off coefficients β for latency cost. From formula (10), β is the weighted parameter of delay cost and $1 - \beta$ computation resource cost. In terms of A-LP, since the computational resource cost is fixed, the performance of A-LP decreases as the trade-off coefficient β increases. However, the other three algorithms consider the trade-off between delay cost and computation resource cost, so the performance of De-DDPG, Ce-DDPG, and A-RSU varies

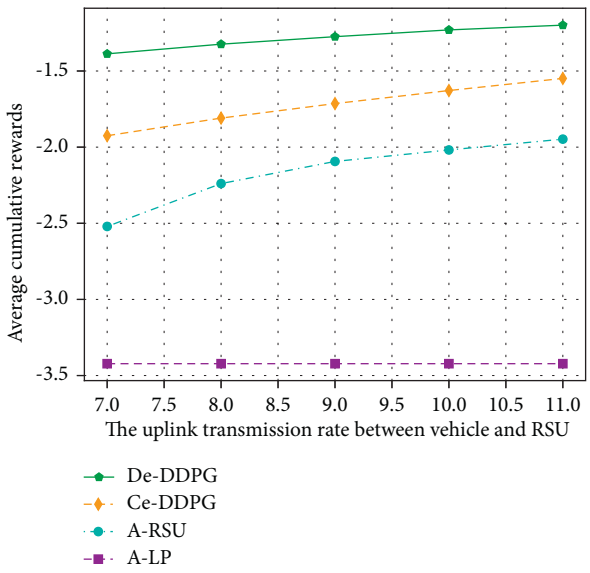

Figure 7: Comparison on all algorithms with different uplink transmission rates r_{ij}^{UL}.

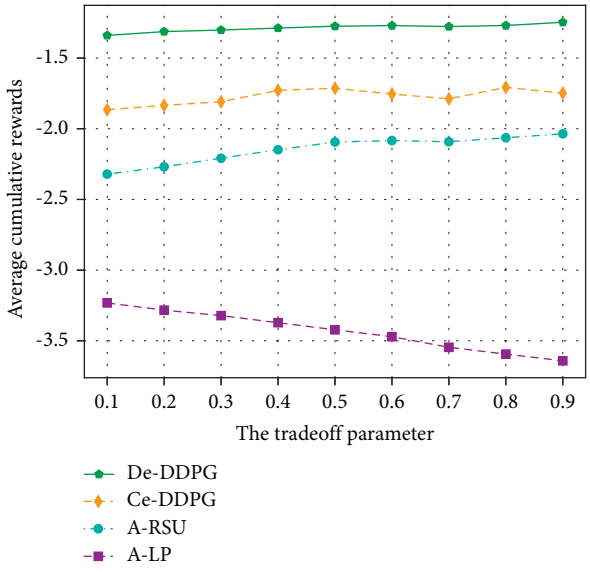

Figure 8: Comparison on all algorithms with different trade-off coefficients β.

with the changing of β. As shown in Figure 8, in terms of the average cumulative rewards, the performance of De-DDPG outperforms that of Ce-DDPG, A-RSU, and A-LP no matter the size of β.

6. Conclusions

We propose a computation offloading and resource allocation scheme based on DRL for the MEC-assisted multiagent with stochastic arrival task model in the IoV environment. To minimize the total weighted cost of the proposed model, we adopt a decentralized multiagent DDPG-based approach (De-DDPG) to solve the nonconvex joint optimization problem. The simulation results demonstrate that our proposed approach has a stable learning capacity and effectively learns the optimal offloading policy and resource allocation to obtain the maximum reward (minimum cost). Compared with the three baseline algorithms, our proposed algorithm has better performance for various parameter configurations. In this paper, the binary offloading decision is used and the task priority is not considered. We will improve these two points in our future work, such as considering partial offloading and task prioritization in this joint optimization problem.

Data Availability

The data used to support the findings of this study are available from the corresponding author upon request.

Conflicts of Interest

The authors declare that there are no conflicts of interest regarding the publication of this paper.

Acknowledgments

This work was supported by China National Natural Science Foundation (61572229 and 6171101066).

References

[1] J. Contreras-Castillo, S. Zeadally, and J. A. Guerrero-Ibanez, "Internet of vehicles: architecture, protocols, and security," *IEEE Internet of Things Journal*, vol. 5, no. 5, pp. 3701–3709, 2018.

[2] J. Lin, W. Yu, N. Zhang, X. Yang, H. Zhang, and W. Zhao, "A survey on internet of Things: architecture, enabling technologies, security and privacy, and applications," *IEEE Internet of Things Journal*, vol. 4, no. 5, pp. 1125–1142, 2017.

[3] O. Kaiwartya, A. H. Abdullah, Y. Cao et al., "Internet of vehicles: motivation, layered architecture, network model, challenges, and future aspects," *IEEE Access*, vol. 4, pp. 5356–5373, 2016.

[4] J. Lin, W. Yu, X. Yang, P. Zhao, H. Zhang, and W. Zhao, "An edge computing based public vehicle system for smart transportation," *IEEE Transactions on Vehicular Technology*, vol. 69, no. 11, pp. 12635–12651, 2020.

[5] K. Zhu, Z. Chen, Y. Peng, and L. Zhang, "Mobile edge assisted literal multi-dimensional anomaly detection of in-vehicle network using LSTM," *IEEE Transactions on Vehicular Technology*, vol. 68, no. 5, pp. 4275–4284, 2019.

[6] J. Lin, L. Huang, H. Zhang, X. Yang, and P. Zhao, "A novel lyapunov based dynamic resource allocation for UAVs-assisted edge computing," *Computer Networks*, vol. 205, no. C, pp. 108710–111286, 2022.

[7] S. Wan, R. Gu, T. Umer, K. Salah, and X. Xu, "Toward offloading internet of vehicles applications in 5G networks," *IEEE Transactions on Intelligent Transportation Systems*, vol. 22, no. 7, pp. 4151–4159, 2021.

[8] K. Zhang, Y. Mao, S. Leng, Y. He, and Y. Zhang, "Mobile-edge computing for vehicular networks: a promising network paradigm with predictive off-loading," *IEEE Vehicular Technology Magazine*, vol. 12, no. 2, pp. 36–44, 2017.

[9] Y. Wang, P. Lang, D. Tian et al., "A game-based computation offloading method in vehicular multiaccess edge computing

networks," *IEEE Internet of Things Journal*, vol. 7, no. 6, pp. 4987–4996, 2020.

[10] J. Zhou, D. Tian, Z. Sheng, X. Duan, and X. Shen, "Distributed task offloading optimization with queueing dynamics in multiagent mobile-edge computing networks," *IEEE Internet of Things Journal*, vol. 8, no. 15, pp. 12311–12328, 2021.

[11] H. Wang, Z. Lin, K. Guo, and T. Lv, "Computation offloading based on game theory in MEC-assisted V2X networks," in *Proceedings of the 2021 IEEE International Conference on Communications Workshops (ICC Workshops)*, Montreal, QC, Canada, June 2021.

[12] J. Zhang, H. Guo, J. Liu, and Y. Zhang, "Task offloading in vehicular edge computing networks: a load-balancing solution," *IEEE Transactions on Vehicular Technology*, vol. 69, no. 2, pp. 2092–2104, 2020.

[13] G. Wang, F. Xu, and C. Zhao, "QoS-enabled resource allocation algorithm in internet of vehicles with mobile edge computing," *IET Communications*, vol. 14, no. 14, pp. 2326–2333, 2020.

[14] H. Ye, G. Y. Li, and B. H. F. Juang, "Deep reinforcement learning based resource allocation for V2V communications," *IEEE Transactions on Vehicular Technology*, vol. 68, no. 4, pp. 3163–3173, 2019.

[15] G. Hong, W. Su, Q. Wen, and P. L. Wu, "RAVEC: an optimal resource allocation mechanism in vehicular MEC systems," *Journal of Information Science and Engineering*, vol. 36, no. 4, pp. 865–878, 2020.

[16] H. Zhang, Z. Liu, S. Hasan, and Y. Xu, "Joint optimization strategy of heterogeneous resources in multi-MEC-server vehicular network," *Wireless Networks*, vol. 28, no. 2, pp. 765–778, 2022.

[17] H. Zhang, Z. Wang, and K. Liu, "V2X offloading and resource allocation in SDN-assisted MEC-based vehicular networks," *China Communications*, vol. 17, no. 5, pp. 266–283, 2020.

[18] W. Fan, J. Liu, M. Hua, F. Wu, and Y. Liu, "Joint task offloading and resource allocation for multi-access edge computing assisted by parked and moving vehicles," *IEEE Transactions on Vehicular Technology*, vol. 71, no. 5, pp. 5314–5330, 2022.

[19] Y. He, N. Zhao, and H. Yin, "Integrated networking, caching, and computing for connected vehicles: a deep reinforcement learning approach," *IEEE Transactions on Vehicular Technology*, vol. 67, no. 1, pp. 44–55, 2018.

[20] Z. Ning, Y. Li, P. Dong et al., "When deep reinforcement learning meets 5G-enabled vehicular networks: a distributed offloading framework for traffic big data," *IEEE Transactions on Industrial Informatics*, vol. 16, no. 2, pp. 1352–1361, 2020.

[21] S. Xiao, S. Wang, J. Zhuang, T. Wang, and J. Liu, "Research on a task offloading strategy for the internet of vehicles based on reinforcement learning," *Sensors*, vol. 21, no. 18, pp. 6058–6067, 2021.

[22] T. Liu, B. Tian, Y. Ai, L. Li, D. Cao, and F. Y. Wang, "Parallel reinforcement learning: a framework and case study," *IEEE-CAA JOURNAL OF AUTOMATICA SINICA*, vol. 5, no. 4, pp. 827–835, 2018.

[23] X. Xu, B. Shen, S. Ding et al., "Service offloading with deep Q-network for digital twinning empowered internet of vehicles in edge computing," *IEEE Transactions on Industrial Informatics*, vol. 18, no. 2, pp. 1414–1423, 2022.

[24] X. Zhu, Y. Luo, A. Liu, M. Z. A. Bhuiyan, and S. Zhang, "Multiagent deep reinforcement learning for vehicular computation offloading in IoT," *IEEE Internet of Things Journal*, vol. 8, no. 12, pp. 9763–9773, 2021.

[25] D. Zhang, L. Cao, H. Zhu, T. Zhang, J. Du, and K. Jiang, "Task offloading method of edge computing in internet of vehicles based on deep reinforcement learning," *Cluster Computing*, vol. 25, no. 2, pp. 1175–1187, 2022.

[26] C. Pan, Z. Wang, Z. Zhou, and X. Ren, "Deep reinforcement learning-based URLLC-aware task offloading in collaborative vehicular networks," *China Communications*, vol. 18, no. 7, pp. 134–146, 2021.

[27] J. Shi, J. Du, J. Wang, J. Wang, and J. Yuan, "Priority-aware task offloading in vehicular fog computing based on deep reinforcement learning," *IEEE Transactions on Vehicular Technology*, vol. 69, no. 12, pp. 16067–16081, 2020.

[28] Z. Wu and D. Yan, "Deep reinforcement learning-based computation offloading for 5G vehicle-aware multi-access edge computing network," *China Communications*, vol. 18, no. 11, pp. 26–41, 2021.

[29] H. Yang, Z. Wei, Z. Feng, X. Chen, Y. Li, and P. Zhang, "Intelligent computation offloading for MEC-based cooperative vehicle infrastructure system: a deep reinforcement learning approach," *IEEE Transactions on Vehicular Technology*, vol. 71, no. 7, pp. 7665–7679, 2022.

[30] J. Zhao, Q. Li, Y. Gong, and K. Zhang, "Computation offloading and resource allocation for cloud assisted mobile edge computing in vehicular networks," *IEEE Transactions on Vehicular Technology*, vol. 68, no. 8, pp. 7944–7956, 2019.

[31] N. Wisitpongphan, F. Bai, P. Mudalige, V. Sadekar, and O. Tonguz, "Routing in sparse vehicular ad hoc wireless networks," *IEEE Journal on Selected Areas in Communications*, vol. 25, no. 8, pp. 1538–1556, 2007.

[32] S. Durrani, X. Zhou, and A. Chandra, "Effect of vehicle mobility on connectivity of vehicular ad hoc networks," in *Proceedings of the 2010 IEEE 72nd Vehicular Technology Conference - Fall*, 2010.

[33] C. Han, M. Dianati, R. Tafazolli, and R. Kernchen, "Throughput analysis of the IEEE 802.11p enhanced distributed channel access function in vehicular environment," in *Proceedings of the 2010 IEEE 72nd Vehicular Technology Conference (VTC 2010-Fall)*, Ottawa, Canada, December 2010.

[34] Y. Dai, D. Xu, S. Maharjan, and Y. Zhang, "Joint load balancing and offloading in vehicular edge computing and networks," *IEEE Internet of Things Journal*, vol. 6, no. 3, pp. 4377–4387, 2019.

[35] Z. Su, Y. Hui, and T. H. Luan, "Distributed task allocation to enable collaborative autonomous driving with network softwarization," *IEEE Journal on Selected Areas in Communications*, vol. 36, no. 10, pp. 2175–2189, 2018.

[36] K. Zhang, Y. Mao, S. Leng et al., "Energy-efficient offloading for mobile edge computing in 5G heterogeneous networks," *IEEE Access*, vol. 4, pp. 5896–5907, 2016.

[37] R. Lowe, Y. Wu, A. Tamar, J. Harb, P. Abbeel, and I. Mordatch, "Multi-Agent Actor-Critic for Mixed Cooperative-Competitive Environments," 2018, https://arxiv.org/abs/1706.02275.

[38] T. P. Lillicrap, J. J. Hunt, A. Pritzel, D. Silver, and D. Wierstra, "Continuous Control with Deep Reinforcement Learning," 2019, https://arxiv.org/abs/1509.02971.