Joint Computation Offloading and Resource Allocation for Edge-Cloud Collaboration in Internet of Vehicles via Deep Reinforcement Learning

Jiwei Huang, *Member, IEEE*, Jiangyuan Wan, Bofeng Lv, Qiang Ye, *Senior Member, IEEE*, and Ying Chen, *Member, IEEE*

***Abstract*—Mobile edge computing (MEC) and cloud computing (CC) have been considered as the key technologies to improve the task processing efficiency for Internet of Vehicles (IoV). In this article, we consider a random traffic flow and dynamic network environment scenario where MEC and CC are collaborated for processing delay-sensitive and computation-intensive tasks in IoV. We study the joint optimization of computation offloading and resource allocation (CORA) with the objective of minimizing the system cost of processing tasks subject to the processing delay and transmission rate constraints. To attack the challenges brought by the dynamic environment, we use the Markov decision process model for formulating the dynamic optimization problem, and apply a deep reinforcement learning (DRL) technique to deal with high-dimensional and continuous states and action spaces. Then, we design a CORA algorithm, which is able to effectively learn the optimal scheme by adapting to the network dynamics. Extensive simulation experiments are conducted, in which we compare the CORA algorithm with both non-DRL algorithms and DRL algorithms. The experimental results show that the CORA algorithm outperforms others with excellent training convergence and performance in processing delay and processing cost.**

***Index Terms*—Cloud computing, computation offloading, deep reinforcement learning (DRL), Internet of Vehicles (IoV), mobile edge computing, resource allocation.**

I. Introduction

With the emergence and rapid development of Internet of Vehicles (IoV), various computation-intensive and delay-sensitive applications are employed to improve the convenience of human–vehicle interaction [1], such as autonomous driving, augmented reality [2], natural language processing, etc. Among the aforementioned applications, autonomous driving is one of the key technologies to improve travel efficiency in the intelligent IoV [3]. In order to perceive the surrounding environment in real time, vehicles need to perceive the environment through on-board cameras. Then, the wireless communication on-board units (OBU) need to extract useful information from the image data captured by cameras to assist autonomous driving. However, with the increase of the amount of data, the computation capacity of the OBU may be not enough to meet the autonomous driving task processing delay requirement, and high delay is a fatal problem in autonomous driving. Therefore, cloud computing (CC) with high-performance computing capability is an effective method to reduce the computing burden of vehicles [4]. Although CC can deal with the challenge of insufficient computing capability of vehicles, massive data will lead to high transmission delay and unstable connection. In order to make up for the deficiency of CC, mobile edge computing (MEC) is introduced to improve the performance of IoV [5].

As an emerging paradigm, MEC can provide auxiliary computing services with low delay for vehicles. Although the MEC can bring advantages like low delay due to short distance between vehicles and edge servers, there are challenges in MEC for IoV. The edge server (ES) needs to process a large number of heterogeneous tasks from vehicles in parallel, which leads to dynamic remaining available computing resources of the ES. Furthermore, the randomness of vehicle density leads to the dynamic task computation load [6]. When the number of tasks increases, the ES with limited computing resources may not be able to handle high volume of tasks. To solve these problems, the roadside unit (RSU) needs to offload tasks partially to cloud server (CS). In this article, we study the scenario where CC and MEC are collaborated in IoV to process tasks with strictly high requirements in performance.

Due to the dynamic network environment of IoV, the available bandwidth of the micro base station (MBS) and RSU to MBS (R2B) wireless channel states are time varying and unpredictable [7]. All RSUs along the road communicate with the MBS and share the available bandwidth of the MBS. In addition, the channel state will affect the transmission delay of R2B connections, and R2B connections sharing the bandwidth of the MBS will cause interference to other connections. Therefore, the bandwidth of MBS and channel state of R2B connections

Manuscript received 12 February 2022; revised 24 December 2022; accepted 19 February 2023. Date of publication 13 March 2023; date of current version 8 June 2023. This work was supported in part by Beijing Nova Program under Grant Z201100006820082; in part by the National Natural Science Foundation of China under Grant 61972414; in part by the Project of Cultivation for young top-motch Talents of Beijing Municipal Institutions under Grant BPHR202203225; in part by R&D Program of Beijing Municipal Education Commission under Grant KM202011232015; and in part by the Young Elite Scientists Sponsorship Program by BAST under Grant BYESS2023031. *(Corresponding author: Ying Chen.)*

Jiwei Huang, Jiangyuan Wan, and Bofeng Lv are with the Beijing Key Laboratory of Petroleum Data Mining, China University of Petroleum, Beijing 102249, China (e-mail: huangjw@cup.edu.cn; 15611562852@163.com; lvbofeng@foxmail.com).

Qiang Ye is with the Department of Computer Science, Memorial University of Newfoundland, St. John's NL A1C 5S7, Canada (e-mail: qiangy@mun.ca).

Ying Chen is with the Computer School, Beijing Information Science and Technology University, Beijing 100101, China (e-mail: chenying@bistu.edu.cn).

Digital Object Identifier 10.1109/JSYST.2023.3249217

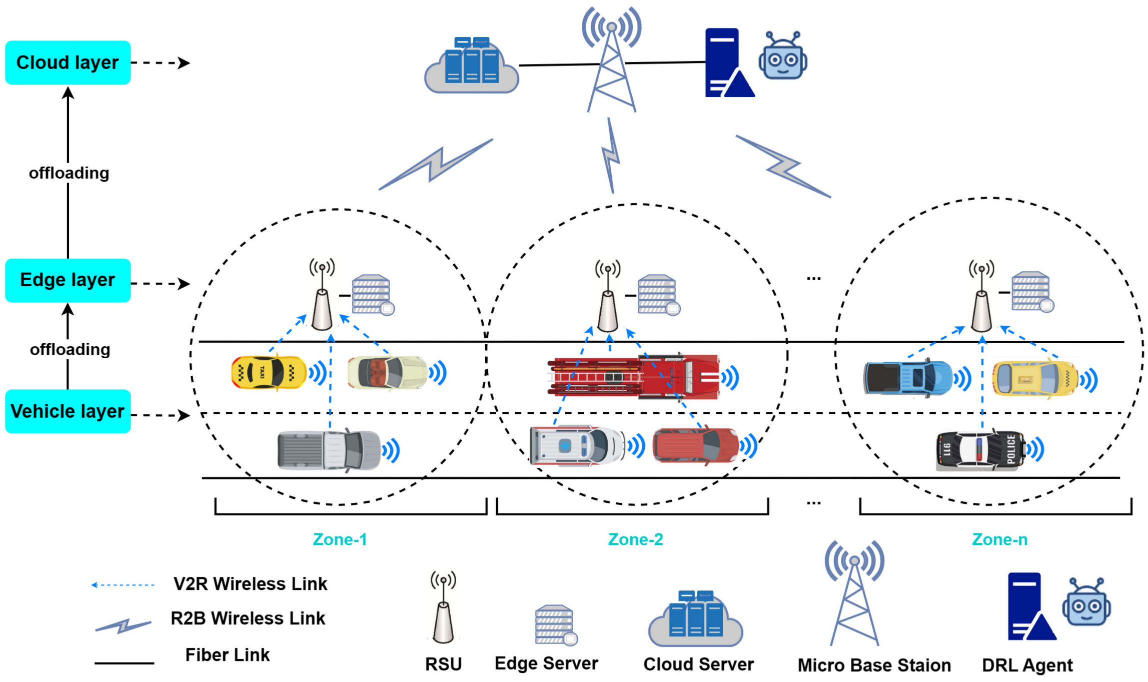

Fig. 1. System architecture.

cannot be ignored. Finally, due to the time-varying network environment, it is difficult for traditional mathematical optimization methods to determine the optimal scheme of computation offloading and resource allocation (CORA) in real time.

To attack the aforementioned challenges, we consider a three-layered architecture of IoV consisting of both MEC and CC. In order to minimize the task processing cost in the dynamic scenario of IoV, we propose a CORA algorithm based on deep reinforcement learning (DRL) supporting the partial offloading strategy, which is able to deal with continuous and large-scale action spaces for effectively obtaining the optimal CORA schemes. The main contributions of this article are as follows.

1) We investigate a three-layered architecture of IoV supporting MEC and CC. In the dynamic scenario, the vehicle layer includes autonomous driving vehicles on a straight road, the edge layer contains RSUs for receiving tasks from vehicles and ESs for processing tasks, and the cloud layer contains one CS for relieving the computing burden of the edge layer, one central controller for collecting system status and making scheduling decisions and one MBS for receiving tasks from RSUs. Furthermore, we consider both binary offloading and partial offloading to make the task offloading strategy more flexible. Then, we propose a joint optimization of computation offloading and bandwidth resource allocation scheme. The objective of our scheme is to minimize the task processing cost with the delay and transmission rate constraints.
2) We consider the dynamic IoV scenario, in which the vehicle density, available bandwidth of MBS, channel states of R2B connections, and computing resources of edge servers are time varying. In order to deal with the large-scale dynamic IoV scenario, we design our CORA algorithm based on DRL to obtain the optimal computing offloading and resource allocation scheme effectively. We take advantage of the collaboration of multiple actor networks and critic networks for preventing from overestimation and stabilizing the training process.
3) We compare our CORA algorithm with traditional non-DRL algorithms and existing DRL algorithms [deep Q network (DQN) and deep deterministic policy gradient (DDPG)] through extensive simulation experiments. The results show that our CORA algorithm outperforms others in learning convergence rate, processing delay, and system cost.

The remainder of this article is organized as follows. In Section II, we propose the system model. The optimization scheme of our problem is illustrated in Section III. Section IV proposes Markov decision process (MDP)-based CORA algorithm. Section V shows the experimental results of our experiment. Section VI discusses the related work. Finally, Section VII concludes this article.

II. System Model

This section proposes the detailed system model, which includes network model, mobility model, task model, computation model, and communication model.

A. Network Model

We consider a three-layer edge-cloud collaboration system in IoV. The vehicle layer includes autonomous driving vehicles on a straight road. The edge layer includes M RSUs and M ESs. The cloud layer includes one CS equipped with one MBS. The system architecture is shown in Fig. 1.

In the vehicle layer, vehicles run on a unidirectional road. Each vehicle is equipped with an OBU for communicating with RSU and processing tasks. The road is divided into M zones with equal length, and each zone is covered by one RSU. The set of RSUs is denoted by $\mathcal{M} = \{R_1, R_2, \ldots, R_M\}$. We assume the coverage areas of RSUs do not overlap, and thus, the vehicles driving in the ith zone can only communicate with RSU R_i.

In the edge layer, there are M RSUs along the road, and each RSU is connected with an ES through wired connection. The set of ES is expressed as $\mathcal{S} = \{S_1, S_2, \ldots, S_M\}$. In each time slot t, each RSU receives arrival tasks within its coverage. According to the dynamic network environment, RSUs can choose to offload partial or all of the tasks to the CS to reduce the computation burden on ES through wireless connections.

In the cloud layer, the CS is connected with one MBS through wired connection. The CS has enough computing resources for processing tasks. The MBS can receive and forward tasks to the CS, which are offloaded by RSUs.

We assume that there is a central controller with full knowledge of the network environment and traffic environment of the IoV [8]. The DRL agent that makes task offload and resource allocation decisions is deployed in the central controller. The central controller collects the status of vehicle tasks, edge server computational resources, channels, bandwidth resources, etc. Then, the DRL agent in the central controller makes offloading decisions for all vehicles intensively. The central controller is deployed with the MBS connected by wired link. For the clarity of the following analysis, we describe in Table I notation mainly used in this article.

TABLE I
NOTATION AND DEFINITIONS

Notations	Definitions
$\mathcal{M}$	Set of RSU
$\mathcal{S}$	Set of ES
M	Number of RSU and ES
R_i	RSU which covers the ith zone
$V_i(t)$	Set of vehicles within coverage of RSU R_i
t	Time slot
$N_i(t)$	Number of vehicles within coverage area of RSU R_i
$\overline{v}$	Average driving speed of vehicles on the road
$\overline{N}$	Average number of vehicles in a zone
$N_{\max}$	Maximum number of vehicles in a zone
p	Probability of task generation
λ	Average task arrival rate
l	Size of one task
c	Number of CPU cycles required to process one bit data
$a_i(t)$	Offloading decision of vehicles in RSU $V_i(t)$
$L_i(t)$	Total size of tasks generated by $V_i(t)$
f_v	CPU frequency of vehicle
$T_i^{\text{local}}(t)$	Average local processing time of load $L_i(t)$
$x_i(t)$	Load slicing proportion of $L_i(t)$
$T_i^{\text{comp}}(t)$	Average processing time of $L_i(t)$ by ES
B_i	Bandwidth of RSU R_i
p_v	Transmission power of vehicles
g_i^{v2r}	Channel power gain of V2R connection in the ith zone
I_i^{v2r}	Interference noise from other V2R connections in $V_i(t)$
σ^2	Average gaussian white noise of wireless connections
$R_i^{\text{v2r}}(t)$	Average transmission rate of V2R connections
R_{min}^{v2r}	Minimum transmission rate of V2R connections
$T_i^{\text{v2r}}(t)$	Average transmission delay of V2R connections
$\eta_i(t)$	Allocation proportion of MBS bandwidth
B_0	Bandwidth of MBS
p_i	Transmission power of RSU R_i
$g_i^{\text{r2b}}(t)$	Channel power gain of R2B connection in the ith zone
I_i^{v2r}	Interference noise from other R2B connections in $V_i(t)$
$R_i^{\text{r2b}}(t)$	Average transmission rate of R2B connections
$R_{\min}^{\text{r2b}}$	Minimum transmission rate of R2B connections
$T_i^{\text{r2b}}(t)$	Average transmission delay of R2B connections
$T_i(t)$	Total processing delay of load $L_i(t)$
γ	Price of processing data per bit for ES
μ	Price of processing data per bit for CS
δ	Price of renting bandwidth per Hz
$C_i^{\text{comp}}(t)$	Computation cost for processing load $L_i(t)$
$C_i^{\text{comm}}(t)$	Communication cost for processing load $L_i(t)$
$C(t)$	Total cost of system during time slot t

B. Mobility Model

Vehicles enter/leave the road as time passes. At time slot t, we denote $V_i(t)$ as the set of vehicles within coverage of RSU R_i. The length of time slot t is usually short (e.g., milliseconds). Thus, the number of vehicles within coverage area of RSU R_i during one time slot can be considered constant, denoted as $N_i(t)$. Nevertheless, $N_i(t)$ can vary over long time scales (e.g., hours) due to dynamics of traffic flow. The average number of vehicles is related to the average speed on the road [9], i.e.,

$$\overline{v} = v_{\text{lim}}\left(1 - \frac{\overline{N}}{N_{\max}}\right) \tag{1}$$

where $\overline{v}$ is the average speed of vehicles in one zone; $\overline{N}$ is the average number of vehicles in one zone; v_{lim} is the maximum speed for vehicles on the road; and $N_{\max}$ is the maximum number of vehicles in one zone.

C. Task Model

We consider that each vehicle generates at most one task at the beginning of each time slot t. We assume that the tasks generated at each time slot can be completed during the current time slot. Therefore, the task generation events are independently and identically distributed at each time slot. Therefore, we consume that task generation follows the Bernoulli distribution [10] with probability p. Thus, the average task arrival rate is expressed as

$$\lambda = \frac{p}{t}. \tag{2}$$

The size of one task is l bits. At time slot t, the total computation load $L_i(t)$ generated by $V_i(t)$ is

$$L_i(t) = N_i(t)l. \tag{3}$$

D. Computation Model

The vehicles do not drive out of one zone during each time slot t. The vehicles in $V_i(t)$ can complete the computing process before driving out of the zone. Task offloading decisions for all vehicles in $V_i(t)$ are identical, denoted by $a_i(t)$. $a_i(t)$ is a binary variable which represents whether or not $L_i(t)$ is offloaded to RSU R_i at time slot t, i.e.,

$$a_i(t) = \begin{cases} 1, & \text{load } L_i(t) \text{ is offloaded to RSU } R_i \\ 0, & \text{processed by vehicles locally.} \end{cases} \tag{4}$$

1) Local Computing Model: We consider that the CPU processing frequency of all vehicles are identical, denoted as f_v. The number of CPU cycles required to process one bit data is c. Then, the average local computing delay of computation load $L_i(t)$ is

$$T_i^{\text{local}}(t) = a_i(t)\frac{cL_i(t)}{f_v N_i(t)},\ i \in \mathcal{M}. \tag{5}$$

2) Edge Computing Model: Each RSU needs to process a large number of multitypes of tasks in parallel. The computing resources of ES connected to RSU is limited. In addition, due to the dynamic network environment in IoV, the remaining available computing resources of ES is not static. At time slot t, the available computing resources of ES i is $f_i(t)$.

If computation load $L_i(t)$ is offloaded to RSU, the processing delay includes vehicle to RSU (V2R) transmission delay and ES processing delay. We adopt task partition technique during task processing. RSU R_i can divide the computation load $L_i(t)$ into two parts, which are processed by ES S_i and CS, respectively. The slicing proportions of $L_i(t)$ processed by ES S_i and CS are $1 - x_i(t)$ and $x_i(t)$, respectively, i.e.,

$$x_i(t) = \begin{cases} 0 & \text{, Processed all by RSU } R_i \\ (0,1) & \text{, Processed jointly by ES } S_i \text{ and CS} \\ 1 & \text{, Processed all by CS.} \end{cases} \tag{6}$$

If computation load $L_i(t)$ is processed all by ES S_i, i.e., $x_i(t) = 0$, the average computing delay of load $L_i(t)$ is

$$T_i^{\text{comp}}(t) = \frac{cL_i(t)}{f_i(t)N_i(t)}. \tag{7}$$

In this article, we consider a dynamic IoV scenario where the tasks generated from the vehicles have to be processed during the time period when the vehicles locating in the coverage area of the RSU. In our model, it is assumed that the task should be processed within one slot; otherwise, it will be discarded by the ES [10]. Therefore, we do not consider the queuing delay of the edge server in the RSU.

3) Cloud Computing Model: The computation capacity of the CS is much larger than that of ES and vehicle. In addition, the computing results of image processing are usually small. Thus, the bottleneck in the total processing delay may be the transmission delay of task offloading. The transmission delay consists of transmission delay of V2R connections and transmission delay of R2B connections. In comparison, the delay of processing tasks by CS and sending results back to vehicles are negligible generally.

E. Communication Model

We define the state of communication link as non-line of sight (NLoS) when the communication link between two equipments is blocked by the buildings; otherwise, it is defined as line of sight (LoS) [11]. Since RSUs that have limited coverages are generally deployed on both sides of the road, the distance between vehicles and RSUs is close. When a vehicle moves at a uniform speed within the coverage of an RSU, the variation of the distance between the vehicle and the RSU is negligible. Furthermore, we assume that the V2R connection is the LoS link and there is almost no path loss. The MBS is farther away from the RSU than the vehicle, and the R2B connection is usually obscured by obstacles in urban areas. There is a nonnegligible path loss and the signal strength obeys Rayleigh distribution [12]. Therefore, we assume that the channel states of V2R connections are unchanged, and the channel states of R2B connections are time varying.

1) Vehicles to RSUs: RSUs use high-speed optical fiber and gigabit Ethernet for data transmission. Thus, RSUs can meet the data transmission requirements of multiple vehicles and multiple scenarios. RSUs work on the 5.9-GHz LTE V2X (vehicle-to-everything) intelligent transportation system spectrum. Therefore, there is no interference from other channels for V2R connections.

For convenience, we consider the transmission powers of vehicles to be identical, denoted as p_v. The channel gain of V2R connections remains constant during one time slot t, denoted as g_i^{v2r}. The interference noise $I_i^{\text{v2r}}(t)$ depends on other V2R connections, which share the same channel, i.e.,

$$I_i^{\text{v2r}}(t) = (N_i(t) - 1)p_v g_i^{\text{v2r}}. \tag{8}$$

Thus, the average transmission rate between vehicles set $V_i(t)$ and RSU R_i is

$$R_i^{\text{v2r}}(t) = \frac{B_i}{N_i(t)}\log_2\left(1 + \frac{p_v g_i^{\text{v2r}}}{I_i^{\text{v2r}}(t) + \sigma^2}\right) \tag{9}$$

where B_i is the bandwidth of RSU R_i, and σ^2 is average Gaussian white noise. In addition, the V2R connections have minimum transmission rate to ensure the real-time positioning of vehicles [10], i.e.,

$$R_{\text{min}}^{\text{v2r}} = \lambda l. \tag{10}$$

The average transmission delay for offloading the computation load $L_i(t)$ is

$$T_i^{\text{v2r}}(t) = \frac{L_i(t)}{N_i(t)R_i^{\text{v2r}}(t)}. \tag{11}$$

2) RSUs to MBS: In our model, we consider the scenario in which the available bandwidth of the MBS is dynamic. At time slot t, the available bandwidth of the MBS is denoted by $B_0(t)$. RSUs can offload computation load to the CS through the MBS. All RSUs share the available bandwidth of the MBS. The allocation proportion of the bandwidth allocated to RSU R_i is $\eta_i(t)$. The channel state between the RSU R_i and the MBS is time varying, denoted by $g_i^{\text{r2b}}(t)$. The interference noise $I_i^{\text{r2b}}(t)$ from other R2B connections is expressed by

$$I_i^{\text{r2b}}(t) = \sum_{j \neq i, j \in M} p_j g_i^{\text{r2b}}(t). \tag{12}$$

Thus, the transmission rate between RSU R_i and the MBS is

$$R_i^{\text{r2b}}(t) = \eta_i(t)B_0(t)\log_2\left(1 + \frac{p_i g_i^{\text{r2b}}(t)}{I_i^{\text{r2b}}(t) + \sigma^2}\right) \tag{13}$$

where p_i is the transmission power of RSU R_i. If RSU R_i chooses to offload all computation load $L_i(t)$ to the MBS, the average transmission delay from RSU R_i to the MBS should be

$$T_i^{\mathrm{r2b}}(t) = \frac{L_i(t)}{R_i^{\mathrm{r2b}}(t)N_i(t)}. \tag{14}$$

According to (5), (7), (11), and (14), we obtain the total processing delay of computation load $L_i(t)$ as

$$\begin{aligned} T_i(t) = (1 - a_i(t))T_i^{\mathrm{local}}(t) + a_i(t)[T_i^{\mathrm{v2r}}(t) \\ + (1 - x_i(t))T_i^{\mathrm{comp}}(t) + x_i(t)T_i^{\mathrm{r2b}}(t)]. \end{aligned} \tag{15}$$

III. Optimization Scheme

A. Cost of Processing Tasks

The total cost of processing tasks includes computation cost and bandwidth leasing cost.

1) Computation Cost: The computation cost consists of ES computation cost and CS computation cost. The prices of computing data per bit for ES and CS are γ and μ, respectively. Therefore, at time slot t, the computation cost for processing $L_i(t)$ is

$$C_i^{\mathrm{comp}}(t) = a_i(t)[\gamma(1 - x_i(t))L_i(t) + \mu x_i(t)L_i(t)]. \tag{16}$$

2) Communication Cost: The RSUs and the MBS need to rent bandwidth resources for communication. The price of renting bandwidth is δ per hertz. Therefore, at time slot t, the communication cost for processing $L_i(t)$ is

$$C_i^{\mathrm{comm}}(t) = a_i(t)[\delta(\eta_i(t)B_0(t) + B_i)]. \tag{17}$$

According to (16) and (17), at time slot t, the total cost of processing tasks of system is

$$C(t) = \sum_{i\in\mathcal{M}} C_i^{\mathrm{comp}}(t) + \sum_{i\in\mathcal{M}} C_i^{\mathrm{comm}}(t). \tag{18}$$

B. Cost Minimization Problem Formulation

With the quantitative analyses presented in the aforementioned discussions, we formulate the cost minimization problem as follows:

$$(P1) \quad \min_{a_i(t),x_i(t),\eta_i(t)} \frac{1}{T}\sum_{t=0}^{T-1} C(t) \tag{19}$$

$$\text{s.t. } C1: a_i(t) \in \{0, 1\}\ \forall i \in \mathcal{M} \tag{20}$$

$$C2: 0 \le \eta_i(t) \le 1\ \forall i \in \mathcal{M} \tag{21}$$

$$C3: \sum_{i\in\mathcal{M}} \eta_i(t) = 1\ \forall i \in \mathcal{M} \tag{22}$$

$$C4: 0 \le x_i(t) \le 1\ \forall i \in \mathcal{M} \tag{23}$$

$$C5: T_i(t) < t\ \forall i \in \mathcal{M} \tag{24}$$

$$C6: R_i^{\mathrm{v2r}}(t) \ge R_{\min}^{\mathrm{v2r}}\ \forall i \in \mathcal{M} \tag{25}$$

$$C7: R_i^{\mathrm{r2b}}(t) \ge R_{\min}^{\mathrm{r2b}}\ \forall i \in \mathcal{M}. \tag{26}$$

The optimization objective is to minimize the average task processing cost of the system in the long timescale. The decision variable $a_i(t)$ represents the offloading decision of task $L_i(t)$. $x_i(t)$ represents the slicing proportion of computation load $L_i(t)$ on RSU R_i. $\eta_i(t)$ represents the slicing proportion of bandwidth allocated to RSU R_i.

C1 indicates the vehicles in $V_i(t)$ can choose local computing or offloading computing. C2 shows that the bandwidth slicing proportion of the MBS should be in range 0–1. C3 illustrates that the summation of the proportions of bandwidth allocated to RSUs should be 1. C4 illuminates that the computation load slicing proportion is in range 0–1. C5 indicates that the average processing delay of load $L_i(t)$ should not exceed the maximum task processing delay (i.e., length of time slot t). Finally, C6 and C7 illustrate that the transmission rates of V2R and R2B connections are not less than the minimum transmission rate, respectively.

IV. DRL-Based CORA Algorithm Design

The aforementioned optimization problem is a mixed integer programming problem with a very high computational complexity. Thus, we choose DRL to solve this problem.

A. MDP-Based Problem Formulation

Due to the randomness of vehicle density and the dynamic of the network environment in the IoV, the computational complexity of solving this problem by traditional mathematical methods will be extremely high. To attack this challenge, we reformulate the problem by an MDP model, and apply the DRL technique to solve this problem. The MDP model is generally represented by a 4-tuple $(\mathcal{S}, \mathcal{A}, \mathcal{T}, \mathcal{R})$. Here, $\mathcal{S}$ represents the system states, and $\mathcal{A}$ represents the system actions. $\mathcal{T} = p(s_{t+1}|s_t, a_t)$ is the state transition probability, but $\mathcal{T}$ is unknown in our problem. $\mathcal{R}$ is the reward of executing a action based on a state. With the aforementioned definitions, our MDP model of this problem can be formulated as follows.

1) State: At time slot t, the agent of the system observes the dynamic environment of IoV. The agent can obtain state information of computation load, channel states, available computing resources of ES, and bandwidth of the MBS. The state s_t consists of the following parameters:

$$\begin{aligned} s(t) = \{&L_1(t), L_2(t), \ldots, L_i(t), \ldots, L_M(t) \\ &f_1(t), f_2(t), \ldots, f_i(t), \ldots, f_M(t), B_0(t) \\ &g_1^{\mathrm{r2b}}(t), g_2^{\mathrm{r2b}}(t), \ldots, g_i^{\mathrm{r2b}}(t), \ldots, g_M^{\mathrm{r2b}}(t) \\ &T_1(t-1), T_2(t-1), \ldots, T_i(t-1), \ldots, T_M(t-1)\} \end{aligned} \tag{27}$$

where $L_i(t)$ is the total size of arrival tasks within coverage of RSU R_i. $f_i(t)$ is the available computing resources of ES S_i. $B_0(t)$ is the available bandwidth of the MBS. $g_i^{\mathrm{r2b}}(t)$ is the channel state of V2R connection between the RSU R_i and the MBS. $T_i(t-1)$ is the average processing delay of the task $L_i(t-1)$ in previous time slot $t-1$.

2) Action: At time slot t, the agent executes action $a(t)$ based on the state $s(t)$. The vehicles within the coverage of RSU R_i make decision of task offloading. RSU R_i makes the decision of computation load slicing ratio. The MBS makes decision of the bandwidth allocation ratio. The action a_t consists of the following parameters:

$$\begin{aligned} a(t) = \{&a_1(t), a_2(t), \ldots, a_i(t), \ldots, a_M(t) \\ &x_1(t), x_2(t), \ldots, x_i(t), \ldots, x_M(t) \\ &\eta_1(t), \eta_2(t), \ldots, \eta_i(t), \ldots, \eta_M(t)\} \end{aligned} \tag{28}$$

where $a_i(t)$ is the decision for tasks offloading of vehicles in $V_i(t)$. $x_i(t)$ is the decisions for tasks slicing of RSU R_i. $\eta_i(t)$ is the decisions for bandwidth allocating of the MBS.

3) Reward: At time slot t, the agent will get reward $r(t)$ after executing action $a(t)$ at state $s(t)$. Whether the training process of the DRL model converges depends on the reward function. The reward function of our problem contains the objective function (19). Thus, the agent need to minimize the long-term cumulative reward. In addition, when quality of service (QoS) requirements (24)–(26) are not satisfied, a penalty with a large positive value is added to the reward function. Thus, the reward function, which consists of two parts of utility function and QoS penalty, is expressed as

$$r(t) = \sum_{i \in \mathcal{M}} E[C_i(t) + P_i(t)]. \tag{29}$$

Considering the long-term minimization of the average reward value, our objective can be expressed as

$$R = \min \frac{1}{T} \sum_{t=0}^{T-1} r(t). \tag{30}$$

B. DRL Algorithm

DRL can be divided into on-policy learning and off-policy learning according to the learning methods. The difference between them is that the method used to update value is an established policy or a new policy. After analyzing our optimization problem P1, we apply off-policy learning and propose a CORA algorithm based on the twin delayed deep deterministic policy gradient (TD3) algorithm [13], which performs well in dealing with continuous action space.

At time slot t, the agent observes the real-time state $s(t)$ of the IoV. According to the state $s(t)$, the agent gets the optimal action $a(t)$ of CORA through the CORA algorithm. The DRL model of the CORA algorithm has two neural networks, which includes main networks and target networks, and both of them include one actor network and two critic networks. The parameters of actor network and critic networks in main networks are denoted as $\phi, \theta_1,$ and θ_2. The parameters of actor network and critic networks in target networks are expressed as $\phi', \theta_1',$ and θ_2'. The actor network π_ϕ is used to generate action $a(t)$, i.e.,

$$a = \pi_\phi(s) + \epsilon, \epsilon \mathcal{N}(0, \tau) \tag{31}$$

ϵ is the noise, which can increase exploration of the DRL model. ϵ follows normal distribution with mean value equal to 0 and variance equal to τ.

After the agent performs action $a(t)$, the system state is updated from $s(t)$ to $s(t+1)$. The agent obtains the system reward $r(t)$ corresponding to $(s(t), a(t))$. Then, the state transition is stored in the reply memory, which is used for updating

Algorithm 1: CORA.

Input: Training episode number N; training step number T; reply memory size D; batch size S; target smoothing coefficient τ; exploration noise ϵ; actor learning rate l_a; critic learning rate. l_c; actor network weights ϕ; actor target network weights ϕ'; critic network weights θ_1, θ_2; critic target network weights ${\theta_1}', {\theta_2}'$

Output: Decisions for tasks offloading of vehicles $a(t)$; decisions for tasks slicing of RSU $x(t)$; decisions for bandwidth allocating of the MBS $\eta(t)$.

```
 1: for $episode \leftarrow 1$ to $N$ do
 2:   Initialize environment parameters, get initial state $s_0$;
 3:   for $step \leftarrow 1$ to $T$ do
 4:     Obtain action $a(t)$ using (31);
 5:     Execute action $a(t)$;
 6:     Obtain the next state $s(t+1)$ and reward $r(t)$
 7:     using (29);
 8:     Store transition $\langle s(t), a(t), s(t+1), r(t) \rangle$ into reply
 9:     memory $D$ for training networks;
10:     if step $\geq S$ then
11:       Randomly sample $N$ experience from reply
12:       memory;
13:       Update $\phi$ with (32);
14:       Update $\theta_1$ and $\theta_2$ with (33);
15:       Update target networks with $\tau = 0.005$:
16:       $\phi' \leftarrow \tau\phi + (1-\tau)\phi'$;
17:       ${\theta_i}' \leftarrow \tau\theta_i + (1-\tau){\theta_i}'$;
18:     end if
19:   end for
20: end for
```

network parameters of TD3. TD3 uses the deterministic policy gradient to update the parameters of the main actor network. The deterministic policy gradient is

$$\nabla_\phi J(\phi) = N^{-1} \sum \nabla_a Q_{\theta_1}(s,a)|_{a=\pi_\phi(s)} \nabla_\phi \pi_\phi(s) \tag{32}$$

where Q_{θ_1} is the first critic network of main network, and π_ϕ is the actor network of main network.

The parameters of main critic networks are updated by following formula:

$$\theta_i = \arg\min_{\theta_i} N^{-1} \sum (y - Q_{\theta_1}(s,a))^2 \tag{33}$$

where $Q_{\theta_1}(s,a)$ is the current value of Q, and y is the target value of Q calculated by target critic networks. y is calculated by

$$y = r + \gamma \min_{i=1,2} Q_{\theta_i'}(s', \widetilde{a}) \tag{34}$$

where r is the reward, γ is the discount factor range in [0, 1], and $\widetilde{a}$ is the action generated by the target actor network π_{ϕ^1} based on state s. The detailed algorithm for solving our problem is shown in Algorithm 1. The specific algorithm flow chart is shown in Fig. 2.

Theoretically, we can prove that our CORA algorithm is able to obtain the (near-)optimal solution of the original problem

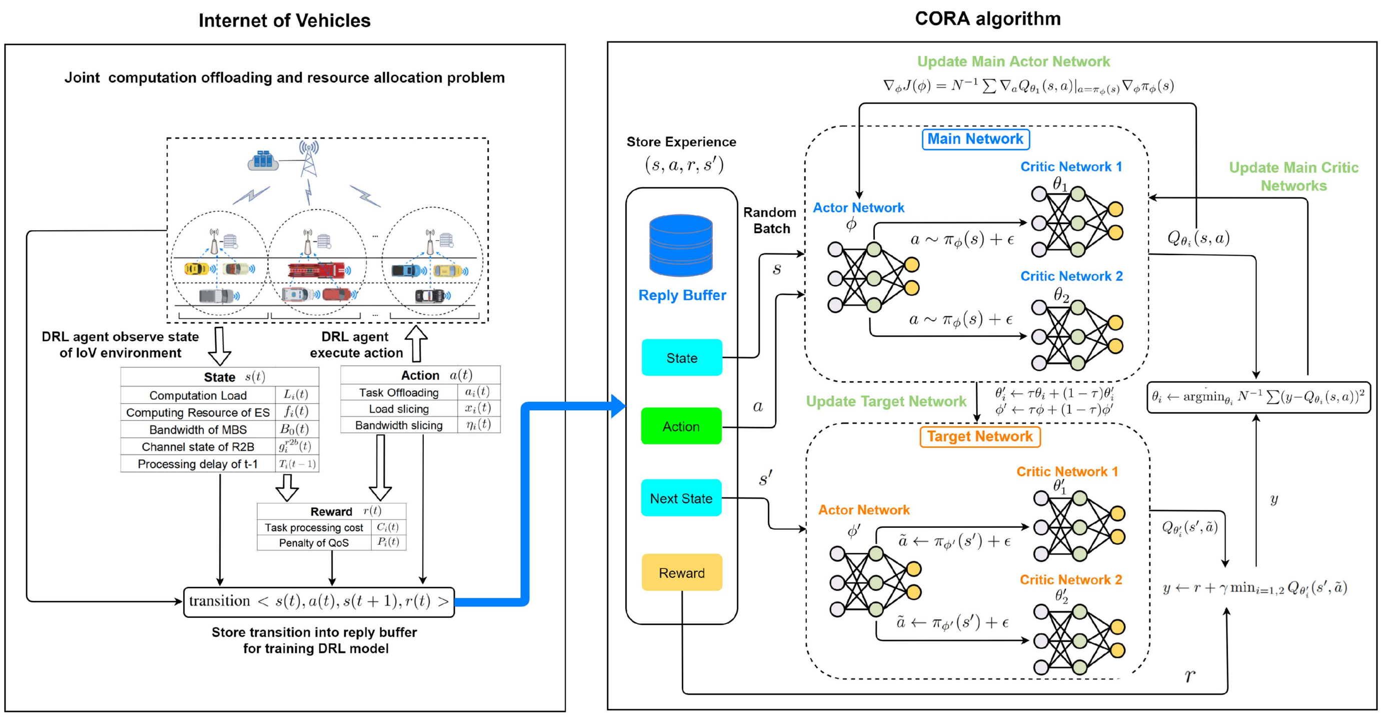
Internet of Vehicles
Joint computation offloading and resource allocation problem
DRL agent observe state of IoV environment
DRL agent execute action
State $s(t)$
Computation Load $L_i(t)$
Computing Resource of ES $f_i(t)$
Bandwidth of MBS $B_0(t)$
Channel state of R2B $g_i^{r2b}(t)$
Processing delay of t-1 $T_i(t-1)$
Action $a(t)$
Task Offloading $a_i(t)$
Load slicing $x_i(t)$
Bandwidth slicing $\eta_i(t)$
Reward $r(t)$
Task processing cost $C_i(t)$
Penalty of QoS $P_i(t)$
transition $< s(t), a(t), s(t+1), r(t) >$
Store transition into reply buffer for training DRL model
CORA algorithm
Update Main Actor Network
$\nabla_\phi J(\phi) = N^{-1}\sum \nabla_a Q_{\theta_1}(s,a)|_{a=\pi_\phi(s)} \nabla_\phi \pi_\phi(s)$
Store Experience
(s,a,r,s')
Random Batch
Reply Buffer
State
Action
Next State
Reward
Main Network
Actor Network
Critic Network 1
Critic Network 2
$a \sim \pi_\phi(s) + \epsilon$
Update Main Critic Networks
$Q_{\theta_i}(s,a)$
$\theta_i \leftarrow \mathrm{argmin}_{\theta_i} N^{-1} \sum (y - Q_{\theta_i}(s,a))^2$
Update Target Network
$\theta_i' \leftarrow \tau\theta_i + (1-\tau)\theta_i'$
$\phi' \leftarrow \tau\phi + (1-\tau)\phi'$
Target Network
$\tilde{a} \leftarrow \pi_{\phi'}(s') + \epsilon$
$Q_{\theta_i'}(s',\tilde{a})$
$y \leftarrow r + \gamma \min_{i=1,2} Q_{\theta_i'}(s',\tilde{a})$

Fig. 2. Overview of CORA.

P1 upon finishing its iterations. To this end, we first build a continuous-time Markov chain (CTMC) to model the dynamic IoV scenario. Then, we assign a reward to each state of the CTMC according to the reward function in the MDP Bellman Equation, and thus, we construct a Markov reward model (MRM). Next, we can prove that the MDP algorithms can obtain the steady-state optimal reward of the MRM, whose details of the proof can be found in our previous work [14].

V. Performance Evaluation

A. Simulation Settings

We consider a scenario in which there is a 400-m long unidirectional straight road, covered by four RSUs with no overlapping areas. The covering radius of each RSU is 100 m. The moving speed of vehicles on the road is in the scale of 20–28 m/s. The maximum number $N_{\max}$ of vehicles in each zone is 40. The size of task is randomly distributed in range 5–10 MB. The detailed parameters of IoV are shown in Table II.

Our simulation experiments are conducted on PCs with 8-core Intel i7-7700H CPU and 32-GB DRAM. We use NVIDIA GTX 1060 GPU with 6-GB memory and CUDA 11.4 to train our DRL model. The algorithm runs on Python 3.6 and pytorch 1.9.

The TD3 model includes six neural networks. Each neural network has three hidden layers with (400, 400, 300) neurons. We add rectified linear unit on each hidden layer. We train the model for 500 episodes, and each episode includes 300 steps. The soft update parameter τ and discounted factor γ are set as 0.005 and 0.99, respectively. The number of neural nodes in the three hidden layers of actor networks and critic networks are set to be 400, 400, and 300, respectively. The activation function

TABLE II
IoV Parameters Setting

Definitions	Notations	Value
Number of RSU and ES	M	4
Speed of vehicles	$\overline{v}$	20m/s∼28m/s
Maximum number of vehicles in a zone	$N_{\max}$	40
Probability of task generation	p	0.7
Task size	l	5MB∼10MB
Processing density of one bit	c	300
CPU frequency of vehicle	f_v	20GHz
CPU frequency of ES	f_i	100GHz
Bandwidth of RSU	B_i	20MHz
Transmission power of vehicle	p_v	200mW
Gaussian white noise	σ^2	-100dBm [16]
Bandwidth of the MBS	B_i	40MHz
Transmission power of RSU	p_v	2W
Price of renting bandwidth per Hz	δ	0.002 [17]
Computing resource price of MEC($/bit)	γ	0.03 [4]
Computing resource price of Cloud($/bit)	μ	0.015 [4]

between hidden layers is selected as rectified linear unit (ReLU), and the optimizer of DRL is Adam. The learning rates of the actor and critic networks are set to 0.001 and 0.002, respectively. The capacity of the memory buffer is set as 5000, and the batch size of each episode is set as 128.

B. Comparison Experiments With Non-DRL Algorithms

We select the following four traditional non-DRL algorithms to verify the effectiveness of our CORA algorithm in finding optimal task offloading and resource allocation polices.

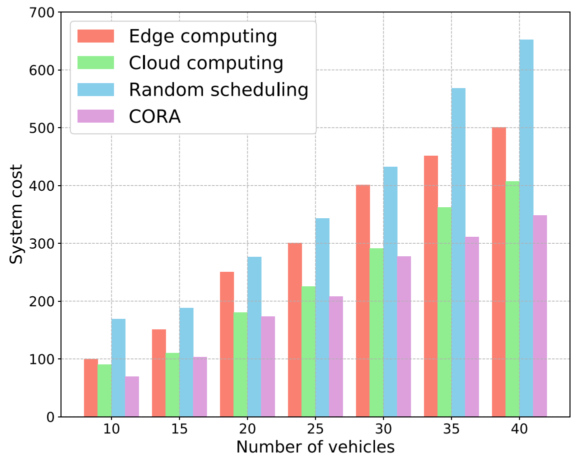

Fig. 3. System cost with different number of vehicles.

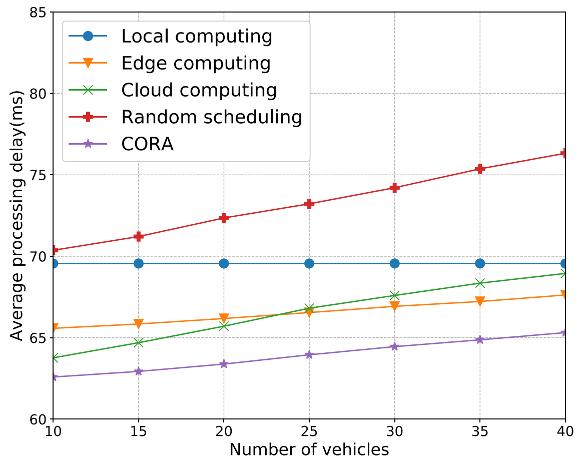

Fig. 4. Average processing delay with different number of vehicles.

1) *Local computing:* All the computation workload of each zone is processed by vehicles locally.
2) *Edge computing:* In each zone, all vehicles submit their workload to ES for processing.
3) *Cloud computing:* The computation workload is uploaded and processed by the CS.
4) *Random scheduling:* The decisions for task offloading and resource allocation are randomly selected from vehicles, ES, or CS within each zone. The task slicing ratio and bandwidth allocation ratio are also random.

Fig. 3 shows the processing cost with different number of vehicles. We can see that, for all the approaches, the processing cost of task processing grows with the increase of the number of vehicles in an area. Among them, random scheduling performs the worst in system cost, because both task offloading and resource allocation policies are randomly selected and such approach is not able to make intelligent decisions according to the dynamic network environment. Besides, since CS has a lower computing price than ES and the renting bandwidth cost of R2B connections is much lower than task computation cost, the cost of cloud computing is lower than that when all tasks are processed by the ES, especially when the workload is heavy (with large number of vehicles). This also proves the scale economy of cloud computing. We can see that our CORA algorithm performs best among different number of vehicles, which validates that our approach dominates the other traditional task offloading and resource allocation approaches.

Fig. 4 shows the average processing delay of different processing methods with different number of vehicles. We find out that the processing delay of task processing increases when the workload generated from each zone gets heavier. Our CORA performs best among any number of vehicles. For the random scheduling, its average processing delay is the highest, because the local computation delay is independent of the number of vehicles in one zone, so the average processing delay of local method is stable. When the number of vehicles is less than or equal to 25, the delay of cloud computing is smaller than that of edge computing. Because the computing resources of CS are much larger than that of ES. The tasks can be processed immediately after being offloaded to the CS. When the number of vehicles exceeds 25, the delay of edge computing is lower than that of cloud computing, because the interference from wireless connections increases with the increasing of vehicles which leads to the increasing of R2B transmission delay. Thus, in our scenario, edge computing is suitable for large amounts of data, while cloud computing is suitable for small amounts of data. We compare Fig. 4 with Fig. 3. Although the cost of cloud computing is lower than that of edge computing under different vehicle numbers, the delay of cloud computing is higher when the number of vehicles increases, and CORA can achieve a good tradeoff between cost and delay.

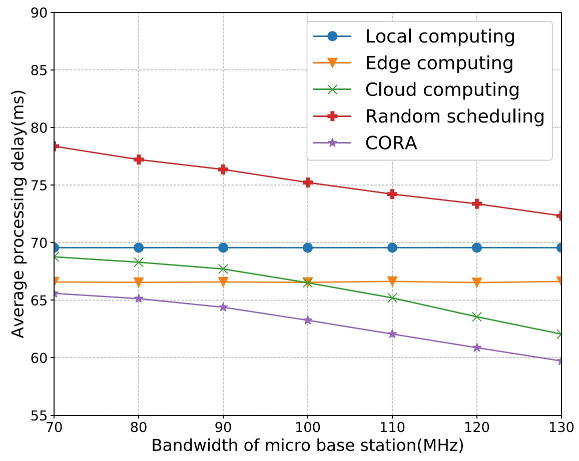

Fig. 5. Average processing delay with different bandwidth of the MBS.

Fig. 5 shows the average processing delay with different bandwidths of the MBS. We find out all average processing delay increases as the bandwidth increases except local computing and edge computing. Because edge computing and local computing do not use the MBS. CORA performs best among different bandwidths of the MBS. Fig. 5 illustrates that in the dynamic network environment scenario we considered, when bandwidth resource is scarce, we tend to offload tasks to ES for processing, because low bandwidth can bring a high R2B transmission delay. In contrast, when bandwidth resource is sufficient, we

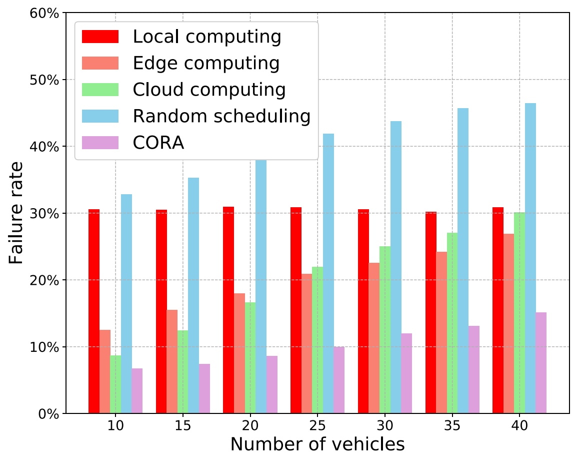

Fig. 6. Failure rate with different number of vehicles.

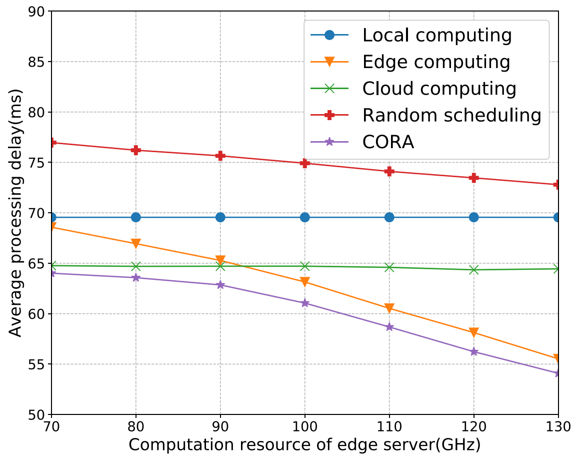

Fig. 7. Average processing delay with different computation resources of ES.

tend to offload tasks to CS for computing, because the the CS with abundant computing resources can quickly process tasks, Fig. 5 shows that CORA can flexibly make decisions of task segmentation and bandwidth allocation according to the real-time network state.

Fig. 6 shows the failure rate with different number of the vehicles. We define the failure rate as the ratio of the number of tasks whose processing time exceeds one time slot to the number of all tasks. As the number of vehicles increases, the failure rate of all algorithms except local computing also increases simultaneously. Because the increasing number of vehicles leads to more intense competition for the computing resources and bandwidth resources. As a result, the number of tasks with processing time exceeding the time limit also increases. Among all the scheduling schemes, the failure rate of our CORA algorithm is consistently the lowest. Because our CORA algorithm is able to flexibly utilize the computational and communication resources of vehicles, edges, and cloud.

Fig. 7 shows the average processing delay with different computation resources of the edge server. We can find out that the processing delay decreases as the computation resource increases of all methods. Random scheduling has the highest

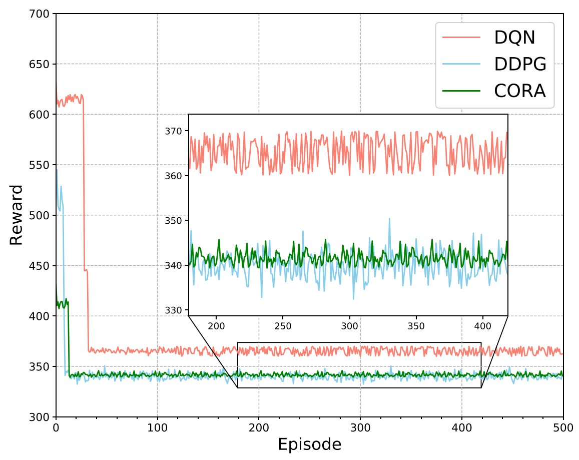

Fig. 8. Reward of the training process.

latency due to improper scheme. When the available computing resources of edge servers are lacking, the processing delay of cloud computing is lower than edge computing. In contrast, when the available computing resources are abundant, the processing delay of edge computing is lower than that of cloud computing. CORA can determine the optimal workload slicing ratio in any case to minimize the overall processing delay. In addition, Fig. 7 also indicates that in our dynamic scenario, edge-cloud collaboration can effectively deal with various workload conditions.

C. Comparison Experiments With DRL Algorithms

Besides traditional scheduling approaches, we further implement two DRL approaches for comparison to verify the convergence and performance of the CORA method. We select DQN and DDPG algorithms, which have been widely applied in unsupervised reinforcement learning.

1) *Deep Q network (DQN):* The action space of the DQN is discrete. Therefore, in our experiments, we have to cut the original continuous value actions into discrete intervals, including the task slicing ratio and the bandwidth allocation ratio. The number of intervals will be further discussed in the following discussions.
2) *Deep deterministic policy gradient (DDPG):* DDPG inherits the advantages of the DQN, and it uses strategy gradient technology to solve the problem caused by a high-dimensional action space. The action space of DDPG is continuous.

Fig. 8 shows the convergence and performance of the three approaches. We can observe from the experimental results that both DDPG and CORA converge rapidly before 20 episodes, while the convergence rate of the DQN is the slowest, because the DQN has low learning efficiency in a high-dimensional action space. Furthermore, we can also see that DQN performs the worst among the three algorithms. The system cost of CORA is slightly better than that of DDPG. In addition, the training process of CORA is more stable, because CORA delays the update of actor network to make the training of the actor network

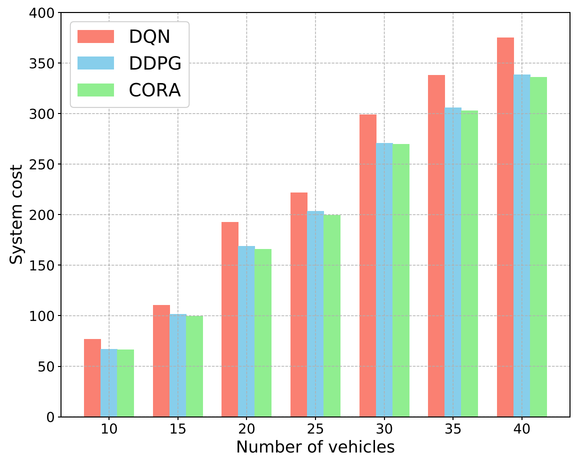

Fig. 9. System cost with different number of vehicles.

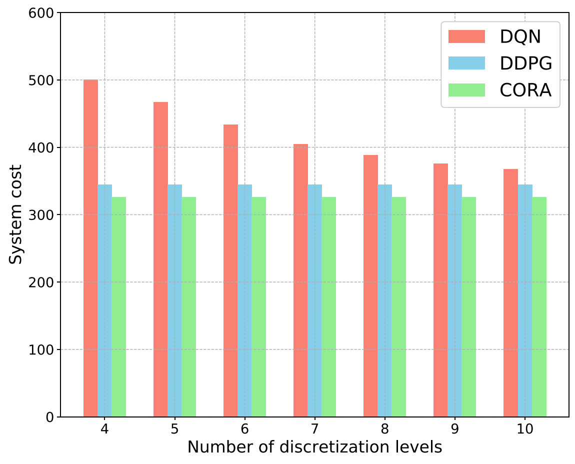

Fig. 11. System cost with different number of discretization levels in the DQN.

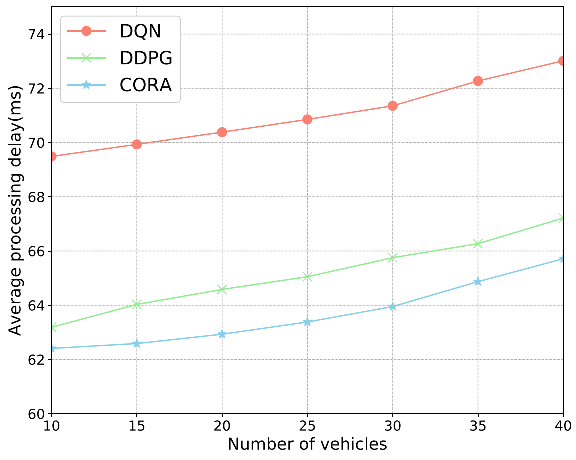

Fig. 10. Average processing delay with different number of vehicles.

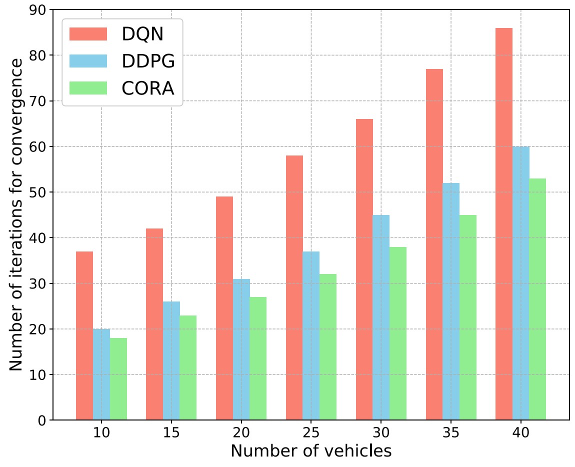

Fig. 12. Number of iterations for convergence with different number of vehicles.

more stable. In summary, our CORA algorithm has the best performance among the three DRL methods.

Fig. 9 illustrates the processing cost with different number of vehicles. We can see that, with the increase of vehicles in each zone, the processing costs grow up for all the three algorithms. Also, the DQN performs the worst due to its inadequate capability in dealing with optimization problems with a high-dimensional action space. Our CORA has the best performance among different vehicle numbers. This is because the network of CORA has two more critic networks than DDPG, then we can choose smaller values to estimate Q value and avoid overestimation. Therefore, our CORA performs slightly better than DDPG.

Fig. 10 illuminates the average processing delay with different number of vehicles. With the increase of vehicles in one zone, more vehicles join the computation offloading process, resulting in more interference between V2R channels. Meanwhile, the bandwidth of the RSUs are fixed. Therefore, each vehicle is allocated less bandwidth resource which leads to processing delays going up. Comparing to DQN and DDPG, our CORA is able to achieve the minimum processing delay by adjusting its policies to the dynamic environment.

Fig. 11 shows the processing cost of system with different number of discretization levels of the DQN. It is shown that the system cost can be further minimized by dividing the continuous action space into more discrete slices in the DQN. However, DQN still performs worse than DDPG and CORA, and suffers from high computational overhead. Due to the strong ability in dealing with continuous action space and fine-grained dynamic optimization, our approach performs the best among the three DRL approaches.

Fig. 12 shows the numbers of iterations for convergence with different number of vehicles. As the number of vehicles increases, the number of iterations required for different DRL algorithms to reach convergence also increases. Because the increase in the number of vehicles causes the state action space of the DRL model to become larger. Therefore, the DRL algorithm requires more training iterations to obtain the optimal policy. Moreover, our CORA algorithm consistently requires the least number of iterations compared to DQN and DDPG. Because we use TD3 as the basis of the CORA algorithm, and TD3 delays the update of the actor network and makes the training of the actor network more stable and faster.

VI. Related Work

Cloud computing, edge computing, and their collaboration can effectively reduce the computation workload of mobile devices, making it possible for various data-intensive and computational-intensive mobile services. In such scenarios, CORA are the most important factors affecting the QoS, which have attracted much attention from the researchers. There have been several existing works dedicating to the modeling and optimization of CORA.

Chen et al. [17] investigated the multiuser task offloading problem in end-edge cloud systems where all user devices compete for limited communications and computing resources. A game-based decentralized task unloading method was proposed to obtain the Nash equilibrium unloading strategy. Feng et al. [18] studied the transmission and offloading strategies in the Internet of Things (IoT) and the fog computing system supported by nonorthogonal multiple access. Li et al. [19] studied a joint RSUs selection and resource allocation problem. The objective was to reduce the total task offloading delay constrained by bandwidth and computing resources. Ni et al. [20] studied the task allocation in mobile crowdsensing. Ning et al. [16] proposed an offloading system for vehicle edge computing to jointly optimize task scheduling and resource allocation strategies to maximize quality of experience (QoE). Zhao et al. [4] designed a cloud-edge cooperative offloading method, which can effectively improve the system utility. Zeng et al. [21] proposed an edge computing model for volunteer-assisted vehicles. The volunteer vehicles were encouraged to help overloaded vehicular edge computing (VEC) servers by receiving rewards from VEC servers. Lyu et al. [22] proposed a semidistributed heuristic offloading decision algorithm to maximize system utility. Liu et al. [23] considered that vehicles can be used as mobile edge servers to provide computing service for nearby devices. The optimization objective was maximizing the long-term utility of IoV. However, most of the aforementioned works used binary schemes for task offloading. Although the efficiency of the optimization algorithms is commonly extremely high, in large-scale systems with complex edge-cloud collaboration, such schemes are so primitive to obtain the optimal policies.

Partial offloading technique has been proposed recently. In large-scale systems with complex edge-cloud collaboration, partial offloading approach can obtain a more flexible computational offloading strategy compared to binary offloading. Li et al. [8] proposed a task partial offloading and scheduling algorithm to determine the execution order of task offloading. Ning et al. [24] considered the partial offloading problem of cloud-edge collaboration, which was solved by the iterative heuristics method. Dai et al. [25] proposed a partial computation offloading scheme where vehicle tasks are divided into two parts and processed as local and edge computation to effectively improve the vehicle performance. Ren et al. [26] proposed a joint computation method for offloading and transmitting the power network allocation scheme, which considers both binary offloading and partial offloading. Zhao et al. [27] considered partial offloading of computational tasks from vehicles to other vehicles, unmanned aerial vehicles (UAVs), and MEC servers at fixed locations. Zhang et al. [28] presented a load balancing of computational resources on edge servers and proposes a load balancing task offloading scheme. Feng et al. [18] proposed a transmission and offloading strategies for IoT, where the tasks of fog nodes can be partially computed locally and another part computed by other idle fog nodes. The dynamic of the network environment and the randomness of vehicle traffic in the IoV scenario usually cannot be ignored. For example, users tend to offload computing when the edge servers have rich computing resources, while they prefer local computing when the edge servers are busy. However, most of the aforementioned works do not take into account these dynamic factors.

Recently, the emergence of DRL has attracted much attention in the community of IoV. As a novel powerful tool for solving mathematical problems with a high computational complexity, some well-designed DRL-based approaches can solve the computing offloading and resource allocation problems with high efficacy. Cui et al. [29] proposed a satellite-assisted vehicle-to-vehicle (V2V) communication scenario. In this work, the Lagrange multiplier method and DRL were used to solve joint offloading and resource allocation problem to maximize the long-term reward of offloading. Lee et al. [30] proposed a circumstance-independent approach to effectively solve the resource allocation problem in different network environments based on DRL. Yang et al. [31] used the DRL algorithm to effectively solve the intelligent transmission scheduling problem of IoT system under high-dimensional variables. Ye et al. [32] proposed a V2V communication distributed resource allocation mechanism based on the DQN. The agent of the DRL model can effectively minimize the interference of vehicle-to-infrastructure connections. Luo et al. [33] presented a joint data scheduling problem for communication and computational resource allocation and uses an enhanced DQN algorithm to solve the problem. Zhan et al. [34] investigated an important computational offloading scheduling problem in a typical VEC scenario and unites the DRL algorithm and convolutional neural network to find the optimal offloading policy. Ke et al. [35] designed a task computational offloading model for a heterogeneous vehicular network. The model is trained using the Dueling DQN algorithm. Tian et al. [36] proposed a multiintelligence DRL-based resource allocation framework to jointly optimize the channel allocation and power control.

Different from the aforementioned works, in this article, we consider the collaboration of CC and MEC for IoV and try to obtain the near-optimal offloading policy with arbitrary proportion between CC and MEC. To make the computation offloading policy more flexible and better adapted to dynamic environment, we consider both binary offloading and partial offloading. Furthermore, we fully consider the dynamics of the computing and communication systems, including random traffic flow, dynamic computing resources of the edge servers, and bandwidth resources of the MBSs. Finally, we construct more actor networks and critic networks in our DRL algorithm to make our training process more stable avoiding overestimation without introducing additional notable computational overhead. Our approach is expected to be more powerful in obtaining the long-term optimal CORA policies, and more practical in real-world dynamic IoV scenarios.

VII. Conclusion

In this article, we study the problem of CORA for edge-cloud collaboration in IoV based on DRL. The objective of the problem is to minimize the total cost of processing tasks with the delay and transmission rate constraints. Furthermore, we consider a dynamic traffic scenario in which the vehicle density, the channel states of R2B connections, the computing resources of the ES, and the bandwidth of the MBS are time varying. We reconstruct the problem into an MDP model, and design a CORA algorithm based on DRL. Extensive simulation experiments are conducted, and we compare our CORA algorithm with existing non-DRL algorithms and DRL algorithms. The results show that our algorithm outperforms other algorithms in processing delay and cost. In the future, we will consider the scenario that multiple vehicles compete for computing resources of edge servers.

References

[1] H. Peng and X. Shen, "Deep reinforcement learning based resource management for multi-access edge computing in vehicular networks," *IEEE Trans. Netw. Sci. Eng.*, vol. 7, no. 4, pp. 2416–2428, Oct.–Dec. 2020.

[2] K. Sasaki, N. Suzuki, S. Makido, and A. Nakao, "Vehicle control system coordinated between cloud and mobile edge computing," in *Proc. 55th Annu. Conf. Soc. Instrum. Control Eng. Jpn.*, 2016, pp. 1122–1127.

[3] Q. Jiang, N. Zhang, J. Ni, J. Ma, X. Ma, and K.-K. R. Choo, "Unified biometric privacy preserving three-factor authentication and key agreement for cloud-assisted autonomous vehicles," *IEEE Trans. Veh. Technol.*, vol. 69, no. 9, pp. 9390–9401, Sep. 2020.

[4] J. Zhao, Q. Li, Y. Gong, and K. Zhang, "Computation offloading and resource allocation for cloud assisted mobile edge computing in vehicular networks," *IEEE Trans. Veh. Technol.*, vol. 68, no. 8, pp. 7944–7956, Aug. 2019.

[5] S. Liu, L. Liu, J. Tang, B. Yu, Y. Wang, and W. Shi, "Edge computing for autonomous driving: Opportunities and challenges," *Proc. IEEE*, vol. 107, no. 8, pp. 1697–1716, Aug. 2019.

[6] F. Lyu et al., "Service-oriented dynamic resource slicing and optimization for space-air-ground integrated vehicular networks," *IEEE Trans. Intell. Transp. Syst.*, vol. 23, no. 7, pp. 7469–7483, Jul. 2022.

[7] H. Wu, F. Lyu, C. Zhou, J. Chen, L. Wang, and X. Shen, "Optimal UAV caching and trajectory in aerial-assisted vehicular networks: A learning-based approach," *IEEE J. Sel. Areas Commun.*, vol. 38, no. 12, pp. 2783–2797, Dec. 2020.

[8] M. Li, J. Gao, L. Zhao, and X. Shen, "Deep reinforcement learning for collaborative edge computing in vehicular networks," *IEEE Trans. Cogn. Commun. Netw.*, vol. 6, no. 4, pp. 1122–1135, Dec. 2020.

[9] W. L. Tan, W. C. Lau, O. Yue, and T. H. Hui, "Analytical models and performance evaluation of drive-thru internet systems," *IEEE J. Sel. Areas Commun.*, vol. 29, no. 1, pp. 207–222, Jan. 2011.

[10] Q. Ye, W. Shi, K. Qu, H. He, W. Zhuang, and X. Shen, "Joint RAN slicing and computation offloading for autonomous vehicular networks: A learning-assisted hierarchical approach," *IEEE Open J. Veh. Technol.*, vol. 2, no. 6, pp. 272–288, Jun. 2021.

[11] L. Zhao, K. Yang, Z. Tan, X. Li, S. Sharma, and Z. Liu, "A novel cost optimization strategy for SDN-enabled UAV-assisted vehicular computation offloading," *IEEE Trans. Intell. Transp. Syst.*, vol. 22, no. 6, pp. 3664–3674, Jun. 2021.

[12] L. Liang, H. Ye, and G. Y. Li, "Spectrum sharing in vehicular networks based on multi-agent reinforcement learning," *IEEE J. Sel. Areas Commun.*, vol. 37, no. 10, pp. 2282–2292, Oct. 2019.

[13] S. Fujimoto, H. Hoof, and D. Meger, "Addressing function approximation error in actor–critic methods," in *Proc. Int. Conf. Mach. Learn.*, 2018, pp. 1587–1596.

[14] J. Huang and C. Lin, "Agent-based green web service selection and dynamic speed scaling," in *Proc. IEEE 20th Int. Conf. Web Serv.*, 2013, pp. 91–98.

[15] J. Huang et al., "AOI-aware energy control and computation offloading for industrial IoT," *Future Gener. Comput. Syst.*, vol. 139, pp. 29–37, 2023.

[16] Z. Ning, P. Dong, X. Wang, J. J. Rodrigues, and F. Xia, "Deep reinforcement learning for vehicular edge computing: An intelligent offloading system," *ACM Trans. Intell. Syst. Technol.*, vol. 10, no. 6, pp. 1–24, 2019.

[17] Y. Chen, J. Zhao, Y. Wu, and X. S. Shen, "QoE-aware decentralized task offloading and resource allocation for end-edge-cloud systems: A game-theoretical approach," *IEEE Trans. Mobile Comput.*, to be published, doi: 10.1109/TMC.2022.3223119.

[18] W. Feng et al., "Energy-efficient collaborative offloading in NOMA-enabled fog computing for Internet of Things," *IEEE Internet Things J.*, vol. 9, no. 15, pp. 13794–13807, Aug. 2022.

[19] S. Li, N. Zhang, H. Chen, S. Lin, O. A. Dobre, and H. Wang, "Joint road side units selection and resource allocation in vehicular edge computing," *IEEE Trans. Veh. Technol.*, vol. 70, no. 12, pp. 13190–13204, Dec. 2021.

[20] J. Ni, K. Zhang, Q. Xia, X. Lin, and X. S. Shen, "Enabling strong privacy preservation and accurate task allocation for mobile crowdsensing," *IEEE Trans. Mobile Comput.*, vol. 19, no. 6, pp. 1317–1331, Jun. 2020.

[21] F. Zeng, Q. Chen, L. Meng, and J. Wu, "Volunteer assisted collaborative offloading and resource allocation in vehicular edge computing," *IEEE Trans. Intell. Transp. Syst.*, vol. 22, no. 6, pp. 3247–3257, Jun. 2021.

[22] X. Lyu, H. Tian, C. Sengul, and P. Zhang, "Multiuser joint task offloading and resource optimization in proximate clouds," *IEEE Trans. Veh. Technol.*, vol. 66, no. 4, pp. 3435–3447, Apr. 2017.

[23] Y. Liu, H. Yu, S. Xie, and Y. Zhang, "Deep reinforcement learning for offloading and resource allocation in vehicle edge computing and networks," *IEEE Trans. Veh. Technol.*, vol. 68, no. 11, pp. 11158–11168, Nov. 2019.

[24] Z. Ning, P. Dong, X. Kong, and F. Xia, "A cooperative partial computation offloading scheme for mobile edge computing enabled Internet of Things," *IEEE Internet Things J.*, vol. 6, no. 3, pp. 4804–4814, Jun. 2019.

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

[26] C. Ren, G. Zhang, X. Gu, and Y. Li, "Computing offloading in vehicular edge computing networks: Full or partial offloading?," in *Proc. IEEE 6th Inf. Technol. Mechatronics Eng. Conf.*, 2022, vol. 6, pp. 693–698.

[27] L. Zhao et al., "Vehicular computation offloading for industrial mobile edge computing," *IEEE Trans. Ind. Inform.*, vol. 17, no. 11, pp. 7871–7881, Nov. 2021.

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

[29] G. Cui, Y. Long, L. Xu, and W. Wang, "Joint offloading and resource allocation for satellite assisted vehicle-to-vehicle communication," *IEEE Syst. J.*, vol. 15, no. 3, pp. 3958–3969, Sep. 2021.

[30] H.-S. Lee, J.-Y. Kim, and J.-W. Lee, "Resource allocation in wireless networks with deep reinforcement learning: A circumstance-independent approach," *IEEE Syst. J.*, vol. 14, no. 2, pp. 2589–2592, Jun. 2020.

[31] H. Yang and X. Xie, "An actor–critic deep reinforcement learning approach for transmission scheduling in cognitive Internet of Things systems," *IEEE Syst. J.*, vol. 14, no. 1, pp. 51–60, Mar. 2020.

[32] H. Ye, G. Y. Li, and B.-H. F. Juang, "Deep reinforcement learning based resource allocation for V2V communications," *IEEE Trans. Veh. Technol.*, vol. 68, no. 4, pp. 3163–3173, Apr. 2019.

[33] Q. Luo, C. Li, T. H. Luan, and W. Shi, "Collaborative data scheduling for vehicular edge computing via deep reinforcement learning," *IEEE Internet Things J.*, vol. 7, no. 10, pp. 9637–9650, Oct. 2020.

[34] W. Zhan et al., "Deep-reinforcement-learning-based offloading scheduling for vehicular edge computing," *IEEE Internet Things J.*, vol. 7, no. 6, pp. 5449–5465, Jun. 2020.

[35] H. Ke, J. Wang, L. Deng, Y. Ge, and H. Wang, "Deep reinforcement learning-based adaptive computation offloading for MEC in heterogeneous vehicular networks," *IEEE Trans. Veh. Technol.*, vol. 69, no. 7, pp. 7916–7929, Jul. 2020.

[36] J. Tian, Q. Liu, H. Zhang, and D. Wu, "Multiagent deep-reinforcement-learning-based resource allocation for heterogeneous QoS guarantees for vehicular networks," *IEEE Internet Things J.*, vol. 9, no. 3, pp. 1683–1695, Feb. 2022.