Deep Reinforcement Learning-Based Energy-Efficient Edge Computing for Internet of Vehicles

Xiangjie Kong, *Senior Member, IEEE*, Gaohui Duan, Mingliang Hou, *Student Member, IEEE*, Guojiang Shen, Hui Wang, Xiaoran Yan, *Member, IEEE*, and Mario Collotta, *Member, IEEE*

***Abstract*—Mobile network operators (MNOs) allocate computing and caching resources for mobile users by deploying a central control system. Existing studies mainly use programming and heuristic methods to solve the resource allocation problem, which ignores the energy cost problem that is really significant to the MNO. To solve this problem, in this article, we design a joint computing and caching framework by integrating deep deterministic policy gradient (DDPG) algorithm. Especially, we focus on the Internet of Vehicles scenario, which needs the support of mobile network provided by MNO. We first formulate an optimization problem to minimize MNO's energy cost by considering the computation and caching energy costs jointly. Then, we turn the formulated problem into a reinforcement learning problem and utilize DDPG methods to solve this problem. The final simulation result shows that our solution can reduce energy costs by more than 15%, while ensuring the tasks can be completed on time.**

***Index Terms*—Computation offloading, deep reinforcement learning (DRL), energy-efficient, Internet of Vehicle (IoV), mobile edge computing (MEC).**

Manuscript received October 12, 2021; revised December 24, 2021 and January 13, 2022; accepted February 21, 2022. Date of publication March 1, 2022; date of current version June 13, 2022. This work was supported in part by the National Natural Science Foundation of China under Grant 62072409 and Grant 62073295, in part by the Zhejiang Provincial Natural Science Foundation under Grant LR21F020003, in part by the Key Research and Development Program of Zhejiang Province under Grant 2022C01050, and in part by the Fundamental Research Funds for the Provincial Universities of Zhejiang under Grant RF-B2020001. Paper no. TII-21-4463. *(Corresponding author: Xiaoran Yan.)*

Xiangjie Kong is with the College of Computer Science and Technology, Zhejiang University of Technology, Hangzhou 310023, China, with the School of Software, Dalian University of Technology, Dalian 116620, China, and also with the Research Institute of Artificial Intelligence, Zhejiang Lab, Hangzhou 311100, China (e-mail: xjkong@ieee.org).

Guojiang Shen is with the College of Computer Science and Technology, Zhejiang University of Technology, Hangzhou 310023, China (e-mail: gjshen1975@zjut.edu.cn).

Xiaoran Yan is with the Research Institute of Artificial Intelligence, Zhejiang Lab, Hangzhou 311100, China (e-mail: yanxr@zhejianglab.com).

Gaohui Duan, Mingliang Hou, and Hui Wang are with the School of Software, Dalian University of Technology, Dalian 116620, China (e-mail: ghuiduan@outlook.com; teemohold@outlook.com; wanghui.sd@outlook.com).

Mario Collotta is with the Faculty of Engineering and Architecture, Kore University of Enna, 94100 Enna, Italy (e-mail: mario.collotta@unikore.it).

Color versions of one or more figures in this article are available at https://doi.org/10.1109/TII.2022.3155162.

Digital Object Identifier 10.1109/TII.2022.3155162

I. Introduction

NOWADAYS, Internet of Things (IoT) technologies have been applied in various fields, including healthcare and transportations [1], [2]. Internet of Vehicles (IoV), a classical application of IoT in the field of transportation systems, has better development prospects in 5G and the upcoming 6G era [3], [4]. Through IoV technologies, vehicles can execute onboard service and safety applications such as speed warning and vehicle tracking, providing a pleasant experience for users and ensuring the safety of drivers and passengers. However, these applications need many computation resources, and some applications are latency-sensitive. Due to the confined computing capabilities, most vehicles may be unable to execute these tasks locally. Although these tasks can be processed in a cloud computing center with abundant computing resources, processing tasks on the cloud computing center will result in high latency due to the distance between the cloud computing center and the vehicle. Therefore, this processing scheme is not suitable for applications that are sensitive to latency.

To tackle this issue, mobile edge computing (MEC) [5]–[8] is presented as a viable solution for fulfilling the low-latency demands by giving cloud computing abilities at the mobile network's edge. The basic principle is to transfer computing tasks proposed by vehicles to MEC servers located in the edge network. Moreover, the contents of the vehicle request have similarities. Some of the requested contents are attractive for vehicles. Therefore, the popular contents could be cached on the MEC servers to save the computation and bandwidth resources. Roadside units (RSUs), equipped with MEC servers, can provide vehicles with computation and caching services. However, the resources of RSU are restricted. Reasonable allocation of computing and caching resources based on the resources of the MEC server is challenging.

There are many studies concentrated on computation offloading and edge caching. Feng *et al.* [9] proposed a distributed iterative algorithm for data caching and computation offloading. They leveraged a dynamic programming algorithm to deal with the cache placement issue. Wang *et al.* [10] adopted a noncooperative computation offloading game strategy. They designed a distributed computation offloading algorithm capable of achieving a stable equilibrium and thus reducing the delay. Hou *et al.* [11] focused on the reliability of computation offloading issues in IoV. They leveraged a heuristic algorithm to optimize the reliability of computation offloading. These works primarily apply programming methods, game theory, and heuristic algorithms to solve computation offloading and edge caching problems. However, it is of great difficulty for these approaches to ensure reliable and effective data transmission due to the heterogeneity of vehicular networks. Furthermore, it is challenging for programming techniques to achieve real-time decision-making, and heuristic algorithms have less robustness. Recently, deep reinforcement learning (DRL) has emerged as an appealing technique to solve the computation offloading and resource allocation problem due to the powerful cognitive and decision-making capabilities [12]. Against the heuristic approaches, DRL methods have more robustness, which can obtain a stable convergence result. DRL methods can achieve real-time, online decision-making compared to programming techniques.

However, performing computation offloading and edge computing in the IoV system faces two significant challenges.

1) *Efficient use of energy will be important in the construction of the 6G-supported IoV system.* First, the energy cost of prospective IoV scenarios will rise due to the increase in IoV devices, as well as the demands of extensive computation. The increased energy demands resulting from 6G's adoption of higher frequency bands will also contribute to higher energy costs. Therefore, energy consumption has become a concern for mobile network operators (MNOs). Second, it is a challenging problem in constructing sustainable vehicular infrastructure as power costs and vehicle fuel emissions place an energy burden on IoV systems. Third, many decision-making algorithms have been deployed in the intelligent application of IoV. Executing these algorithms will lead to tremendous energy consumption, which affects energy efficiency improvement.
2) *DRL methods may not be used directly in the IoV system.* Traditional DRL-based resource allocation approaches such as deep Q-network (DQN) are challenging to discover the best policy in the IoV scenario, since this scenario has continuous state and action space. Furthermore, the action space of DQN is discrete. Discretizing the continuous action space will trigger the curse of dimensionality.

To improve energy efficiency, tiny machine learning (TinyML) is considered a potential technique. Compared with MEC approaches that must deploy machine learning (ML) algorithms such as DRL away from the IoT device, TinyML concentrates on deploying ML models on low-powered IoT devices such as microcontroller units (MCUs). In recent years, there have been some studies that applied TinyML methods to the research of IoV [13], [14]. However, the vehicles' low-powered embedded devices usually have low computation capability and memory. Take MCU as an example. Although MCU is equipped with advanced processors, the processing speed of MCU is relatively slow in comparison with MEC-based systems, which may affect the quality of service. Additionally, the expansion of the research of TinyML puts increasing demands on MCU in terms of performance and reliability in some situations. It may be tough to lower the cost of MCU while maintaining the performance and reliability of MCU.

Therefore, this article proposes an energy-efficient computation offloading and content caching scheme based on MEC and caching to overcome the above two challenges. Explicitly speaking, to overcome the first challenge, we take the energy consumption in the process of computation offloading and edge caching into account. We formulate an optimization problem by considering the computation and caching energy cost jointly. Then, to overcome the second challenge, motivated by [15], we turn the formulated optimization problem into a reinforcement learning (RL) problem and adopt the deep deterministic policy gradient (DDPG) to solve this problem. The following are the major contributions of this article.

1) We design a joint computation and caching framework by considering the computation and caching resources of MEC servers. In this framework, RSU and base station (BS) cooperatively allocate resources for the vehicles. Through joint consideration of computation energy cost and caching energy cost, we formulate an optimization problem aimed at minimizing the energy cost of MNO.
2) We turn the formulated energy cost optimization problem into a RL problem, since DRL can achieve real-time online decision-making and has more robustness than programming and heuristic methods. We propose a DDPG-based algorithm to solve this problem. DDPG is more applicable to discovering the optimal resource allocation strategy in the IoV system compared with DQN.
3) We design simulation experiments to assess the performance of our algorithm. Results demonstrate that our method effectively promotes IoV systems' performance, decreasing the energy cost of MNO.

The rest of this article is organized as follows. Section II displays the related work. Section III is the detailed description of the system model. Section IV gives the problem formulation. In Section V, we use the DDPG-based algorithm to solve the energy cost optimization problem. Section VI presents the simulation results. Finally, Section VII concludes this article.

II. Related Work

This section reviews the current research progress of IoV, which are summarized as follows.

A. MEC in IoV

With the intelligentization of vehicle applications, traditional cloud computing has higher network delay since the computing center is far from users. The network latency has become a concern for the application providers [16]–[18]. MEC is taken as a promising technique to tackle this problem. MEC can decrease the network delay since the MEC servers are located closer to users. Hence, the user experience is enhanced.

Because of the low-latency advantage, MEC has been adopted in the research of IoV. Liu *et al.* [19] proposed a software-defined vehicular network framework with the assistance of MEC. This framework can enhance the scalability of network. Ji *et al.* [20] improved the architecture of VANET by bringing MEC, providing low-latency IoV transmission services. Zhao *et al.* [21]

presented a cloud-MEC cooperative method to tackle the challenge of computation offloading. They designed a collaborative scheme called CCORAO. This scheme can lower the computation time and increase the utility of IoV system.

B. Computation Offloading in IoV

6G maintains the capacity to connect large-scale devices, while ensuring service performance [22]. Moreover, 6G is envisioned to support more intelligent vehicle applications. However, as vehicles become more intelligent, more resources are required. Computation ability and storage space are also factors that should be considered in the 6G era. Currently, the available computing resources provided to vehicles are restricted. Offloading computing tasks to edge servers is critical.

Xu *et al.* [23] designed a new offloading method called ACOM. This method can minimize the transmission delay, while also increasing resource utility. Xiong *et al.* [24] proposed a smart task offloading method that can guarantee the reliability of offloading. In addition to considering the budgets of communication and computation, they also took the failure possibility of offloading into account. Huang *et al.* [25] proposed an offloading strategy named CTOSO, which can lower the energy consumption and delay.

C. DRL Methods for IoV

DRL has a powerful ability to perceive and make decisions because of the combination of RL and deep learning. DRL analyzes the long-term impact of current resource allocation on subsequent time points compared with traditional IoV resources allocation methods such as convex optimization and heuristic algorithm.

Currently, many scholars have applied DRL technologies to the research of IoV. Ye *et al.* [26] considered the challenge of delay limitation on the V2V link in two scenarios. They proposed a decentralized mechanism for resource allocation by leveraging DRL technologies. With the introduction of MEC, vehicles can offload their computing tasks to edge servers. Ning *et al.* [27] utilized the imitation learning and Lyapunov optimization technique to minimize the network delay. Peng *et al.* [28] considered that the vehicular network includes UAVs, which can assist in allocating resources. They utilized the DDPG approach to enhance the delay satisfaction ratio.

Although MEC technology can reduce the transmission delay in the vehicular network, this technology must deploy ML algorithms such as DRL away from the IoT device, and energy consumption is a major problem due to the high power consumption of MEC servers. Recently, TinyML has emerged as an energy-efficient approach in IoV. ML model can be deployed in low-power, computationally confined embedded devices using TinyML technology [14]. However, the processing speed of embedded devices in the vehicle is slower than the MEC server due to the limited compute and memory capabilities, affecting the quality of service. Furthermore, it may be hard to lower the cost of these embedded devices without sacrificing their performance or reliability in some situations.

Finally, in Table I, we give the comparison of TinyML and DRL. Unlike existing studies that mostly concentrated on delay, we focus primarily on the energy cost of the network operators, including computing energy cost and caching energy cost. We construct an intelligent IoV system, allocating computation and caching resources dynamically based on service migration. An optimization problem is formulated to minimize the energy cost of MNOs. Our work put forward a DRL-based algorithm. This algorithm can allocate resources efficiently.

TABLE I
COMPARISON OF TINYML AND DRL

Feature	TinyML [13], [14]	DRL [26]–[28]
Machine learning	yes	yes
Deployed away from IoT device	no	yes
Deployed in low-powered devices	yes	no
Deployed in high computation ability devices	no	yes

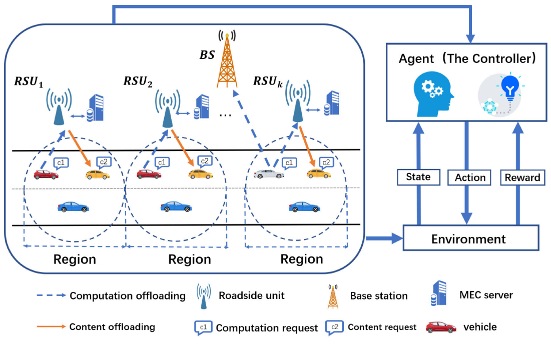

Fig. 1. Intelligent IoV system model.

III. SYSTEM MODEL

We construct an intelligent IoV system. As featured in Fig. 1, the constructed system consists of one BS, a set of vehicles denoted as $v = \{1,\ldots,V\}$ and some RSUs, which are denoted by $k = \{1,\ldots,K\}$. Each RSU is equipped with edge servers that can process the task proposed by every vehicle. RSU and BS are both capable of communicating with vehicles. In reality, not all the areas have RSU, or the RSU of the current area may not be available or vacant. When the vehicle is in these areas, the communication requests are only received by BS.

We consider requests from vehicles, including computing task requests and content download requests. We assume that BS owns adequate computing and caching resources, while RSUs are usually restricted. Hence, to make the best use of resources, BS and RSU can cooperate to process the requests from the vehicles. Vehicles can upload their tasks to the BS and RSUs through the full-duplex radio, as well as download contents from them. Table II illustrates the main notation settings of this article.

We assume that each RSU covers one particular region, the coverage region of each RSU does not overlap. The BS's communication range covers all vehicles in the IoV system. Time is divided into discrete timeslots, and the vehicular network works in every timeslot. When vehicles are moving along the road at a certain speed and driving into the communication range of RSU, MNO needs to determine how to allocate resources reasonably for vehicles at each timeslot. When a vehicle goes away from the present RSU, the existing service is transferred to another RSU.

TABLE II
NOTATION SETTING

Notation	Description
v	A set of vehicles
k	A set of RSUs
$P_{ij}(t)$	The transmission power of vehicle i
$G_{ij}(t)$	The channel gain between RSU j and vehicle i
$G_{i0}(t)$	The channel gain between BS and vehicle i
$\gamma_{ij}(t)$	The SNR between RSU j and vehicle i in timeslot t
$\gamma_{i0}(t)$	The SNR between BS and vehicle i in timeslot t
$b_{ij}(t)$	The bandwidth allocated by RSU j to the vehicle i
$b_{i0}(t)$	The bandwidth allocated by BS to the vehicle i
ω_i	The computation task size of vehicle i
q_i	The required number of CPU cycles for vehicle i to compute the task
B_j	The overall bandwidth of RSU j
M_j	The overall computation resource of RSU j
N_j	The overall cache space of RSU j
$g_i(t)$	The completed computation task size of vehicle i in timeslot t
$D_i(t)$	The caching resources allocated to vehicle i in timeslot t
$\xi_{i0}(t)$	The caching resources allocated by BS for vehicle i in timeslot t
$\xi_{ij}(t)$	The caching resources allocated by RSU j for vehicle i in timeslot t

A. Communication Model

In our constructed model, we consider that vehicles could communicate with both BS and RSUs simultaneously. BS and RSUs cooperate in allocating resources for the vehicles. In our model, the transmission mode can be grouped into vehicle-to-network (V2N) and vehicle-to-infrastructure (V2I).

1) V2N: Vehicles can communicate with BS in this transmission mode. The signal-to-noise ratio (SNR) between BS and vehicle i in timeslot t is defined as

$$\gamma_{i0}(t) = \frac{P_{i0}(t)G_{i0}(t)}{\sigma^2} \tag{1}$$

where $P_{i0}(t)$ is the transmission power of BS, and $G_{i0}(t)$ is the channel gain between BS and vehicle i. The white Gaussian noise is σ^2. Therefore, according to the Shannon theory, the transmission rate between vehicle i and BS in timeslot t can be stated as

$$R_{i0}(t) = b_{i0}(t)\log_2(1+\gamma_{i0}(t)) \tag{2}$$

where $b_{i0}(t)$ denoted the bandwidth allocated by BS to the vehicle i.

2) V2I: When a vehicle enters the coverage region of an RSU, a reliable channel is created between the vehicle and RSU for data transmission. Since the vehicle uploads computing tasks to the MEC server and downloads content from the server, MEC server needs to provide vehicles with the bandwidth required for these two requests. The SNR between RSU j and vehicle i in timeslot t is defined as

$$\gamma_{ij}(t) = \frac{P_{ij}(t)G_{ij}(t)}{\sigma^2} \tag{3}$$

where $P_{ij}(t)$ is the transmission power of vehicle i, and $G_{ij}(t)$ is the channel gain between RSU j and vehicle i. Thus, the transmission rate between RSU j and vehicle i in timeslot t can be denoted as

$$R_{ij}(t) = b_{ij}(t)\log_2(1+\gamma_{ij}(t)) \tag{4}$$

where $b_{ij}(t)$ is the bandwidth allocated by RSU j to the vehicle i.

B. Computation Model

The computational task requests of every vehicle consist of two parts: data size of the computing task and required CPU cycles. When vehicles make a computing request to the MNO, MNO needs to formulate a scheme to allocate computing resources for vehicles according to the available resources of servers. We aim to ensure that computing tasks proposed by vehicles can be processed within a specific time range. BS and RSU closest to the vehicle cooperate in allocating computing resources for vehicles, since the resource of RSU is limited. Let $u_{i0}(t)$ and $u_{ij}(t)$ be the CPU resources allocated to the vehicle i by the BS and RSU j in timeslot t. We denote q_i as the required number of CPU cycles for the vehicle i to compute the task. The computing task size of vehicle i is ω_i. Therefore, in timeslot t, the size of the computation task performed by vehicle i in RSU j and BS can be calculated by

$$\eta_{ij}(t) = \frac{u_{ij}(t)\omega_i}{q_i}$$
$$\eta_{i0}(t) = \frac{u_{i0}(t)\omega_i}{q_i}. \tag{5}$$

We denote the completed computation task size of vehicle i in timeslot t as $g_i(t)$, where $g_i(t) = \eta_{ij}(t) + \eta_{i0}(t)$.

C. Caching Model

The download contents requested by vehicles can be cached on the BS and RSU, decreasing the delay. According to the caching space, the network operator develops a caching scheme to determine the cache content size on these two devices. We denote the request content size of vehicle i as m_i, and the caching resources allocated to vehicle i in timeslot t are denoted as $D_i(t)$. We denote the caching resources allocated by BS and RSU j for vehicle i in timeslot t as $\xi_{i0}(t)$ and $\xi_{ij}(t)$, where $\xi_{i0}(t) + \xi_{ij}t = D_i(t)$.

RSU and BS allocate bandwidth to vehicles for downloading content when vehicles make a download request. Nevertheless, RSU is unable to cache the entire download content required by vehicles since the caching ability of one MEC server is restricted.

IV. Problem Formulation

A. Optimization Objective

Our optimization goal is to minimize the energy cost during the computation offloading and caching process. The entire energy cost includes two components: computation energy cost and caching energy cost. We assume that BS and RSU have the same unit computational energy consumption and cache energy consumption and denote e_0 as the unit computation energy consumption. Thus, the overall computation energy cost in timeslot t can be calculated as follows:

$$D_{\text{cp}}(t) = \sum_{i=1}^{V}\sum_{j=1}^{K}(\eta_{ij}(t)e_0C_j + \eta_{i0}(t)e_0C_0) \tag{6}$$

where C_j is the unit energy cost of RSU, while C_0 is the unit energy cost of BS.

Caching energy cost is generated when BS and RSU cache the content needed for the vehicles. We denote a_0 as the unit caching energy consumption. Caching energy consumption $E_{i0}(t)$ generated by BS and cache energy consumption $E_{ij}(t)$ generated by RSU j, which can be stated as

$$\begin{aligned} E_{i0}(t) &= \xi_{i0}(t)a_0 \\ E_{ij}(t) &= \xi_{ij}(t)a_0. \end{aligned} \tag{7}$$

The overall caching energy cost in timeslot t can be calculated by

$$D_{\text{ca}}(t) = \sum_{i=1}^{V}\sum_{j=1}^{K}(E_{i0}(t)C_0 + E_{ij}(t)C_j). \tag{8}$$

Based on (6) and (8), the energy cost in timeslot t can be stated as

$$f_c(t) = D_{\text{cp}}(t) + D_{\text{ca}}(t). \tag{9}$$

Therefore, the entire energy cost can be represented by

$$S_{\text{En}} = \sum_{t=1}^{T} f_c(t) \tag{10}$$

where T is the mean sojourn duration of vehicles in the IoV system.

B. Constraint Analysis

The allocation scheme must be appropriate when allocating resources for vehicles, whether computational resources or cache resources. That is, certain constraints must be met. The main constraints of the system model are summarized as follows:

$$\sum_{t=1}^{T}\sum_{i=1}^{V} b_{ij}(t) \le B_j \quad \forall j \in k \tag{11}$$

$$\sum_{t=1}^{T}\sum_{i=1}^{V} u_{ij}(t) \le M_j \quad \forall j \in k \tag{12}$$

$$\sum_{t=1}^{T}\sum_{i=1}^{V} \xi_{ij}(t) \le N_j \quad \forall j \in k \tag{13}$$

$$\sum_{t=1}^{T_i} g_i(t) \le \omega_i \quad \forall i \in v \tag{14}$$

$$\sum_{t=1}^{T_i} D_i(t) \le m_i \quad \forall i \in v \tag{15}$$

$$b_{ij}(t), u_{ij}(t), \xi_{ij}(t) \ge 0 \quad \forall i \in v \forall j \in k \tag{16}$$

where constraints (11)–(13) ensure the total bandwidth, computation, and caching resources allocated to vehicles should be within the ability of RSU. Constraint (14) ensures the aggregate of allocated computation resources cannot surpass the requirements of vehicles. Meanwhile, constraint (15) indicates that the aggregate of provided caching resources should be within the range of vehicles' requirements. Constraint (16) guarantees the allocated bandwidth, computation, and caching resources for vehicles ought to be great than or equal to zero.

However, all the above constraints that our constructed system should satisfy may be unrealistic. We assume that all the RSU can provide computation and caching services for vehicles when the vehicle is in the service area of the IoV system and propose the computation and content downloading requests. In reality, not all the RSU is available or vacant. When the vehicle proposes a request, the IoV system should determine whether the RSU in the current area is available. Therefore, the above constraints may be unrealistic. However, if we take this situation into account, the final optimization problem will be tough to deal with. For example, When a vehicle enters the coverage area of an RSU, some computing resources of the RSU may have been occupied. Therefore, there is no unified standard for the computing resources available to the RSU in constraint (12), making this constraint becomes very complex and hard to formulate. Therefore, we simplify the constraints of the system model, which only considers the situation that all the RSU is available and resources of all the RSU are not occupied. The final optimization problem is formulated as

$$\begin{aligned} &\min \quad S_{\text{En}}. \\ &\text{s.t.} \quad \text{Constraints} \quad (11)-(16). \end{aligned} \tag{17}$$

However, the situation mentioned above cannot be ignored. Therefore, in Section VI, we conducted corresponding experiments.

V. DRL-Based Method for Computation Offloading and Edge Caching

This section, first, presents an overview of RL techniques. Then, we present a DRL-based approach to address the computation offloading and resource allocation problem.

A. RL: An Overview

RL is a field of ML that is essential. The essence of RL technology is to maximize the discount between an immediate reward and a future reward. RL adopts a continuous interactive trial and error mechanism to learn the optimal and practical strategies through continuous interaction with the environment. RL aims to discover the optimal strategy given a Markov decision process (MDP).

MDP refers to the sequential decision process with the concept of time and dependence between states, which involves five essential elements: environment state, action, reward, state transition probability matrix, and discount factor. Only the current state has an impact on the next state in MDP. In general, the policy π is a mapping from states to a probability distribution. Given a state S_t, the agent performs an action through a specific strategy π according to the state transition probability matrix. Then, the environment moves into a new state S_{t+1} by executing this action, providing the agent with a reward. Every policy execution accumulates a reward from the environment and brings out the return reward. RL intends to discover the optimum policy π^*. Adopting this policy will ensure that all states receive the maximum intended return.

However, RL needs much time to achieve the best policy since it has to explore and gain knowledge of an entire system.

Algorithm 1: DDPG-Based Computation and Caching Resource Allocation Algorithm.

```
Input: Initial status of vehicles, BS and RSUs, action
   space $A$;
Output: Maximum reward $R_i$;
1:  Initialize the weights of actor network $\mathcal{L}(s|w^{\mathcal{L}})$ and
    critic network $Q(s,a|w^Q)$ with $w^Q$ and $w^{\mathcal{L}}$.
2:  Initialize the weights of actor-target network $\mathcal{L}'$ and
    critic-target network $Q'$: $w^{Q'} \leftarrow w^Q, w^{\mathcal{L}'} \leftarrow w^{\mathcal{L}}$.
3:  Initialize experience replay buffer $F$ to be empty.
4:  for episode=1,..., E do
5:      Initialize a process for the exploration of action.
6:      Initialize observation state $s_{ini}$ and $R_i = 0$.
7:      for t=1,..., T do
8:          Choose action $a_t = \mathcal{L}(s_t|w^{\mathcal{L}}) + N_t$ to decide
            the computation and caching resources allocated
            to vehicles in accordance with the present policy
            and noise.
9:          Carry out action $a_t$, calculate instant reward $r_t$
            and gain next state $s_{t+1}$.
10:         $R_i = R_i + r_t$.
11:         Place transition$(s_t, a_t, r_t, s_{t+1})$ in $F$.
12:         Gain a batch of M transitions $(s_t, a_t, r_t, s_{t+1})$
            from $F$.
13:         Obtain the target Q-value $y_t$ according to (22).
14:         Update the parameters of critic network by
            minimizing the loss based on (23).
15:         Update the resource allocation policy based
            on (24).
16:         Update the target networks $\mathcal{L}'$ $Q'$ based on (25).
17:     end for
18: end for
```

Therefore, RL is not appropriate and applicable to large-scale networks. DRL conquers these disadvantages with the introduction of deep learning. DRL has both the perception ability of deep learning and the decision-making capability of RL.

B. Problem Transformation

In the IoV system, the vehicle is constantly moving. Therefore, the state of wireless channels, edge servers, and vehicles constantly changes over time. MNO needs to make different decisions at different times to reasonably allocate resources to each vehicle. Therefore, the resource allocation problem can be regarded as a sequential decision problem and modeled as a MDP. Although the traditional dynamic programming methods can be used to solve the sequential decision problem, the calculation cost of the dynamic programming algorithm is too high since the dynamic algorithm needs to list all possible resource allocation strategies and make decisions to retain those that may reach the optimal resource allocation strategy. Therefore, it is not easy to realize real-time decision-making.

Compared with dynamic programming method, RL algorithm can make decisions according to the current state of the IoV system and realize real-time decision making. After the agent defines the reward function, the RL algorithm optimizes the resource allocation strategy according to the reward value through continuous interaction with the IoV environment to find the relatively optimal resource allocation strategy. Therefore, we turn the formulated optimization problem in (17) into a DRL problem.

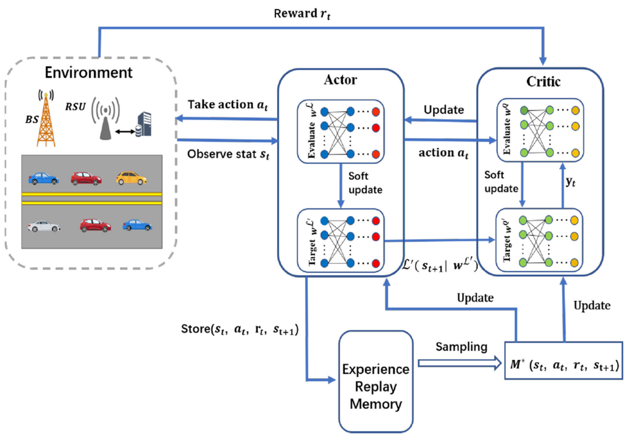

Fig. 2. DDPG-based framework for the vehicular network.

The traditional DQN method is mainly oriented to discrete control. On the IoV scenario, because the bandwidth, computing, and cache resources allocated to the vehicle by the system cannot exceed the bandwidth, computing power, and cache space of the RSU, and the computing and cache resources allocated to the vehicle cannot exceed the request of the vehicle. Therefore, the computation and caching resources allocated to the vehicle by the BS and RSU are continuous values. Based on the above analysis, traditional DQN method is not suitable for this scenario. Compared with DQN, the output action of DDPG method is continuous. Therefore, we design a DDPG-based method to find the optimal resource allocation strategy. The DDPG-based framework for the vehicular network is presented in Fig. 2. In the following, three crucial elements are illustrated.

1) *Environment state*: The environment state consists of the status of vehicles, BS and RSUs. At each timeslot t, The state s_t is represented as

$$s_t = \{U_i(t), N_i(t), M_i(t)\} \tag{18}$$

where $U_i(t)$, $N_i(t)$, and $M_i(t)$ denote the status of vehicles, BS, and MEC server, respectively. Vehicular status is composed of the vehicle's location, speed, the total size of computing tasks and required contents, the popularity of requested contents and required CPU cycles for the task computing, the residual computation task size and remaining contents. Besides, BS status and RSUs status include their computing ability, caching space, and useable bandwidth.

2) *Actions*: The constructed system needs to determine how to allocate computation and caching resources appropriately for vehicles. The system schedules resources on different RSUs and BS to provide computation and caching services for the vehicles. Hence, the action in timeslot *t* can be expressed as

$$a_t = \{u_{ij}(t), \xi_{ij}(t), b_{ij}(t), u_{i0}(t), \xi_{i0}(t), b_{i0}(t)\}. \tag{19}$$

3) *Rewards*: Our aim is to keep the whole energy costs as low as possible. However, the objective of the agent is to maximize the reward. Thereby, the reward function is set as follows:

$$R_i = -S_{En}. \tag{20}$$

C. DDPG-Based Computation and Caching Resource Allocation Algorithm

We propose a DDPG-based computation and caching resource allocation (DCRA) algorithm. The following illustrates the steps of the proposed DCRA algorithm.

At first, initialize the parameters of DDPG. Notably, the actor-target network and critic-target network parameter settings are consistent with the actor and critic network.

For every episode, at present state s_t, the actor network selects an action a_t following the existing action policy and the noise of exploration. On the basis of this action, vehicles conduct computation offloading and download content. Meanwhile, RSU and BS allocate corresponding computing, caching resources, and communication bandwidth for vehicles. Then, next state s_{t+1} can be observed and gain an instant reward r_t.

Next, a transition (s_t, a_t, r_t, s_{t+1}) is placed in the replay buffer as the training sets to train the actor network. The critic network gives the Q value for the online policy through the Bellman equation. The formula for calculating Q value is given as follows:

$$Q^{\mathcal{L}}(s_t, a_t) = E\left[R(s_t, a_t) + \gamma Q^{\mathcal{L}}(s_{t+1}|w^Q)\right]. \tag{21}$$

Then, the target Q-value is calculated by the critic-target network. The critic-target network is leveraged to train the critic network. The target Q-value can be obtained by the following formula:

$$y_t = r_t + \gamma Q'(s_{t+1}, \mathcal{L}'(s_{t+1|w^{\mathcal{L}'}})|w^{Q'}) \tag{22}$$

where γ is the discount factor. After that, the critic network is updated by minimizing the loss. The loss function can be stated as

$$L = \frac{1}{M}\sum_{t}^{M}\left(y_t - Q\left(s_t, a_t|w^Q\right)\right)^2. \tag{23}$$

Next, update the resource allocation policy through policy gradient

$$\nabla w^{\mathcal{L}} J(\mathcal{L}) = \frac{1}{M}\sum_{t=1}^{M}\left(\nabla a Q\left(s, a, |w^Q\right)|_{s=s_t, a=\mathcal{L}(s_t)} \cdot \nabla w^{\mathcal{L}}\mathcal{L}\left(s|w^{\mathcal{L}}\right)|_{s=s_t}\right). \tag{24}$$

Finally, a soft updating mechanism is leveraged to update the target networks. In this way, the target values do not vary so fast, resulting in a more steady learning process. The equation for soft updating is as follows:

$$w^{Q'} \leftarrow \beta w^Q + (1-\beta) w^{Q'}$$
$$w^{\mathcal{L}'} \leftarrow \beta w^{\mathcal{L}} + (1-\beta) w^{\mathcal{L}'} \tag{25}$$

where β is the coefficient for soft updating. The pseudocode of DCRA is presented in Algorithm 1.

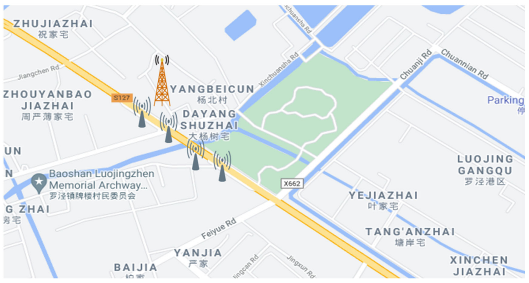

Fig. 3. Illustration of area selection on the S127 Wenchuan Highway in Shanghai.

TABLE III
PARAMETER SETTING

Parameter	Value
The overall bandwidth of BS/RSU	20/10MHz
The overall computing capacity of BS/RSU	10/5GHz
The overall caching space of BS/RSU	10/5GB
The unit computation energy consumption	$9*10^{-5}$W.h/Mb
The unit caching energy consumption	$2*10^{-5}$W.h/Mb
The unit energy cost of BS/RSU	1/0.3units/J

VI. Performance Evaluation

A. Simulation Setup

As stated in Fig. 3, we consider that the IoV system is deployed in a real-world highway road (S127 Wenchuan Highway in Shanghai, China). One BS and four RSUs are deployed in our established system to provide IoV service. We assume that BS and RSU have the same unit computational energy consumption and cache energy consumption. Each RSU covers a 250 × 250 m^2 area, whereas BS can cover the entire region served by the IoV system. We consider the traffic flow on the road is about 20 in every 2 min, and the average velocity is 30 Km/h [29]. According to [27] and [29], the sizes of computing tasks and requested contents are distributed at random in [0.15,0.25,0.3,0.4,0.45,0.6] GB, the CPU cycles required for computing tasks are chosen at random from [0.5,0.6,0.7,0.8,0.9,1.2] Gcycles. The popularity of the required content is distributed in a uniform manner in [1, 6]. The other parameters are illustrated in Table III. PyTorch 1.8.1 is adopted with Python 3.7 on Windows 10 to accomplish the proposed algorithm DCRA.

To verify the efficiency of DCRA, following three schemes are utilized for comparison.

1) Noncooperative scheme (NCS): This scheme does not consider collaborative computing and caching, i.e., only BS provides computing and caching services for vehicles.
2) Computing offloading scheme (COS): In this scheme, only BS is used to cache the content needed for the vehicle.
3) Edge caching scheme (ECS): This scheme does not consider offloading, i.e., only BS provides computing service for vehicles.

B. Results

The convergence and average reward performance comparison between our proposed scheme and the other three schemes

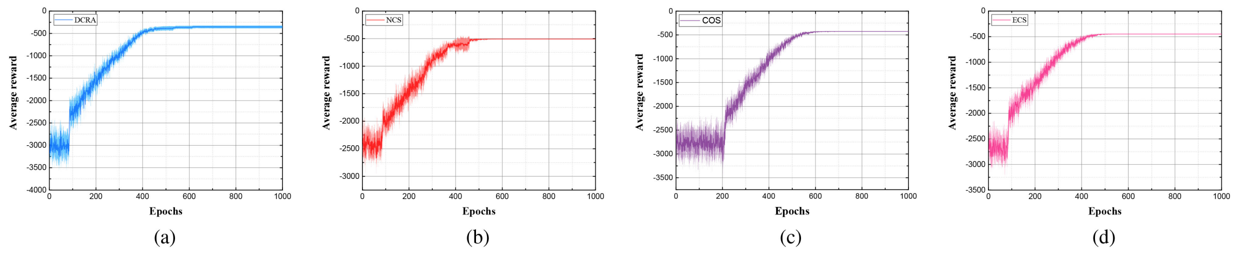

Fig. 4. Convergence performance and the average reward of different schemes. (a) Average reward of DCRA. (b) Average reward of NCA. (c) Average reward of COS. (d) Average reward of ECS.

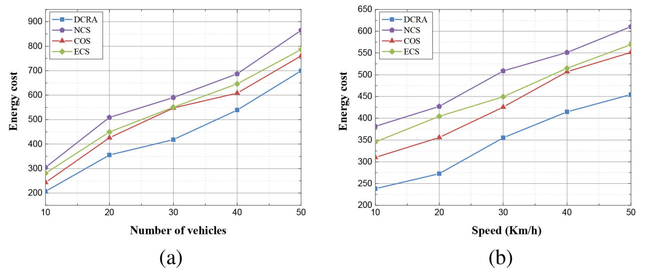

Fig. 5. Energy cost with different numbers of vehicles and different speeds.

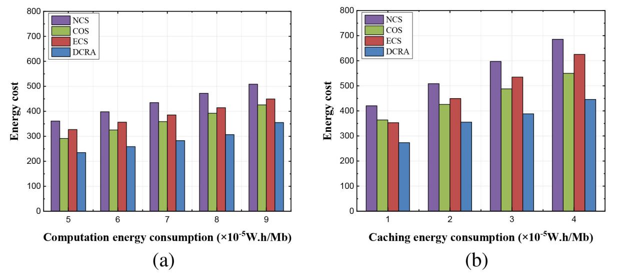

Fig. 6. Energy cost with different computation and cache energy consumption.

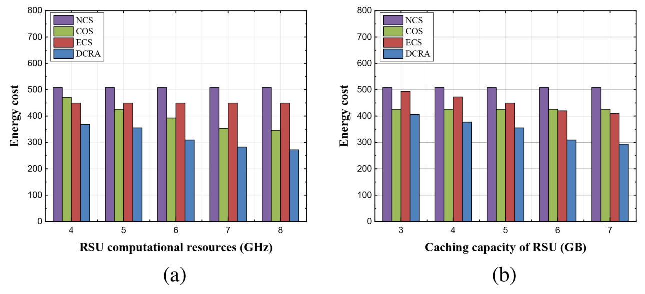

Fig. 7. Energy cost with different computation resources and caching capacities of RSU.

is provided in Fig. 4. As previous work done [30], the average reward is obtained after 10 independent runs of 1000 epochs. The shadow areas in Fig. 4 represent a 95% confidence interval. As can be observed from Fig. 4 that these four schemes do not have many deviations on the part of convergence. However, our proposed scheme can get a better reward which means that our scheme surpasses other schemes in reducing energy costs. Moreover, we can observe that all the methods' energy costs are high at the starting of the training stage, decreasing rapidly as the number of training steps increases gradually. The reward tends to be sustained when the number of training steps reaches around 600. The reward convergence illustrates that the agent in our construct system can gain a more efficient resource allocation strategy.

As described in Fig. 5(a), we expand the number of vehicles from $V = 10$ to $V = 50$. The number of computing tasks and cache content that the server needs to perform also increases due to the increasing number of vehicles. In consequence, the energy cost is increasing.

We also investigated the impact of the vehicle speed on the results. From Fig. 5(b), we can observe that the whole energy costs increase when vehicles are going faster. As the vehicles go faster, vehicles stay in the IoV system for a short time. BS needs to provide more resources for vehicles to guarantee that the task requests from the vehicle can be completed promptly since the resource of MEC server is limited.

There are differences in the unit computing and cache energy consumption of BS and RSU for different MNOs. As can be noticed from Fig. 6(a) and (b), the energy cost of all the method increase with the growth of unit computation energy consumption and caching energy consumption. However, our method can get lower energy costs compared with other schemes.

Performance comparisons with various caching capacities and computation resources of RSU are displayed in Fig. 7(a) and (b). From Fig. 7(a), we can observe that the energy cost of NCS and ECS are stable since these schemes do not offload computation tasks to the RSU. The energy cost of DCRA and COS decreases when the computation resources of RSU are increased. RSU has the capacity to deal with more computation tasks when the computation resources are increasing. The amount of tasks undertaken by BS has also decreased, thereby decreasing the whole energy cost. Correspondingly, RSU can cache more contents as the caching ability of RSU grows. Therefore, the energy cost of DCRA and ECS decreases when increasing the caching capacity of RSU in Fig. 7(b).

In practice, the above situation only considers that all the RSUs can provide computation and caching services for vehicles. However, in reality, the situation that the RSU in the current area is not available or no RSU is deployed in the current area cannot be ignored. Therefore, we explored the effects of different numbers of RSUs on the results. From Fig. 8, we can observe that when no RSU is deployed in the area covered by the IoV system, that is, only BS provides services for the vehicles, the energy cost of MNO tends to be high. When the number of RSUs is 1, most of the tasks are processed in the BS. Therefore, the reduction in energy cost is not obvious. As the number of RSUs increases,

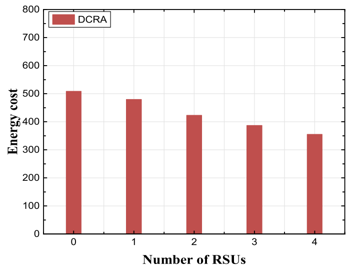

Fig. 8. Energy cost with different numbers of RSUs.

RSUs can provide more computation and caching resources for vehicles, thus decreasing the energy cost.

VII. Conclusion

In this article, we designed a joint computing and caching framework. By considering the computation and caching energy cost, we formulated an optimization problem to minimize the energy costs of MNO. To tackle this problem, we designed a resource allocation algorithm based on DDPG. Finally, we executed simulation experiments to assess the performance of our approach. Numerical results proved that our solution can effectively decrease energy costs compared with other schemes. The results also indicated that RSU and BS coordinate resource allocation is an energy-efficient scheme, and DDPG method is suitable for sequential decision-making problems.

However, the assumptions used in our constructed IoV system are ideal situations, and there is a certain gap with reality. Furthermore, we do not consider the interference between channels when multiple vehicles communicate to the same server. In future work, we will consider a more realistic scenario to build appropriate IoV models.

References

[1] X. Ai, H. Chen, K. Lin, Z. Wang, and J. Yu, “Nowhere to hide: Efficiently identifying probabilistic cloning attacks in large-scale RFID systems,” *IEEE Trans. Inf. Forensics Secur.*, vol. 16, pp. 714–727, Sep. 2021.

[2] Y. Huang *et al.*, “OPAT: Optimized allocation of time dependent tasks for mobile crowdsensing,” *IEEE Trans. Ind. Informat.*, vol. 18, no. 4, pp. 2476–2485 Apr. 2022.

[3] X. Kong *et al.*, “A federated learning-based license plate recognition scheme for 5G-enabled Internet of Vehicles,” *IEEE Trans. Ind. Informat.*, vol. 17, no. 12, pp. 8523–8530, Dec. 2021.

[4] X. Zhou, W. Liang, J. She, Z. Yan, and K. I.-K. Wang, “Two-layer federated learning with heterogeneous model aggregation for 6G supported Internet of Vehicles,” *IEEE Trans. Veh. Technol.*, vol. 70, no. 6, pp. 5308–5317, Jun. 2021.

[5] X. Huang, R. Yu, J. Kang, Y. He, and Y. Zhang, “Exploring mobile edge computing for 5G-Enabled software defined vehicular networks,” *IEEE Wireless Commun.*, vol. 24, no. 6, pp. 55–63, Dec. 2017.

[6] X. Kong *et al.*, “Real-time mask identification for COVID-19: An edge computing-based deep learning framework,” *IEEE Internet Things J.*, vol. 8, no. 21, pp. 15929–15938, Nov. 2021.

[7] Y. C. Hu, M. Patel, D. Sabella, N. Sprecher, and V. Young, “Mobile edge computing–A key technology towards 5G,” *ETSI White Paper*, vol. 11, no. 11, pp. 1–16, 2015.

[8] X. Zhou, X. Yang, J. Ma, and K. I.-K. Wang, “Energy efficient smart routing based on link correlation mining for wireless edge computing in IoT,” *IEEE Internet Things J.*, to be published, doi: 10.1109/JIOT.2021.3077937.

[9] H. Feng, S. Guo, L. Yang, and Y. Yang, “Collaborative data caching and computation offloading for multi-service mobile edge computing,” *IEEE Trans. Veh. Technol.*, vol. 70, no. 9, pp. 9408–9422, Sep. 2021.

[10] Y. Wang *et al.*, “A game-based computation offloading method in vehicular multiaccess edge computing networks,” *IEEE Internet Things J.*, vol. 7, no. 6, pp. 4987–4996, Jun. 2020.

[11] X. Hou *et al.*, “Reliable computation offloading for edge-computing-enabled software-defined IoV,” *IEEE Internet Things J.*, vol. 7, no. 8, pp. 7097–7111, Aug. 2020.

[12] N. C. Luong *et al.*, “Applications of deep reinforcement learning in communications and networking: A survey,” *IEEE Commun. Surv. Tut.*, vol. 21, no. 4, pp. 3133–3174, Oct.–Dec. 2019.

[13] G. Signoretti, M. Silva, P. Andrade, I. Silva, E. Sisinni, and P. Ferrari, “An evolving TinyML compression algorithm for IoT environments based on data eccentricity,” *Sensors*, vol. 21, no. 12, 2021, Art. no. 4153.

[14] P. Andrade *et al.*, “An unsupervised TinyML approach applied for pavement anomalies detection under the Internet of Intelligent Vehicles,” in *Proc. IEEE Int. Workshop Metrology Industry 4.0 IoT*, 2021, pp. 642–647.

[15] K. Yang, C. Shen, and T. Liu, “Deep reinforcement learning based wireless network optimization: A comparative study,” in *Proc. IEEE IEEE Conf. Comput. Commun. Workshops*, 2020, pp. 1248–1253.

[16] F. Xia, A. M. Ahmed, L. T. Yang, J. Ma, and J. J. Rodrigues, “Exploiting social relationship to enable efficient replica allocation in ad-hoc social networks,” *IEEE Trans. Parallel Distrib. Syst.*, vol. 25, no. 12, pp. 3167–3176, Dec. 2014.

[17] N. Abbas, Y. Zhang, A. Taherkordi, and T. Skeie, “Mobile edge computing: A survey,” *IEEE Internet Things J.*, vol. 5, no. 1, pp. 450–465, Feb. 2018.

[18] F. Xia, A. M. Ahmed, L. T. Yang, and Z. Luo, “Community-based event dissemination with optimal load balancing,” *IEEE Trans. Comput.*, vol. 64, no. 7, pp. 1857–1869, Jul. 2015.

[19] J. Liu, J. Wan, B. Zeng, Q. Wang, H. Song, and M. Qiu, “A scalable and quick-response software defined vehicular network assisted by mobile edge computing,” *IEEE Commun. Mag.*, vol. 55, no. 7, pp. 94–100, Jul. 2017.

[20] X. Ji, W. Xu, C. Zhang, and B. Liu, “A three-level routing hierarchy in improved SDN-MEC-VANET architecture,” in *Proc. IEEE Wireless Commun. Netw. Conf.*, 2020, pp. 1–7.

[21] 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.

[22] F. Guo, F. R. Yu, H. Zhang, X. Li, H. Ji, and V. C. M. Leung, “Enabling massive IoT toward 6G: A. comprehensive survey,” *IEEE Internet Things J.*, vol. 8, no. 15, pp. 11 891–11 915, Aug. 2021.

[23] X. Xu, X. Zhang, X. Liu, J. Jiang, L. Qi, and M. Z. A. Bhuiyan, “Adaptive computation offloading with edge for 5G-Envisioned Internet of Connected Vehicles,” *IEEE Trans. Intell. Transp. Syst.*, vol. 22, no. 8, pp. 5213–5222, Aug. 2021.

[24] K. Xiong, S. Leng, C. Huang, C. Yuen, and Y. L. Guan, “Intelligent task offloading for heterogeneous V2X communications,” *IEEE Trans. Intell. Transp. Syst.*, vol. 22, no. 4, pp. 2226–2238, Apr. 2021.

[25] M. Huang, W. Liu, T. Wang, A. Liu, and S. Zhang, “A Cloud–MEC collaborative task offloading scheme with service orchestration,” *IEEE Internet Things J.*, vol. 7, no. 7, pp. 5792–5805, Jul. 2020.

[26] 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.

[27] Z. Ning *et al.*, “Intelligent edge computing in Internet of Vehicles: A joint computation offloading and caching solution,” *IEEE Trans. Intell. Transp. Syst.*, vol. 22, no. 4, pp. 2212–2225, Apr. 2021.

[28] H. Peng and X. S. Shen, “DDPG-based resource management for MEC/UAV-Assisted vehicular networks,” in *Proc. IEEE 92nd Veh. Technol. Conf.*, 2020, pp. 1–6.

[29] Z. Ning *et al.*, “Joint computing and caching in 5G-Envisioned Internet of Vehicles: A. deep reinforcement learning-based traffic control system,” *IEEE Trans. Intell. Transp. Syst.*, vol. 22, no. 8, pp. 5201–5212, Aug. 2021.

[30] F. L. Da Silva, P. Hernandez-Leal, B. Kartal, and M. E. Taylor, “Uncertainty-aware action advising for deep reinforcement learning agents,” in *Proc. Assoc. Adv. Artif. Intell. Conf. Artif. Intell.*, 2020, pp. 5792–5799.