A Digital Twin-Assisted Intelligent Partial Offloading Approach for Vehicular Edge Computing

Liang Zhao, *Member, IEEE*, Zijia Zhao, Enchao Zhang, Ammar Hawbani, Ahmed Y. Al-Dubai, *Senior Member, IEEE*, Zhiyuan Tan, *Senior Member, IEEE*, and Amir Hussain

***Abstract*— Vehicle Edge Computing (VEC) is a promising paradigm that exposes Mobile Edge Computing (MEC) to road scenarios. In VEC, task offloading can enable vehicles to offload the computing tasks to nearby Roadside Units (RSUs) that deploy computing capabilities. However, the highly dynamic network topology, strict low-delay constraints, and massive data of tasks of VEC pose significant challenges for implementing efficient offloading. Digital Twin-based VEC is emerging as a promising solution that enables real-time monitoring of the state of the VEC network through mapping and interaction between the physical and virtual worlds, thus assisting in making sound offload decisions in the physical world. Thus, this paper proposes an intelligent partial offloading scheme, namely, Digital Twin-Assisted Intelligent Partial Offloading (IGNITE). First, to find the optimal offloading space in advance, we combine the improved clustering algorithm with the Digital Twin (DT) technique, in which unreasonable decisions can be avoided by reducing the size of the decision space. Second, to reduce the overall cost of the system, Deep Reinforcement Learning (DRL) algorithm is employed to train the offloading strategy, allowing for automatic optimization of computational delay and vehicle service price. To improve the efficiency of cooperation between digital and physical spaces, a feedback mechanism is established. It can adjust the parameters of the clustering algorithm based on the final offloading results in this clustering. To the best of our knowledge, this is the first study on DT-assisted vehicle offloading that proposes a feedback mechanism, forming a complete closed loop as prediction-offloading-feedback. Extensive experiments demonstrate that IGNITE has significant advantages in terms of total system computational cost, total computational delay, and offloading success rate compared with its counterparts.**

***Index Terms*— Mobile edge computing, vehicle edge computing, digital twin network, deep reinforcement learning, task offloading.**

I. Introduction

NOWADAYS, the proliferation of vehicular applications has emerged with the continued development of connected and autonomous vehicles. These applications require massive data processing with complexity and diversity and pose serious challenges for cloud-based networks. Specifically, the remote distance between vehicles and the cloud server puts a critical burden on delay-sensitive tasks [1]. Accordingly, a promising paradigm, namely, Vehicular Edge Computing (VEC) is proposed to meet the challenges of remote distance. In VEC, task offloading enables vehicles to offload the computing tasks to nearby Roadside Units (RSUs) with computing capabilities [2]. In recent years, task offloading became a hot issue that attracts attention from academia. However, the traditional vehicle-to-roadside units (V2R) offloading approach may bring additional delay and energy consumption. For instance, a large number of tasks occupy the limited RSUs computational resources, resulting in an excessive computational burden on the RSUs and thus requiring additional waiting time [3]. At the same time, vehicles can be considered a distributed computing system as they are equipped with increasing computing resources [4], [5]. Therefore, vehicles with limited resources can offload their tasks to vehicles with available resources for execution through onboard communication [6].

The vehicle-to-vehicle (V2V) offloading is a promising computing approach that enables vehicles to offload their tasks to other vehicles on the road. However, it still faces a number of challenges [7], [8]. First, due to the dynamics of VEC, the available resources for vehicles are constantly changing, and the link connections between vehicles are unstable. However, most existing vehicular applications require real-time response, implying that the task is time-sensitive. Therefore, the performance of task offloading may be affected by the above-mentioned issues, resulting in larger delay. Second, it is unrealistic to assume that vehicles on the road are always willing to assist others for task offloading [9]. This means that in V2V offloading, the issue of vehicle cooperation is crucial. Third, the high density of vehicles on the road makes V2V offloading decisions difficult. According to our extensive

Manuscript received 14 October 2022; revised 21 May 2023; accepted 4 August 2023. Date of publication 30 August 2023; date of current version 26 October 2023. This work was supported in part by the Liaoning Province Applied Basic Research Program under Grant 2023JH2/101300194, in part by the U.K. Engineering and Physical Sciences Research Council (EPSRC) funded by the COG-MHEAR Program under Grant EP/T021063/1, in part by the Liaoning Revitalization Talents Program, and in part by the National Natural Science Foundation of China under Grant 62372310. *(Corresponding author: Liang Zhao.)*

Liang Zhao, Zijia Zhao, and Enchao Zhang are with the School of Computer Science, Shenyang Aerospace University, Shenyang 110136, China (e-mail: lzhao@sau.edu.cn; zijiazhaostu@163.com; enchaozhang1998@163.com).

Ammar Hawbani is with the School of Computer Science and Technology, University of Science and Technology of China, Hefei 230026, China (e-mail: anmande@ustc.edu.cn).

Ahmed Y. Al-Dubai, Zhiyuan Tan, and Amir Hussain are with the School of Computing, Edinburgh Napier University, EH11 4BN Edinburgh, U.K. (e-mail: a.al-dubai@napier.ac.uk; z.tan@napier.ac.uk; a.hussain@napier.ac.uk).

Color versions of one or more figures in this article are available at https://doi.org/10.1109/JSAC.2023.3310062.

Digital Object Identifier 10.1109/JSAC.2023.3310062

research, most existing studies of V2V only investigate how to transform the V2V offloading problem into a sequential decision problem to reduce delay, but they did not fundamentally consider how to solve the above-proposed problem. Thus, the offloading effectiveness is still limited by the dynamic vehicle network [10]. To address the above problems, it is critical to construct a specified preliminary space as a pre-processing process before task offloading. Digital Twin (DT) technology can create multi-scale digital mapping of physical objects, and its deployment in VEC can effectively solve the above problems.

DT has emerged as an extremely promising and cutting-edge technology that enables a virtual model to accurately reflect a physical object, facilitating the mapping, iterative operation, and interaction between the virtual model and physical object [11], [12]. This mapping provides comprehensive information about the VEC network, which allows feature mining and prediction of physical vehicles and polymorphic environments [13], [14]. Given the above advantages of DT, some studies combining DT and VEC have recently emerged. Among them, Zhang et al. [15] combine DT with vehicle edge computing to reveal the relationship between different vehicles to reduce the offloading cost. Dai et al. [16] build a DT-based adaptive VEC network with two AI-powered closed loops to perform autonomous offloading, thus minimizing delay.

In fact, the above DT solutions for VEC are still in the early stage and there is still much space for improvement. First, there is still a lack of research on using DT for prediction in VEC to optimize offloading decisions. Second, in current schemes that use DT-assisted vehicle task offloading, the offloading methods adopted are primarily binary offloading and just have a single optimization goal, which is typically to reduce delay. Considering Deep Reinforcement Learning (DRL) has become an effective and feasible research method to address task offloading and resource allocation in dynamic environments [17]. Therefore, a partial offloading approach, namely, Digital Twin-Assisted Intelligent Partial Offloading (IGNITE) is proposed to provide more autonomous and efficient offloading decisions. First, this scheme utilizes digital twin networks (DTN) and an improved k-means algorithm for clustering to achieve the effect of predicting the offloading space. Second, partial offloading is performed within the aggregation group using DRL with system delay and service price of the vehicle as joint optimization objectives. Finally, a new clustering will be performed using feedback from the offloading results. The main contributions of this paper are as follows.

- A feedback mechanism is proposed to adjust the parameters of the clustering algorithm in DTN based on the final offloading target selection and resource allocation results, thus making the clustering algorithm using DT more accurate and efficient. To the best of our knowledge, this is the first study on DT-assisted vehicle offloading that proposes a feedback mechanism, forming a complete closed loop as prediction-offloading-feedback.
- In VEC, a new vehicle clustering scheme, Gk-means, is introduced in DTN with the advantages of (i) factors such as resources, communication, and social relationships among vehicles are taken into account during clustering, and (ii) the optimal offload space found by clustering can improve the efficiency of task offloading.
- An autonomous dynamic pricing and partial offloading scheme is proposed, which is trained by the Deep Deterministic Policy Gradient (DDPG) algorithm. The objective is to jointly optimize task offloading delay and vehicle service price by combining V2V and V2R offloading scenarios to minimize overall system cost.

The rest of this paper is organized as follows. In Section II, we present the related work. Section III introduces the system model and the problem formulation. Section IV presents the algorithm design. Simulation results and conclusions are given in Sections V and VI, respectively.

II. Related Work

In this section, we review the literature on task offloading in VEC using RSUs and nearby vehicles. Besides, we present some studies that utilise DT techniques in task offloading. This section compares the existing studies in Table I to further discuss them in various terms.

A. Vehicle Task Offloading With RSUs

In recent years, RSUs-based vehicle task offloading schemes are extensively investigated. Ouyang et al. [18] consider multiple server-equipped RSUs on top of a base station (BS) at the same time, so the vehicle can choose to execute locally or offload to the BS/RSUs, thereby optimizing the computational resource allocation problem. Zhan et al. [19] divide the roads using the service range of the RSUs, on the basis of which the vehicles with insufficient computing power are queued, allowing them to optimize long-term costs. Li et al. [20] propose a location-aware task offloading scheme based on which vehicles can offload tasks to the nearest RSUs for computation, which can choose to process it alone by itself or jointly with other RSUs for collaborative processing. Peng et al. [21] consider the resource allocation problem from two different perspectives: the MacroeNodeB (MeNB) and the edge node, where both nodes are equipped with RSUs, thus improving the QoS for users.

B. Vehicle Task Offloading With Nearby Vehicles

The problem of limited RSUs coverage with a high computational burden can be alleviated by using vehicles with computing capabilities around the vehicle as edge servers. Many researchers discuss V2V-based vehicle task offloading schemes in recent years. For instance, Tang et al. [7] consider the migration cost problem due to vehicle location changes, so the proposed sequential decision problem is solved by introducing Bayesian inference combined with the DRL algorithm. Shi et al. [8] jointly consider the free computational resources of neighboring vehicles and the mobility of vehicles, and describe the offloading problem as a sequential decision problem, thus solving the proposed resource allocation dilemma. Chen et al. [10] consider the resources of the surrounding vehicles as a resource pool (RP) and propose an offloading

TABLE I
DISCUSSION OF SOLUTIONS FOR TASK OFFLOADING

Ref.	Algorithm	Offloading	Considered Factors			Optimization Objective
			DT	Social	Scenario	
[7]	DQN	Partial			V2V	Minimize delay and energy consumption.
[8]	SAC	Partial			V2V	Maximize the average utility of all tasks.
[10]	DDQN	Partial			V2V	Minimize delay of the task.
[18]	Dueling-DQN	Binary			V2R	Minimize energy consumption.
[19]	PPO	Binary			V2R	Minimize delay and energy consumption.
[20]	DDPG	Partial			V2R	Minimize the computing delay.
[13]	DDQN	Binary	✓		D2R	Minimize the computing delay.
[15]	MADDPG	Binary	✓		V2R	Minimize delay and energy consumption.
[25]	AAC	Partial	✓		D2R	Minimize the long-term energy efficiency.
[17]	AC	Binary	✓		D2R	Minimize the offloading delay.
[9]	TA-MCTS	Binary		✓	D2D	Minimize energy consumption.
Our work	DDPG	Partial	✓	✓	V2R and V2V	Minimize delay and service price.

scheme based on the DRL algorithm and the first-come, first-served (FCFS) principle to solve the task allocation problem.

C. Task Offloading With DT

As a new promising technology that can map the physical space to the virtual space, the digital twin have been gaining massive momentum, particularly in the manufacturing and intelligent transportation [22], [23], [24]. Among them, researches on the application of DT in task offloading has been emerging in recent years. Liu et al. [13] propose a DT-based scheme for mutual cooperation between edge server selection and user task offloading, with the goal of reducing the overall power overhead and system delay. Dai et al. [16] construct the virtual models of the vehicles and RSUs using DT and then propose an adaptive VEC network aiming at minimizing the total delay of vehicle task offloading. Dai et al. [25] divide the DITEN framework into a user layer, an edge layer, and a digital twin layer, where the DT of the client user is constructed in the base station (BS) using its local data. Sun et al. [17] minimize the offloading delay by using the DTs of the edge servers to estimate the state of the edge servers and the DTs of the overall MEC system to provide training data for offloading decisions.

D. Summary

Through analyzing some of the previous work, we can conclude the related work as the following points. First, the DRL algorithm can effectively solve the task offloading problem in complex and dynamic environments. Second, DT helps to provide preparatory solutions for some risky or challenging tasks in the physical world. However, the key challenge of existing work on vehicle task offloading is the ineffectuality about providing real-time services in complex and ever-changing vehicle environments. Therefore, this paper optimizes the task offloading decisions by combining DT and DRL in VEC. First, a feasible prediction scheme is designed by monitoring the state of the vehicle network in DTN in real time. Then, DRL-based task offloading is performed to improve the performance of the system. This program certainly has great potential.

III. SYSTEM MODEL

In this section, we present the system models investigated in this paper in terms of the DT model, the social relationship model, the network model, and the task offloading model, followed by a propose optimization problem framework. The main notations in this paper are listed in Table II.

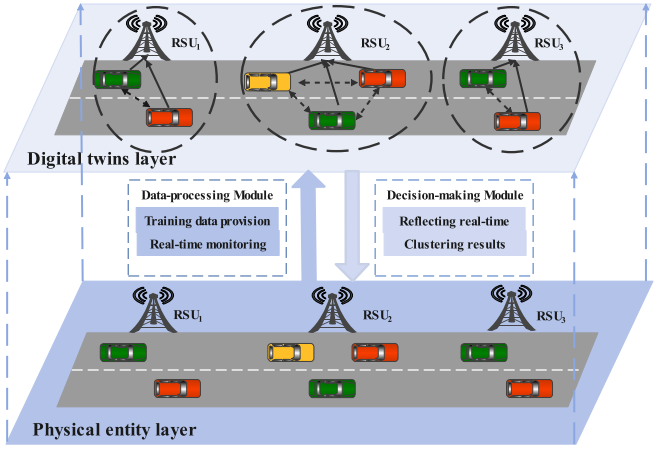

Fig. 1. Digital twin empowered VEC networks.

A. DT Model

The DTN framework is shown in Fig. 1, which consists of a physical entity layer and a digital twins layer.

The physical vehicle edge computing network consists of a user layer, an edge layer, and a cloud layer. The framework of VEC is illustrated in Fig. 2. Each vehicle in the user layer collects and analyzes data from a variety of applications and sensors, with the majority of these tasks being computationally intensive and time-sensitive. At the same time, vehicles in recent years have been equipped with more and more computing resources, which can be shared between vehicles. When the vehicle's computing power is limited, it can transfer the task to other vehicles with free resources to process, reducing the burden on RSUs while providing high-quality computing service. The edge layer consists of RSUs, which are distributed near the vehicles and can be connected to the vehicles in their coverage area through wireless communication, and it has real-time information about the vehicles on the road. Meanwhile, the digital twins of the vehicles are modeled in the RSUs. The cloud server in the cloud layer consists of servers with ultra-high performance computing power and storage capacity that can manage multiple edge servers efficiently.

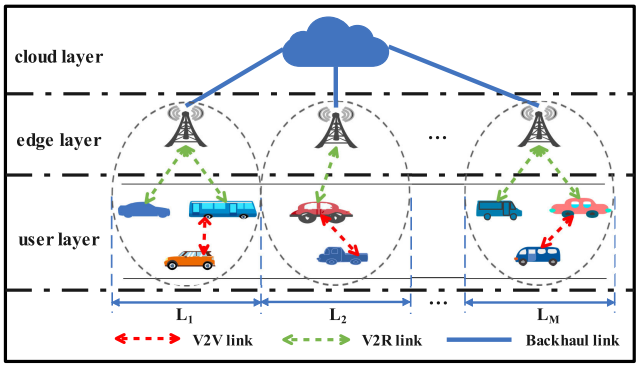

Fig. 2. The framework of VEC.

TABLE II
KEY NOTATIONS LIST

Notation	Description
$\omega_{n,m}^{d}$	Social similarity between vehicle n and m
$\omega_{n,m}^{c}$	Frequency of social contact between vehicle n and m
$\varepsilon_1\varepsilon_2$	Weighting of social similarity and frequency of social contact
H	Social Trust Matrix
K	Number of vehicles in the aggregation group
N	Number of TVs
M	Number of SVs
R_n	Task of TV_n
D_n	Input data size of TV_n
C_n	The amount of CPU cycles required for TV_n
T_n	Maximum tolerable delay of TV_n
ρ_n	The TV_n offloading ratio
f_n	Computing power of the vehicle
f_{rsu}	Computing power of RSUs
t_n^{loc}	Local computational delay of TV_n
t_n^{up}	The upload delay of TV_n
t_n^{comp}	The computing delay of TV_n
t_n^{down}	The download delay of TV_n
t_n^{off}	The total unloading delay of T
r_n	The rate of uploading data of TV_n
W_n	Bandwidth of wireless channels
P_n	Transmission power of uploaded data
d_n	Distance between TV_n and server
β	Path loss index
h_n	Wireless channel gain
N_0	Gaussian noise power inside the channel
I_n	Interference from other transmissions
p_n	Price of unit services provided by TV_n
M_i	Weight of vehicle i
$F_{i,j}$	Gravitational force between vehicle i and j
S	State space
A	Action space
$R(S,A)$	Reward function

The real-time reflection of the characteristic state of the physical system and the prediction of the system can be achieved by constructing the virtual representation in the DT network. The construction of the DT network mainly includes three key components: data storage, model mapping, and digital twin management [13]. Data storage is the storage of vehicle state, RSUs state, network state, and channel state in the physical network, which provides the raw material for building the model. Model mapping entails mapping the virtual model of the vehicle, including the steering wheel, driving speed, tire pressure, and other information; the virtual model of the RSUs, including location, resources, and other information; and the virtual model of the network, including road conditions, congestion, and other information. Digital twin management means updating and managing the mapping between the physical and digital twin layers in real-time.

We mainly utilize the DTN framework in this paper to assist in the design of clustering since DTN can learn global information about all vehicles on the road, such as vehicle resources and road conditions. Specifically, the DTN framework is formed in the RSUs based on the clustering algorithm and is then used to divide the vehicles on the road into different aggregation groups. When the vehicle resources within the group are insufficient to require offloading, the constraint is offloaded within the group, i.e., the range of objects selected to be offloaded is narrowed in advance to achieve the goal of offloading prediction.

B. Social Relationship Model

Due to the high dynamic nature of VEC network, it is crucial to maintain the stability of the communication links between vehicles. Broken links may lead to the extremely high delay and energy consumption. Consequently, we have conducted extensive research on the solution of dynamics of vehicles. As a promising architecture to aggregate the multi-scale factors of VEC network, social network can extract raw data from vehicles to construct social relationships and further meet the challenge of vehicle mobility [9]. Therefore, we propose a model to describe the stability of inter-vehicle link connections using a social relationship model. And the social trust value is proposed as a measurement of the social relationship between vehicles, and quantify it as a social trust matrix. Among them, the social trust value between vehicles is measured by two main metrics, namely directional similarity and speed similarity. The main design principle is the measurement of proximity of two vehicles. The specified details of above parameters are defined as follows.

Directional similarity. By checking whether the directions of the two vehicles are the same, we let $\omega_{n,m}^{d}$ to express the directional similarity between the two vehicles as follows.

$$\omega_{n,m}^{d}=\begin{cases}1, & \text{if } m \text{ has the same direction as } n;\\ 0, & \text{if } m \text{ has no the same direction as } n.\end{cases} \tag{1}$$

where $\omega_{n,m}^{d}=1$ indicates that vehicle m has the same orientation as vehicle n. Conversely, $\omega_{n,m}^{d}=0$ indicates that vehicle m does not have the same orientation as vehicle n.

Speed similarity. The speed similarity between vehicle n and vehicle m is calculated by examining whether the speeds of two vehicles are the same. We let $\omega_{n,m}^{v}$ to denote the speed similarity of the two vehicles as below.

$$\omega_{n,m}^{v}=\begin{cases}0, & \text{if } v_m=0 \text{ or } v_n=0;\\ \dfrac{v_m}{v_n}, & \text{if } v_m<v_n;\\ \dfrac{v_n}{v_m}, & \text{if } v_n<v_m;\\ 1, & \text{if } v_n=v_m.\end{cases} \tag{2}$$

where v_n and v_m represent the respective travel speeds of vehicle n and vehicle m.

Based on the above indicators, we can use $\omega_{n,m}$ to express the social trust value between vehicle n and vehicle m. From

the perspective of vehicle n, the social trust value between it and vehicle m is as follows.

$$\omega_{n,m} = \varepsilon_1 \omega_{n,m}^{d} + \varepsilon_2 \omega_{n,m}^{v} \tag{3}$$

where ε_1 and ε_2 are tuning parameters, representing the weights of directional and speed similarities, respectively. $(\varepsilon_1 + \varepsilon_2 = 1)$. A larger value of $\omega_{n,m}$ indicates a greater trust between vehicle n and vehicle m. Conversely, a smaller trust between vehicle n and vehicle m. We then introduced a social trust matrix $\mathrm{H} = [\omega_{n,m}]_{M\times M}$ to represent the social relationships, where M denotes the number of vehicles in the region.

Considering the highly dynamic nature of vehicle networks and the fact that not all vehicles on the road are willing to share their resources to help other vehicles perform their tasks, several studies have emerged to solve the vehicle mobility problem through clustering method [26]. In this paper, we focus on clustering vehicles using a social relationship model, where link connections between vehicles within the same aggregation group will be more stable and can improve the efficiency of V2V computational offloading. Thus, considering the social relationship model in the improved clustering algorithm can achieve better prediction results.

C. Network Model

The vehicles on the road are clustered into aggregation groups using the DT-based clustering algorithm. Each aggregation group is made up of an RSUs with a server and K vehicles, the set of vehicles is represented by $\mathcal{K} = \{1, 2, \ldots, K\}$, and all vehicles in the group have access to the RSUs. We assume that two-thirds of the vehicles in the aggregation group lack the computational resources to complete the computationally intensive tasks, forcing them to offload some of their tasks [27]. Specifically, these vehicles transmit service requests to the RSUs, which then decides whether to execute them directly at the RSUs or assign them to other vehicles with available resources to reduce delay, depending on the burden on the RSUs.

We denote the vehicles that need to offload tasks as task vehicles (TVs) and the vehicles that can provide free resources within the aggregation group as service vehicles (SVs). Assume that there are N vehicles that need to be offloaded during the task offloading and denoted as $\mathcal{TV} = \{TV_1, T\ V_2, \ldots, T\ V_N\}$. There are M vehicles available for service within the aggregation group and indicated as $\mathcal{SV} = \{SV_1, S\ V_2, \ldots, S\ V_M\}$. We denote the task of TV_n as $R_n = \{D_n, C_n, T_n\}$, where D_n is the input data size of the task, C_n is the amount of CPU cycles required to complete task R_n and T_n is the maximum tolerable delay.

Unlike other binary computation offloading schemes, our scheme is that $TV_n\,(TV_n \in \mathcal{TV})$ can be offloaded to RSUs or $SV_m\,(SV_m \in \mathcal{SV})$ in any proportion, which places higher demands on our algorithm. We assume that TV_n can offload the tasks from part ρ_n and the rest of $(1-\rho_n)$ will be computed locally, where $\rho_n \in [0, 1]$. $\rho_n = 0$ signifies that the task will be executed locally, $\rho_n = 1$ means that it will be offloaded, and $0 < \rho_n < 1$ signifies that it will be partially offloaded and partially executed locally in parallel.

D. Task Offloading Model

1) Local Computing Model: When the vehicles on the road perform tasks locally, that is, when they all use local resources to compute, the tasks are performed independently of other vehicles on the road and the MEC server. The local computation delay is denoted by t_n^{loc}, which is represented as follows.

$$t_n^{loc} = \frac{C_n}{f_n} \tag{4}$$

where f_n is the computational power of TV_n (the number of CPU cycles per second of TV_n).

2) Edge Computing Model: When the task of TV_n is offloaded out, we consider offloading the ρ_n part of task R_n for execution while the remaining $(1-\rho_n)$ part is executed locally. Upload delay, compute delay, and data download delay are all included in part ρ_n and are denoted by t_n^{up}, t_n^{comp}, and t_n^{down}, respectively. t_n^{up} is calculated as follows.

$$t_n^{up} = \frac{\rho_n \times D_n}{r_n} \tag{5}$$

where r_n denotes the data transmission rate, and r_n is calculated as shown below.

$$r_n = W_n \log_2\left(1 + \frac{P_n d_n^{-\beta} h_n^2}{N_0 + I_n}\right) \tag{6}$$

where W_n denotes the bandwidth of the wireless channel, P_n refers to the transmission power of the data uploaded by the vehicle, d_n represents the distance between TV_n and the server, β is the path loss index, h_n denotes the wireless channel gain, N_0 denotes the Gaussian noise power inside the channel, and I_n denotes the interference from other transmissions [28]. t_n^{comp} is estimated when TV_n chooses to offload to SV_m, as shown below.

$$t_{n,m}^{comp1} = \frac{\rho_n C_n}{f'_m} \tag{7}$$

where f'_m represents the computational resources provided by SV_m for TV_n, where f'_m should be less than f_m, and f_m represents SV_m's computational power (the number of CPU cycles per second of SV_m). The calculation of t_n^{comp} when TV_n chooses to offload to RSUs is shown below.

$$t_n^{comp2} = \frac{\rho_n C_n}{f'_{rsu}} \tag{8}$$

where f'_{rsu} denotes the computing resources provided by the MEC for TV_n, where f'_{rsu} should be less than f_{rsu}, and f_{rsu} is the computing power of RSUs (the number of CPU cycles per second of RSUs). In summary, when TV_n is offloaded out, t_n^{comp} is computed as follows.

$$t_n^{comp} = \begin{cases} t_n^{comp1}, & \text{If offloading to } SV_m; \\ t_n^{comp2}, & \text{If offloading to RSUs .} \end{cases} \tag{9}$$

t_n^{down} is determined as shown below.

$$t_n^{\text{down}} = \frac{\delta D_n}{r_n} \tag{10}$$

δ is the ratio of the size of the output data to the size of the input data, where $\delta \ll 1$ [29], [30], therefore the data

downloading delay can be omitted when calculating the delay, where t_n^{loc} is the time delay of portion $(1-\rho_n)$. The result of t_n^{loc} 's calculation is displayed below.

$$t_n^{loc} = \frac{(1-\rho_n)\, C_n}{f_n} \tag{11}$$

Based on Equations (5) (9) (10) (11) above, we use t_n^{off} to denote the total delay during partial offloading, and t_n^{off} is calculated as shown below.

$$t_n^{off} = \max\left\{\left(t_n^{up} + t_n^{comp}\right), t_n^{loc}\right\} \tag{12}$$

since partial offloading causes task R_n to execute in parallel, the total delay should be the greater of the delay of part ρ_n and part $(1-\rho_n)$ of R_n [8].

3) Economic Factors: According to the above computational model, when vehicle TV_n decides to offload the task, it will also decide the service price to be paid to the resource provider at the same time, where the service price is p_n and the price to be paid by TV_n is expressed as

$$P_n = \rho_n p_n \tag{13}$$

The total consumption cost of TV_n is the weighted sum of t_n^{off} and P_n.

E. Problem Formulation

In Equation (12), when $\rho_n = 0$ indicates that the tasks are not offloaded and are all executed locally, $t_n^{off} = t_n^{loc}$ is used. When $\rho_n = 1$ indicates that all tasks are offloaded, then $t_n^{\text{loc}} = 0$. In conclusion, we can combine the delay of three strategies using Equation (12): local computation, partial offload computation, and full offload computation. As a result, the total cost of TV_n is the sum of t_n^{off} and P_n. The total cost of all vehicles in the aggregation group can be denoted by Ω, and Ω can be defined as

$$\Omega = \sum_{n=1}^{N}\left(\alpha t_n^{off} + \beta\eta P_n\right) \tag{14}$$

where $\alpha, \beta \in [0,1]$ denotes the relative weights accounted for by delay and service price, which can be adjusted based on the actual situation, and η is a regulator. Our goal is to minimize Ω, which can be defined as

$$(P): \min\ \Omega = \sum_{n=1}^{N}\left(\alpha t_n^{off} + \beta\eta P_n\right) \tag{15}$$

$$\boldsymbol{s.t.}\ N \cup M = K, \tag{15a}$$

$$t_n^{off} \le T_n, \quad \forall n \in N, \tag{15b}$$

$$\rho_n \in [0,1], \quad \forall n \in N, \tag{15c}$$

$$0 \le (f'_{rsu} \mid \rho_n \ne 0) \le f_{rsu}, \quad \forall n \in N, \tag{15d}$$

$$0 \le (f'_m \mid \rho_n \ne 0) \le f_m, \quad \forall n \in N,\ \forall m \in M, \tag{15e}$$

$$\sum_{n=1}^{i}(f'_{rsu} \mid \rho_n \ne 0) \le f_{rsu}, \quad \forall n \in N, \tag{15f}$$

where condition (15a) indicates that TV_n must be restricted to select SV_m within the aggregation group to help it perform the computation. The completion time of each task should not exceed its maximum time delay, according to condition (15b). Condition (15c) states that each TV_n has the option of offloading locally or offloading out of computational execution in any proportion. Condition (15d) and (15e) imply that other vehicles' or RSUs' resources allocated to TV_n are positive but should not exceed their total computational capacity. Condition (15f) indicates that all resources allocated by the RSU for the TVs should not exceed its total computing capacity.

The time complexity of finding the best solution increases exponentially with the number of TVs when $\rho_n = 0/1$. This problem has been shown to be an NP problem [6]. This NP problem becomes more complex as we introduce a partial offloading scheme (i.e., the value of ρ_n is varied arbitrarily between 0 and 1), so in the next section we solve this problem using DRL.

IV. Proposed Solution

As shown in Fig. 3, we detail the solution to the problem and the steps need to be taken to solve it in this section. First, we address the vehicle clustering problem using the enhanced Gk-means algorithm based on DITEN. Second, after generating the aggregation group, we design the state space, action space and reward function for the vehicle partial offloading problem, based on this, we train the neural network using the DDPG algorithm to obtain the optimal solution to the task offloading problem. Finally, we design a feedback mechanism to readjust the clustering process parameters based on the offloading results in order to achieve better clustering results.

A. Solution for Vehicle Clustering: Gk-means

Due to the massive distribution of vehicles on the road, directly scheduling the task offload of the entire edge network on a large scale is both costly and impossible. To address this problem, we devise an edge service aggregation scheme that employs the DTN and gravity model to determine the optimal offloading space. The scheme efficiently aggregates vehicles based on possible matching relationships between computing resource availability and demand, considerably reducing task offload scheduling complexity.

To lead edge service aggregation, the digital twin nodes of the vehicular edge network are constructed in the RSU. Each RSU gathers the computational power and communication topology of the vehicles in its immediate surroundings, then communicates this data through wired transmission to build a vehicle edge DTN. The DTN is considered as a mirror image of the physical space, and with this in mind, we define the elements of the DTN as $DTN = \{\mathrm{Z}, \Phi, \mathrm{T}\}$. where Z denotes the digital model of the vehicle in the physical system, consisting of the vehicle task set $\{R_i\}$, the computational power set $\{f_i\}$, the available transmission rate set $\{r_i\}$, and the social trust set $\{\omega_{i,j}\}$. $\Phi = \{\phi_1, \phi_2, \phi_3\}$ is the modeling parameter that represents the relative relevance of the three aspects of resources, social trust, and communication that clusters in DTN account for. T stands for the serial number of the cycle, as the parameters are constantly updated.

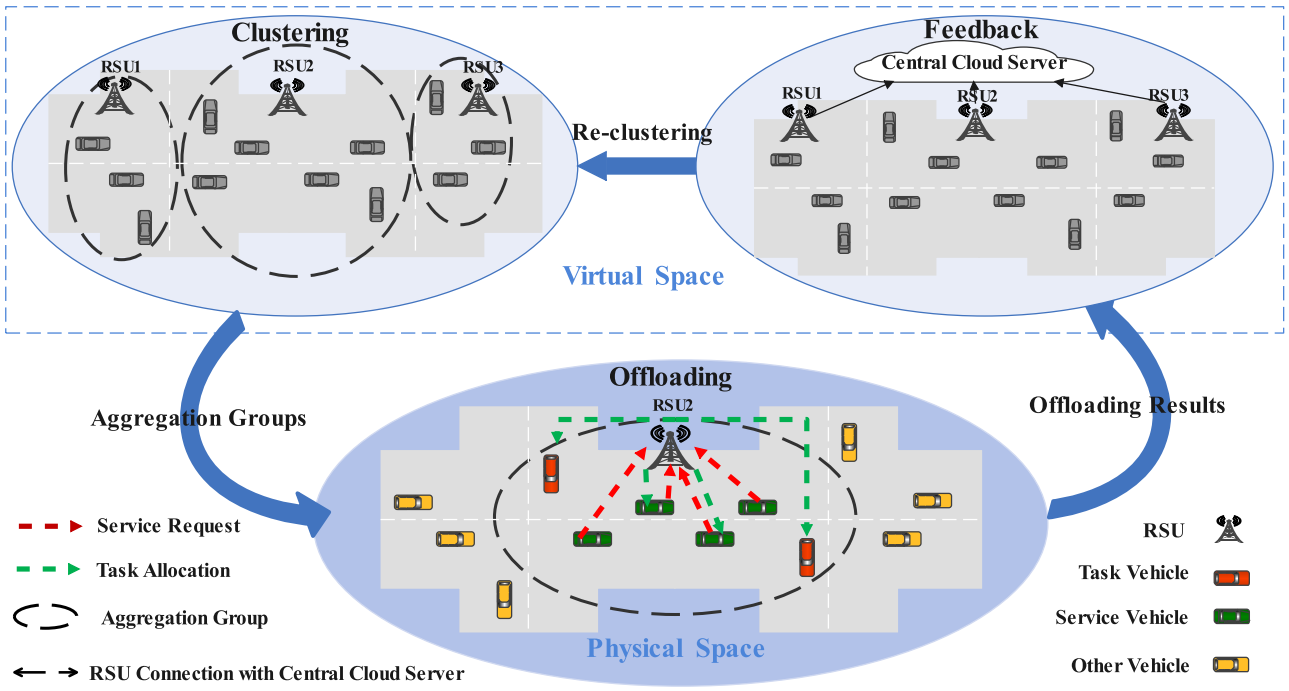

Fig. 3. A comprehensive strategy for offloading in VEC networks.

With the assistance of DTN, we design a gravity model-based algorithm to improve the original k-means algorithm [31], and then we produce aggregation groups using the innovative Gk-means algorithm, with each aggregation group containing an RSU and N vehicles. where Isaac Newton's law of gravitation is enhanced [32], allowing it to be used to define the supply and demand for vehicle edge services on the road. The weight of vehicle i requesting service is defined as

$$M_i = \frac{C_i}{f_i} \tag{16}$$

where C_i is the amount of CPU cycles required by the vehicle to complete task R_i, and f_i is the number of CPU cycles per second for the vehicle. It estimates the need of vehicle i and consequently relieves its own resource limits via offloading. When vehicle i acts as an edge service vehicle, its mass M_i' is indicated as $M_i' = M_i^{-1}$.

Applying the definition of mass to the gravitational model, the gravitational force between vehicle i and vehicle j on the road is defined as

$$F_{i,j} = \frac{\phi_1 \max\left(M_i/M_j, M_j/M_i\right)}{\left(\phi_2/\omega_{i,j} + \phi_3/r_{i,j}\right)^2} \tag{17}$$

where a maximum function is used in the numerator to obtain the strongest supply and demand relationship between two vehicles, and the denominator replaces the original distance element with the social trust value and the communication rate, two factors that affect the edge service performance.

Finally, we use the gravitational formula defined by Eq.(17) to replace the original distance element in the k-means algorithm to first cluster the vehicles on the road, then the distance from the clustering center to the RSUs is considered for judgment, and the RSUs closest to the clustering center within the group is merged into the group, thus each aggregation group consists of one RSUs equipped with MEC server and N vehicles. We assume that the set consisting of the vehicles in the region with the clustering centers removed is $\bar{N} = \{v_1, v_2, \ldots, v_n\}$, the clustering center set is $\bar{K} = \{v_1', v_2', \ldots, v_k'\}$. Finally, the set of aggregation groups generated by the clustering algorithm is $\bar{G} = \{\bar{g_1}, \bar{g_2}, \ldots, \bar{g_k}\}$. The specific clustering scheme is shown in Algorithm 1. First, the system initializes the parameters in Eq.(17) and the number of aggregation groups (Lines 1-3). Second, the clustering centers are randomly selected and the initial aggregation groups are generated (Lines 4-7). At last, the gravitational force between each vehicle and each clustering center is calculated using Eq.(17). Each vehicle then selects the clustering center with which it has the greatest gravitational force and adds it to this aggregation group (Lines 8-16).

B. Solution for Partial Offloading of Vehicles

In this subsection, we apply the DRL-based algorithm to the issue of task offloading [33]. First, we present the definition of Markov Decision Process (MDP). Second, we describe the training process based on the model-free DDPG algorithm [34].

1) Deep Reinforcement Learning Framework: The state space, action space, and reward function for the joint V2R and V2V offloading problem are designed as follows.

The state space needs to be determined by observing the whole system after DT-assisted clustering, including the number of vehicles in the aggregation group, the task size of TVs, and the size of computational resources of MEC servers and SVs, so we can represent the state of the system as

$$S = \{S_n, S_m\} \tag{18}$$

Algorithm 1 Edge Vehicle Aggregation Algorithm Based on Gravity Model

Require: Expected number of aggregation groups k

Ensure: The set $\bar{G} = \{\bar{g_1}, \bar{g_2}, \ldots, \bar{g_k}\}$ of k aggregation groups

Initialize modeling parameters ϕ_1, ϕ_2, and ϕ_3;
Initialize the value of k;
Initialize the set of aggregation groups $\bar{G} = \varnothing$;
Randomly select k vehicles as initial clustering centers to form $\bar{K} = \{v'_1, v'_2, \ldots, v'_k\}$;
for m=[1,2,..., k] **do**
    Update $\bar{g_m}$ to $\bar{g_m} = \{v'_m\}$;
end for
for i=[1,2,..., n] **do**
    According to Eq.(17), calculate the gravitational set $\{F_{v_i,v_j}\}$,$\forall v_j \in \bar{K}$;
    Select vehicle v_j, where $F_{v_i,v_j} = \max\{F_{v_i,v_j}\}$;
    Update $\bar{g_j}$ to $\bar{g_j} = \bar{g_j} \cup \{v_i\}$;
    $i = i + 1$;
end for
for $p = [1, 2, \ldots, k]$ **do**
    Update $\bar{G}$ to $\bar{G} = \bar{G} \cup \{\bar{g_p}\}$;
end for

where S_n denotes the task information of T_n and S_n equals to (D_n, C_n, f_n), D_n is the size of T_n, C_n is the size of the computing resources required by T_n, f_n is the size of the computing power of T_n. S_m indicates the information of the server and S_m equals to $(f'_m, C^m_{\text{remain}})$, f'_m is the size of the computing power of RSU or S_m, C^m_{remain} denotes the amount of energy remaining in the server.

Since each vehicle autonomously chooses that target to offload to, how much to offload, and the price of the services it can provide, where $\rho_n \in [0, 1]$, encompasses all cases of local computation, partial offloading, and full offloading, this offloading in any proportion optimizes the total delay to a large extent. As a result, the action space is defined as

$$A = \{\rho_n, p_n, x^n_m\} \tag{19}$$

where ρ_n and p_n represent the offloading percentage and the price of services provided by TV_n respectively. x^n_m represents the offload of the TV_n selection to server m, where $m \in \{0, 1, .., M\}$ represents the RSU and SVs.

The reward obtained by executing the computational offloading strategy A in the state space S is denoted as $R(S, A)$. In DRL, the reward function is seen as a guide for algorithm learning, while the goal of DRL is to maximize the cumulative reward, so the goal and the desired reward should be the same. As a result, the reward function can be expressed as

$$R(S, A) = -\sum_{n=1}^{N} \left(t_n^{off} + \rho_n p_n\right) \tag{20}$$

which indicating that the reward is greater when the offloading cost is lower. We have now built the three components of DRL: state space, action space, and reward function, and we will already introduce the task offloading strategy.

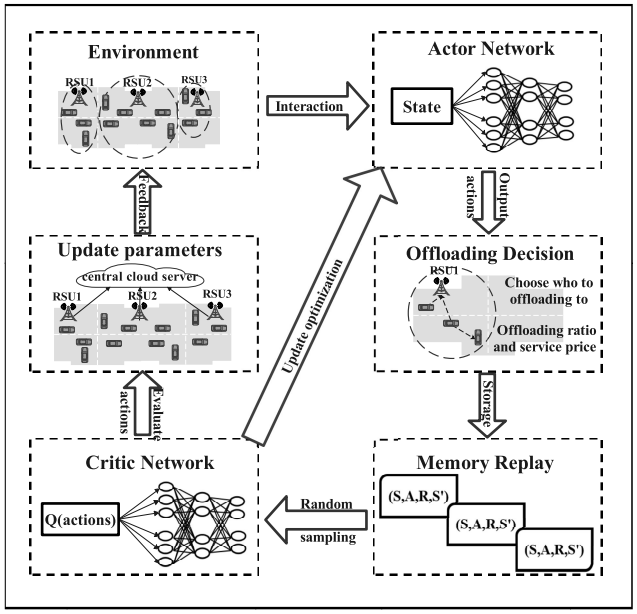

Fig. 4. The VEC offloading framework based on DDPG.

2) DDPG-Based Partial Offloading of Vehicles: The DDPG algorithm combines deterministic policy gradient (DPG) with a deep network through the Actor-Critic framework, thus extending the action space of DRL to a continuous domain; it also uses empirical playback and a dual-network approach to improve the difficult convergence problem of Actor-Critic. Fig. 4 depicts the VEC offloading framework and training process based on DDPG.

First, we use the current actor network to interact with the environment. The state space is obtained from the information in DT as the input to the actor network. The actor network is responsible for selecting the current action a_t based on the current state s_t, which is used to interact with the environment to generate s_{t+1} and r_t. And the quaternion (s_t, a_t, r_t, s_{t+1}) is then placed into the experience pool.

Second, we update the parameters of the current critic network. We first take random samples from the experience pool for training; then the s_t and a_t in (s_t, a_t, r_t, s_{t+1}) are fed into the critic network to evaluate the action in the current state to obtain a realistic Q value Q^{real}_t ; we finally input s_{t+1} from (s_t, a_t, r_t, s_{t+1}) to the actor network to obtain a_{t+1}, and we input s_{t+1} and a_{t+1} to the target critic network to get the target Q value, where the calculation of the target Q value is given by

$$Q^{\text{target}}_t = r_t + \gamma Q'\left(s_{t+1}, \mu'\left(s_{t+1} \mid \theta^{\mu'}\right) \mid \theta^{Q'}\right) \tag{21}$$

We want Q^{real}_t to be equal to Q^{target}_t, so we use gradient descent to update the current critic network's parameter θ^Q to bring Q^{real}_t closer to Q^{target}_t. Its update formula is given by

$$Loss = \frac{1}{T}\sum_{i=1}^{T}\left(Q^{\text{target}}_i - Q\left(s_i, a_i \mid \theta^Q\right)\right)^2 \tag{22}$$

where γ and T in the above equation represent the decay factor and the size of the mini-batch, respectively.

Third, we update the parameters of the current actor network. Since the action output by the current actor network gives the Q value in the current critic network, we maximize the Q output by updating the parameter θ^{μ} of the current actor network, which is updated using the gradient ascent method. Its update method is given by

$$\nabla_{\theta^{\mu}} J \approx \frac{1}{T} \sum_{i} \nabla_{a} Q\left(s_i, a_i \mid \theta^{Q}\right) \nabla_{\theta^{\mu}} \mu\left(s_i \mid \theta^{\mu}\right) \tag{23}$$

Finally, we update the parameters of the target actor network and the target critic network. At regular intervals, soft target updates are performed on the target network using the parameters of the actor network and the critic network. The soft target update is performed as

$$\theta^{\mu'} \leftarrow \tau\theta^{\mu} + (1-\tau)\theta^{\mu'} \tag{24}$$

$$\theta^{Q'} \leftarrow \tau\theta^{Q} + (1-\tau)\theta^{Q'} \tag{25}$$

where τ is the soft update speed of the target network and it generally takes a relatively small value, such as 0.01. The detailed procedure of the DDPG algorithm assisted by DT in the VEC system is summarized in Algorithm 2. The replay buffer and neural network are first initialized by the system (Lines 1-5). Second, at the beginning of each episode, initialize the VEC network (Lines 6-8). Third, at each time slot, the action is executed and the reward is obtained with the state of the next moment and (s_t, a_t, r_t, s_{t+1}) is stored into B (Lines 9-14). Fourth, the policy gradient is leveraged to update the parameters of the actor and target network (Lines 16-21).

3) Algorithm Complexity Analysis: In this section, we analyze the time complexity and space complexity of our proposed IGNITE partial offloading scheme. First, based on the description of the previous work [35], we can determine the time and space complexity of the DDPG algorithm as follows.

$$O\left(\sum_{j=0}^{J-1} u_{\text{actor},j} u_{\text{actor},j+1} + \sum_{k=0}^{K-1} u_{\text{critic},k} u_{\text{critic},k+1}\right) + O(\mathcal{N}(s)) \tag{26}$$

$$O\left(\sum_{j=0}^{J-1} u_{\text{actor},j} u_{\text{actor},j+1} + \sum_{k=0}^{K-1} u_{\text{critic},k} u_{\text{critic},k+1}\right) + O(\mathcal{N}(s)) + O(N) \tag{27}$$

where J and K represent the number of fully connected layers in the actor and critic networks, respectively.

However, Algorithm 2 is not identical to the DDPG algorithm. Line 7 in Alg. 2 indicates that the vehicles on the road are clustered using the Gk-means algorithm through the DT technique to form an optimal offloading space. By filtering the offloading decision in this step, the action space is reduced and the complexity of the space is decreased. But clustering the vehicles also incurs additional time and space complexity, and both the time and space complexity of the clustering algorithm is $O(N)$, meaning that both are linear. This demonstrates that, compared with the original DDPG algorithm, our suggested IGNITE offloading scheme can somewhat increase network learning efficiency. However, it also adds some additional time and space complexity.

Algorithm 2 IGNITE Partial Offloading Scheme

Require: DT-assisted VEC network
Ensure: Offloading decisions
1: Initialize actor network $\mu\left(s \mid \theta^{\mu}\right)$ and critic network $Q\left(s, a \mid \theta^{Q}\right)$ with parameters θ^{μ}, θ^{Q};
2: Initialize the parameters of target actor network with $\theta^{\mu'} \leftarrow \theta^{\mu}$;
3: Initialize the parameters of target critic network with $\theta^{Q'} \leftarrow \theta^{Q}$;
4: Initialize replay buffer B, the mini-batch T;
5: Initialize discounter γ and soft update factor τ;
6: **for** episode $= 1$ to 1000 **do**
7: Reset simulation environment of the DT and the obtained clustering results;
8: Observe an initial state s_0;
9: **for** $t = 1$ to T **do**
10: Choose action $a_n(t)$ with actor network actor network $\mu\left(s \mid \theta^{\mu}\right)$;
11: Execute $a_n(t)$ and receive reward r_t;
12: Observe the next state s_{t+1};
13: **if** B is not full **then**
14: Save the record (s_t, a_t, r_t, s_{t+1}) into replay buffer B;
15: **else**
16: Randomly replace the record (s_t, a_t, r_t, s_{t+1}) in B and sample mini-batch records from B;
17: Compute the target Q value by Eq.(21);
18: Update the parameters of the critic network by Eq.(22);
19: Update the actor policy by Eq.(23);
20: **if** t mod d **then**
21: Soft update the parameters of target network by Eq.(24) and Eq.(25);
22: **end if**
23: **end if**
24: **end for**
25: **end for**

C. Solutions for Feedback Mechanism

When the vehicles within the aggregation group complete their computational offloading, the selection of offloading targets and the results of resource allocation affect the performance of the VEC network. To improve clustering results, we need to re-tune the parameter set of gravity in the clustering algorithm during the next mapping week of the DTN. The specific adjustment is as follows.

We use $\Phi_T = \{\phi_{1,T}, \phi_{2,T}, \phi_{3,T}\}$ to denote the set of parameters of the gravitational force in the period T. They are all updated in a way that depends on the results of the comparison of the two cycles before and after. Parameter $\phi_{1,T}$ reflects the influence of resource factors on the gravitational operation and it is updated as

$$\phi_{1,T} = \phi_{1,T-1} \cdot \frac{\sum_{i=1}^{k \cdot K} C_i}{\sum_{g_i \in \bar{G}} \cdot \sum_{n=1}^{N} C'_n} \tag{28}$$

where the coefficient in the update equation represents the ratio of computational resources required within the aggregation group during the mapping cycle of the current and last DTN. When the ratio is greater than 1, it indicates that the average computational resources required within the aggregation group increases in the new mapping period, so $\phi_{1,T}$ needs to be increased to improve the sensitivity of computing resource matching in the clustering process.

The parameter $\phi_{2,T}$ indicates the degree of influence of social trust on the gravitational operation and it is updated as

$$\phi_{2,T} = \phi_{2,T-1} \cdot \frac{\sum_{g_i \in \bar{G}} \sum_{m=1}^{M} \sum_{n=1}^{N} \omega'_{m,n}}{\sum_{i=1}^{k \cdot K} \sum_{j=1}^{k \cdot K} \omega_{i,j}} \tag{29}$$

where $\omega'_{m,n}$ and $\omega_{i,j}$ denote the social trust values between two vehicles in cycles $T-1$ and T, respectively. When the coefficient is less than 1, it indicates that the social trust value within the aggregation group is enhanced in the new mapping cycle, so the role of social trust in the clustering process can be improved by reducing $\phi_{2,T}$.

The parameter $\phi_{3,T}$ indicates the role of data transfer in the gravitational operation, which is updated as

$$\phi_{3,T} = \phi_{3,T-1} \cdot \frac{\sum_{g_i \in \bar{G}} \sum_{m=1}^{M} \sum_{n=1}^{N} r'_n T'_{n,m}}{\sum_{i=1}^{k \cdot K} D_i} \tag{30}$$

where $T'_{n,m}$ denotes the transmission time between vehicle n and vehicle m in period $T-1$. Hence, the coefficient in the update equation represents the ratio of the data transmission capacity during the mapping period between the last and the current DTN; and when the ratio is less than 1, the weight of data transmission in the clustering process can be increased by reducing $\phi_{3,T}$.

V. Performance Evaluation

In this section, we evaluate the IGNITE scheme based on the Gk-means clustering algorithm in the VEC system through numerical simulations. First, the settings of simulation parameters and network structure are described in detail. Second, the comparison and analysis results of different parameters in DDPG algorithm are given. Finally, we compare the evaluation results with the other four different baseline schemes on different metrics.

A. Simulation Setup

1) Simulation Parameters Setting: To perform the simulation experiments, we generated our simulation platform in Python 3.7 and implemented the experiments using libraries such as numpy, matplotlib and pandas, and finally we made the open source code of the platform available at https://github.com/NetworkCommunication/IGNITE. In our simulation, the main parameters used are described below. Assume the targeted area consists of two intersections that are $200m$ in length and width, and there are 25 vehicles in the area. The value of k in the Gk-means algorithm is defined as 3. Then, after one clustering, there are 15 vehicles within one of the aggregation groups to be studied. As for the channel model, we let the transmission bandwidth of the channel be $W = 10$ MHz, the noise power density be $N_0 = -174$ dBm/Hz, and the channel gain between TV and SV vary from moment to moment between $\left(10^{-8}, 10^{-7}\right)$. In addition, we set the power of the vehicle uploading data to $P = 0.1$ W. In terms of the task model, it is assumed that the data size of task is $(0.8, 1.2)$ MB, and the amount of CPU cycles required to execute the task is $(0.1, 1.0)$ G. Regarding the computing power, it is assumed that the CPU frequency of the VEC server is randomly distributed between 2.0G and 2.8G cycles/s, while the CPU frequency of the vehicle is 0.5G cycles/s. Table III provides a list of the specific simulation parameters.

TABLE III
Main Parameters Setting

Parameter	Symbol	Value
Transmission bandwidth	W_n	10 MHz
Noise power density	N_0	-174 dBm/Hz
Channel gain	h_n	$\left(10^{-8}, 10^{-7}\right)$
Data upload power	P_n	0.1 W
Data size of tasks	D_n	$(0.8, 1.2)$ MB
CPU cycles of tasks	C_n	$(0.1, 1.0)$ G
CPU frequency of the server	f_{rsu}	$(2.0, 2.8)$ GHz
CPU frequency of the vehicle	f_n	0.5 GHz
Number of clustering centers	k	3
Number of vehicles	K	15
Number of task vehicles	N	10
Number of service vehicles	M	5
Discount Factor	γ	0.99
Experience pool size	e	6400
Batch Size	b	64
Maximum delay tolerance	T_n	1.5
Learning Rate	α, β	0.0001,0.001

2) Network Structure: The network structure of DDPG in the VEC offloading algorithm is shown in Fig. 4. The input layer dimension of the actor network is 1, which represents the state input of the VEC system after clustering; the final output of the actor network is whether to offload, to whom to offload, the offloading ratio and the price of the service provided, so the output layer dimension is 4. The output of the actor network is scaled using the sigmoid activation function. And the structure of the target actor network is identical to its structure. The critic network's input layer is divided into two parts, one part is the state of the VEC system after clustering, and the other part is the output content of the actor network; the output layer of the critic network has a dimension of 1, indicating the prediction of Q values. Similarly, the structure of the target critic network is exactly the same as that of the critic network.

B. Parametric Comparsion for DDPG

1) Parameter Introduction: To identify the optimal values for these parameters, we compare the effects of various parameters on simulation experiments.

(a) Experience pool size is the experience data that can be stored. The size of the experience pool needs to be set reasonably because if it is too big, it might add unnecessary experience to the training.

(b) Batch size is the quantity of samples used in each neural network training cycle. Large batches typically allow

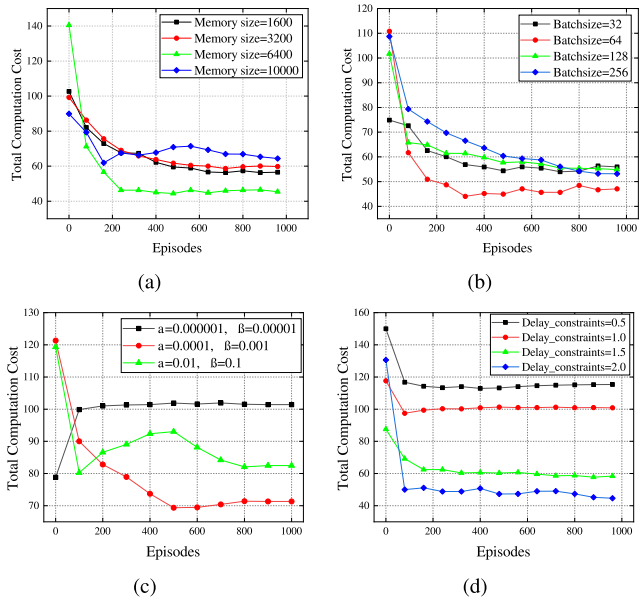

Fig. 5. Total Computation Cost under different algorithm parameters.

the network to converge faster, but they are limited by memory resources.

(c) Learning rate is the magnitude of updating the network weights in the optimization algorithm. Too large may lead to non-convergence of the model, too small may lead to slow convergence.

(d) Time delay constraint is the maximum time limit that a vehicle can tolerate to complete a computational task, beyond which it is considered to have failed. If the delay constraint value is set too high, it may not have a constraining effect on the processing time of the task. If the delay constraint value is set too low, it may lead to a high offload failure rate.

2) Description of Comparative Results: Fig. 5 depicts the impact of different parameter values on experimental performance. In Fig. 5(a), the total computation cost of the system is lowest when the memory size is 6400. Furthermore, it stabilizes the computation cost at about 200 episode, so its convergence performance is better compared with others. Therefore, we set the memory size to 6400 in the next simulation. In Fig. 5(b), when the batch size (=64), the overall computational cost of the system is the lowest. Although different batch sizes eventually converge and the differences are not large. However, when the batch is too small, it does not make good use of the data in the experience pool; when the batch is too large, it leads to the use of "old data" and the training time will increase accordingly. Therefore, we will set the batch size to 64 in the next experiments. Fig. 5(c) depicts the effect of different learning rates in the actor and critic networks on experimental performance. The system has the lowest computational cost when $\alpha = 0.0001$, $\beta = 0.001$ and eventually converges to around 70. When $\alpha = 0.000001$, $\beta = 0.00001$, it makes the loss function in the algorithm change very slowly, resulting in slow convergence speed as well, and eventually stabilizes around 100, the computational cost is relatively large. A good convergence cannot be finalized when $\alpha = 0.01$, $\beta = 0.1$. As a result, in the following simulation experiments, we will use $\alpha = 0.0001$, $\beta = 0.001$ as the actor and critic networks' learning rate. As shown in Fig. 5(d), we consider the performance impact of different delay constraints on the simulated experiments. When T is large, the system's overall computing cost is small; when T is small, the cost is high. However, if the delay constraint is too large, the system may not be suitable for delay-sensitive tasks. As a result, after balancing the computational cost and delay sensitivity, we will set the delay constraint to 1.5 in the following experiments.

C. Performance Evaluation of Offloading Schemes

1) Counterparts: In this section, we compare our proposed scheme IGNITE to the four baseline schemes listed below. This includes no clustering + DDPG (NC-DDPG), k-means clustering + DDPG (K-DDPG), Gk-means clustering + DQN (GK-DQN) and Gk-means clustering + local computing(GK-Local), where our scheme is defined as GK-DDPG. And set the required hyperparameters to the same values as in our scenario.

(a) NC-DDPG: This scheme does not take into account the prediction of offloading space by DT with clustering algorithm before offloading. In comparison, it can be demonstrated that the overall cost of the system can be reduced by pre-determining the optimal offloading space in the DT network in our scheme. Furthermore, it reduces delay much more than the time required to perform the clustering step. That is, the cost of using DT for clustering is worthy.

(b) K-DDPG: This scheme uses the traditional k-means algorithm for clustering on the basis of DT, and then uses the DDPG algorithm for offloading. Comparing it to the improved clustering method in our scheme will show how much superior it performs. Our proposed Gk-means algorithm takes into account factors such as vehicle resources, distance, and social relationships in order to improve vehicle collaboration. Thus, by comparison, it can be highlighted that our clustering algorithm is more innovative.

(c) GK-DQN: This scheme employs the improved Gk-means algorithm for clustering, followed by the DQN algorithm for offloading. In comparison, the use of the DDPG algorithm in a continuous action space, such as vehicle task offloading, performs better if one wants to achieve autonomous partial offloading rather than relying on a predefined discrete offloading ratio.

(d) GK-Local: This scheme does not consider offloading the tasks for execution, but executes them all locally, and the computation fails if the local computational resources are insufficient or the delay constraint is exceeded. By contrast, the importance of task offloading techniques in VEC can be demonstrated, and it is through offloading that the needs of delay-sensitive applications can be met.

2) Evaluation Metrics: We introduce the following performance metrics to fully evaluate the simulation results.

(a) Total Computation Cost (TCC), which is the optimization target of the system, represents the weighted sum of the delay consumed and the service price paid to the service provider during offloading. The value of TCC reflects

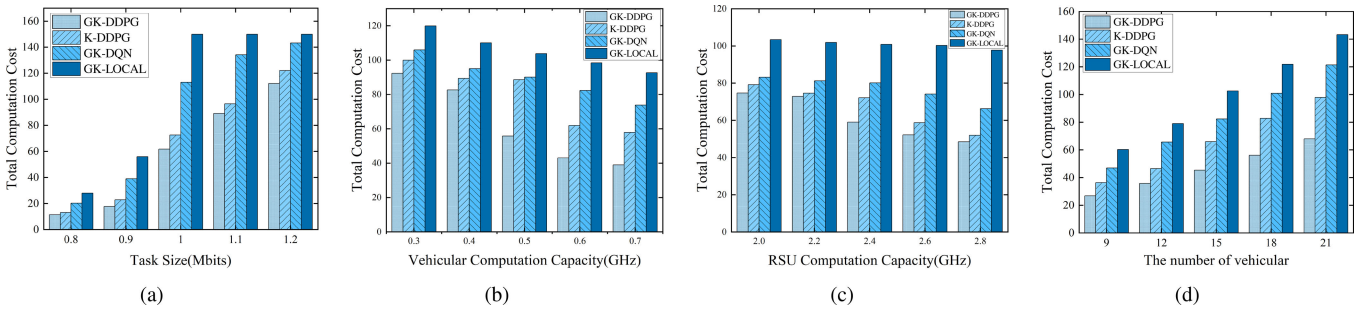

Fig. 6. (a) Total computation cost with different task size. (b) Total computation cost with different vehicular computation capacity. (c) Total computation cost with different RSU computation capacity. (d) Total computation cost with different vehicular number.

the performance of the offloading system and the training efficiency of the intelligent algorithm.

(b) Total Computation Delay (TCD) is the sum of the computational delay of all vehicles in the aggregation group. It can reflect the efficiency and communication quality of the network, the network performs better the lower its value.

(c) Average Computation Delay (ACD) represents the average computational delay of the vehicle, so it can directly reflect the efficiency of the vehicle network, and as with TCD, the smaller its value, the better.

(d) Offloading Ratio (OR) indicates the ratio of completed tasks to total tasks requested to be processed within the delay constraint, the system is more stable and reliable overall when the value is higher.

3) Performance Evaluation: In this section, we compare our scheme IGNITE with several of the above baseline schemes and analyze the comparison results. The results presented here are the average of ten independent runs of the same configuration. Figure 5 shows the performance evaluation under different parameter constraints, and Figure 6 shows the performance under different evaluation measures.

In Fig. 6(a), we compare the computational costs of the four schemes for different task sizes. The horizontal coordinate indicates the task size, and as task size increases, so do computation and communication resources. The vertical coordinate represents the total computational cost of the system, and it can be seen that as the horizontal coordinate increases, the total computational cost of these scenarios increases. However, the computation cost of the GK-DDPG scheme is obviously always lower than that of other schemes, owing to the fact that, as the task size increases and the system environment becomes more complex, GK-DDPG can obtain the best action by GKMEANS and DDPG algorithms, resulting in better performance than other schemes.

In Fig. 6(b), we contrast the computation cost of the four scenarios for different vehicle computing capabilities. This figure illustrates that the computational cost of all four scenarios tends to decrease as the computational power of the vehicle increases, and the computational cost of the GK-DDPG scenario is always the smallest regardless of the computational power of the vehicle. This is primarily due to the fact that increasing the vehicle's computational power results in more computational resources and faster processing speed, whereas local offloading cannot be executed in parallel using the server, the GK-DQN scheme cannot act in a continuous action space, and the K-DDPG scheme's clustering algorithm does not take into account factors such as resources and socialization. In conclusion, the GK-DDPG scheme performs better the others.

In Fig. 6(c), we analyze the impact of the computational power of the RSUs on the computing cost of the four methods. The findings demonstrate that as RSUs processing capacity increases, the computational cost of each strategy reduces. It suggests that boosting RSUs computational power can enhance the system's overall performance. Furthermore, the GK-DDPG scheme consistently has the lowest computational cost in both cases, while the local computation scheme has the highest. The principle reason is that our scheme utilizes an enhanced clustering algorithm, DDPG algorithm, and feedback mechanism to deliver dependable computational support in various environments, which together improve the system's overall performance and give this scheme an advantage over others.

In Fig. 6(d), we examine the effects of various vehicle counts on the overall system cost and confirm the scalability of our experiments. According to the experimental findings, the computational cost of each of the four schemes rises as the number of vehicles does, but the GK-DDPG scheme's computational cost is consistently the lowest. The key reasons are as follows: first, the performance of the offloading system may be influenced by a number of variables, whereas our suggested scheme incorporates numerous variables including the distance between vehicles on the road, resources, and social interactions when clustering. Second, as the number of vehicles grows, the tasks that must be handled grow as well, making the overall offloading environment complex. The GK-DDPG scheme, in comparison to other schemes, can obtain the best action space and state space and exhibit exceptional performance in the continuous action space.

Fig. 7(a) compares the total costs incurred by the five computation schemes discussed above. All programs are trained in 1000 epsiodes, among them, the local computing scheme is the most expensive because tasks cannot be offloaded even when the vehicle's own computing resources are insufficient, and neither the RSU nor other vehicles can provide help to it, thus leading to a high TCC for local computing. The TCC of other schemes decreased with the increase of episod, and finally the best convergence is reached when the episode number is

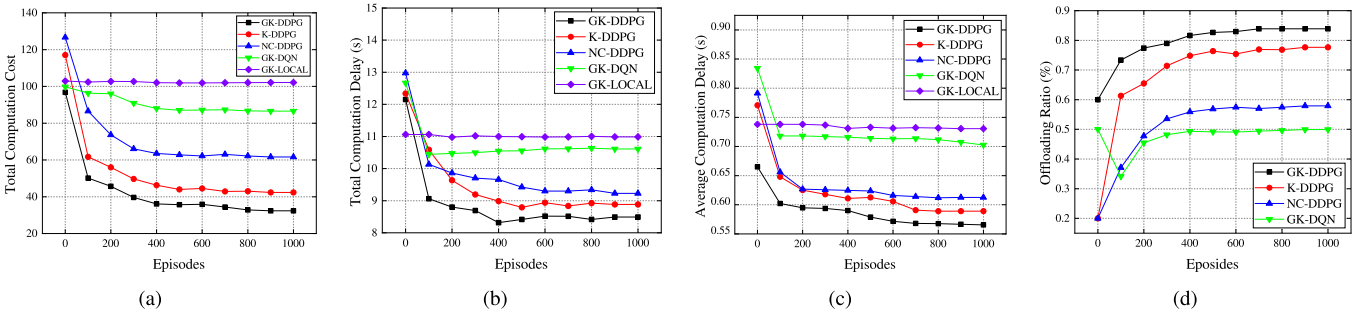

Fig. 7. Performance comparison of different schemes in terms of (a) TCC, (b) TCD, (c) ACD, and (d) OR.

around 400. This is primarily due to the fact that reinforcement learning can accumulate experience during interaction with the environment, and the effectiveness of TCC improves as the strategy improves. Compared with the other schemes, the GK-DDPG scheme can obtain a better strategy and thus achieve the best performance for the following main reasons. First, explain the reason why the GK-DDPG scheme performs better than the K-DDPG and NC-DDPG schemes, based on the improved clustering scheme, the optimal offloading space can be obtained in advance by taking into account all resource, communication and social factors, which both reduces the complexity of the state space and avoids some wrong decisions. As a result, the overall efficiency of the computation can be improved. Next, discuss the advantages of the GK-DDPG scheme over the GK-DQN scheme, the DDPG algorithm incorporates the benefits of the DQN algorithm for Q-value estimation and the single-step update of the policy gradient in Actor-Critic, and is capable of learning more efficiently on successive actions, resulting in improved performance. Therefore, it can be concluded that the GK-DDPG scheme has the best performance in the evaluation of TCC.

Fig. 7(b) and Fig. 7(c) present the comparative results of the TCD and the ACD for different schemes, respectively. The horizontal axis represents the number of episodes, and the vertical axis shows the computational delay. With the increase of the number of episodes, the computation delay of all schemes decrease until it ultimately stabilizes at 600. Among them, the GK-LOCAL scheme has the highest delay because, the servers' computation resources are not proper used while the computation resources of vehicles are limited. In contrast, the GK-DDPG scheme consistently achieve the lowest computation delay, because the GK-DDPG scheme employs an improved clustering algorithm for offloading space prediction in DT networks, thereby reducing the state space and action space. As a result, the GK-DDPG scheme can efficiently maximize the utility of the computation resource and minimize the computational delay. Furthermore, it is evident that the delay of clustering in the DT network prior to offloading is lower than the delay avoided during the offloading process. In conclusion, the proposed GK-DDPG scheme has the potential to significantly improve offloading efficiency.

Fig. 7(d) compares the offloading rates between the four schemes GK-DDPG, K-DDPG, NC-DDPG, and GK-DQN. The comparison results demonstrated that all of the schemes ultimately converge to a certain fixed value. The GK-DDPG system has both the highest offloading rate and the quickest convergence rate. On the other side, the GK-DQN scheme has a fluctuating offloading rate in the early stage and the slowest convergence rate. The main reason is that it is challenging to acquire the ideal values when we utilise the offloading ratio in action space to the DQN scheme because the high-dimensional states input to the DNN have a tendency to output greater values. Furthermore, in the proposed GK-DDPG scheme, a negative reward will be obtained as a penalty if the task is not completed within the time constraint. And our goal is to minimize the total cost, i.e., to obtain the largest possible reward. As a result, GK-DDPG can obtain the better performance. Moreover, the GK-DDPG scheme implements a feedback mechanism through which, based on the offloading results in the current cycle, the parameters of the clustering algorithm in the next cycle can be adjusted to achieve better clustering results, thereby improving the overall QoS of the system. As a result, the GK-DDPG scheme can achieve the greatest results in evaluating OR.

VI. Conclusion and Future Work

This paper presents the IGNITE partial offloading scheme in VEC systems and define the total system computational delay and service price as a combined optimization problem. The aim is to provide a real-time and effective partial offloading method for complex service requests in vehicular networks as follows. First, the vehicles on the road are clustered using the DT technique and the enhanced Gk-means clustering algorithm to find the best offloading space. Second, we offer a DRL-based partial offloading system that employs the DDPG algorithm to train the offloading strategy and address the combined optimization problem by jointly optimizing the task offloading ratio and the vehicle service price. Finally, to adapt to the constantly changing vehicle network, we investigate a feedback mechanism based on the final offloading results, by adjusting the parameters of the clustering algorithm in the DTN, a new round of clustering is carried out. The simulation results indicate that our proposed IGNITE scheme outperforms other task offloading approaches in VEC systems in terms of total system computation cost, total computation delay, and offloading success rate.

As a future direction, in order to further expand the current research, we will develop a scheme in DTN where computation and caching collaborate with each other for better research results. This will build concrete steps towards a number of interesting and challenging research lines.

REFERENCES

[1] 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.

[2] J. Zhang, L. Zhao, K. Yu, G. Min, A. Y. Al-Dubai, and A. Y. Zomaya, "A novel federated learning scheme for generative adversarial networks," *IEEE Trans. Mobile Comput.*, early access, May 22, 2023, doi: 10.1109/TMC.2023.3278668.

[3] 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.

[4] P. Liu, X. Wang, A. Hawbani, B. Hua, L. Zhao, and Z. Liu, "BETA: Beacon-based traffic-aware routing in vehicular ad hoc networks," *IEEE Trans. Intell. Transp. Syst.*, vol. 23, no. 12, pp. 24206–24219, Dec. 2022.

[5] A. Hawbani et al., "A novel heuristic data routing for urban vehicular ad hoc networks," *IEEE Internet Things J.*, vol. 8, no. 11, pp. 8976–8989, Jun. 2021.

[6] Z. Zhou et al., "Learning-based URLLC-aware task offloading for Internet of Health Things," *IEEE J. Sel. Areas Commun.*, vol. 39, no. 2, pp. 396–410, Feb. 2021.

[7] L. Zhao et al., "MESON: A mobility-aware dependent task offloading scheme for urban vehicular edge computing," *IEEE Trans. Mobile Comput.*, early access, Jun. 26, 2023, doi: 10.1109/TMC.2023.3289611.

[8] J. Shi, J. Du, J. Wang, and J. Yuan, "Deep reinforcement learning-based V2V partial computation offloading in vehicular fog computing," in *Proc. IEEE Wireless Commun. Netw. Conf. (WCNC)*, Mar. 2021, pp. 1–6.

[9] S. Yu, B. Dab, Z. Movahedi, R. Langar, and L. Wang, "A socially-aware hybrid computation offloading framework for multi-access edge computing," *IEEE Trans. Mobile Comput.*, vol. 19, no. 6, pp. 1247–1259, Jun. 2020.

[10] C. Chen, Y. Zhang, Z. Wang, S. Wan, and Q. Pei, "Distributed computation offloading method based on deep reinforcement learning in ICV," *Appl. Soft Comput.*, vol. 103, May 2021, Art. no. 107108.

[11] J. Lai et al., "Deep learning based traffic prediction method for digital twin network," *Cogn. Comput.*, May 2023.

[12] Z. Zhou et al., "Secure and latency-aware digital twin assisted resource scheduling for 5G edge computing-empowered distribution grids," *IEEE Trans. Ind. Informat.*, vol. 18, no. 7, pp. 4933–4943, Jul. 2022.

[13] T. Liu, L. Tang, W. Wang, Q. Chen, and X. Zeng, "Digital-twin-assisted task offloading based on edge collaboration in the digital twin edge network," *IEEE Internet Things J.*, vol. 9, no. 2, pp. 1427–1444, Jan. 2022.

[14] E. Zhang, L. Zhao, N. Lin, W. Zhang, A. Hawbani, and G. Min, "Cooperative task offloading in cybertwin-assisted vehicular edge computing," in *Proc. IEEE 20th Int. Conf. Embedded Ubiquitous Comput. (EUC)*, Dec. 2022, pp. 66–73.

[15] K. Zhang, J. Cao, and Y. Zhang, "Adaptive digital twin and multi-agent deep reinforcement learning for vehicular edge computing and networks," *IEEE Trans. Ind. Informat.*, vol. 18, no. 2, pp. 1405–1413, Feb. 2022.

[16] Y. Dai and Y. Zhang, "Adaptive digital twin for vehicular edge computing and networks," *J. Commun. Inf. Netw.*, vol. 7, no. 1, pp. 48–59, Mar. 2022.

[17] W. Sun, H. Zhang, R. Wang, and Y. Zhang, "Reducing offloading latency for digital twin edge networks in 6G," *IEEE Trans. Veh. Technol.*, vol. 69, no. 10, pp. 12240–12251, Oct. 2020.

[18] Y. Ouyang, "Task offloading algorithm of vehicle edge computing environment based on dueling-DQN," *J. Phys., Conf. Ser.*, vol. 1873, no. 1, Apr. 2021, Art. no. 012046.

[19] 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.

[20] 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.

[21] H. Peng and X. Shen, "Multi-agent reinforcement learning based resource management in MEC- and UAV-assisted vehicular networks," *IEEE J. Sel. Areas Commun.*, vol. 39, no. 1, pp. 131–141, Jan. 2021.

[22] H. Liao et al., "Cloud-edge-device collaborative reliable and communication-efficient digital twin for low-carbon electrical equipment management," *IEEE Trans. Ind. Informat.*, vol. 19, no. 2, pp. 1715–1724, Feb. 2023.

[23] L. Zhao, C. Wang, K. Zhao, D. Tarchi, S. Wan, and N. Kumar, "INTERLINK: A digital twin-assisted storage strategy for satellite-terrestrial networks," *IEEE Trans. Aerosp. Electron. Syst.*, vol. 58, no. 5, pp. 3746–3759, Oct. 2022.

[24] L. Zhao, Z. Bi, A. Hawbani, K. Yu, Y. Zhang, and M. Guizani, "ELITE: An intelligent digital twin-based hierarchical routing scheme for softwarized vehicular networks," *IEEE Trans. Mobile Comput.*, vol. 22, no. 9, pp. 5231–5247, Sep. 2023.

[25] Y. Dai, K. Zhang, S. Maharjan, and Y. Zhang, "Deep reinforcement learning for stochastic computation offloading in digital twin networks," *IEEE Trans. Ind. Informat.*, vol. 17, no. 7, pp. 4968–4977, Jul. 2021.

[26] W. Wang et al., "Vehicle trajectory clustering based on dynamic representation learning of Internet of Vehicles," *IEEE Trans. Intell. Transp. Syst.*, vol. 22, no. 6, pp. 3567–3576, Jun. 2021.

[27] A. Bozorgchenani, F. Mashhadi, D. Tarchi, and S. A. S. Monroy, "Multi-objective computation sharing in energy and delay constrained mobile edge computing environments," *IEEE Trans. Mobile Comput.*, vol. 20, no. 10, pp. 2992–3005, Oct. 2021.

[28] W. Feng et al., "Latency minimization of reverse offloading in vehicular edge computing," *IEEE Trans. Veh. Technol.*, vol. 71, no. 5, pp. 5343–5357, May 2022.

[29] G. Ma, X. Wang, M. Hu, W. Ouyang, X. Chen, and Y. Li, "DRL-based computation offloading with queue stability for vehicular-cloud-assisted mobile edge computing systems," *IEEE Trans. Intell. Vehicles*, vol. 8, no. 4, pp. 2797–2809, Apr. 2023.

[30] H. Lu, C. Gu, F. Luo, W. Ding, and X. Liu, "Optimization of lightweight task offloading strategy for mobile edge computing based on deep reinforcement learning," *Future Gener. Comput. Syst.*, vol. 102, pp. 847–861, Jan. 2020.

[31] S. Lloyd, "Least squares quantization in PCM," *IEEE Trans. Inf. Theory*, vol. IT-28, no. 2, pp. 129–137, Mar. 1982.

[32] J. E. Anderson, "The gravity model," *Annu. Rev. Econ.*, vol. 3, no. 1, pp. 133–160, 2011.

[33] J. Liu et al., "RL/DRL meets vehicular task offloading using edge and vehicular cloudlet: A survey," *IEEE Internet Things J.*, vol. 9, no. 11, pp. 8315–8338, Jun. 2022.

[34] X. Zhu, Y. Luo, A. Liu, N. N. Xiong, M. Dong, and S. Zhang, "A deep reinforcement learning-based resource management game in vehicular edge computing," *IEEE Trans. Intell. Transp. Syst.*, vol. 23, no. 3, pp. 2422–2433, Mar. 2022.

[35] C. Qiu, Y. Hu, Y. Chen, and B. Zeng, "Deep deterministic policy gradient (DDPG)-based energy harvesting wireless communications," *IEEE Internet Things J.*, vol. 6, no. 5, pp. 8577–8588, Oct. 2019.

Liang Zhao (Member, IEEE) received the Ph.D. degree from the School of Computing, Edinburgh Napier University, in 2011. He is currently a Professor with Shenyang Aerospace University, China. Before joining Shenyang Aerospace University, he was an Associate Senior Researcher with Hitachi (China) Research and Development Corporation from 2012 to 2014. He was also a JSPS invitational Fellow in 2023. He was listed as Top 2% of scientists in the world by Standford University in 2022. His research interests include ITS, VANET, WMN, and SDN. He has published more than 150 articles. He was a recipient of the Best/Outstanding Paper Awards at 2015 IEEE IUCC, 2020 IEEE ISPA, 2022 IEEE EUC, and 2013 ACM MoMM. He served as the Chair of several international conferences and workshops, including the Steering Co-Chair for 2022 IEEE BigDataSE, the Program Co-Chair for 2021 IEEE TrustCom, the Program Co-Chair for 2019 IEEE IUCC, and the Founder for NGDN Workshop (2018–2022). He is an Associate Editor of *Frontiers in Communications and Networking* and *Journal of Circuits Systems and Computers*. He is/has been a Guest Editor of IEEE TRANSACTIONS ON NETWORK SCIENCE AND ENGINEERING and *Journal of Computing* (Springer).

Zijia Zhao is currently pursuing the M.S. degree in computer science with Shenyang Aerospace University. Her research interests mainly include vehicle edge computing, computation offloading, and digital-twins.

Enchao Zhang is currently pursuing the M.S. degree with Shenyang Aerospace University. His research interests include mobile edge computing, computation offloading, digital-twins, and resource allocation.

Ammar Hawbani received the B.S., M.S., and Ph.D. degrees in computer software and theory from the University of Science and Technology of China (USTC), Hefei, China, in 2009, 2012, and 2016, respectively. He is currently an Associate Professor in networking and communication algorithms with the School of Computer Science and Technology, University of Science and Technology of China, China. From 2016 to 2019, he was a Post-Doctoral Researcher with the School of Computer Science and Technology, USTC. His research interests include the IoT, WSNs, WBANs, WMNs, VANETs, and SDN.

Ahmed Y. Al-Dubai (Senior Member, IEEE) received the Ph.D. degree in computing from the University of Glasgow in 2004. He is currently a Professor in networking and communication algorithms with the School of Computing, Edinburgh Napier University, U.K. His research interests include communication algorithms, mobile communication, the Internet of Things, and future internet. He received several international awards.

Zhiyuan Tan (Senior Member, IEEE) received the Ph.D. degree from the University of Technology Sydney, Australia, in 2014. He was a Post-Doctoral Researcher with the University of Twente, The Netherlands, from 2014 to 2016. He is currently an Associate Professor with the School of Computing, Engineering and the Built Environment, Edinburgh Napier University, U.K. He is a member of ACM. His current research interests include cyber security, machine learning, cognitive computing, and intelligent transportation. His research has been funded by the Royal Society (U.K.), the Scottish Informatics and Computer Science Alliance, the ENU Development Trust, the Commonwealth Scientific and Industrial Research Organization (Australia). He is an Associate Editor of IEEE TRANSACTIONS ON RELIABILITY and an Academic Editor of *Security and Communication Networks*.

Amir Hussain received the B.Eng. and Ph.D. degrees in electronic and electrical engineering from the University of Strathclyde, Scotland, U.K., in 1992 and 1997, respectively. He is currently a Professor with the School of Computing, Edinburgh Napier University (ENU), Scotland, U.K. Following post-doctoral and senior academic positions with the West of Scotland (1996–1998), Dundee (1998–2000) and Stirling Universities (2000–2018) respectively, he joined Edinburgh Napier University as the founding Head of the Cognitive Big Data and Cybersecurity (CogBiD) Research Laboratory and the Centre for AI and Data Science. His research interests include cognitive computation, machine learning, and computer vision.