IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received 3 December 2023, accepted 20 December 2023, date of publication 1 January 2024, date of current version 24 January 2024.

Digital Object Identifier 10.1109/ACCESS.2023.3349029

RESEARCH ARTICLE

Computing Resource Allocation Strategy Based on Cloud-Edge Cluster Collaboration in Internet of Vehicles

XIANHAO SHEN, LI WANG, PANFENG ZHANG, XIAOLAN XIE, YI CHEN, AND SHAOFANG LU

College of Information Science and Engineering, Guilin University of Technology, Guilin 541006, China
Guangxi Laboratory of Embedded Technology and Intelligent System, Guilin University of Technology, Guilin 541006, China

Corresponding author: Panfeng Zhang (panf_zhang@glut.edu.cn)

This work was supported in part by the National Natural Science Foundation of China under Grant 62362017, Grant 62341117, and Grant 662262011; and in part by the Natural Science Foundation of Guangxi under Grant AB23075175.

ABSTRACT Edge computing plays a crucial role in the field of the Internet of Vehicles (IoV), meeting the resource and latency requirements of time-sensitive vehicle applications. However, the emergence of numerous compute-intensive and latency-sensitive applications, such as augmented reality and autonomous driving, has led to a situation where traditional edge computing architectures cannot meet the increasing application demands of the IoV. This paper extends the paradigm of vehicular edge computing to a collaborative cloud-edge cluster resource provisioning framework. Integrating compute resources from multiple Edge Service Providers (ESP) and the cloud enables horizontal and vertical collaborative computation offloading among service nodes. To facilitate resource sharing among different ESPs, we introduce a dynamic pricing model and utilize software-defined networking (SDN) to tackle this scenario's complex resource management challenges. Furthermore, with the optimization objectives of minimizing task computation latency and maximizing the profits of ESPs, we establish a mathematical model. Before resource allocation, we employ a clustering algorithm to determine initial offloading decisions, reducing the dimensionality of the action space. Subsequently, we employ the Double Deep Q-Network (DDQN) algorithm to achieve a rational allocation of compute resources. Simulation results demonstrate that compared to the Deep Q-Network (DQN) algorithm and greedy strategy, the proposed approach reduces latency by 18.18% and 34.85%, respectively, while increasing the profits of edge service providers by 16.25% and 33.33%, respectively.

INDEX TERMS Internet of Vehicles, mobile edge computing, resource allocation, reinforcement learning, task offloading.

I. INTRODUCTION

With the advent of the Internet of Everything era, Internet of Things technology has been widely used in many fields, including health care, smart home and transportation. In the area of transportation, the Internet of Vehicles (IoV) is a typical application form that has received significant attention from academia in recent years. Vehicular networks have been widely recognized as having great potential to improve driving safety and traffic mobility for both manual and autonomous vehicles [1]. However, with the emergence of various in-vehicle applications, especially delay-sensitive and computation-intensive applications, IoV systems face severe challenges due to the limited resources of spectrum and computing [2].

Cloud computing is widely regarded as a powerful solution to address the shortage of computing resources in vehicular networks, as it provides a more abundant pool of computing resources. However, cloud centers are typically distant from the vehicle terminals, making it challenging to meet the latency-sensitive demands of vehicles. In this context, edge computing has gained increasing attention. Many researchers

The associate editor coordinating the review of this manuscript and approving it for publication was Mehdi Sookhak.

in the field of IoV have extensively studied vertical offloading [3], [4], transferring the computational tasks of vehicles to Mobile Edge Computing (MEC) servers within the edge network, enabling data and applications to be closer to the vehicles and achieving real-time processing of in-vehicle computational tasks [5], [6]. This approach satisfies the requirements for expanding vehicular computing capabilities and compensates for the shortcomings of long delays caused by remote cloud computing.

In fact, in congested urban traffic, the resource supply-demand conflict intensifies when many vehicle users simultaneously engage in task offloading and decision-making. Specifically, during the task offloading process, edge servers need to continuously obtain the latest network information, such as vehicular positions and traffic conditions. Considering the high vehicle density, edge servers require frequent communication to coordinate data and resource allocation. However, due to the limitations of computational resources, such frequent communication may overload the servers, leading to delays in responding to vehicular requests or even communication congestion, further exacerbating the performance degradation of IoV. Therefore, it is a research problem worthy of in-depth investigation to explore how to effectively address the growing vehicular tasks with limited computational resources through collaborative cooperation among the cloud, edge, and terminal devices to achieve complex task offloading and resource allocation decisions.

A feasible solution has been proposed in recent research. It involves vehicles offloading their computational tasks to edge servers and then horizontally expanding resources through collaboration between edge servers or fog nodes. When the capacity of an edge server is insufficient, tasks can be migrated to auxiliary edge servers to cope with the growing demands of vehicles. This collaborative expansion approach can potentially improve resource utilization and the performance of IoV. However, it still faces some challenges:

(i) In real scenarios, since each Edge Service Provider (ESP) tends to establish a private edge environment to provide services for users, the horizontal expansion of edge resources is often limited to the private edge environment supplied by each ESP. The lack of cooperation among ESPs limits the edge network's cooperative advantages and system performance. In addition, users are limited to a single-edge environment and cannot enjoy services and resources across edge environments, resulting in low resource utilization and potential resource idleness.
(ii) In a multi-ESP IoV architecture, effectively managing and optimizing computational resources to achieve task offloading and resource allocation across two layers is a challenge that needs to be addressed.

In conclusion, this study addresses the limitations of vertical offloading by constructing a model for resource sharing and service among multiple ESPs. We achieve horizontal resource expansion at the edge layer by integrating the private edge environments provided by multiple ESPs into a shared resource pool. Additionally, we integrate the abundant computing resources of remote clouds into the shared resource pool through vertical integration, which can supplement the areas that edge nodes may not be able to serve [7]. Typically, in the systems of IoV, the network and resource states exhibit dynamic variations, necessitating the handling of complex scenarios such as interactions between vehicles and infrastructure, and dynamic changes in network topology [8], [9], to deal with the complexity of resource management in this scenario, we introduce software-defined networking (SDN) to perceive the network state and collect device information from a global perspective, enabling vehicles to choose suitable access methods for task offloading. We propose a joint optimization strategy for task offloading and resource allocation, considering the latency requirements of different tasks. Before using the Double Deep Q-Network (DDQN) algorithm for resource allocation, we employ clustering algorithms to segment the offloading modes of vehicles, that is, choosing between local or Vehicle-to-Infrastructure (V2I) offloading mode, thereby reducing the dimension of the action space and improving resource allocation efficiency. This effectively reduces task latency and energy consumption, significantly enhancing the service experience of vehicle users while maximizing the profit of ESP. The main contributions of this study are as follows.

(i) This paper proposed a Cloud-edge clusters collaboration IoV architecture with integrated SDN, for the proposed architecture can provide horizontal and vertical offloading between service nodes. At the same time, we analyzed the response delay of processing task requests and the profit of ESP when processing tasks under this architecture. Considering the time-varying resources in the system, a dynamic pricing model is introduced to adjust the price dynamically to promote cooperation among ESPs, and to avoid high computing latency caused by resource constraints.
(ii) In this scenario, to improve the profit of ESP while ensuring the task delay performance, the optimization problem is transformed into the optimal task offloading and computing resource allocation problem, and it is modelled as a Markov decision process. We proposed a joint task offloading and resource allocation algorithm that combines K-means ++ and DDQN algorithm (KDDQN-JOARA). Where K-means ++ is used to determine the initial offloading node, and the DDQN algorithm is used to allocate computing resources reasonably.
(iii) A series of simulation experiments are conducted to verify the necessity of considering cluster cooperation and the advantages of the proposed approach in terms of delay and profit.

The rest of the paper is organized as follows: we review the related work in Section II. Section III presents the network and system model and defines and formulate the delay

minimization of task and the profit maximization of ESP problem. In Section IV, we propose the KDDQN-JOARA to solve this problem. In Section V, we give the experimental values and discussion and analyze the performance enhancement of the proposed scheme. In Section VI, we conclude with a discussion of the paper. The conclusions of our study are presented in Section VII.

II. RELATED WORK

To address the task offloading and resource allocation problem in the context of MEC-based IoV, Fan et al.) [10] proposed a vehicle-road cooperative communication-based algorithm for joint task offloading and resource allocation, aiming to minimize overall latency and energy consumption. Ren et al. [11] investigated the mutual communication and computation resource allocation problem under cloud and edge computing coordination. Yang et al. [12] studied efficient task offloading strategies in vehicular edge computing networks, considering optimal decisions for offloading time selection, communication, and computation resource allocation. Zhao et al. [13] proposed a distributed computation offloading and resource allocation algorithm in cloud-assisted scenarios. Most of the research above work focuses on achieving global resource management through a vertical collaborative architecture. However, in the context of IoV, when there is a surge in task demand, considering only V2I offloading can lead to server overload, while relying solely on Vehicle-to-Vehicle (V2V) offloading may result in communication interruptions or difficulties in establishing connections. Even though some studies have attempted to integrate both types of offloading frameworks, it remains challenging to alleviate the resource constraints in IoV.

Some researchers have addressed the limitations of resource constraints by employing a hybrid architecture that combines both horizontal and vertical offloading. In this approach, vehicles offload computational tasks to edge servers, and then collaborative horizontal resource scaling is achieved among the edge servers or fog nodes. For instance, Al-Hammadi et al. [14] primarily investigated a MEC network with a complete offloading strategy, utilizing collaborative edge servers for task migration when the MEC network becomes overloaded. Phan et al. [15] proposed a dynamic offloading service among fog nodes in a fog computing system, aiming to select the optimal offloading node and assist with offloading paths. However, from a cost perspective, edge servers may belong to different ESPs, and deploying servers by ESPs incurs costs related to network infrastructure coverage, energy consumption, maintenance, and management. ESPs are not likely to provide resources to others without compensation. Existing research has paid limited attention to the integration of resources from multiple ESPs and remote clouds into a unified resource pool for collaborative computation. Therefore, this paper devises a rational pricing model to ensure collaboration among ESPs and integrate the private edge resources offered by different ESPs. A resource-sharing pool is created at the edge layer to facilitate collaborative resource-sharing across different ESPs.

However, despite the expansion of computing capabilities in IoV systems brought by this hybrid offloading architecture, it may lead to increased complexity in resource management. SDN can effectively address this challenge. Cao et al. [16] proposed a novel 5G vehicular networking architecture based on fog computing and SDN to meet the requirements of IoV, and they tackled the resource allocation problem in the fog-cloud system within this architecture. Phan et al. [15] presented a dynamic fog-to-fog offloading scheme for SDN-based fog computing systems. Zhang et al. [17] constructed a new SDN-assisted MEC network architecture for IoV, aiming to minimize overall latency and energy consumption through joint optimization of offloading strategies and resource allocation. Therefore, introducing an SDN controller on top of the hybrid architecture enables centralized management of vehicles and resource information. Based on the information collected from the network and vehicles, an intelligent selection of the most suitable offloading nodes can be made, facilitating the management of the offloading process.

In the IoV environment, inappropriate task offloading and resource allocation strategies can lead to high response latency and low resource utilization. Traditional algorithms, which primarily focus on optimizing the performance of relatively stable quasi-static systems, may not be suitable for IoV systems due to various uncertainties, such as real-time requirements of computational tasks and dynamic changes in resource demands, both of which impact the performance of task offloading.

To tackle these challenges, Liu et al. [18] proposed a computation offloading and resource allocation algorithm based on DDQN to minimize the latency and energy consumption of data fusion computational tasks. Liu et al. [19] introduced a Deep Reinforcement Learning algorithm that allocates resources using vehicles as edge devices to provide computing services to nearby users. Huang et al. [20] presented a Deep Q-Network (DQN) based algorithm to address the complex problem of jointly optimizing task offloading and bandwidth allocation in mobile edge computing networks. Motivated by the above research, this paper utilizes the K-means++ algorithm to initially classify task offloading models, improving the efficiency of offloading and resource allocation strategies. Additionally, the DDQN algorithm is employed to obtain optimal offloading decisions and resource allocation strategies for tasks within classified edge node clusters.

The current research on task offloading and resource allocation strategies in IoV scenarios is relatively mature. However, there are still blind spots in some particular scenarios. This paper constructs a hybrid horizontal and vertical offloading framework in an SDN-integrated IoV scenario. It adopts a reasonable resource pricing mechanism to promote the collaboration between ESPs. Finally, the task offloading and resource allocation problem under the framework was

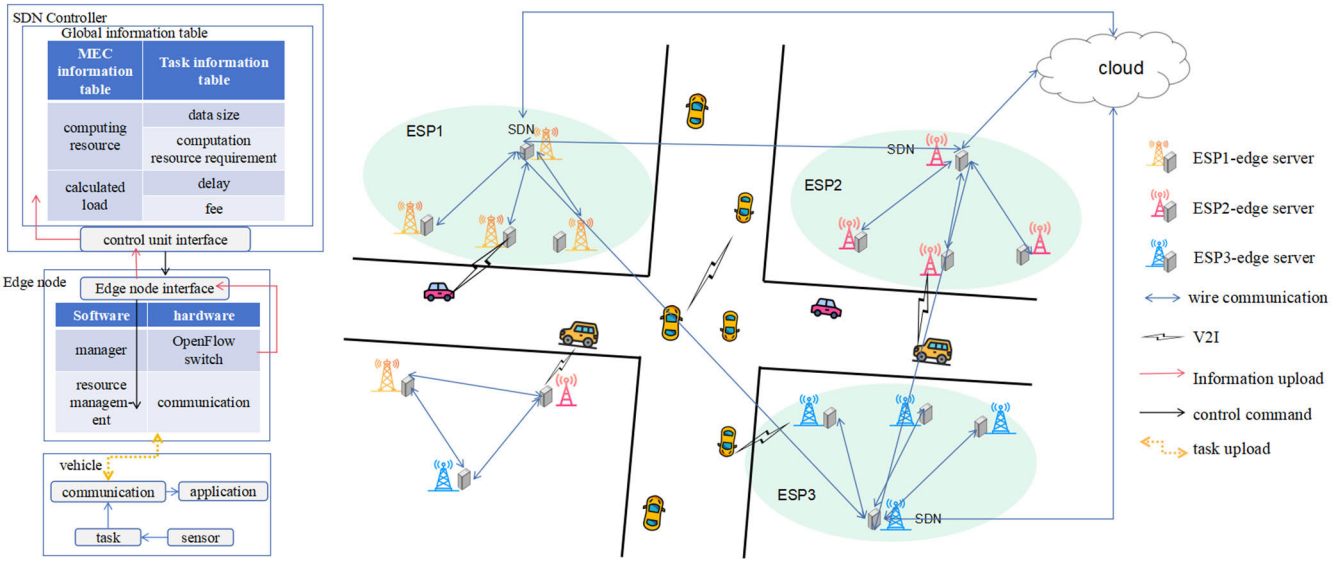


FIGURE 1. Cloud-edge clusters collaboration IoV architecture with integrated SDN.

established, the task completion time and the profit of ESP were jointly optimized, and the Deep Reinforcement Learning (DRL) algorithm obtained the optimal task offloading decision and resource allocation strategy.

III. SYSTEM OVERVIEW AND PROBLEM FORMULATION

A. SYSTEM ARCHITECTURE

As illustrated in Fig. 1, we consider the Cloud-edge clusters collaboration IoV architecture with integrated SDN. There are multiple ESPs in the architecture, and the edge nodes of each ESP are coordinated and controlled by their own deployed SDN, aiming to reduce the system delay and improve the overall performance. SDN divides the IoV system into data and control layers through software definition and virtualization technology. The physical communication channels of the control layer and the physical communication channels of the data layer are independent of each other. The SDN controller broadcasts the global state, including channel state information and available resources. After receiving the offloading request from the vehicle, SDN looks for the best solution in the control layer, including offloading decision and resource allocation, and then sends control instructions. The data layer transfers data according to the instructions of the control layer. Communication between SDNs deployed by different ESPs is established through standard network communication protocols (such as TCP/IP) or interfaces between specific SDN controllers. This allows for the exchange of resource information from other ESPs in the cluster, which can be used for collaborative computation and further optimization of the current resource allocation strategies.

The detailed description is as follows: suppose the cluster cooperation scenario consists of multiple ESPs, the set of ESPs is K= {1, 2, 3,...k}, each ESP deploys m edge servers in its managed area to provide services for IoV users. The edge servers deployed by ESP_k are denoted as $E_k = \{E_{k,1}, E_{k,2}, E_{k,3}, \ldots E_{k,m}\}$, and the vehicles served are denoted by $V = \{v_1, v_2, v_3, \ldots, v_n\}$, and each ESP is equipped with an SDN controller, in which the collection and management of vehicular data and information of all edge nodes are completed, and the task offloading and resource allocation are uniformly scheduled. Assuming that each vehicle sends only one computing task request per time slot t, and all vehicles are travelling at a constant speed, the computing task requests issued by vehicle users can be represented by a quadruple as follows: $W_i = \{t_i^{\max}, C_i, d_i, fee_i\}$, where $t_i^{\max}$ represents the maximum tolerable execution delay of the task, C_i represents the amount of computation required to execute the task, d_i represents the data size required to upload the computation task, fee_i represents the cost price that IoV users are willing to pay to compute task W_i. Define X as the task offloading decision, $X = \{x_1, x_2, x_3 \ldots x_n\}$, where $x_i = \{x_i^l, x_i^e, x_i^c\}$, $x_i^l, x_i^e, x_i^c \in \{0, 1\}$ and $x_i^l + x_i^e + x_i^c = 1$. x_i^l is executed locally in the vehicle, x_i^e is to offload the task to the edge server provided by ESP, and x_i^c is to offload the task to the remote cloud for execution. The illustration of main notations is summarized in Table 1.

B. COMMUNICATION MODEL

In this paper, when the resources of vehicle cannot handle the computing tasks, the computing tasks are offloaded to the local edge servers deployed by ESP_k or the edge resources provided by other ESPs in the cluster leased by ESP_k. The orthogonal frequency division multiple access (OFDMA) technique is used for wireless communication between the vehicle and the MEC server to avoid severe interference.) [10]. Therefore, according to Shannon's formula, the data transmission rate when vehicle n offloads tasks to the edge

server m for processing is given by

$$R_n^m = Blog_2(1 + \frac{p_n h_{n,m}}{\delta^2}) \tag{1}$$

where B represents the total bandwidth, let δ^2 denote the noise power, $h_{n,m}$ is the channel gain between vehicle n and edge server m, and p_n is the transmit power of vehicle n. In addition, we use R^{E2E} to denote the transmission rate between edge servers, and R^{E2C} to denote the transmission rate between edge servers and cloud servers.

C. DYNAMIC PRICING MODEL

When the resource-demand vehicle offloads tasks to the edge server, it will occupy the computing resources of the edge server. Considering the limited resources of the edge server deployed by a single ESP, it will rent resources from other ESPs in the cluster scenario at a certain fee when the resource usage is at its peak. ESP in the cluster should dynamically adjust the unit resource price with the change in resource utilization and its load status.

In this paper, the computing resources of the edge server are dynamically priced according to user demand and computing resource load, and the price of unit computing resources purchased by the resource demander ESP_k is given by

$$U_k = U_{s,m}(\frac{\pi_k}{1 - L_{s,m}}) \tag{2}$$

where $U_{s,m}$ is the initial unit cost price of the resources of edge server m set by ESP_s, $L_{s,m}$ is the current resource load percentage of edge server m deployed by ESP_s, which is calculated as follows:

$$L_{s,m} = \frac{total_{s,m} - res_{s,m}}{total_{s,m}} \tag{3}$$

where $total_{s,m}$ represents the total amount of computing resources of edge server m provided by ESP_s, and $res_{s,m}$ represents the remaining amount of its current available resources.

π_k is a willingness value of resource demand, the larger π_k is, the higher demand willingness of ESP_k. Use $\pi_k \in [0, 1]$ to denote the current degree of computational demand of ESP_k, where π_k is calculated as follows.

$$\pi_k = \frac{\sum_{i=1}^{l} C_{k,i}}{\sum_{e=1}^{m} f_{k,e}} \tag{4}$$

where $C_{k,i}$ represents the computing demand of task W_i served by ESP_k, and $f_{k,e}$ represents the available computing resources of the e^{th} edge server of ESP_k.

D. COMPUTATION MODEL

1) LOCAL COMPUTING

Since the tasks are executed locally, the latency only needs to account for the computation latency, denoted by t_i^l, given by

$$t_i^l = \frac{C_i}{f_i^l} \tag{5}$$

TABLE 1. Notation definitions.

Notation	Definition
ESP_k	edge service providers K
E_k	the edge servers deployed by ESP_k
W_i	computation task of vehicles
N	number of vehicles
t_i^{max}	the maximum tolerable execution delay of the task
C_i	the amount of computation required to execute the task
d_i	the data size required to upload the computation task
fee_i	the cost price that Vehicles are willing to pay to compute task
X	the task offloading decision
R_n^m	uplink rate per vehicles
R^{E2E}	the transmission rate between edge servers
R^{E2C}	the transmission rate between edge servers and cloud servers
f_{mec}	computing resources of the edge server
τ	resource allocation vector
$A_{k,i}$	local edge server computing or renting another edge server computing
π_k	the degree of demand for rented resources
P_{oe}	the device power of the edge server which ESP_k leased
P_{mec}	the device power of the ESP_k local edge server
P_c	the device power of the cloud server

where C_i represents the computing demand of task W_i, and f_i^l is the local computing power provided by the vehicle. The corresponding energy consumption is calculated as follows:

$$E_i^l = k_l C_i (f_i^l)^2 \tag{6}$$

where k_l is an energy consumption parameter depending on the chip architecture [21].

2) EDGE CLUSTER COMPUTING

We denote the computational resource allocation vector in time slot t as$\tau = \{\alpha_1, \alpha_2, \alpha_3 \ldots \alpha_n\}$, where $\alpha_i \in [0, 1]$ is the computing resource allocated by the edge server to complete the task W_i. Since the total computing resources of the MEC server are f_{mec}, $\alpha_i f_{mec}$ represents the computing resources allocated to vehicle user i in time slot t. When a higher proportion of computing resources is assigned to a particular vehicle user, the task execution time becomes more minor, but the energy consumption may increase.

If vehicle i decides to perform the task W_i by offloading. There are two options. The task can be executed directly at the local edge node, when the local edge resources of ESP_k are sufficient. If the load is too high and the resources are insufficient, the task can be executed by using the edge resources provided by other ESPs by renting resources. Therefore, the

decision vector $A_{k,i}$ is used to represent where ESP_k should execute task W_i, where $A_{k,i} = 1$ means the execution is performed at the local deployed edge node of ESP_k, and $A_{k,i} = 0$ indicates the task is offloaded to the edge server provided by other ESPs. Since the data size of the calculated result is much smaller than the input data, the backhaul delay of the calculated result can be ignored.

(1) When vehicle i offloads task W_i to the local deployment server of ESP_k to perform. The total delay includes transmission and computation delays. Where the transmission delay is calculated as follows:

$$t_{i,e}^{tran1} = \frac{d_i}{R_n^m} \tag{7}$$

The computation delay is defined as:

$$t_{i,e}^{comp} = \frac{C_i}{\alpha_i f_{mec}} \tag{8}$$

The energy consumption of offloading to local edge server of ESP_k to execute task W_i is given by

$$E_i^e = E_{i,e}^{tran1} + E_{i,e}^{comp} = p_n t_{i,e}^{tran1} + P_{mec}\alpha_i t_{i,e}^{comp} \tag{9}$$

where P_{mec} represents the device power when the local edge server of ESP_k uses all computing resources to calculate the task [22].

Assuming that the unit energy consumption cost of executing a task at the local edge server of ESP_k is θ_e, the cost of offloading a task to the local edge server of ESP_k is calculated as follows:

$$C_i^e = \theta_e E_i^e = \theta_e(E_{i,e}^{tran1} + E_{i,e}^{comp}) \tag{10}$$

Assuming that the unit resource cost of the local edge server of ESP_k is R_i^e, The profit of ESP_k from executing the task on the local server is calculated as follows:

$$Q_i^e = fee_i - (C_i^e + R_i^e \alpha_i f_{mec}) \tag{11}$$

(2) Task is executed by using the edge resources provided by other ESPs. Therefore, the total delay when using other ESPs resources for calculation is calculated as follows:

$$\begin{aligned} t_i^{oe} &= (t_{i,e}^{tran1} + t_{i,oe}^{tran2} + t_{i,oe}^{comp}) \\ &= (\frac{d_i}{R_n^m} + \frac{d_i}{R^{E2E}} + \frac{C_i}{\alpha_i f_{oe}}) \end{aligned} \tag{12}$$

where $\alpha_i f_{oe}$ represents the computing resources of other ESPs rented by ESP_k to perform task W_i, $t_{i,e}^{tran1}$ is the transmission delay for the vehicle to upload the task W_i to the connected edge node, $t_{i,oe}^{tran2}$ is the transmission delay for the RSU to transmit the task to the edge server provided by other ESPs in the selected cluster, $t_{i,oe}^{comp}$ is the computation delay.

We ignore the energy consumption of offloading tasks from edge server k to cooperative edge server s, because the energy consumption between edge servers is much smaller than the energy consumption between vehicles and edge servers [14]. Therefore, the total energy consumption of offloading tasks is expressed as follows:

$$\begin{aligned} E_i^{oe} &= E_{i,e}^{tran1} + E_{i,oe}^{comp} \\ &= p_n t_{i,e}^{tran1} + t_{i,oe}^{comp}\alpha_i P_{oe} \end{aligned} \tag{13}$$

where P_{oe} is the device power of the edge server which ESP_k leased.

Assuming θ_{oe} is the unit energy cost of ESP_k when it offloads tasks to the edge nodes provided by other ESPs in the cluster, the total energy cost is as follows:

$$C_i^{oe} = \theta_{oe} E_i^{oe} = \theta_{oe}(E_{i,e}^{tran1} + E_{i,oe}^{comp}) \tag{14}$$

Therefore, the total cost of ESP_k renting resources to perform task W_i is defined as:

$$Z_i^{oe} = \frac{U_{s,m}\pi_k \alpha_i f_{oe}}{1 - L_{s,m}} \tag{15}$$

Then the profit of ESP_k by leasing cluster resources to compute the task W_i is given by

$$Q_i^{oe} = fee_i - (C_i^{oe} + Z_i^{oe}) \tag{16}$$

Therefore, the total delay to execute the task W_i by offloading to the edge server is given by

$$t_i^{mec} = A_{k,i} t_i^{\mathrm{e}} + (1 - A_{k,i}) t_i^{\mathrm{oe}} \tag{17}$$

The profit of ESP_k executing task W_i is as follows:

$$Q_i^{mec} = A_{k,i} Q_i^e + (1 - A_{k,i}) Q_i^{oe} \tag{18}$$

3) CLOUD COMPUTING

The vehicle will offload its computing tasks to the cloud server through the core network. Therefore, the upload time to transfer the input data from the RSU to the cloud server must be considered. and the latency to perform the task through the cloud server is expressed as:

$$\begin{aligned} t_i^c &= t_{i,e}^{tran1} + t_{i,c}^{tran3} + t_{i,c}^{comp} \\ &= \frac{d_i}{R_n^m} + \frac{d_i}{R^{E2C}} + \frac{C_i}{\alpha_i f_c} \end{aligned} \tag{19}$$

where the $\alpha_i f_c$ represents the cloud server resources rented by ESP_k to perform tasks, and $t_{i,c}^{tran3}$ represents the upload time of task transmission from RSU to cloud server. $t_{i,c}^{comp}$ is the computation delay.

The total energy consumption is denoted by E_i^c is given by

$$\begin{aligned} E_i^c &= E_i^{tran1} + E_{i,c}^{comp} \\ &= p_n t_{i,e}^{tran1} + P_c \alpha_i t_{i,c}^{comp} \end{aligned} \tag{20}$$

where P_c indicates the device power of the cloud server.

Assuming θ_c is the unit energy cost when the vehicle chooses the offloading task W_i to the cloud, the total energy cost is given by

$$C_i^c = \theta_c E_i^c = \theta_c(E_i^{tran1} + E_{i,c}^{comp}) \tag{21}$$

The number of resources in the cloud is usually to be huge. Therefore, considering the static pricing model, that is, setting a constant and using d_c to represent the unit computing cost of the cloud server, the total profit of scheduling tasks to the cloud is defined as:

$$Q_i^c = fee_i - (C_i^c + d_c \alpha_i f_i^c) \tag{22}$$

Therefore, the total delay and the total profit to process task W_i are respectively,

$$T_i = x_i^l t_i^l + x_i^e t_i^{mec} + x_i^c t_i^c \tag{23}$$

$$Q_i = x_i^e Q_i^{mec} + x_i^c Q_i^c \tag{24}$$

E. PROBLEM FORMULATION

In summary, given a set X of task offloading policies and a set τ of computing resource allocation policies, we can define the system cost function as the sum of the profit and the reciprocal of total delay. Where T_i and Q_i represent the delay of executing the task and the profit of the ESP, respectively, and $\lambda \in [0, 1]$ is used as the weight coefficient to regulate T_i and Q_i. Considering the constraints of variable and delay tolerance, we can formulate a joint task offloading and resource allocation optimization problem to obtain the optimal task offloading and computing resource allocation strategy, Therefore, the corresponding optimization problem can be formulated as follows:

$$\max_{X,\tau} C(X, \tau) = \lambda \sum_{e=1}^{m} \sum_{i=1}^{n} Q_i + (1 - \lambda)\left(\frac{1}{\sum_{e=1}^{m} \sum_{i=1}^{n} T_i}\right)$$

$$\begin{aligned}
\text{s.t.} \quad & C1 : x_i = \{x_i^l, x_i^e, x_i^c\}, \quad \forall i \in N \\
& C2 : t_i^{exe} \le t_i^{\max}, \quad \forall i \in N \\
& C3 : A_{k,i} \in \{0, 1\}, \quad \forall i \in N, \quad \forall k \in K \\
& C4 : 0 \le \sum_{i=1,}^{n} x_i \alpha_i \le 1, \quad \forall i \in N \\
& C5 : Q_i^k \ge 0, \forall i \in N, \quad \forall k \in K
\end{aligned} \tag{25}$$

where C1 indicates that the task of the vehicle user can only choose one of three decisions: local computing, offloading to the edge server provided by ESP, and cloud server, C2 means that the computing delay of the task does not exceed the maximum tolerable delay of the task, C3 means whether to rent resources from the edge service provider, C4 means that the resources provided by the edge server deployed in ESP are less than the total resources on the edge server, C5 means that the profit of the computing task in ESP cannot be negative.

IV. OPTIMIZATION SOLUTION

In this section, we propose a joint computation offloading and resource allocation algorithm based on K-means ++ and DDQN for delay minimization and profit maximization and discuss its time complexity. The algorithm meets the delay requirements of vehicular tasks and improves the profit of edge providers.

A. MARKOV DECISION PROCESS

In real scenarios, the channel state, the available computing resources of MEC servers, and the IoV environment all change dynamically, and at each moment, the system needs to decide which resources to allocate to which users based on the current system state. When solving such problems, the traditional methods need to select the optimal strategy from the vast decision space, which is challenging to complete. Deep reinforcement learning can search for effective sample features from many training samples to train the agent, dramatically shortening decision-making time. Therefore, this paper proposes a DDQN-based joint optimization strategy for task offloading and resource allocation. The DDQN algorithm can estimate the future environmental changes in interacting with the environment and maximize the long-term system performance by constantly learning and adjusting the strategy, even if the total delay of all vehicles is minimized and the profit of ESP is maximized. In this subsection, the system performance maximization problem is first transformed into a Markov decision process, and the DRL method is used to solve the optimal policy. A typical MDP model consists of a tuple with five elements, namely $< S, A, P, R, \gamma >$. S represents the state space, and A is a finite action space. P is the probability transition distribution, which characterizes the probability. that the current state of the environment will change to another state under a specific action. R stands for the reward function. The parameter $\gamma \in [0, 1]$ is a discount factor for future rewards.

State: The SDN controller with the ability of global information instantaneous collection, can just assume the role of agent in DRL. In each time slot, the SDN controller monitors the IoV environment within its communication range. It also collects state information, including vehicular tasks and resource availability information of edge servers and cloud servers provided by ESP under coverage. Defines the state space $as\ S(t) = \{t_i^{max}, C_i, d_i, fee_i, 1 - \sum_{i=1}^{n} x_i \alpha_i\}$. Respectively, the maximum tolerable delay of the vehicle user to calculate the task, the number of tasks needed to calculate the task, the data size required to calculate the task, the cost price that the vehicle user is willing to pay to calculate the task W_i and the available state of resources.

Action: According to the observed state $S(t)$ in the time slot, the agent will output an action as the offloading strategy taken in the current time slot and determine the percentage corresponding to the resources allocated to the vehicle user in each time slot t. Therefore, the action space is defined as $a(t)$, and the action $a(t) = \{X, \tau\}$ in time slot t. X denotes offloading decision-making, $X = \{x_1, x_2, x_3 \ldots x_n\}$, τ denotes the edge server computing resources allocation strategy,$\tau = \{\alpha_1, \alpha_2, \ldots \alpha_n\}$.

Reward: The goal of agent is to maximize the rewards for good behavior. These rewards evaluate the goodness of action $a(t)$ and guide the updating of the value network parameters. Our reward function aims to minimize the total latency of all vehicles and maximize the profit of ESP, and we define the reward function as follows:

$$R = \lambda \sum_{e=1}^{m} \sum_{i=1}^{n} Q_i + (1 - \lambda)\left(\frac{1}{\sum_{e=1}^{m} \sum_{i=1}^{n} T_i}\right) \tag{26}$$

B. THE KDDQN-JOARA BASED COMPUTATION OFFLOADING AND RESOURCE ALLOCATION ALGORITHM

The task offloading method in this paper adopts binary offloading, and the computing task request issued by the vehicle either chooses the local computing of the vehicle or offloads to the server in the cluster for computing. Considering the complex computing environment in the cluster collaboration scenario, suppose that the ESP_r deploys M MEC servers in its service area, and the number of vehicles served is represented by N. The dimension of the action space will increase with the increase in the number of vehicles communicating with the MEC server in the cluster. In this case, there will be 2^N possible actions in the action space. With the exponential growth of the number of vehicles in IoV, the first part of KDDQN-JOARA uses the K-means ++ algorithm to preprocess the action space so as to reduce the dimension of the action space and improve the efficiency of task offloading and resource allocation algorithm.

The specific description is as follows: K-Means++ changes the selection of the initial centre of mass, improving the performance and stability of the algorithm. First of all, the initial centroid is determined. Considering that the tasks of local computing generally have high real-time and low computing resource requirements, two different centroids, k_{loc} and k_{off}, are selected as the centres of local cluster w^l and offload cluster w^o according to this characteristic. The characteristics of the task are set as a combination of the data size d_i of the data needed to compute the task and the delay requirement t_i^{max}, where the delay is a hard constraint. Then, its task characteristics are extracted for each task, the vehicle computing task is assigned to two different clusters, and the centre point of the cluster is updated until convergence.

Furthermore, after the clustering operation is completed, the tasks that are clustered to local computing mode are provided with computing resources locally by the vehicle. In contrast, for the tasks in offloading computing mode, the cluster collaboration scenario is composed of multiple edge servers provided by multiple ESPs, and the environment is more complex. Therefore, The DDQN-based task offloading and resource allocation joint optimization method was used to make the optimal decision for the task. Our goal is to maximize the profit of ESP while meeting the delay requirements of task by making the optimal resource allocation scheme under various offloading decisions.

In the system scenario, SDN is regarded by us as an agent in the system, and at each time slot t, SDN observes the current system environment $S(t) = \{t_i^{max}, C_i, d_i, fee_i, 1 - \sum_{i=1}^{n} x_i\alpha_i\}$, the environment includes the current usage of system resources, the unit price of system resources, the requirements of pending tasks, and the corresponding joint action $a(t) = \{X, \tau\}$ is taken, including the selection of offloading mode and resource allocation scheme for tasks based on the current strategy. Then, after the selection of the action, the state of the environment will be transferred from $s(t)$ to the next state, $s(t+1)$. At the same time, the experience tuples ($s(t)$, $a(t)$, $R(t)$, $s(t+1)$) obtained by the agent are stored in the experience buffer for parameter updating, and the agent will receive the corresponding reward R. The algorithm uses two neural networks to approximate two action value functions: the main network and the target network, respectively. Among them, the parameters of the main network are updated in real time. In contrast, the parameters of the target network are updated every Z step, which can alleviate the overestimation of the Q value caused by the DQN algorithm. At the same time, the playback experience pool is added to solve the correlation between the data. Throughout the training, the DDQN agent randomly selects a subset of samples from the experience replay buffer to train, and in each iteration, we, according to

$$L(\omega) = \mathrm{E}\left\{Y^{DDQN} - Q\left[S(t), a(t); \omega^2\right]\right\} \tag{27}$$

calculate the loss. ω is the parameter of the main network, and Y^{DDQN} is the output of the target network, which is used to calculate the target Q value after taking all possible actions, $Q(S(t), a(t); \omega)$ is the output of the main network, which is used to calculate the Q value of the execution action in the current state. In the DDQN algorithm, the output of the target network

$$Y^{DDQN} = R(t) + \gamma Q(S(t+1), a^*; \omega^-) \tag{28}$$

The primary network is first used to find the action with the largest Q value

$$a^* = \underset{a}{argmax}\, Q(S(t+1), a; \omega) \tag{29}$$

Then the target network is used to calculate the Q value of the action a^*. Finally, the loss function is calculated according to Equation (27), and the main network parameter ω is inversely updated using gradient descent. The Q value of action a^* may not be the largest in the target network. Choosing this avoids choosing an overrated action. Through this algorithm, we can obtain the optimal network parameter ω, so that we can obtain the optimal offload resource allocation in the case of changes in the size of the input data. The details of this algorithm are briefly given in Algorithm 1.

C. COMPUTATIONAL COMPLEXITY ANALYSIS

The computational complexity of the task clustering method based on K-means ++ in Algorithm 1 usually depends on the number of data points, the number of clusters, and the dimension of the data points. When initializing the cluster centers, the complexity of this step is $O(K \times D)$, where K is the number of centers and D is the dimension of the data points. Secondly, the K-means ++ algorithm is constantly updated and recalculation in the form of iterations to achieve convergence. Therefore, the number of iterations also affects the computational complexity of the algorithm. Assuming that N is the number of tasks to be processed in the cluster collaboration scenario, and I represents the number of iterations, which usually stops when the algorithm converges, it can be

concluded that the computational complexity of K-Means++ is $O(K \times D \times I \times N)$.

The time complexity of deep reinforcement learning in Algorithm 1 is usually measured in terms of the number of multiplications computed in iterations. We assume that the number of layers of the deep neural network in the network architecture is denoted by L, and these hidden layers have $n_1, n_2, \ldots, n_l$ neuron each. Also, we assume that the number of iterations of deep reinforcement learning in Algorithm 1 in the training phase is E; Therefore, the computational complexity of deep reinforcement learning in Algorithm 1 is: $O(E \sum_{i=1}^{L} n_i \times n_{i-1})$.

The final implementation of KDDQN-JOARA algorithm first needs to execute the algorithm k-means++ to obtain the classification result, and then use DDQN to find the optimal solution. Then the computational complexity of KDDQN-JOARA is the sum of the two algorithms.

V. SIMULATION EVALUATION

This section uses Python to simulate the computational offloading and resource allocation algorithm. The simulation experiments are conducted on a personal computer with 2.4 GHz CPU and 16 GB memory.

A. SIMULATION SCENARIO

The simulation scenario is set to a street area range of 3kmx3km, including MEC servers provided by 4 ESPs and multiple vehicles, and vehicle users follow the Poisson distribution and drive at a constant speed of [40km/h, 60km/h]. The number of MEC servers provided by each ESP is [5], [6], [7], [8], [9], and [10] and is evenly distributed within the region. The value of computational resources of an edge server ranges from [10] and [20] GHz. The computing resource costs of edge servers deployed with different ESPs are heterogeneous, the value is between [2], [3] $. The computing resources of the cloud server are set to 1000Ghz and the resource price is 6$. The data transmission rate between edge servers provided by different ESPs ranges from [90,100] MB/s [23], the data transmission rate between edge servers and cloud server is set to 20 MB/s, in the cluster collaboration scenario. All emulation settings comply with the IEEE 802.11p in-vehicle networking standard and specifications in the MEC white paper. The specific simulation parameters are set according to the definition in [24] and [25], as shown in Table 2.

In order to evaluate the effectiveness of the proposed algorithm, the performance comparison is carried out through the following four algorithms:

Random: randomly select local or offload computation and randomly allocate resources to execute tasks under the condition of meeting the task delay requirement.

Greedy: In this scheme, the system greedily selects the action in each time slot based on the currently observed MEC system state without a tradeoff between exploitation and exploration.

Algorithm 1 The KDDQN-JOARA Based Computation Offloading and Resource Allocation Algorithm

Initialize the environment
Initialize the experience replay buffer
Initializes task clusters: $W^l = W^o = \emptyset$
Choose the initial centroid k_{loc} and k_{off}
for each task W_i, $i = [1, 2, \ldots, N]$ **do**
  compute the Euclidean distance between the task W_i and each centroid
  Assign the task to the cluster with the shortest distance
end for
  Initialize the main deep neural network with random weight ω
Initialize the target deep neural network with $\omega = \omega^-$
for each episode **do**
  Obtain the initial observation S1 and as the beginning state
for each step of episode **do**
  for each task W_j in W^o **do**
  Choose $a(t)$ with a random probability β as
  if $\beta \leq \epsilon$ then
  randomly select an action $a(t)$
else
  select $a(t) = \text{argmax}_a (s(t), a(t); \omega)$
end if
  Execute action $a(t)$ to offload task and allocate resource, obtain the reward $R(t)$ by calculating the corresponding delay and benefit, receive the next observation s $(t+1)$
  Store the experience into the experience replay buffer
  Randomly select a minibatch of R sample from the experience replay buffer
  Calculate the target Q value using Equation (28)
  Calculate the loss function using Equation (27)
  Perform a gradient descent step on $L(\omega)$ with respect to the parameter ω by $\partial L(\omega)/\partial\omega$
  Update the target deep neural network parameter ω^- by using weight ω
  end for
end for
end for

QoS aware Genetic Algorithm(*QGA*): Different user QoS requirements are considered by improving the traditional GA, and resources are allocated within a given budget.

DQN: An offloading strategy based on deep Q-network is designed to dynamically fine-tune the offloading ratio of each user, so as to realize a low-cost MEC system.

B. PERFORMANCE COMPARISON

Fig. 2 shows the total reward value obtained by the two methods in relation to the number of iterations. It can be seen from the figure that both methods gradually converge with the

TABLE 2. Simulation parameters.

Parameter	Value
The input data size of the task d_i	[1,5] MB
the maximum tolerable execution delay of the task t_i^{max}	[200,1000] ms
the cost price that Vehicles are willing to pay to compute task fee_i	[5,10] $
Task computation density	0.297Gigacycles/MB
The computational resources of the vehicle f_i^l	2.5GHz
vehicular communication bandwidth B	20MHz
Noise power σ^2	-114dBm
Vehicular transmission power p_n	24dBm
discount factor	0.9
ϵ-greedy exploration method	0.9
learning rate	0.01

change of the number of iterations, but the KDDQN-JOARA algorithm proposed in this paper gradually stabilizes and eventually converges through the interaction between the agent and the environment when it is iterated to about 750 times. DQN only converges after 1250 iterations. This is because KDDQN-JOARA uses the k-means++ algorithm to determine the initial offloading decision of the task and obtains the offloading decision of the task in the local cluster and the cluster cooperation cluster, which reduces the scale of the problem and reduces the dimension of the action set. Therefore, the KDDQN-JOARA algorithm can realize efficient offloading decisions and resource allocation strategies.

1) THE IMPACT OF CLOUD-EDGE CLUSTERS COLLABORATION

In order to analyze the profits of cluster cooperation, this section evaluates the delay and profit in two scenarios, respectively. Fig 3 shows the comparison of ESP profit in the clustered collaboration scenario and the non-clustered collaboration scenario, from which it can be seen that as the number of tasks increases, the profit of ESP in the scenario with collaboration considered are consistently higher than those in the scenario without collaboration considered. When the number of tasks is small, both scenarios have enough resources to process the tasks. Therefore, the profit in both scenarios shows an increasing trend, and when the number of tasks increases to a certain number, the resources in the usual scenario are not enough to compute more tasks. Therefore, the profit of the ESP does not increase anymore. In the collaborative computing scenario, ESPs can have more choices in computing tasks, and when the resources are insufficient, they can rent the resources of other ESPs at a certain cost to obtain higher revenue. At the same time, during the low peak period of task processing, ESPs can also sell their idle resources at a certain price to increase the additional revenue. Therefore, in collaborative scenarios, ESP profits show a

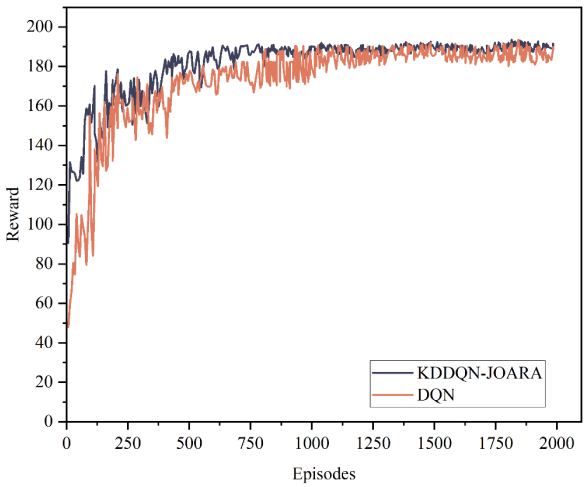


FIGURE 2. Convergence of the proposed DDQN method combined with k-means and the original DDQN method.

continuous upward trend and are always higher than those in non-collaborative scenarios.

Fig.4 shows that the average latency of tasks in both cluster collaboration and non-collaboration scenarios is upward with the increasing number of tasks. When the number of tasks is less, in the stage of 10-50, there is not much difference between the average latency of the two scenarios. However, when the number of tasks exceeds 50, the average latency in the non-cluster collaboration scenario increases steeply. When the number of tasks reaches 150, the average latency in the cluster collaboration scenario decreases by 30% compared to the non-collaboration scenario. This is because one of the essential features of clustered collaboration is the diversity of edge resources, which can be utilized to reduce the average latency of tasks even further. In contrast, in the non-collaboration scenario, when the local resources of the ESP are used up, even though there is an option to offload the tasks to the cloud for execution, this also incurs a high transmission latency due to the geographic location of the remote cloud. Therefore, as the number of tasks grows, the average latency in cluster collaboration scenarios can increase relatively slowly. In contrast, non-collaboration scenarios, when a certain number of tasks are reached, inevitably incur a higher average latency.

2) THE IMPACT OF CHANGE IN THE NUMBER OF TASKS

In order to test the impact of the number of tasks change on the average delay of system tasks, the number of ESPs is set to 4, where the number of servers deployed in each ESP is evenly distributed between 5 and 10. The experiment considers indivisible tasks, that is, tasks generated by vehicles are either computed locally or entirely offloaded. The number of test tasks is the experimental comparison of increasing from 10 to 150.

Figures 5 and 6 show the performance comparison of the five methods regarding the average execution delay of tasks and the profit of ESP as the number of tasks in the system

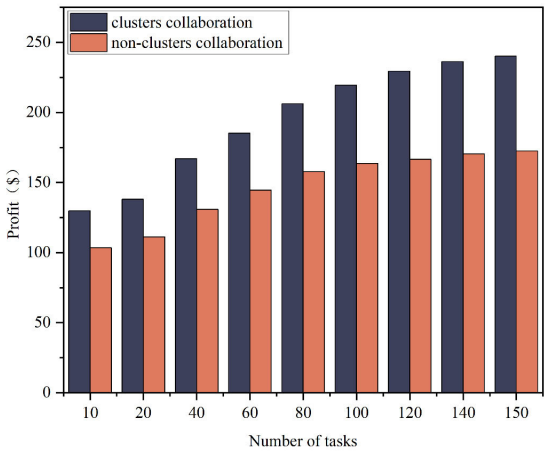


FIGURE 3. The impact of cluster collaboration on ESP profits.

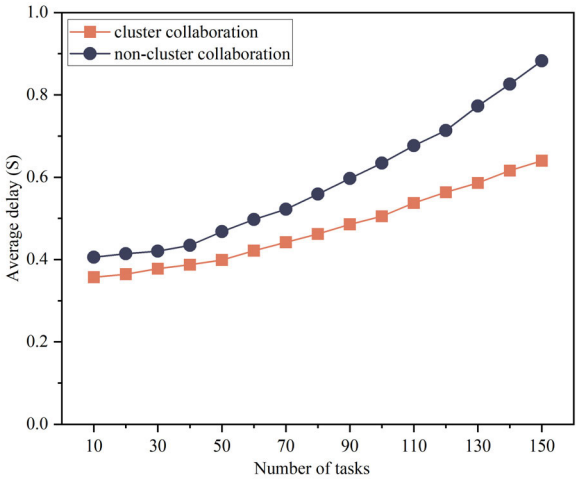


FIGURE 4. The impact of cluster collaboration on the average delay.

increases. As the number of tasks increases, the latency of the five methods will show a certain upward trend. When the number of tasks reaches 150, the resource utilization rate will increase, the load on each edge node will increase, and the resource selectivity during task execution will decrease. Therefore, the average latency of each method will increase accordingly. The Greedy method always selects the current optimal method for resource allocation, which is prone to falling into local optima. However, the Random method randomly assigns tasks when the delay threshold is satisfied, and the uncertainty is too high. Therefore, it performs poorly in terms of average delay performance. The QGA and DQN methods do not perform joint optimization of task offloading and computing resource allocation to minimize latency. Therefore, increasing the number of tasks can lead to a significant increase in average latency. Compared with other schemes, our scheme preprocesses the offloading decision for computing tasks, accelerating the efficiency of task offloading and resource allocation. Compared with the DQN and QGA algorithms, the average latency is reduced by 16% and 27%, respectively. Compared to Greedy and Random algorithms, it has decreased by 37% and 39%, respectively.

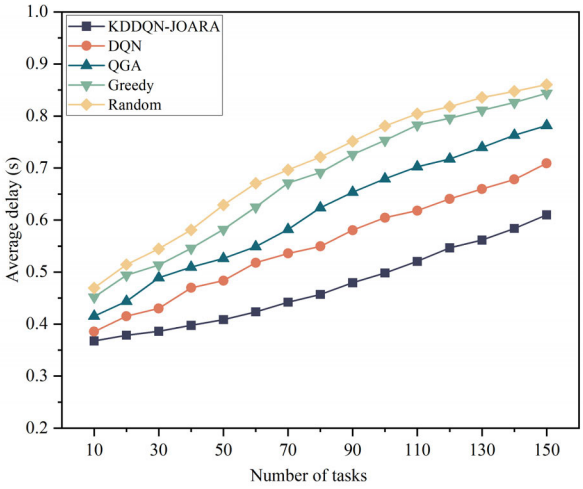


FIGURE 5. The effect of task number change on delay.

Fig. 6 depicts the relationship between the profit of ESP and the number of tasks. From this figure, the ESP's profit of the five methods gradually grows as the number of tasks increases. When the number of tasks is small, at this time, the system resources are abundant, and there are more choices in resource allocation. The advantages of the KDDQN-JOARA method proposed in this paper are not obvious compared with DQN and QGA. Compared with Greedy and Random methods, the advantages are apparent. As the number of tasks increases, this advantage will continue to increase because the system's available resources gradually decrease as the number of tasks increases. The proposed KDDQN-JOARA method can continuously interact with the environment through the agent and optimize the computation offloading strategy and computation resource allocation strategy simultaneously to obtain better performance and generate higher revenue.

Figure 7 shows the performance comparison of the five methods in terms of total energy consumption under different strategies as the number of tasks sent by the vehicle increases. From the figure, as the number of tasks continues to increase, the total energy consumption of the system will also continue to increase. At the same time, the KDDQN- JOARA method proposed in this article can generate lower energy consumption compared to the other four algorithms. This is because the DQN algorithm is easy to lead to over-estimation when obtaining the Q value, and the convergence is slow and unstable, so it performs generally in terms of energy consumption. Greedy schemes choose fixed actions without learning, which can easily fall into local optima, leading to significant performance losses. The QGA algorithm performs worse than the KDDQN-JOARA method in solving large-scale edge resource allocation problems like cluster collaboration. Therefore, Figures 5, 6, and 7 show that compared with the other four methods, the KDDQN-JOARA method can minimize the average execution delay of task and reduce

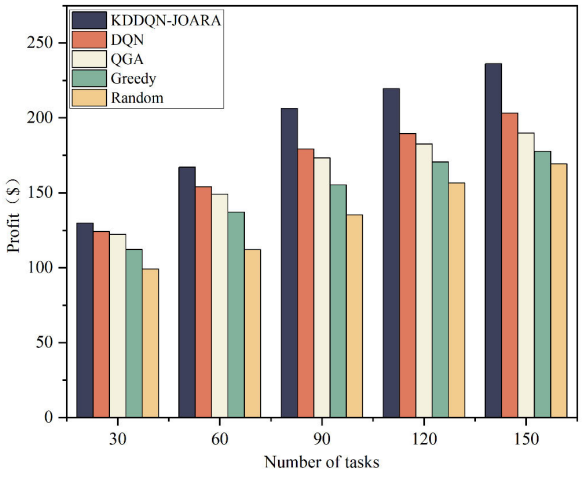


FIGURE 6. The effect of task number change on profit.

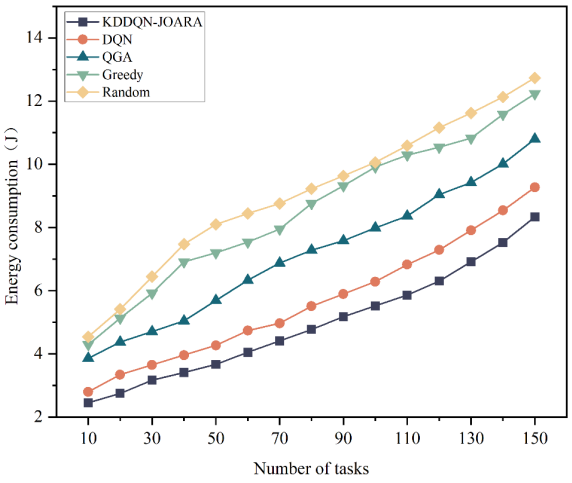


FIGURE 7. Energy consumption versus the number of tasks.

the energy consumption of system while balancing the profit of ESP.

3) THE IMPACT OF CHANGE IN THE COMPUTATION RESOURCE REQUIRED OF TASKS

Next, we compare the impact of different computational resource demands on latency, where the number of tasks is set to 70, and the number of ESPs is set to 4. shown in Fig. 8, we tested the delay variation results of the amount of computing resources required by the task from 1500 to 4000 Megacycles, respectively. It can be observed that for all methods, the average latency grows as the computational resources required by the task increase. It can be seen from the figure that the proposed KDDQN-JOARA-based method consistently outperforms the other methods. Specifically, when the amount of required computing resources is as high as 4000 trillion cycles, compared with DQN, QGA, Greedy and Random, KDDQN-JOARA reduces the average delay by 18%, 24%, 35% and 39%, respectively.

Similarly, for the ESP's profit of executing the task, as shown in Fig. 9, the ESP's profit of all methods increases

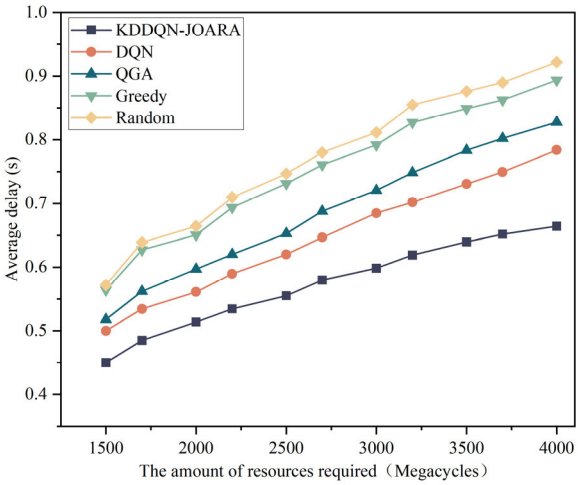


FIGURE 8. Average delay versus the number of CPU cycles required by tasks.

with the computational resources required. This is because an increase in the amount of computing resources required for task computing means that resources with higher computing power must be allocated to tasks, and resources with higher computing power will be priced higher. The Random scheme will randomly choose between high and low benefits. The overall revenue will significantly reduce when too many low benefits are selected. At the same time, the greedy algorithm makes it difficult to achieve the optimal solution. In addition, both DQN and QGA methods are inferior to the KDDQN-JOARA algorithm in terms of revenue and average delay because, through the interaction between agent and environment, the KDDQN-JOARA algorithm can realize efficient offloading decision and resource allocation strategy. Our algorithm can better perceive the global environment and obtain higher revenue through the interaction between the agent and the environment.

As shown in Fig. 10, when the resource demand of computing tasks increases, the energy consumption generated by the KDDQN-JOARA algorithm is still less than that of the other four methods. Specifically, when the required computing resources are up to 4000Megacycles, compared with DQN, QGA, greedy algorithm and random method, The total energy consumption generated by the KDDQN-JOARA method is reduced by about 14%, 24%, 37% and 43%, respectively, because introducing our SDN concept can more reasonably realize the unified scheduling of global variables. At the same time, the proposed method can achieve efficient offloading decisions and resource allocation strategies. Therefore, we show that the proposed method is the best among the five methods.

VI. DISCUSSION

This paper proposes an SDN-driven cloud-edge cluster collaborative IoV architecture. In our scenario, there are multiple ESPs, which form a rich shared resource pool. Aiming at solving the resource shortage problem faced by traditional

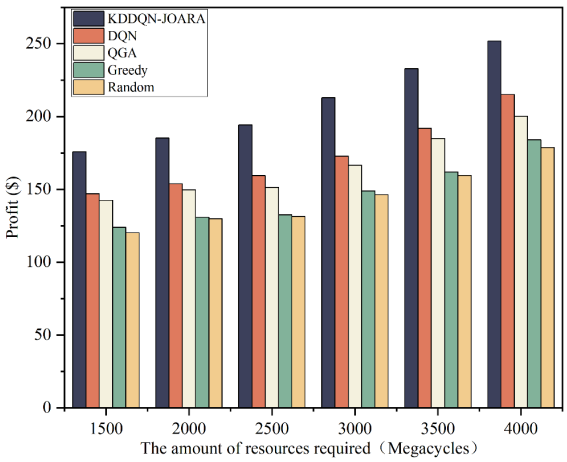


FIGURE 9. Profit versus the number of CPU cycles required by tasks.

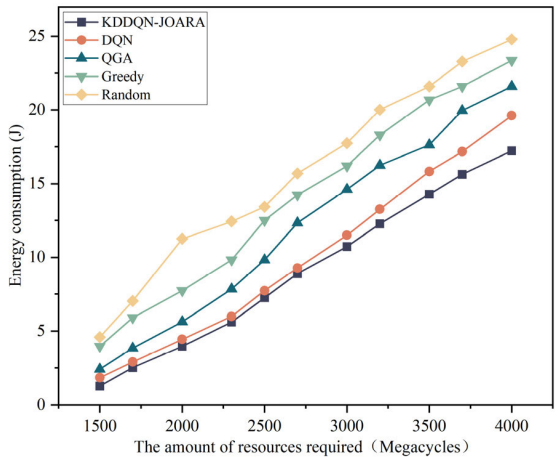


FIGURE 10. Energy consumption versus the number of CPU cycles required by tasks.

edge computing architecture when dealing with the growing in-vehicle applications, a dynamic pricing model for resource sharing in the cluster is adopted. It reflects the dynamic nature of the resource provision market and makes ESP more flexible and effective in setting resource prices and participating in resource-sharing mode. In order to realize the efficient search for the best offloading and computing resource allocation decision in this complex scene, this paper proposed a task offloading and resource allocation algorithm based on KDDQN-JOARA, which adopted the idea of clustering algorithm and took the task requirements and delay as the basis, so as to quickly determine the task should choose local computing mode or offloading computing mode in the early stage of the algorithm. The formation of task-offloading computing clusters provides more specific environmental information for the DDQN algorithm. By learning resource allocation decisions under different states, DDQN can gradually optimize the algorithm to improve task efficiency and resource utilization and finally obtain the optimal offloading and computing resource allocation decision. Finally, the simulation and analysis of our proposed method are carried out. The simulation results show that considering the cloud-edge cluster collaboration scenario, ESP has more choices when executing tasks to allocate computing resources and can select the computing resources with lower execution costs when meeting the task delay requirements. Compared with other existing algorithms, our algorithm has great advantages in terms of task's average response delay and ESP's profit.

Face the future, considering the resource-sharing mode of cloud-edge cluster collaboration scenario, it can be further applied to intelligent traffic management systems to help cities monitor and manage traffic flow, reduce congestion, and improve energy efficiency. However, there are also limitations, such as cost considerations, infrastructure requirements, and security and privacy issues. Since security is a very important section, therefore, next, we must consider the authentication and authorization mechanism when resources are shared between different ESPs to ensure that only legitimate users and devices can access edge computing resources and data. Use multi-factor authentication for added security.

VII. CONCLUSION

In the IoV scenario, this paper proposed a collaboration architecture integrating SDN into IoV cloud-edge clusters to expand resources and effectively handle task offloading during peak hours. A dynamic pricing model was introduced to facilitate resource sharing among different ESPs. Based on this framework, the joint optimization problem of task offloading and resource allocation was formulated with the objective of minimizing the completion delay of tasks and maximizing the profit of ESP. The KDDQN-JORAR algorithm was employed to solve this problem. Firstly, the task delay was treated as a hard constraint, and the initial offloading node was selected. Then, the DDQN algorithm was used to find the optimal solution for task offloading and resource allocation, satisfying the requirements of both vehicles and ESPs. Experimental results demonstrated that the proposed algorithm achieved significant improvements in reducing the delay of task, decreasing the energy consumption of system, and enhancing the profit of ESP compared to other algorithms.

REFERENCES

[1] X. Deng, L. Wang, J. Gui, P. Jiang, X. Chen, F. Zeng, and S. Wan, "A review of 6G autonomous intelligent transportation systems: Mechanisms, applications and challenges," *J. Syst. Archit.*, vol. 142, Sep. 2023, Art. no. 102929, doi: 10.1016/j.sysarc.2023.102929.

[2] L. Liu, C. Chen, Q. Pei, S. Maharjan, and Y. Zhang, "Vehicular edge computing and networking: A survey," *Mobile Netw. Appl.*, vol. 26, no. 3, pp. 1145–1168, Jun. 2021, doi: 10.1007/s11036-020-01624-1.

[3] B. Li and R. Wu, "Joint perception data caching and computation offloading in MEC-enabled vehicular networks," *Comput. Commun.*, vol. 199, pp. 139–152, Feb. 2023, doi: 10.1016/j.comcom.2022.12.021.

[4] K. Zhang, Y. Mao, S. Leng, Y. He, and Y. Zhang, "Mobile-edge computing for vehicular networks: A promising network paradigm with predictive offloading," *IEEE Veh. Technol. Mag.*, vol. 12, no. 2, pp. 36–44, Jun. 2017, doi: 10.1109/MVT.2017.2668838.

[5] L. Zhao, E. Zhang, S. Wan, A. Hawbani, A. Y. Al-Dubai, G. Min, and A. Y. Zomaya, "MESON: A mobility-aware dependent task offloading scheme for urban vehicular edge computing," *IEEE Trans. Mobile Comput.*, pp. 1–15, 2023, doi: 10.1109/TMC.2023.3289611.

[6] L. Wang, X. Deng, J. Gui, X. Chen, and S. Wan, "Microservice-oriented service placement for mobile edge computing in sustainable Internet of Vehicles," *IEEE Trans. Intell. Transp. Syst.*, vol. 24, no. 9, pp. 10012–10026, Sep. 2023, doi: 10.1109/TITS.2023.3274307.

[7] X. Cao, G. Tang, D. Guo, Y. Li, and W. Zhang, "Edge federation: Towards an integrated service provisioning model," *IEEE/ACM Trans. Netw.*, vol. 28, no. 3, pp. 1116–1129, Jun. 2020, doi: 10.1109/TNET.2020.2979361.

[8] M. A. U. Rehman, M. Salah ud din, S. Mastorakis, and B.-S. Kim, "FoggyEdge: An information-centric computation offloading and management framework for edge-based vehicular fog computing," *IEEE Intell. Transp. Syst. Mag.*, vol. 15, no. 5, pp. 78–90, Sep. 2023, doi: 10.1109/MITS.2023.3268046.

[9] B. Hazarika, K. Singh, S. Biswas, and C.-P. Li, "DRL-based resource allocation for computation offloading in IoV networks," *IEEE Trans. Ind. Informat.*, vol. 18, no. 11, pp. 8027–8038, Nov. 2022, doi: 10.1109/TII.2022.3168292.

[10] W. Fan, Y. Su, J. Liu, S. Li, W. Huang, F. Wu, and Y. Liu, "Joint task offloading and resource allocation for vehicular edge computing based on V2I and V2V modes," *IEEE Trans. Intell. Transp. Syst.*, vol. 24, no. 4, pp. 4277–4292, Apr. 2023, doi: 10.1109/TITS.2022.3230430.

[11] J. Ren, G. Yu, Y. He, and G. Y. Li, "Collaborative cloud and edge computing for latency minimization," *IEEE Trans. Veh. Technol.*, vol. 68, no. 5, pp. 5031–5044, May 2019, doi: 10.1109/TVT.2019.2904244.

[12] C. Yang, Y. Liu, X. Chen, W. Zhong, and S. Xie, "Efficient mobility-aware task offloading for vehicular edge computing networks," *IEEE Access*, vol. 7, pp. 26652–26664, 2019, doi: 10.1109/ACCESS.2019.2900530.

[13] 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, doi: 10.1109/TVT.2019.2917890.

[14] I. Al-Hammadi, M. Li, and S. M. N. Islam, "Independent tasks scheduling of collaborative computation offloading for SDN-powered MEC on 6G networks," *Soft Comput.*, vol. 27, no. 14, pp. 9593–9617, Jul. 2023, doi: 10.1007/s00500-023-08091-2.

[15] L.-A. Phan, D.-T. Nguyen, M. Lee, D.-H. Park, and T. Kim, "Dynamic fog-to-fog offloading in SDN-based fog computing systems," *Future Gener. Comput. Syst.*, vol. 117, pp. 486–497, Apr. 2021, doi: 10.1016/j.future.2020.12.021.

[16] B. Cao, Z. Sun, J. Zhang, and Y. Gu, "Resource allocation in 5G IoV architecture based on SDN and fog-cloud computing," *IEEE Trans. Intell. Transp. Syst.*, vol. 22, no. 6, pp. 3832–3840, Jun. 2021, doi: 10.1109/TITS.2020.3048844.

[17] H. Zhang, Z. Wang, and K. Liu, "V2X offloading and resource allocation in SDN-assisted MEC-based vehicular networks," *China Commun.*, vol. 17, no. 5, pp. 266–283, May 2020, doi: 10.23919/JCC.2020.05.020.

[18] Q. Liu, R. Luo, H. Liang, and Q. Liu, "Energy-efficient joint computation offloading and resource allocation strategy for ISAC-aided 6G V2X networks," *IEEE Trans. Green Commun. Netw.*, vol. 7, no. 1, pp. 413–423, Mar. 2023, doi: 10.1109/TGCN.2023.3234263.

[19] Y. Liu, H. Yu, S. Xie, and Y. Zhang, "Deep reinforcement learning for offloading and resource allocation in vehicle edge computing and networks," *IEEE Trans. Veh. Technol.*, vol. 68, no. 11, pp. 11158–11168, Nov. 2019, doi: 10.1109/TVT.2019.2935450.

[20] L. Huang, X. Feng, C. Zhang, L. Qian, and Y. Wu, "Deep reinforcement learning-based joint task offloading and bandwidth allocation for multi-user mobile edge computing," *Digit. Commun. Netw.*, vol. 5, no. 1, pp. 10–17, Feb. 2019, doi: 10.1016/j.dcan.2018.10.003.

[21] K. Cheng, Y. Teng, W. Sun, A. Liu, and X. Wang, "Energy-efficient joint offloading and wireless resource allocation strategy in multi-MEC server systems," in *Proc. IEEE Int. Conf. Commun. (ICC)*, May 2018, pp. 1–6, doi: 10.1109/ICC.2018.8422877.

[22] H. Zhou, K. Jiang, X. Liu, X. Li, and V. C. M. Leung, "Deep reinforcement learning for energy-efficient computation offloading in mobile-edge computing," *IEEE Internet Things J.*, vol. 9, no. 2, pp. 1517–1530, Jan. 2022, doi: 10.1109/JIOT.2021.3091142.

[23] Q. Wang, Y. Mao, Y. Wang, and L. Wang, "Computing task offloading based on multi-cloudlet collaboration," *J. Comput. Appl.*, vol. 40, no. 2, p. 328, Feb. 2020, doi: 10.11772/j.issn.1001-9081.2019081367.

[24] L. Wang and G. Zhang, "Joint service caching, resource allocation and computation offloading in three-tier cooperative mobile edge computing system," *IEEE Trans. Netw. Sci. Eng.*, vol. 10, no. 6, pp. 3343–3353, Nov. 2023, doi: 10.1109/TNSE.2023.3259030.

[25] M. Tang and V. W. S. Wong, "Deep reinforcement learning for task offloading in mobile edge computing systems," *IEEE Trans. Mobile Comput.*, vol. 21, no. 6, pp. 1985–1997, Jun. 2022, doi: 10.1109/TMC.2020.3036871.

XIANHAO SHEN received the Ph.D. degree in mechanical and electronic engineering from the Beijing Institute of Technology, Beijing, in 2008.

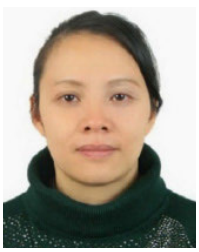
He is currently the Director of the Teaching and Research Department, Electronic Information Engineering, School of Information Science and Engineering, Guilin University of Technology. His research interests include wireless sensor networks and internet.

LI WANG is currently pursuing the master's degree in software engineering with the Guilin University of Technology, Guilin, Guangxi.

Her research interests include the Internet of Vehicles and edge computing.

PANFENG ZHANG received the Ph.D. degree in computer system architecture from the Huazhong University of Science and Technology, in 2017.

He is currently an Associate Professor and a Master Tutor with the Guilin University of Technology. His research interests include information storage technology, information security, big data, and artificial intelligence.



XIAOLAN XIE received the Ph.D. degree from Xidian University Xi'an, Shanxi. She is currently the Dean of the School of Information Science and Engineering, Guilin University of Technology. Her research interests include cloud computing and manufacturing informatization.

YI CHEN is currently pursuing the master's degree in electronic information with the Guilin University of Technology Guilin, Guangxi.

His research interests include reinforcement learning and the Internet of Things.

SHAOFANG LU is currently pursuing the master's degree in electronic information with the Guilin University of Technology, Guilin, Guangxi.

Her research interests include IRS assisted communication and anti-jamming communication.