TSINGHUA SCIENCE AND TECHNOLOGY
ISSN 1007-0214 02/14 pp433–451
DOI: 10.26599/TST.2022.9010006
Volume 28, Number 3, June 2023

Collaborative Offloading Method for Digital Twin Empowered Cloud Edge Computing on Internet of Vehicles

Linjie Gu, Mengmeng Cui*, Linkun Xu, and Xiaolong Xu

Abstract: Digital twinning and edge computing are attractive solutions to support computing-intensive and service-sensitive Internet of Vehicles applications. Most of the existing Internet of Vehicles service offloading solutions only consider edge–cloud collaboration, but the collaboration between small cell eNodeB (SCeNB) should not be ignored. Service delays far lower than offloading tasks to the cloud can be obtained through reasonable collaborative computing between nodes. The proposed framework realizes and maintains the simulation of collaboration between SCeNB nodes by constructing a digital twin that maintains SCeNB nodes in the central controller, thereby realizing user task offloading positions, sub-channel allocation, and computing resource allocation. Then an algorithm named AUC-AC is proposed, based on the dominant actor–critic network and the auction mechanism. In order to obtain a better command of global information, the convolutional block attention mechanism (CBAM) is used in the digital twin of each SCeNB node to observe its environment and learn strategies. Numerical results show that our experimental scheme is better than several baseline algorithms in terms of service delay.

Key words: digital twin; cloud-edge computing; reinforcement learning; actor-critic network

1 Introduction

With the rise of the 5G era, various computing-intensive and delay-sensitive services in the field of internet of vehicles (IoV) are constantly emerging. Simultaneously, demands for the data rate and timelines become substantially stricter than traditional IoV scenes because the images or videos are transmitted and processed in the network[1].

However, some vehicles may not even be equipped with computing devices due to the performance limitations of these devices on vehicles[2], and the usual scheme chooses to offload the service request from the IoV users to the cloud service layer for execution[3]. Nevertheless, the process of offloading the service request to the cloud server and then returning constantly causes serious delays considering the long distance between the cloud and users, which will lead to the reduction in quality of service (QoS). Time delays may also cause serious consequences, such as traffic accidents[4].

As a paradigm between the rapid growth of computing tasks and the limitation of computing power, mobile edge computing (MEC) uses MEC servers to sink information services from the cloud to the edge of the wireless network access point to create high-quality service environments with high bandwidth and low latency for users[5]. Consequently, the explosive growth of application data and the limitations of the channel and computing resources of most MEC servers may lead to problems, such as channel interference and server overload, when faced with intensive service requests. This phenomenon may lead to a reduction in QoS and

• Linjie Gu is with Changwang School of Honors, Nanjing University of Information Science and Technology, Nanjing 210044, China. E-mail: 201983320030@nuist.edu.cn.
• Mengmeng Cui and Xiaolong Xu are with School of Computer and Software, Nanjing University of Information Science and Technology, Nanjing 21044, China. E-mail: cuimengmeng@nuist.edu.cn; xlxu@nuist.edu.cn.
• Linkun Xu is with School of Artificial Intelligence, Nanjing University of Information and Technology, Nanjing 210044, China. E-mail: 201983320033@nuist.edu.cn.
* To whom correspondence should be addressed.
Manuscript received: 2021-10-18; revised: 2022-01-21; accepted: 2022-03-01

failure to meet constraints of service requests from multiple users simultaneously.

This paper assumes that our edge server deployment scheme is based on small cell eNodeB (SCeNB), small edge servers are directly connected to base stations. Computing tasks can be further offloaded from the current edge computing node to other nodes during peak hours based on the connectivity between SCeNB nodes, thereby easing the burden on the current edge node. The result of the service request is executed by other nodes and then returned to the current node; this result is finally sent back to the IoV user side[6]. The idle computing resources in the system can be effectively utilized through the cooperation between SCeNB nodes, and a load of edge computing nodes can be effectively balanced. Meanwhile, the overall service quality of the system will be improved. However, the cooperation between SCeNB nodes can be affected by many factors, such as communication topology and network status between nodes. Therefore, designing a scheme under the constraints of edge computing resources for decision making considering the task offloading destination and server computing resource allocation is challenging.

The emergence of digital twin technology can simultaneously solve the problem of MEC selection and task offloading[7]. In addition, the digital twin utilizes machine learning and Internet of Things technology to bring the data-driven representation of the physical world into the mirrored virtual space. The state mapping between the real and virtual dimensions provides a global perspective for system scheduling, thereby guiding the scheduling and optimization for overall systems[8]. Using the feature of digital twin can help effectively explore collaboration methods between SCeNB nodes and assist IoV users in task offloading. The state of the entire edge computing network can also be monitored in real-time through the combination of MEC and digital twin, and perception data can be directly provided for the decision-making module[9].

In addition, network environments, computing resources, channel conditions, and service requests of users are dynamic under the IoV environment, and the process of offloading decision making can be abstracted as a Markov decision process[10]. As part of artificial intelligence, reinforcement learning has achieved remarkable success in the fields of game confrontation, robot control, and human–machine games[11]. Deep reinforcement learning is the combination of deep and reinforcement learning. It integrates the powerful comprehension of perception problems of deep learning and the decision-making capability of reinforcement learning to achieve end-to-end learning. The emergence of deep reinforcement learning contributes to the practicality of reinforcement learning technology and helps solve complex problems in real-world scenarios.

Thus far, some research works have applied deep reinforcement learning to the edge computing scenarios of the IoV; however, two main problems are found in these works. On the one hand, some studies, such as Ref. [12], only consider offloading of computing tasks to edge or cloud servers and disregard the allocation of channels and computing resources. On the other hand, some studies, such as Ref. [13], consider the channel and computing resource allocation issues. However, the designed computational offloading method is only for a single server and disregards the collaboration issue between edge servers. As the deployment of edge servers becomes increasingly intensive and the demand for user service offloading rises, cooperation between edge servers is necessary. Some studies[14, 15] used multi-agent reinforcement learning methods, such as multi-agent deep deterministic policy gradient (MADDPG), to investigate the cooperative offloading problem between edge servers. However, the application of multi-agent reinforcement learning to cooperative computing offloading will inevitably encounter problems regarding dimensional disaster and reliability distribution[16]. Simultaneously, the dynamics and non-stationarity of the IoV environment will also affect the convergence of multi-agent strategy.

Maximizing the cooperation between edge servers to achieve a reasonable allocation of channels and computing resources while considering limited computing resources at the edge servers to provide long-term and stable low latency services to users is generally a major challenge for edge computing in the current IoV. We propose a task offloading decision and resource allocation scheme for cloud-edge collaboration computing in this paper. SCeNB nodes can allocate base station sub-channels and edge server computing resources, realizing edge–cloud and edge–edge collaborations. The main contributions of this question include the following three aspects.

- A cloud-edge collaborative IoV edge computing task offloading model is proposed. A digital twin of the SCeNB node group is constructed in this model to simulate the process of task forwarding and collaborative computing between SCeNB nodes.

• A convolutional attention mechanism-based environmental state representation method that can identify the states of global nodes in the system is proposed. The digital twins of the SCeNB nodes independently observe the global environment and follow the same rules to extract features from the environment through the proposed method.

• The synchronous advantage actor–critic network is used to solve the distributed decision-making problem, and a method combined with auction mechanism, which can realize the collaboration between SCeNB nodes to complete the decision of the offloading destination of the forwarded tasks and the resource allocation, is proposed.

The remaining chapters of this article are arranged as follows. Section 2 introduces the related work and presents a certain analysis of the results of these studies and existing problems. Section 3 establishes a detailed system model for the task offloading collaboration in IoV edge computing environments. Section 4 discusses the algorithm AUC–AC based on A2C and auction mechanism. Section 5 introduces the experimental parameter settings and results. Finally, the full text is summarized, and the future research work direction is indicated in Section 6.

2 Related Work

2.1 Digital twin empowered edge computing issues

Thus far, many studies have focused on the combination of digital twins and edge computing. Reference [13] proposed a deep learning architecture for user association, which trains algorithms offline by establishing a digital twin of the MEC network environment on a central server. Reference [7] studied the establishment of a digital twin network in the 6G scenario to predict the mobility of users and changes in the MEC environment, minimizing service delays under the constraints of the cumulative service migration costs during the user's movement. A new vehicle edge computing network based on digital twins and multi-agents is proposed in Ref. [17] to improve the collaboration of agents and optimize the efficiency of task offloading. A DT-enabled multiuser offloading system, which uses deep Q network (DQN) networks to improve service quality, is proposed in Ref. [12]. Reference [18] studied the problem of mobile users regarding offloading intelligent tasks to collaborative mobile edge servers using digital twins. Specifically, blockchain technology is used to realize the selection of MES. However, none of these mentioned studies considers the scene including edge–edge and edge–cloud collaborations, which may be the task offloading trend in 5G communication architecture.

2.2 Multi-resource allocation problem via cooperative computing

Many studies have been conducted on the computing resource allocation of task offloading in MEC, including single-node and multi-node computing resource allocations. However, even in multi-node computing resource allocation, most studies adjust single-node resource allocation strategies by sensing the status of other nodes. For example, RSU is used in Ref. [19] to perceive vehicle status information, especially speed information. Therefore, a vehicle speed perception delay constraint model based on different speeds and task types is established, thus achieving satisfactory performance considering energy cost and task computing delay. The potential user service demand is predicted in Ref. [20] through the deep spatio-temporal residual network. The cloud server cooperates with each edge server to obtain the status of traffic flow and acquires the service offloading strategy through the distributed asynchronous advantage actor–critic network. However, these studies have ignored the potential cooperation between edge servers in the IoV edge computing scenarios. The computing and storage resources in the edge computing nodes are modeled in Ref. [21], and a multi-resource allocation system for IoT collaborative computing based on deep reinforcement learning is designed. A noncooperative game was developed in Ref. [6] to simulate the computing offloading of different types of tasks and balance the calculation delay of each task on MEC-BS, and the existence and convergence of the game were proven. However, neither of the two aforementioned articles considered the problem of channel allocation during data transmission, which will substantially influence the performance of the system.

2.3 Application of deep reinforcement learning in edge computing

Deep reinforcement learning has been widely recognized with the rapid development of large-scale parallel graphics processing. Reference [22] proposed a parallel RL paradigm of asynchronous advantage actor–critic based on DQN by training multiple agents in multiple environments. The actor–critic method is improved

in Ref. [23], and a multi-agent reinforcement learning framework with basic strategies is proposed for the agent to control trajectory planning, data scheduling, and bandwidth allocation. The federated learning model is adopted to develop the corresponding edge federation and online joint collaboration algorithms. References [15, 24, 25] all studied the energy supply and computing resource allocation issues. Among these studies, Refs. [24, 25] used the multi-agent deep reinforcement learning of the asynchronous advantage actor–critic algorithm to solve the risk-sensitive energy analysis problem of the micro grid-enabled MEC network and determine the best energy dispatch strategy between MEC nodes. Reference [15] solved the problem of task offloading for mobile devices in large-scale heterogeneous MEC clusters through an improved algorithm based on MADDPG and SAC, focusing on the selection of target servers and the size of the offloaded data. The multi-agent actor–critic algorithm is used in Ref. [26] to solve the problem of load balancing in IoV network, which can quickly adapt to the fluctuation of network traffic. In Ref. [27], a distributed double dueling deep Q-network (D3QN) based algorithm is proposed, by integrating double DQN and dueling DQN, to implement an MEC-enabled distributed cooperative microservice caching scheme. In Ref. [28], a method is proposed to improve the adaptability of offloading in various application scales. The experiment result shows that the proposed method enables the system to realize improved performance in transmission rate and computing capability. Deep reinforcement learning generally has a good prospect in the scenario of edge computing.

3 System Model and Problem Definition

First, this section proposes an edge–cloud collaborative task offloading model of IoV. Then, a mathematical model is established for network communication and user request delay in the system. Finally, the problem of service unloading and resource allocation in networked edge computing is abstracted as an integer programming problem. The notation and the meanings of some important variables in the system model are shown in Table 1.

Table 1 Notations and definitions.

Notation	Definition
N_{node}	Number of the SCeNB nodes
i	ID of the SCeNB node
N_i^{cha}	Number of sub-channels that the SCeNB node can allocate
N_i^{cal}	Number of computing resource that the SCeNB node can allocate
G^{geo}	Graph used to represent the length of the optical fiber between SCeNB nodes
G^{topo}	Graph used to represent the topology relationship between SCeNB nodes

3.1 System model

The edge–cloud task offloading model in IoV environments is shown in Fig. 1. The MEC server deployment in this article is based on the SCC deployment scheme as in Ref. [29]. The core idea of SCC is to enhance the functionalities of small base stations by adding computing and storage capabilities and provide cloud-based SCeNB for realizing edge computing[30, 31]. The deployment scheme introduces a small cell manager named SCM to assist in SCeNB computing and storage resource management to migrate the concept of SCC into the mobile network architecture. The system is divided into two layers: the edge layer of the IoV and the cloud service layer. Most of the base stations in the current 5G architecture are connected in a ring mode. Thus, the scheme discussed in this paper also assumes that the base stations are connected as a ring. The edge layer in the designed system is a cluster of SCeNB nodes connected into a ring. The nodes are connected with others and the cloud service layer through a wired channel. SCeNB nodes with tight computing resources can decide to forward computing tasks to other nodes in the edge computing cluster for low service delay. Simultaneously, we assume that each user will only connect to the SCeNB node closest to it. In each time slot, if an IoV user generates a service request, then the request will be sent to a nearby SCeNB node deployed nearby. In particular, we assume that 5G communication equipment is installed on the 5G edge computing node and the vehicle of the user. Through the 5G wireless channel, the vehicle of the user and the corresponding edge computing node are connected through high bandwidth communication. In this article, we assume that each SCeNB node has pre-installed various vehicle applications in the edge server, which can meet the service request offloaded to the SCeNB nodes. Simultaneously, the service requests of users can also be executed in the cloud service layer. The cloud server in this layer contains sufficient computing and storage resources and is directly connected to the service provider, which can efficiently and quickly meet

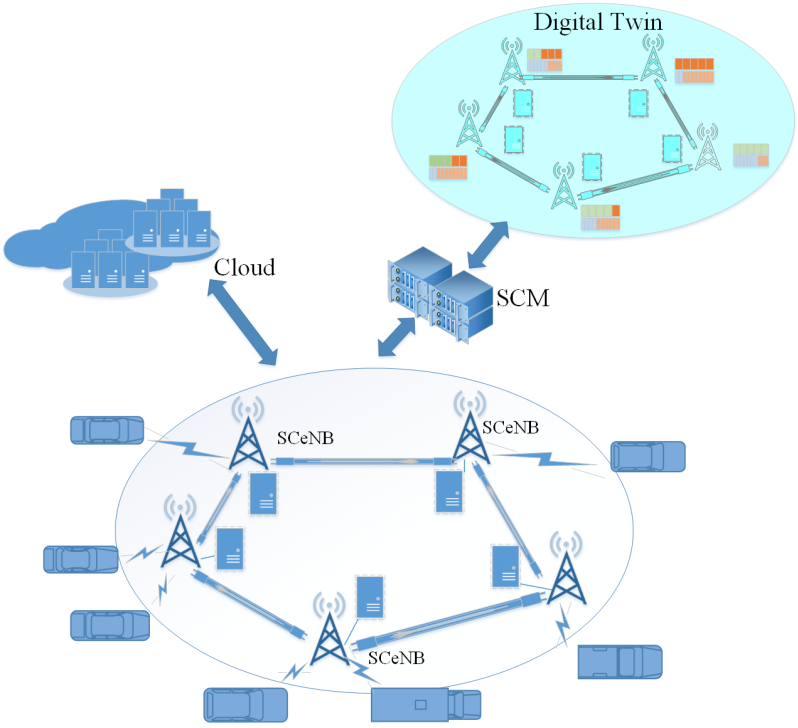

Fig. 1 Architecture of DT-empowered collaborative task offloading system.

the service requests of users.

In addition, we built a digital twin network of edge server nodes in SCM. The digital twin network records the task operation status, computes resource allocation, channel status, and other data of each edge computing node, and utilizes deep reinforcement learning methods to provide intelligent solutions to task offloading and resource allocation for the edge computing cluster.

Time is discretized into different time slots in this system, and the time interval is sufficiently short. Therefore, we can assume that one edge server can only receive at most one service request on each time slot. Meanwhile, the computing resources of each edge server can be virtualized into independent computing resources and allocated to users through virtualization technology. The base station will receive task requests from nearby vehicles at the beginning of each time slot and then send the task information to the SCM. The SCM will make the offloading decision of all tasks in the edge computing cluster according to SCeNB node information recorded by their corresponding digital twins in SCM. Simultaneously, the digital twins in SCM will be updated in accordance with previous decisions and information. Afterward, all decisions are returned to their corresponding SCeNB nodes, and then each SCeNB node allocates sub-channels and computing resources for the service requests of users according to the strategy given by SCM and collaborates with other edge nodes.

The decision in SCM can mainly be divided into two stages. In the first stage, SCM decides on the service requests directly received by each SCeNB node. This stage decides on whether to forward the task or send it to the cloud service layer or execute it in the local server. In the second stage, all forwarded tasks will be auctioned among the digital twins of all SCeNB nodes via the designed auction mechanism. These nodes will provide the bid of computing resources they are willing to allocate for the forwarded tasks. Finally, the auctioned task will be sent to the edge node with the highest bid; this indicates that this node will generate the lowest execution and transmission latency for the forwarded task. If the task is forwarded to any other node, then it cannot generate higher revenues than directly uploading the task to the cloud layer. Thus, the auction will be deemed to have failed, and the forwarded task will be offloaded to the cloud service layer for execution.

3.2 Network communication model

First, we define the request of IoV users to establish a communication model for the network in the IoV edge computing collaborative offloading system accurately. The service request information of a user received by the SCeNB node $Node_i$ in the time slot t is a four-

tuple $r_i(t) = (v_i, p_i, d_i, c_i)$, where v_i represents the maximum data transmission rate between the user and the SCeNB node, p_i represents the signal transmission power of the user, d_i represents the size of application data contained in the service requirement, and c_i represents the amount of computing resource required for the service. If in a certain time slot ξ, then the SCeNB node $Node_i$ does not receive any service request of the user. Thus, let $r_i(\xi) = (0, 0, 0, 0)$.

Users and SCeNB node $Node_i$ are connected wirelessly in this system. Assume that users employ orthogonal frequency division multiplexing multiple access technology[32] to connect to SCeNB nodes for two-way data transmission, wherein the bandwidth of each sub-channel is w. Assuming that the communication interference between different users and the SCeNB node is negligible during the communication between each other, the maximum data transmission rate v_i between the user and the corresponding edge computing node $Node_i$ is

$$v_i = n_i^{\text{cha}} \cdot w \cdot \log(1 + p_i\, h_i\, \sigma^2) \tag{1}$$

where n_i^{cha} represents the number of sub-channels allocated to the service request r_i, and h_i is the channel gain from the user to Node$_i$, which is subject to a Gaussian distribution with a mean value of 0 and a unit variance of σ^2.

The base stations are connected through wired channels, and all the base stations are connected in a ring topology to form a cluster. Therefore, the communication topology between the base stations can be represented by the graphs G^{geo} and G^{graph}, where G^{geo} is used to represent the length of the optical fiber between SCeNB nodes, and G^{graph} is used to represent the distance between nodes on the topological graph. For example, two nodes $Node_m$ and $Node_n$ are directly adjacent, and the geographic distance between the two nodes is 0.9 km; thus, $G_{m,n}^{\text{geo}} = 0.9$, $G_{m,n}^{\text{graph}} = 1$. Another example is the connection between the nodes $Node_p$ and $Node_q$ through the node $Node_r$: the geographic distance between $Node_p$ and $Node_r$ is 0.7 km, and that the geographic distance between $Node_q$ and $Node_r$ is 0.5 km. Thus, $G_{p,q}^{\text{geo}} = 1.2$ and $G_{p,q}^{\text{graph}} = 2$.

We assume that the communication conditions of the links between all base stations are the same. The transmission delay of the wired channel between the base stations is highly correlated with the transmission distance[33], and each transition in a base station will produce additional delay. Let φ_{wire} be the delay required during the transmission process per unit distance for unit data. φ_{BS} is the delay generated after each base station conversion. Then let $Ts_i^{m,n}$ be the wired transmission delay $Ts_i^{m,n}$ between $Node_m$ and $Node_n$, which is given by

$$Ts_i^{m,n} = \varphi_{\text{wire}} \cdot G_{m,n}^{\text{geo}} \cdot d_i + \varphi_{\text{BS}} \cdot G_{m,n}^{\text{graph}} \tag{2}$$

3.3 Service execution delay model

The service request of the user in this system can be executed locally at the SCeNB node. This node receives the request or forwards it to other nodes in the cluster for collaborative computing; the request can also be offloaded to the cloud service layer for execution. The task information of the request, $task_i^k$, will be added to a set $Task_i$, which contains the task information of all tasks running in $Node_i$, after the destination of the user's request $r_i(t)$ has been decided by the SCM. k indicates that it is the k-th task running in $Node_i$. In particular, $task_i^k$ is a seven-tuple that records the running information of the task:

$$task_i^k = (ncha_i^k,\ ncal_i^k,\ pos_i^k,\ lcha_i^k,\ lcal_i^k,\ tcha_{\text{end}}^k,\ tcal_{\text{end}}^k) \tag{3}$$

where $ncha_i^k$ is the number of allocated sub-channels, $ncal_i^k$ is the number of allocated computing resources, pos_i^k is the location for task calculation, $lcha_i^k$ is the number of remaining channels, $lcal_i^k$ is the number of remaining computing resources, $tcha_{\text{end}}^k$ is the end time of the channel occupied by the task, and $tcal_{\text{end}}^k$ is the end time of the computing resource occupied by the task.

3.3.1 Service delay of offloading the task to the local node for computing

Considering the definition of service requests and task running information, the following defines the service delay required to meet user needs in different situations and finally provides the calculation formula of the total service delay of the system.

The delay to be considered when the service request of the user is offloaded to the nearest SCeNB node for calculation includes the transmission delay of uploading service request information r_i and SCeNB node running information $task_i^k$ from the user's vehicle to the SCeNB node, uploading time of r_i and $task_i^k$ to SCM, the time of decision making by digital twins in SCM, the return time of the decisions in SCM to the SCeNB node, the time delay for the SCeNB node to receive the service request data, the calculation delay for processing the service request on the node, and the return delay of the service result returned to the user. The seven processes are represented in Fig. 2. The backhaul delay

Fig. 2 Illustration of the interactive process, where the green line represents the transmission of the information of the request, and orange line represents the transmit of the data of the request data.

can be neglected considering that the data size of the service result is generally smaller than that of the input parameters. Simultaneously, r_i and $task_i^k$ to be uploaded to the digital twin network as well as that for SCM to return the decision to the SCeNB node can also be ignored because of the small data-size. Therefore, the delay generated in processes 1, 2, 4, and 7 in this system is negligible. Moreover, the time required for users to upload their service request $r_i = (d_i, e_i)$ to the SCeNB node can be calculated by Eqs. (4) and (5):

$$Ts_i^{\text{u2e}} = \frac{d_i}{v_i} \tag{4}$$

$$Tc_i^{\text{local}} = \frac{e_i}{n_i^{\text{cal}} \cdot f_{\text{mec}}} \tag{5}$$

where Eq. (4) represents the data transmission delay from the user's vehicle to the edge node, and Eq. (5) represents the calculation delay of processing service request r_i on the local SCeNB node. In addition, n_i^{cal} is the number of computing resources allocated to the service request r_i, and f_{mec} represents the computing rate of a single edge server computing resource. Therefore, the resulting service delay generated in the local edge server can be calculated by the following:

$$T_i^{\text{loacl}} = Ts_i^{\text{u2e}} + Tc_i^{\text{local}} + T_i^{\text{twin}} \tag{6}$$

where T_i^{twin} represents the time for the digital twin network in SCM to make task offloading decisions.

3.3.2 Service delay of offloading the service to the cloud service layer for computing

The interactive delay model in Section 3.3.1 indicates that if the service request of the user is to be offloaded to the cloud server for execution, then the data of the service request must first be uploaded to the SCeNB node through the wireless channel and then sent to the cloud layer by the corresponding SCeNB node through the wired channel. The cloud server has strong computing power; thus, the calculating delay is slightly negligible compared with the transmission delay. Simultaneously, we choose to ignore several extremely small delays. Therefore, the service delay when deciding on offloading the service to the cloud for execution mainly includes the transmission delay of uploading service request data to the edge computing node and the return delay caused by the data transmission between the SCeNB node and the cloud server. The calculation formula can be described as follows:

$$T_i^{\text{cloud}} = Ts_i^{\text{u2e}} + \text{RTT} + T_i^{\text{twin}} \tag{7}$$

The cloud server is geographically far away from the edge computing node. Thus, the forwarding process of the input parameter data by the edge computing node to the cloud and the cloud returning process of the service processing result generally have a similar delay, and the delay has no relationship with the input parameter. Therefore, RTT can be written as

$$\text{RTT} = 2Ts^{\text{cloud}} \tag{8}$$

where Ts^{cloud} denotes the delay of sending the data from the SCeNB node to the cloud server.

3.3.3 Service delay when forwarding tasks to other SCeNB nodes for execution

The IoV edge computing model in Section 3.1 shows that if the service requests of the user must be forwarded to other SCeNB nodes in the cluster for execution, then the raw service data must be uploaded from the vehicle of the user to the SCeNB node through the wireless channel, resulting in delays in local offloading in Section 3.3.1. The delay caused by the data transmission process with a small data size is also ignored. Therefore, the resulting delay when the service request is offloaded to other SCeNB nodes via the directly connected SCeNB node mainly includes the transmission delay of uploading the service request data to the edge computing node, the data transmission delay between nodes through optical fibers, and the time consumption of executing computing tasks on the assisting node. Consequently, the service delay of the forwarded task to other nodes can be expressed as

$$T_i^{\text{forward}} = Ts_i^{\text{u2e}} + Ts_i^{\text{e2e}} + Tc_i^{\text{forward}} + T_i^{\text{twin}} \tag{9}$$

3.3.4 Total delay of the system

The request received by each node in this study has three choices: local offloading, forwarding to other nodes for assistance in computing, and offloading to the cloud server. 0 variables use 0/1 variables, a_i and b_i, to indicate the offloading method of a certain service, where a_i represents execution on the local node:

$$a_i = \begin{cases} 1, & \text{if execute } r_i \text{ in local node;} \\ 0, & \text{if do not execute } r_i \text{ in local node} \end{cases} \tag{10}$$

Meanwhile, b_i represents forwarding of the task to other SCeNB nodes

$$b_i = \begin{cases} 1, & \text{forward } r_i \text{ to other nodes;} \\ 0, & \text{upload } r_i \text{ to cloud service layer} \end{cases} \tag{11}$$

Therefore, the service delay for any service request is

$$T_i = a_i \cdot T_i^{\text{local}} + (1-a_i)\left[b_i T_i^{\text{forward}} + (1-b_i)T_i^{\text{cloud}}\right] \tag{12}$$

Thus, the calculation formula of the global service delay of the system for all users in the time slot t is as follows:

$$T_{\text{total}}(i) = \sum_{i=1}^{N_{\text{node}}} T_i \tag{13}$$

3.4 Problem definition

The purpose of the proposed system is to minimize the average system delay over a long period. The problem can then be modeled as follows:

$$\min_{a,b,n_i^{\text{cal}},n_i^{\text{cha}}} \frac{1}{N_{\text{node}}} \sum_{i=0}^{N_{\text{node}}} \frac{\sum_{t=0}^{\tau} T_i(t)}{\sum_{t=0}^{\tau} Sgn(r_i(t))} \tag{14}$$

$$\text{s.t.} \sum_{task_i^k \in Task_i} n_i^{\text{cal}} \leqslant N_{\text{cal}}, \ \forall i \tag{15}$$

$$\sum_{task_i^k \in Task_i} n_i^{\text{cha}} \leqslant N_{\text{cha}}, \ \forall i \tag{16}$$

$$T_i^{\text{forward}} \leqslant T_i^{\text{cloud}} \tag{17}$$

where $Sgn(r_i(t))$ is the symbol of $r_i(t)$. The value is 1 when $r_i(t)$ is not empty; otherwise, it is 0. The target of the objective function is to minimize the average delay of long-term users. a, b, n_i^{cal}, and n_i^{cha} represent the decision variable in the optimization problem. Constraint (15) means that the number of subchannels allocated to users cannot exceed the number of those owned by the connected SCeNB node in any time slot. Constraint (16) means that the number of computing resources that can be allocated to users in any time slot cannot exceed those owned by the connected SCeNB node. Constraint (17) indicates that the delay caused by the forwarded task must be less than that of the directly uploaded task to the cloud service layer.

4 AUC–AC for DT–Empowered Edge–Cloud Cooperative Offloading System

A deep reinforcement learning method named AUC–AC is designed in this section by combining the traditional A2C algorithm with the auction mechanism to solve the collaborative offloading problem in the IoV edge computing environment. First, the overall architecture of the algorithm is given. Then, the detail of each part is described. Finally, the complete algorithm of AUC–AC is provided.

4.1 Analysis of the auction mechanism model

Part of the edge server nodes chooses to forward their tasks after all edge computing decides to offload directly received tasks, and all the forwarded tasks form a set:

$$task^{\text{forward}} = (task_m^{\text{forward}}, task_n^{\text{forward}}, \ldots) \tag{18}$$

where m and n represent the ID of the SCeNB nodes corresponding to the forwarded task. Differences are observed in the computing and storage resources available for collaboration in each SCeNB node. Thus, we assume that all idle computing resources of each edge server can be used for collaborative computing to reduce the complexity of the problem. At present, many existing schemes mostly adopt the scheme of offloading tasks to their neighboring nodes[7, 18]. However, the deployment of SCeNB nodes becomes increasingly intensive, and data transmission between nodes through optical fibers only generates a small delay. Therefore, only considering the cooperation between neighboring nodes may ignore the potential excellent cooperative offloading scheme.

We believe that any two nodes in the SCeNB node cluster can implement collaborative computing via data transmission through direct optical fiber between two adjacent nodes or forwarding by the medium nodes. Simultaneously, we aim to help SCeNB nodes prioritize the tasks within their coverage areas.

For each forwarded task, the communication delay between the base stations due to forwarding and the number of computing resources that the other server is willing to allocate for cooperation must be comprehensively considered to obtain a corresponding destination edge server for offloading. Meanwhile, the node receiving the computing task must also consider the service requests within its service area that may be sent later. Therefore, through the reinforcement learning combined with the auction mechanism, the tasks that must be forwarded are auctioned among all SCeNB nodes, and the number of computing resources in the action given by the buyer node is the price it is willing to provide. Combined with the data transmission time in the optical fiber between the buyer and seller nodes, the price of all buyers relative to the seller node $Price_i^j$ can be calculated. This price is expressed as follows:

$$Price_m^n = \begin{cases} -(Ts_n^{\text{m2n}} + Tc_n^{\text{forward}}), & m \neq n; \\ -2Ts_m^{\text{cloud}}, & m = n \end{cases} \tag{19}$$

Therefore, the bids of all buyers to the seller node $Node_m$ form a set:

$$Price_m(t) = (Price_m^1, Price_m^2, \ldots, Price_m^i) \tag{20}$$

The ID of the node to which $task_m^{\text{forward}}$ will eventually be offloaded is

$$kw = \text{argmax} Price_m(t) \tag{21}$$

4.2 Framework of reinforcement learning in AUC–AC

4.2.1 State space

The state space of an SCeNB node includes not only the operating status of the node but also its observation of the running status of other nodes. However, if the running state of all nodes is directly used as a state, then the state space dimension will be excessively high, which will lead to the convergence difficulty problem in the reinforcement learning algorithm. We propose a global state representation method based on the convolutional block attention mechanism to solve this problem.

First, we created an environment for each SCeNB node, in which only the information of the corresponding SCeNB node is directly contained. Second, we aggregated the task running status of all edge computing nodes into a three-dimensional graph $F_i^{3D}(t) \in \mathbb{R}^{C\times H\times W}$. Then at time slot t, the state $st_i(t)$ of the SCeNB node $Node_i$ at time slot t can be expressed as

$$st_i(t) = (F_i^{3D}(t), r_i(t), L_i^{\text{cha}}(t), L_i^{\text{cal}}(t), P_i(t)) \tag{22}$$

where $L_i^{\text{cha}}(t)$ and $L_i^{\text{cal}}(t)$ respectively represent the remaining number of sub-channels and computing resources in the SCeNB node, and $P_i(t)$ represents the statistical probability of the node's recent task arrival of the node.

For the task running feature map $F_i^{3D}(t)$, we define $task_i^k$ as the k-th running task in the SCeNB node $Node_i$, and tf_i^k records the running information of the task:

$$tf_i^k = (Ncha_i^k, Ncal_i^k, pos_i^k, Lcha_i^k, Lcal_i^k, \Delta tcha_i^k, \Delta tcal_i^k, R_i^k, T_i^k) \tag{23}$$

where $Ncha_i^k$ is the ratio of the number of allocated sub-channels to the total number of sub-channels, $Ncal_i^k$ is the ratio of the number of allocated computing resources to the total number of resources, and pos_i^k is the location for task calculation, $Lcha_i^k$ is the ratio of the number of remaining channels to the total number of channels, $Lcal_i^k$ is the ratio of the number of remaining computing resources to the number of computing resources of the total computing resources, $\Delta tcha_i^k = tcha_{\text{end}}^k - t_{\text{sys}}$ is the difference between the end time of the channel occupied by the task and the current system time, and $\Delta tcal_i^k = tcal_{\text{end}}^k - t_{\text{sys}}$ is the difference between the computing resource time occupied by the task and the current system time. R_i^k is the reward obtained by the task, and T_i^k is the time delay generated by the task.

Later experiments revealed that the tasks running in the edge server simultaneously will not exceed M. All the tasks added to the SCeNB node can then form a two-dimensional graph. If the number of tasks running in the node does not reach M, then the missing task features will be filled with 0. The size of the final two-dimensional task feature map in the edge server is $9\times M$. Combining the two-dimensional task feature maps of all nodes can help obtain a $9 \times M \times N_{\text{node}}$ feature map $F_i^{3D}(t)$, thus containing each task feature in the channel direction. In particular, considering the difference in the relative position relationship with other nodes for the task feature map of each node, the splicing method should be consistent with all nodes when splicing two-dimensional feature maps. Thus, we will arrange the two-dimensional feature maps of the node $Node_i$ in the center of the graph, and the two-dimensional feature maps of other nodes are spliced on both sides according to the geographical distance relationship between the nodes. Afterward, we extract the features of each task feature through the convolutional neural network and stitch them as part of the state with other values in the system state to form a new system state representation.

Considering the complex relationship between different features in the feature map, and the entire feature map is filled with a large number of all-zero task features, we focus on the notable information when encoding. Moreover, we intend to provide attention to the information in the direction of the task feature and the spatial direction between different SCeNB nodes. Therefore, we introduce the convolutional attention mechanism CBAM[34], and the newly generated environment state representation is $st_i^{\text{cbam}}(t)$ after the features are extracted by the convolutional block attention module.

Figure 3 demonstrates that forwarding the channel and spatial attention modules helps obtain an overall attention map for a feature map inputted into CBAM. Specifically, for the three-dimensional task feature map of a node, its one-dimensional channel attention map is first multiplied element by element. Then, the feature map and its two-dimensional spatial attention map are multiplied element by element to obtain the final overall

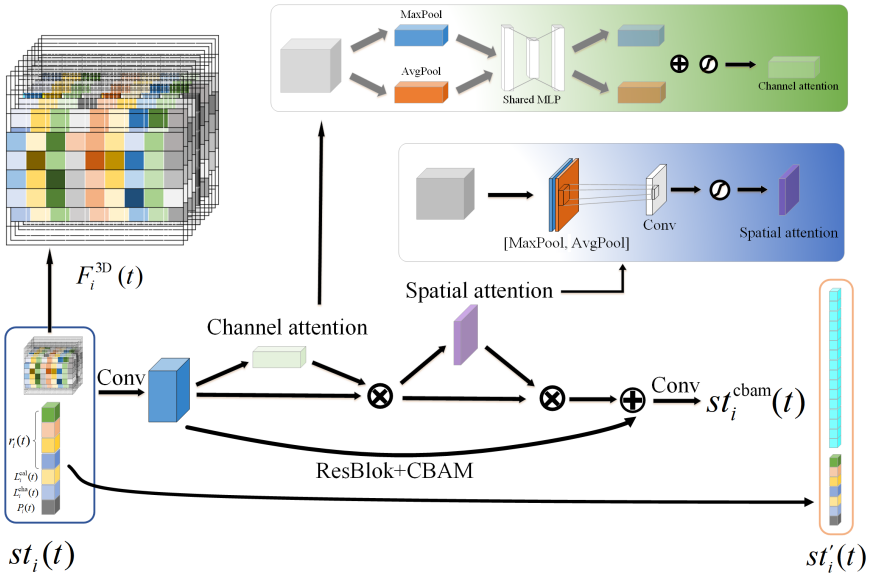

Fig. 3 Global state representation based on convolutional attention mechanism.

attention map. Finally, the overall attention and feature maps are added to obtain the task feature map of the SCeNB node. A 16-dimensional state representation $st_i^{\text{cbam}}(t)$, which is spliced with other state values in the state space $st_i(t)$ to form a new 23-dimensional state representation $st_i(t)$, is obtained after passing two convolutional layers.

4.2.2 Action space

The action space of MDP in the current study contains the following three elements: the position of user task offloading, the number of sub-channels, and the number of sub-resources allocated to the service request of the user. We use one-hot vectors $(0/1)^{\text{pos}}$, $(0/1)^{\text{cha}}$, and $(0/1)^{\text{cal}}$ for representation. Assume that the maximum number of sub-channels and the maximum number of computing resources are 4 and 3, and the task is executed locally, then $(0/1)^{\text{pos}} = (1,0,0)$, $(0/1)^{\text{cha}} = (0,0,1,0,0)$, and $(0/1)^{\text{cal}} = (0,1,0,0)$. Therefore, the action space is the Cartesian product of the three, which is defined as

$$A = (0/1)^{\text{pos}} \times (0/1)^{\text{cha}} \times (0/1)^{\text{cal}} \tag{24}$$

where $\times$ represents the Cartesian product, and $a(t) \in A$ is used to represent an action performed during the time slot t in the action space.

4.2.3 Reward function

The purpose of offloading the service in this paper is to reduce the sum of the long-term average service delay of all users. We define the instant reward $R(t)$ obtained by the action $a(t)$ in the system state $st(t)$. Each decision is divided into two stages: the first stage is the decision on the service request directly received, and the second stage is the auction of forwarded tasks. Simultaneously, we hope that each edge server will pay attention to its service first. The priority of the directly received service request is higher than that of the forwarded task. In addition, we attempt to modify the situation where the allocated resources exceed the existing resources via the reward function to meet the Constraints (15) and (16).

We divide the reward into two parts: one part is the reward of saved time compared with the task execution on the user side, which is called time reward; and the other part is the error penalty for the allocated resources exceeding the allocatable resources, which is called error reward.

For the time reward in the process of dealing with $task_i^k$ generated by the service request $r_i(t)$ sent directly, the calculation methods $Rt_i^{\text{direct}}(t)$ corresponding to the three kinds of offloading decisions are as follows:

$$Rt_i^{\text{direct}}(t) = \begin{cases} T_i^{\text{user}} - T_i^{\text{loc}}, & a_i = 1, b_i = 0; \\ T_i^{\text{user}} - T_i^{\text{cloud}}, & a_i = 0, b_i = 1; \\ T_i^{\text{user}} - T_i^{\text{forward}} \times (1 + \lambda_{\text{forward}}), & a_i = 0, b_i = 0 \end{cases} \tag{25}$$

where T_i^{user} represents the time delay caused by the task directly executed on the user side. The time saved by maximizing computational offloading compared with executing tasks directly on the client-side is essentially the same as minimizing the average computational offloading service latency. Thus, the time rewards obtained from the task running in the local edge node

or the cloud service layer have difference in service delay between the IoV user side and the corresponding offloading SCeNB node.

For the time reward from the forwarded task, the reward must be multiplied by a coefficient to suppress the blind forwarding task of the node. Therefore, for the SCeNB node which sends the task, the time reward $Rt_{\text{send}}^{\text{forward}}$ in the auction stage is

$$Rt_{\text{seller}}^{\text{forward}} = \begin{cases} T_i^{\text{forward}} \times \lambda_{\text{forward}}, & a_i = 1; \\ T_i^{\text{cloud}} \times \lambda_{\text{cloud}}, & a_i = 0 \end{cases} \tag{26}$$

Therefore, the total reward of two stages R^{forward} through forwarding for the forwarded task $task_i^{\text{forward}}$ can be indicated as follows:

$$Rt_i^{\text{forward}} = \begin{cases} T_i^{\text{user}} - T_i^{\text{forward}}, & a_i = 1; \\ T_i^{\text{user}} - T_i^{\text{forward}} \times (1 + \lambda_{\text{forward}}) + \\ \quad T_i^{\text{cloud}} \times \lambda_{\text{cloud}}, & a_i = 0 \end{cases} \tag{27}$$

where λ_{cloud} represents the discount generated by offloading the task to the cloud layer if the forwarded task auction fails. For the SCeNB node which receives the forwarded task, the time reward $Rt_{\text{receive}}^{\text{forward}}$ is

$$Rt_{\text{buyer}}^{\text{forward}} = \begin{cases} \rho \times Rt^{\text{forward}}, & a_i = 1, b_i = 1; \\ 0, & a_i = 0 \text{ or } b_i = 0 \end{cases} \tag{28}$$

where ρ is also a discount factor for increasing the priority of the service request received by itself. Therefore, the edge server can give priority to ensuring that the service request directly received from its service area can be offloaded first.

The error reward is then discussed. The error of allocating resources can be divided into two parts: one is the error of allocating channels, and the other is the error of allocating computing resources. We set the error of allocating channels as δ_i^{cha} and the error of allocating computing resources as δ_i^{cal}. Then,

$$\delta_i^{\text{cha}} = \begin{cases} \sum_{k=1}^{K} n_k^{\text{cha}} - N_i^{\text{cha}}, & \sum_{k=1}^{K} n_k^{\text{cha}} \geqslant N_i^{\text{cha}}; \\ 0, & \sum_{k=1}^{K} n_k^{\text{cha}} < N_i^{\text{cha}} \end{cases} \tag{29}$$

$$\delta_i^{\text{cal}} = \begin{cases} \sum_{k=1}^{K} n_k^{\text{cha}} - N_i^{\text{cal}}, & a = 1,\ b = 0; \\ n_i^{\text{cal}}, & a = 0; \\ 0, & \text{other} \end{cases} \tag{30}$$

where K indicates the number of the forwarded tasks.

The channel error δ_i^{cha} in all cases only considers the number of allocated channels exceeding the number of available channels. The error of computing resources δ_i^{cal} is considered to be the allocated resources that exceed the number of the available computing resources when the task is directly executed on the local node. However, the error does not consume computing resources when it is uploaded to the cloud service layer or forwarded to other nodes. In the two cases, the error is equal to the number of computing resources allocated in the action, Therefore, the error reward of action $a(t)$ is

$$\delta_i = -\eta \times [(1 - \varepsilon) \times \delta_i^{\text{cha}} + \varepsilon \times \delta_i^{\text{cal}}] \tag{31}$$

where η is the coefficient for the error reward, which is used to indicate the importance of the error reward in the reward; ε is used to represent the weight between channel allocation error and computing resource allocation errors.

4.2.4 MDP process

The digital twin of each SCeNB node continuously interacts with the reinforcement learning model in this paper. The service offloading strategy is the probability of the action $a_i(t)$ made in the state $st_i(t)$, which is denoted as $\pi(a_i(t)|st_i(t))$. Let $st(D)$ denotes the system termination state, D is the time step of the termination state, and the cumulative reward with discount formed by the MDP can be calculated as

$$R = \sum_{t=0}^{D-1} \gamma^D \sum_{i=1}^{N_{\text{node}}} (Rt_i(t) + \delta_i) \tag{32}$$

where γ represents the discount rate, which is a constant between [0,1].

4.2.5 AUC-AC with actor-critic network

Reinforcement learning algorithms can generally be divided into two categories: value-based and strategy-based methods. The actor–critic algorithm combines value function- and strategy-based methods. First, the actor observes the actions generated in accordance with the strategy-based method by the environment and then sends the generated action to the critic network based on the value function for evaluation, thus guiding the agent to find the optimal action strategy.

Simultaneously, for the edge computing cooperative offloading problem in the IoV scene, the synchronization of decisions made by all nodes should be ensured to meet the proposed design with the two decision-making stages and the auction mechanism. We choose A2C to solve the aforementioned problem considering the synchronous interaction of all the agents with the corresponding environment in A2C. The proposed architecture, namely the A2C-based reinforcement learning collaboration framework, is shown in Fig. 4. However, problems caused by sparse rewards may emerge in this process.

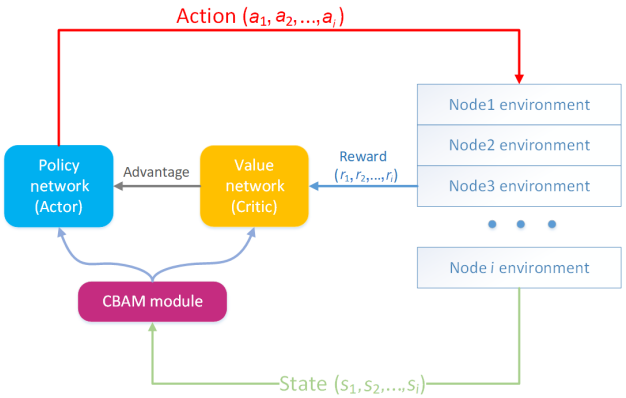

Fig. 4 Architecture of the proposed A2C-based reinforcement learning collaboration framework.

On the one hand, if a node does not receive a service request directly sent to it in the time slot in the first stage, then an empty service request will still be generated and sent to SCM for decision making. If empty service requests account for most service requirements received, then the problem of sparse rewards will emerge. On the other hand, among all the nodes participating in the auction in the second stage of the auction mechanism, only the seller and buyer node with a successful bid will be rewarded. Thus, all other buyer nodes cannot obtain the reward, which will also lead to the phenomenon of sparse reward.

N-step bootstrap has been proven to be effective when faced with such problems[35]. One-step TD can only learn when the reward information is sampled in one step when the reward is remarkably sparse, while the N-step bootstrap can sample N steps. The information can be learned as long as a rewarding step is available. We attempt to change the length in the N-step bootstrap to realize variability in our algorithm, combining the step of the direct decision in a time slot with that in the auction process as a bootstrap. The bootstrap set of each time slot can be presented as follows

$$ST(t)=(st_{\text{direct}}(t), st^1_{\text{forward}}(t), \ldots, st^k_{\text{forward}}(t)) \quad (33)$$

Expanding the calculation method of the target value in the one-step method to the N-step method, the target value $G_i^{t:(t+K)}(t+k)$ can be estimated by the following

$$G_i^{t:(t+K)}(t+k)=R_i^{\text{direct}}(t)+\gamma R_i^{\text{forward}}(t+1)+\cdots+ \gamma^k R_i^{\text{forward}}(t+k)+\gamma^{k+1}V(st_i(t+k+1)) \quad (34)$$

Therefore, the error of the critic network is expressed as

$$L(\theta^v)=\frac{1}{N_{\text{node}}}\sum_{n=1}^{N_{\text{node}}}\sum_{k=0}^{K}(G_i^{t:t+K}(t+k)-V(st_i(t)))^2 \quad (35)$$

For a single-action policy network $\pi_\theta(a(t)|st(t))$, the actor's loss function can be expressed as

$$L(\theta^\pi)=\frac{1}{N_{\text{node}}}\sum_{n=1}^{N_{\text{node}}}\sum_{k=0}^{K}\log P_{\pi_\theta}(a_{t+k}|st_i(t+k))A_i(t) \quad (36)$$

where $A_i(t)=G_i^{t:(t+K)}(t+k)-V_i(st(t))$ represents the error between the target value of the action performed and the true value function when the strategy π_θ with the parameter θ is used. In addition, the action space is the Cartesian product of three elements, which leads the complexity of the action space dimension to be $O(N^{\text{pos}}\cdot N^{\text{cha}}\cdot N^{\text{cal}})$. The action space dimension will be excessively large as the number of available sub-channels in the system and the number of computing resources increase, resulting in the slow of the training process and difficulty in convergence. This article refers to the method in Ref. [20] to change the traditional single-action output network to a policy network that can output the probability distribution of multiple sub-actions simultaneously, i.e., $\pi_\theta(a^{\text{pos}}(t), a^{\text{cha}}(t), a^{\text{cal}}(t)|st(t))$, to solve these problems. Therefore, the dimension complexity of the action space changes from $O(N^{\text{pos}}N^{\text{cha}}N^{\text{cal}})$ to $O(N^{\text{pos}}+N^{\text{cha}}+N^{\text{cal}})$. The loss function of the multi-action output strategy network is then computed as follows:

$$\begin{aligned}L(\theta^\pi)=\frac{1}{N_{\text{node}}}\sum_{n=1}^{N_{\text{node}}}\sum_{\tau=0}^{K}&(\log P_{\pi_\theta}(a^{\text{pos}}_{t+\tau}|st_i(t+k))+\\ &\log P_{\pi_\theta}(a^{\text{cha}}_{t+\tau}|st_i(t+k))+\\ &\log P_{\pi_\theta}(a^{\text{cal}}_{t+\tau}|st_i(t+k)))\cdot A_i(t)\end{aligned} \quad (37)$$

In addition, A2C may converge to a specific action. We have added entropy loss to allow the network update to process the generation of additional randomness to address this issue[36], thereby encouraging further exploration. The entropy can be expressed as

$$E=-\sum_{j}\sum_{i}^{N_{\text{node}}}P_{\pi_\theta}(a_t^j)\log(P\pi_\theta(a_t^j)),\quad j\in\{\text{pos},\text{cal},\text{cha}\} \quad (38)$$

Thus, the loss function of the total network becomes

$$L_{\text{total}}=L(\theta^\pi)+\alpha L(\theta^v)+\beta E \quad (39)$$

where α and β are the weights of the critic network and entropy, respectively.

Finally, we describe our proposed A2C-based service offloading method, namely AUC–AC, in Algorithm 1. Digital twins of all SCeNB nodes are created in SCM in the IoV edge computing scene, and their respective

Algorithm 1 AUC–AC

Input: $N^{\text{cha}}, N_{\text{cal}}, N_{node}, t_{\max}$
Output: $\pi_\theta(a^{\text{pos}}(t), a^{\text{cha}}(t), a^{\text{cal}}(t)|st(t))$

Randomly initialize the policy network parameters θ^π, critic network parameter θ^v;
Initialize the time slot $t = 0$;
repeat
  reset gradient $d\theta^\pi \leftarrow 0, d\theta^v \leftarrow 0$;
  generate $st(t) = (st_1(t), st_2(t), \ldots, st_{N_{node}}(t))$ by Eq. (22);
  input $st(t)$ into Algorithm 2 to generate offloading decision;
  calculate loss function by Eq. (39);
  backward the loss and update θ^π, θ^v;
  $t \leftarrow t + 1$;
until $t > t_{\max}$;

reinforcement learning environments are created by the digital twins to interact with the AUC–AC network. Each SCeNB node receives user request information from their respective service area at the beginning of each time slot and uploads it to the SCM. Simultaneously, the information regarding tasks running in the SCeNB node is uploaded to update the digital twin in the SCM. The network in the SCM first makes decisions for service requests sent directly from all nodes, and then all the forwarded tasks in the previous decision will be auctioned individually through the auction mechanism in Algorithm 2. The process will be repeated until all forwarded tasks have been auctioned. Afterward, the decisions in the SCM are returned to each node, and each node performs task offloading and calculation according to the decision delivered by the SCM. The above process will be repeated in the later time slots. The critic network calculates the expected value of the cumulative reward of the process, uses N-step bootstrap to approximate the value function of each decision, and calculates the cumulative gradient of the two networks according to Eq. (39). The gradient is retrograded to update the network parameter and start the next round of the training process until the number of training steps is exceeded. We generate the requests of IoV users through the simulated environment for algorithm training. The policy network can be deployed on the SCM after the training is completed, and collaborative task offloading is executed in the real environment.

5 Experimental Results and Analysis

This section first describes the experimental platform and the simulation environment used in the experiment.

Algorithm 2 Auction mechanism

Input: $st(t)$
Output: $(G_i^{t:t+k}(t), \ldots, G_i^{t:(t+k)}(t+K))$

for $i \in N_{\text{node}}$ **do**
  choose actions $a_i(t)$ according to $\pi_\theta(a_t|s_t)$
  get a_i and b_i by Eqs. (10) and (11)
  obtain reward Rt_i^{direct} by Eq. (25);
  update the state to $st_i(t+1)$;
  if $a_i^{\text{pos}}(t) == 2$ **then**
    Add $task_i^t$ to $task^{\text{forward}}(t)$;
  end
end
let K represent the number of the element in set $task^{\text{forward}}(t)$;
for $k = 1, 2, \ldots, K$ **do**
  for $i \in N_{\text{node}}$ **do**
    take action $a_i(t+k)$ according to $\pi_\theta(a_t|s_t)$;
    calculate $Price_k^i(t)$ by Eq. (19);
    add $Price_k^i(t)$ to the set $Price_k(t)$;
    store the value function $V(st_i(t+k))$;
  end
  $kw = \text{argmax}(Price_k(y))$;
  get a_i and b_i by Eqs. (10) and (11);
  obtain Rt_i^{forward} by Eqs. (26)–(28);
  update the state to $st_i(t+k+1)$;
  store $R_i(t+k)$ and $st_i(t+k+1)$;
end
for $i \in N_{\text{node}}$ **do**
  for $k = K, K-1, \ldots, 1$ **do**
    if $k == K$ **then**
      $G_i^{t:(t+K)}(t+k) = R_i(t+k) + \gamma V(st_i(t+K+1))$
    end
    $G_i^{t:(t+K)}(t+k) = R_i(t+k) + \gamma G_i^t(t+k+1)$
  end
end

This section then briefly introduces the experimental parameter settings and then conducts a convergence experiment on the AUC–AC method, which proves the feasibility of AUC–AC. Therefore, we conducted a detailed performance experiment and evaluation of AUC–AC for different system resources status. Table 2 shows the occurrence probability and data size for different tasks and computation sizes based on the comparison and analysis with the existing service offloading baseline algorithms.

5.1 Experiment platform and dataset

We collect the data from the real vehicle-mounted RSU communication in Nanjing, China. The data of the data set are from September 1st to 29th, 2014. The data format has four-tuple (*t*, *id*, *speed*, *color*), where *t* represents the time to establish a connection with

the SCeNB node, *id* denotes the ID of the connected SCeNB node, *speed* corresponds to vehicle speed of the user, and *color* is the color of the user's vehicle. An information record of IoV user data indicates that the user is connected to the SCeNB node once. We assume that the duration for each user to connect with each base station is 2 min. Within 2 min, task requests with a certain probability will be randomly generated in each time slot. In addition, the ID of each SCeNB node in the data set corresponds to a specific SCeNB node and its geographic latitude and longitude coordinates. The distribution and communication topology of the SCeNB nodes specifically selected in the experiment are shown in Fig. 5.

Intermittent probability of user requests is used to simulate the user request generation among real roads to evaluate the offloading performance of the proposed AUC–AC and other baseline algorithms in different parameter settings. The user vehicle establishes a connection with the edge; thus, it obeys a certain probability to generate tasks every 0.5 s randomly. After determining the generation of a request, one of six different tasks is produced in accordance with the set probability to generate the specific task based on the parameter setting in Ref. [37]. The parameters for task generation are shown in Table 2.

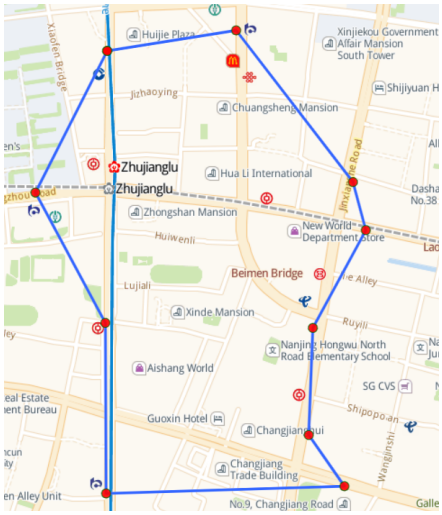

Fig. 5 Position of the ten selected SCeNB nodes.

Table 2 Occurrence probability and data size for different tasks and computation sizes.

Data size	Gigacycles	Probability	Data size	Gigacycles	Probability
60	1.2	0.10	30	0.7	0.30
45	0.8	0.15	25	0.6	0.20
40	0.7	0.15	15	1.2	0.10

The number of channels and computing resources and their division can have a considerable impact on the decision of the SCeNB node. Referring to Refs. [36, 37], the bandwidth size of different 5G IoV is set to 20, 30, and 40 GHz, and the computational capacity of the edge server is 2.5, 5, and 7.5 Gigacycles/s, respectively. The sub-channel bandwidth size is 1 GHz, and the computational sub-resource is 0.5 Gigacycles/s. The other parameters in the training are shown in Table 3.

5.2 Experiment results

5.2.1 Convergence analysis

In reality, using SCeNB nodes aims to reduce latency and improve user experience. Therefore, this section will use the time difference between the AUC–AC and local execution in processing the same service request as the criterion in each experiment.

Every bandwidth of the SCeNB node is set to 20 GHz, and the capability is calculated to be 10 Gigacycles/s to estimate the convergence of AUC–AC. The tested learning rate is 5×10^{-4}, 1×10^{-4}, 5×10^{-5}, 3×10^{-5}, and 1×10^{-5}. Figure 6 shows that the rewards under other learning rates have reached convergence after training for 20 000 epochs, except for the case where the learning rate is 1×10^{-5}. Considering time saving tests, time savings compared to local computing under the five learning rates are shown in Fig. 7. Time savings converge at different rates. However, they all converged to around 2.4 s after approximately 15 000 epochs, and

Table 3 Experiment parameters.

Variable	Value	Variable	Value
p	2 ± 0.2 W	f	0.3 ± 0.03 Gigacyle/s
h	144 ± 14.4 dB	$t_{\text{off}}^{\text{cloud}}$	1000 ms
σ^2	1.5×10^{-8}		

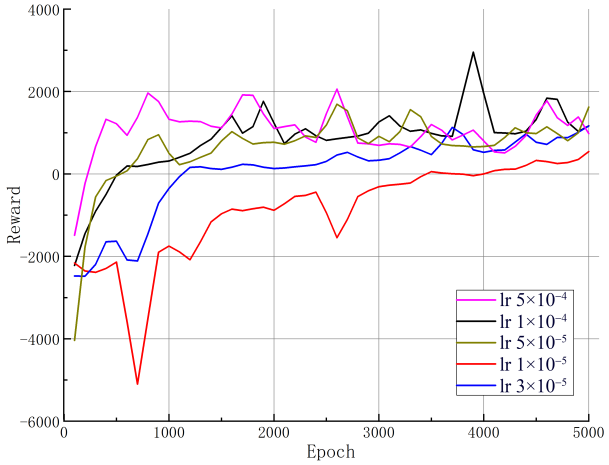

Fig. 6 Reward convergence results.

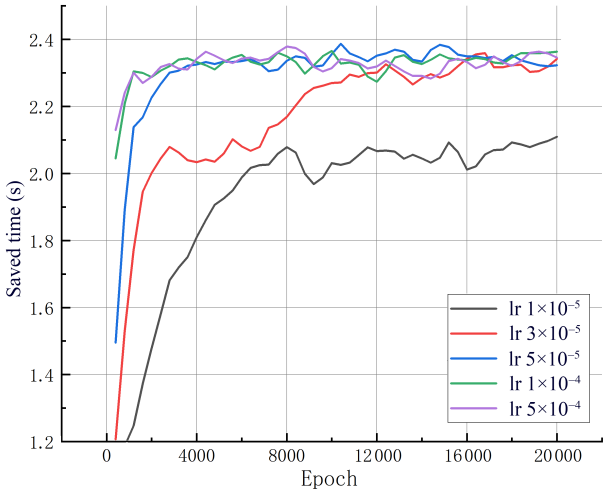

Fig. 7 Time saving convergence results.

the relatively best results were achieved when the learning rate is 3×10^{-5}. 3×10^{-5} is also selected as the training learning rate in the subsequent experiments.

5.2.2 SCeNB node quantity analysis

In practical application scenarios, the 5G MEC network is presented differently. Different numbers of 5G SCeNB nodes can form various edge computing rings. The number of nodes in the communication ring can influence the decision of AUC–AC. Hence, this section will show the decision capability of AUC–AC in different SCeNB node numbers. Figure 8 shows that the collaboration effect significantly increases from 1 to 6, then the effect of the increase gradually diminishes, and the performance decreases slightly when the number of nodes is 10. Therefore, increasing the number of nodes can significantly reduce the response delay of the system to user requests when the number of nodes in the SCeNB node cluster is small. However, the effect is insignificant and even side effects may occur when the number of nodes is large.

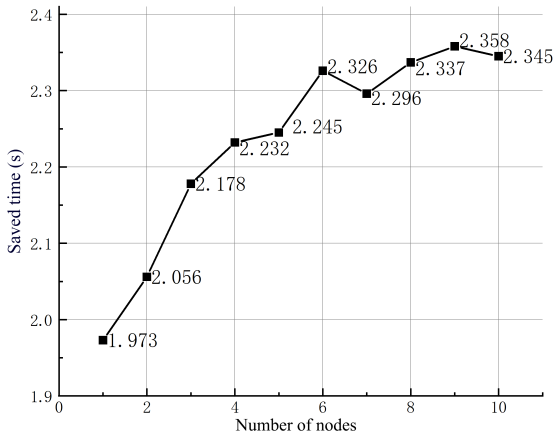

Fig. 8 Impact of different SCeNB node quantities.

5.2.3 Task density and collaboration frequency

The collaboration mainly aims to cope with high task density. Particularly, when a single SCeNB node receives a large number of tasks, this node can forward the tasks received beyond its carrying capacity to other nodes to reduce the burden on its SCeNB node and improve the utilization of channel and computational resources of the entire SCeNB node cluster. In our simulation experiment, tasks are generated at each node with a certain probability p in each 0.5 s. We generate random noise for the task generation interval to further explore the practical role of the collaboration mechanism. We then attempted to train the A2C model under different task generation probabilities, and the result is shown in Fig. 9. With a low probability of task appearance when p is 0.4, 0.5, and 0.6, the frequency of task forwarding decreases significantly until 0 as the epochs of training progress. The nodes have additional free channels and computational resources in the case of fewer task occurrences. Therefore, cooperative offloading is unnecessary at this time. However, as the probability of task generation increases, the total amount of forwarding tasks in the system shows a trend of increasing and gradually stabilizing. Thus, the collaboration mechanism plays an active role in the rising number of tasks and effectively adjusts the task distribution in different SCeNB nodes.

5.2.4 Comparison with other algorithms

We choose six different algorithms in this section for comparison with the AUC–AC.

- Random (random offloading): When the user sends

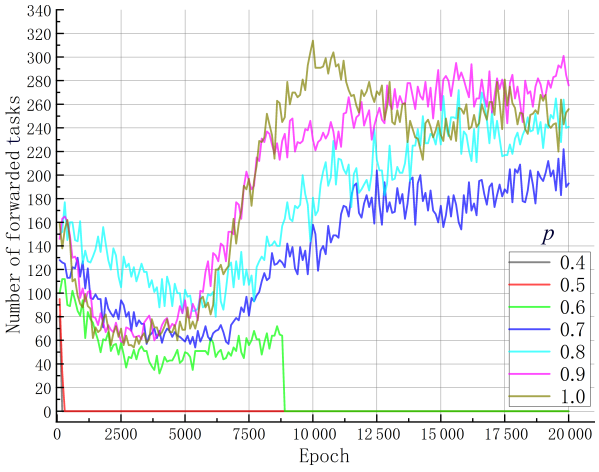

Fig. 9 Number of forwarded tasks with different probabilities of task generation.

a request, the MEC will offload the task, allocate channels, and calculate resources randomly.

• GN (greedy algorithm with no collaboration): When the user sends a request, MEC will allocate all channels and calculate resources to the task. If no free calculation resource is available, then the request will be sent to the cloud layer.

• GC (greedy algorithm with collaboration): When the user sends a request, the SCeNB node will choose the offloading position, which saves the most time among local nodes, other nodes, and cloud layers. If the task is decided to be forwarded, then it will be sent to the node with the relatively least service delay. The process of forwarding the task will also be stimulated to obtain the instant maximum benefit.

• A2C–NC (A2C without collaboration): We still use ten environments to stimulate ten SCeNB nodes but remove the task feature map, and each node would not cooperate.

• MADDPG: As mentioned in Section 3, that if one of the representative algorithms in multi-agent reinforcement learning, MADDPG can utilize global agent state information to make decisions, and multi-agents will attempt to achieve global optimality through cooperation.

The variation of the global average user service latency generated by different service offloading methods is shown in Fig. 10. In three experiments, SCeNB nodes were configured with 40, 20, and 30 sub-channels and 5, 10, and 15 computing resources, respectively. The task generation probability is 0.6. The results indicate that AUC–AC outperforms five other methods under three kinds of SCeNB node configurations. The time saving remains stable around 2.4 s when AUC–AC is applied, while the five other methods were affected to varying degrees by configuration changes. Among these methods, another two reinforcement learning methods demonstrate advantages over the three remaining methods, and the largest advantage of AUC–AC emerges when computing resources are insufficient. This condition can be attributed to the forwarding and allocation of tasks by AUC–AC due to an improved cooperation mechanism when computing resources are scarce. By contrast, the difference is small between AUC–AC and the two other reinforcement learning methods when the computing resource is sufficient. The SCeNB node can effectively execute most of the services locally; therefore, the advantages of the cooperation mechanism at this time are not evident.

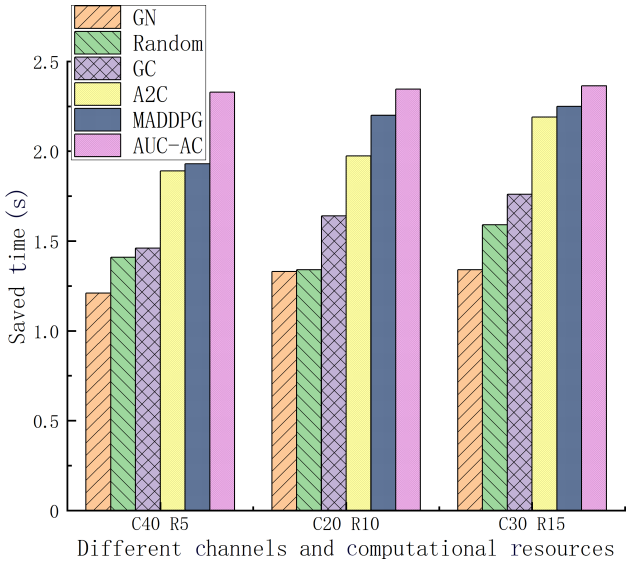

Fig. 10 Comparison of saved time from six strategies with different numbers of channels and computing resources.

In addition to the performance comparison between methods under different edge node configurations, we also compare the performance of various methods under different task densities. In this experiment, the edge node is equipped with 20 sub-channels and 10 computing resources. The plot of task generation probability versus time saving in Fig. 11 shows that AUC–AC has good adaptability to user requests with different task densities, which has remained stable at approximately 2.4 s. Meanwhile, the performance of MADDPG is also stable but slightly worse than AUC–AC, which is around 2.2 s. The performance of A2C is poor when task density is low. However, the time saving gradually increases as the task density grows, even surpassing MADDPG when the task probability is over 0.9. This phenomenon may be due to a certain side effect of global

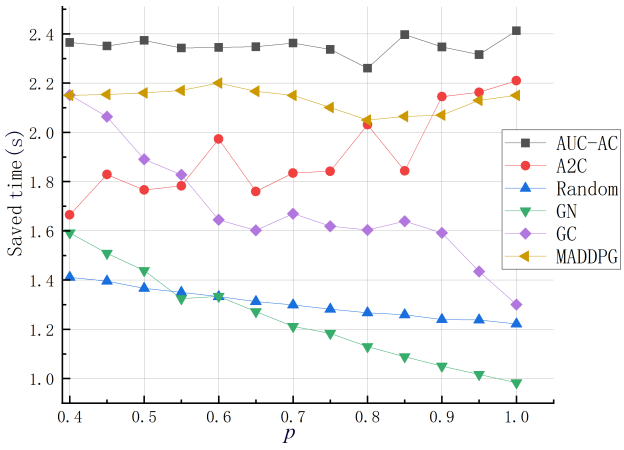

Fig. 11 Comparison of saved time from six different strategies with different task generation probabilities.

information by multi-agents on its decision making when the task density is high. Moreover, GC has good performance when the task density is low, but the decision making caused by the single greedy strategy is no longer excellent when the task density is high. The remaining two algorithms, Random and GN, perform worse than the others in all task densities.

5.2.5 Ablation studies

We conduct ablation experiments to verify their impact on the decision making of the model and demonstrate the effectiveness of the proposed representation method for global information and attention mechanism. Global state representation and global state with CBAM are investigated with different edge node configurations. The experimental results are shown in Fig. 12. Compared with not using global state representation, using global information improves the time saving by approximately 2.2%. The time saving is further improved by approximately 1.7% after adding the attention mechanism. Therefore, the overall system performance was improved by 4% through the global state representation and attention mechanism. This phenomenon is due to the effective discovery of useful information in the global information by the attention mechanism and its assignment of high weights to such information to extract information from the global state efficiently and convert it into a state vector.

Overall, the AUC–AC method proposed in this paper is effective in reducing the average service response delay in MEC networks with different numbers of computational and channel resources. Compared with MADDPG, greedy, random, and some non-collaborative algorithms, AUC–AC has a 9%–70% performance improvement. Therefore, AUC–AC can effectively improve user experience in different 5G edge computing environments.

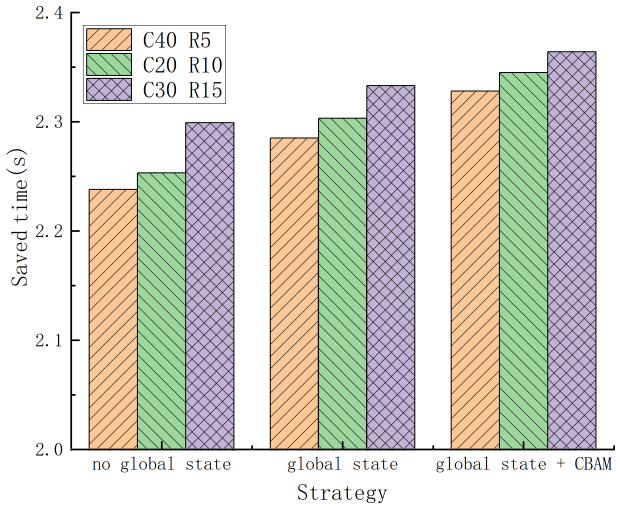

Fig. 12 Comparison of using global state information and the attention mechanism.

6 Conclusion

We investigated the DT-empowered SCeNB collaboration system and proposed a reinforcement learning framework, namely AUC–AC, for sub-channel and sub-computing resource allocation and the collaboration strategy in edge–edge and edge–cloud. The collaboration process can be simulated in SCM by adopting the auction mechanism and digital twins, and the application of the CBAM-empowered global state representation method helps the digital twin of each SCeNB node effectively observe their environment. The results showed that the proposed method has a lower service delay compared with other baseline task offloading methods. Moreover, some other advantages and the relationship between the number of SCeNB nodes and improvement of delays are discussed in accordance with the simulation result. Additional optimization targets, such as the system energy consumption and loading balancing, will be considered on the basis of the proposed AUC–AC framework for future work.

Acknowledgment

This research was supported by the Natural Science Foundation of Jiangsu Province of China (No. BK20211284), the Financial and Science Technology Plan Project of Xinjiang Production and Construction Corps (No. 2020DB005), the National Natural Science Foundation of China (No. 61872219), and NUIST Students' Platform for Innovation and Entrepreneurship Training Program (No. 202110300569).

References

[1] M. Patel, D. Sabella, N. Sprecher, and V. Young, Contributor, Huawei, Vice Chair ETSI MEC ISG, Chair MEC IEGWorking Group, p. 16, 2015.

[2] H. M. Song, H. R. Kim, and H. K. Kim, Intrusion detection system based on the analysis of time intervals of CAN messages for in-vehicle network, in *Proc. 2016 Int. Conf. on Information Networking* (*ICOIN*), Kota Kinabalu, Malaysia, 2016, pp. 63–68.

[3] W. Y. Zhang, Z. J. Zhang, and H. C. Chao, Cooperative fog computing for dealing with big data in the internet of vehicles: Architecture and hierarchical resource management, *IEEE Commun. Mag.*, vol. 55, no. 12, pp. 60–67, 2017.

[4] X. L. He, Z. Y. Ren, C. H. Shi, and J. Fang, A novel load balancing strategy of software-defined cloud/fog networking in the Internet of Vehicles, *China Commun*., vol. 13, no. 2, pp. 140–149, 2016.

[5] J. K. Ren, G. D. Yu, Y. H. He, and G. Y. Li, Collaborative cloud and edge computing for latency minimization, *IEEE Trans. Veh. Technol*., vol. 68, no. 5, pp. 5031–5044, 2019.

[6] W. B. Fan, L. Yao, J. T. Han, F. Wu, and Y. A. Liu, Game-based multitype task offloading among mobile-edge-computing-enabled base stations, *IEEE Internet Things J*., vol. 8, no. 24, pp. 17691–17704, 2021.

[7] W. Sun, H. B. 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, 2020.

[8] Y. L. Lu, X. H. Huang, K. Zhang, S. Maharjan, and Y. Zhang, Low-latency federated learning and blockchain for edge association in digital twin empowered 6G networks, *IEEE Trans. Industr. Inform*., vol. 17, no. 7, pp. 5098–5107, 2021.

[9] C. Gehrmann and M. Gunnarsson, A digital twin based industrial automation and control system security architecture, *IEEE Trans. Industr. Inform*., vol. 16, no. 1, pp. 669–680, 2020.

[10] K. R. Alasmari, R. C. Green II, and M. Alam, Mobile edge offloading using Markov decision processes, in *Proc. 2nd Int. Conf. on Edge Computing - EDGE 2018*, Seattle, WA, USA, 2018, pp. 80–90.

[11] K. Arulkumaran, M. P. Deisenroth, M. Brundage, and A. A. Bharath, Deep reinforcement learning: A brief survey, *IEEE Signal Process. Mag*., vol. 34, no. 6, pp. 26–38, 2017.

[12] X. L. Xu, B. W. Shen, S. Ding, G. Srivastava, M. Bilal, M. R. Khosravi, V. G. Menon, M. A. Jan, and M. L. Wang, Service offloading with deep Q-network for digital twinning-empowered internet of vehicles in edge computing, *IEEE Trans. Industr. Inform*., vol. 18, no. 2, pp. 1414–1423, 2022.

[13] R. Dong, C. Y. She, W. Hardjawana, Y. H. Li, and B. Vucetic, Deep learning for hybrid 5G services in mobile edge computing systems: Learn from a digital twin, *IEEE Trans. Wirel. Commun*., vol. 18, no. 10, pp. 4692–4707, 2019.

[14] Z. L. Cao, P. Zhou, R. X. Li, S. Q. Huang, and D. P. Wu, Multiagent deep reinforcement learning for joint multichannel access and task offloading of mobile-edge computing in industry 4.0, *IEEE Internet Things J*., vol. 7, no. 7, pp. 6201–6213, 2020.

[15] H. F. Lu, C. H. Gu, F. Luo, W. C. Ding, S. Zheng, and Y. F. Shen, Optimization of task offloading strategy for mobile edge computing based on multi-agent deep reinforcement learning, *IEEE Access*, vol. 8, pp. 202573–202584, 2020.

[16] S. Gronauer and K. Diepold, Multi-agent deep reinforcement learning: A survey, *Artif. Intell. Rev*., vol. 55, no. 2, pp. 895–943, 2022.

[17] K. Zhang, J. Y. Cao, and Y. Zhang, Adaptive digital twin and multiagent deep reinforcement learning for vehicular edge computing and networks, *IEEE Trans. Ind. Inform*., vol. 18, no. 2, pp. 1405–1413, 2022.

[18] T. Liu, L. Tang, W. L. Wang, Q. B. Chen, and X. P. 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, 2022.

[19] X. Y. Huang, L. J. He, and W. Y. Zhang, Vehicle speed aware computing task offloading and resource allocation based on multi-agent reinforcement learning in a vehicular edge computing network, in *Proc. 2020 IEEE Int. Conf. on Edge Computing* (*EDGE*), Beijing, China, 2020, pp. 1–8.

[20] X. L. Xu, Z. J. Fang, L. Y. Qi, W. C. Dou, Q. He, and Y. C. Duan, A deep reinforcement learning-based distributed service off loading method for edge computing empowered internet of vehicles, (in Chinese), *Chin. J. Comput*., vol. 44, no. 12, pp. 2382–2405, 2021.

[21] X. Q. Zhang, H. J. Cheng, Z. Y. Yu, and N. Xiong, Design and analysis of an efficient multi-resource allocation system for cooperative computing in internet of things, *IEEE Internet Things J*., doi: 10.1109/JIOT.2021.3094507.

[22] V. Mnih, A. P. Badia, M. Mirza, A. Graves, T. Harley, T. P. Lillicrap, D. Silver, and K. Kavukcuoglu, Asynchronous methods for deep reinforcement learning, in *Proc. 33rd Int. Conf. on Machine Learning*, New York City, NY, USA, 2016, pp. 1928–1937.

[23] Z. Q. Zhu, S. Wan, P. Y. Fan, and K. B. Letaief, Federated multiagent actor-critic learning for age sensitive mobile-edge computing, *IEEE Internet Things J*., vol. 9, no. 2, pp. 1053–1067, 2022.

[24] S. Munir, S. F. Abedin, D. H. Kim, N. H. Tran, Z. Han, and C. S. Hong, A multi-agent system toward the green edge computing with microgrid, in *Proc. 2019 IEEE Global Communications Conf.* (*GLOBECOM*), Waikoloa, HI, USA, 2019, pp. 1–7.

[25] S. Munir, S. F. Abedin, N. H. Tran, Z. Han, E. N. Huh, and C. S. Hong, Risk-aware energy scheduling for edge computing with microgrid: A multi-agent deep reinforcement learning approach, *IEEE Trans. Netw. Serv. Manag*., vol. 18, no. 3, pp. 3476–3497, 2021.

[26] T. L. Mai, H. P. Yao, Z. H. Xiong, S. Guo, and D. T. Niyato, Multi-agent actor-critic reinforcement learning based in-network load balance, in *Proc. GLOBECOM 2020 – 2020 IEEE Global Communications Conf*., Taipei, China, 2020, pp. 1–6.

[27] H. Tian, X. L. Xu, T. Y. Lin, Y. Cheng, C. Qian, L. Ren, and M. Bilal, DIMA: Distributed cooperative microservice caching for internet of things in edge computing by deep reinforcement learning, *World Wide Web*, doi: 10.1007/s11280-021-00939-7.

[28] Q. H. Huang, X. L. Xu, and J. H. Chen, Learning-aided fine grained offloading for real-time applications in edge-cloud computing, *Wirel. Netw*., doi: 10.1007/s11276-021-02750-8.

[29] R. C. Xie, X. F. Lian, Q. M. Jia, T. Huang, and Y. J. Liu, Survey on computation offloading in mobile edge computing, (in Chinese), *J. Commun*., vol. 39, no. 11, pp. 138–155, 2018.

[30] I. P. Chochliouros, I. Giannoulakis, T. Kourtis, M.

Belesioti, E. Sfakianakis, A. S. Spiliopoulou, N. Bompetsis, E. Kafetzakis, L. Goratti, and A. Dardamanis, A model for an innovative 5G-oriented architecture, based on small cells coordination for multi-tenancy and edge services, in *Proc. 12th IFIP WG 12.5 Int. Conf. and Workshops Artificial Intelligence Applications and Innovations*, Thessaloniki, Greece, 2016, pp. 666–675.

[31] I. Giannoulakis, E. Kafetzakis, I. Trajkovska, P. S. Khodashenas, I. Chochliouros, C. Costa, I. Neokosmidis, and P. Bliznakov, The emergence of operator-neutral small cells as a strong case for cloud computing at the mobile edge, *Trans. Emerg. Telecommun. Technol.*, vol. 27, no. 9, pp. 1152–1159, 2016.

[32] M. Morelli, C. C. J. Kuo, and M. O. Pun, Synchronization techniques for orthogonal frequency division multiple access (OFDMA): A tutorial review, *Proc. IEEE*, vol. 95, no. 7, pp. 1394–1427, 2007.

[33] Y. L. Lu, S. Maharjan, and Y. Zhang, Adaptive edge association for wireless digital twin networks in 6G, *IEEE Internet Things J.*, vol. 8, no. 22, pp. 16219–16230, 2021.

[34] S. Woo, J. Park, J. Y. Lee, and I. S. Kweon, CBAM: Convolutional block attention module, in *Proc. 15th European Conf. Computer Vision* (*ECCV*), Munich, Germany, 2018, pp. 3–19.

[35] R. S. Sutton and A. G. Barto, Reinforcement learning: An introduction, *IEEE Trans. Neural Netw.*, vol. 16, no. 1, pp. 285–286, 2005.

[36] N. Zhang, N. Cheng, A. T. Gamage, K. Zhang, J. W. Mark, and X. M. Shen, Cloud assisted HetNets toward 5G wireless networks, *IEEE Commun. Mag.*, vol. 53, no. 6, pp. 59–65, 2015.

[37] J. Zhang and K. B. Letaief, Mobile edge intelligence and computing for the internet of vehicles, *Proc. IEEE*, vol. 108, no. 2, pp. 246–261, 2020.

Linjie Gu is currently pursing the BS degree, Nanjing University of Information Science and Technology. His research interests include edge computing and deep reinforcement learning.

Mengmeng Cui obtained the BS degree at Nanjing Normal University, China, in 2002. She received the MS and PhD degrees from Nanjing University of Information Science and Technology, China, in 2007 and 2019, respectively. Now, she is an associate professor in Nanjing University of Information Science and technology, China. Her main research interests include high performance computing, optimization algorithms, and meteorological data mining.

Linkun Xu is currently pursing the BS degree, Nanjing University of Information Science and Technology. His research interests include deep learning and IoT.

Xiaolong Xu received the PhD degree from Nanjing University, China, in 2016. He was a research scholar at Michigan State University, USA, from April 2017 to May 2018. He is currently a professor with School of Computer and Software, Nanjing University of Information Science and Technology. He has published more than 80 peer-review articles in international journals and conferences. He received the Best Paper Award from the IEEE CBD 2016, IEEE CPCSCom 2020, and SPDE 2020. His research interests include edge computing, IoT, cloud computing, and big data.