IEICE TRANS. COMMUN., VOL.E107–B, NO.6 JUNE 2024

PAPER

Federated Deep Reinforcement Learning for Multimedia Task Offloading and Resource Allocation in MEC Networks*

Rongqi ZHANG[†,††], *Nonmember*, **Chunyun PAN**[†,††a)], *Member*, **Yafei WANG**[†,††], **Yuanyuan YAO**[†,††], *and* **Xuehua LI**[†,††], *Nonmembers*

SUMMARY With maturation of 5G technology in recent years, multimedia services such as live video streaming and online games on the Internet have flourished. These multimedia services frequently require low latency, which pose a significant challenge to compute the high latency requirements multimedia tasks. Mobile edge computing (MEC), is considered a key technology solution to address the above challenges. It offloads computation-intensive tasks to edge servers by sinking mobile nodes, which reduces task execution latency and relieves computing pressure on multimedia devices. In order to use MEC paradigm reasonably and efficiently, resource allocation has become a new challenge. In this paper, we focus on the multimedia tasks which need to be uploaded and processed in the network. We set the optimization problem with the goal of minimizing the latency and energy consumption required to perform tasks in multimedia devices. To solve the complex and non-convex problem, we formulate the optimization problem as a distributed deep reinforcement learning (DRL) problem and propose a federated Dueling deep Q-network (DDQN) based multimedia task offloading and resource allocation algorithm (FDRL-DDQN). In the algorithm, DRL is trained on the local device, while federated learning (FL) is responsible for aggregating and updating the parameters from the trained local models. Further, in order to solve the not identically and independently distributed (non-IID) data problem of multimedia devices, we develop a method for selecting participating federated devices. The simulation results show that the FDRL-DDQN algorithm can reduce the total cost by 31.3% compared to the DQN algorithm when the task data is 1000 kbit, and the maximum reduction can be 35.3% compared to the traditional baseline algorithm.

key words: *multimedia transmission, computing offloading, resource allocation, federated learning, deep reinforcement learning*

1. Introduction

In recent years, with the continuous development of 5G networks, the number of multimedia services and smart terminals in mobile networks has increased rapidly, leading to a significant increase in mobile data volume [1]. According to Cisco's latest forecast report [2], much of the new significant traffic will originate from mobile multimedia services, which will rapidly increase as a percentage of total traffic due to the sheer volume of data. In 2017, mobile multimedia services accounted for 59% of all mobile data traffic. By 2023, this figure will jump to 79%.

Ultra-low latency, intensive computing and massive transmission are the distinctive features of most mobile multimedia services, for example, webcasting, virtual reality services (VR), augmented reality services (AR), cloud computers, and online games. These mobile multimedia services often have high requirements on network latency, bandwidth and computing power. Meanwhile, due to the amount of traffic and computing on mobile users and devices increasing dramatically, multimedia devices need to handle many intensive mobile multimedia tasks such as video compression and transcoding [3], [4]. The huge amount of computation caused by intensive computing tasks puts a lot of pressure on users. However, due to the limited computing resources and storage capacity of multimedia devices, these devices cannot handle tasks locally with low latency as well as low power consumption.

In view of these problems, Mobile Cloud Computing (MCC) has been proposed as a solution, where large amounts of data are centralized in cloud servers to alleviate the burden on local devices [5]. However, traditional cloud computing suffers from problems such as high latency, high load, and core network congestion, as cloud servers are typically deployed at a distance from multimedia devices. In contrast, Mobile Edge Computing (MEC) offers a promising approach by deploying edge servers at edge nodes or base stations. This enables mobile terminals to offload their computing tasks to nearby edge nodes for processing, using wireless channels to reduce task processing delays, improve network utilization efficiency, and enhance Quality of Service [6]. Nevertheless, compared to cloud computing, edge computing is limited by offload decisions, wireless resources and computing resources. Wireless resources mainly include bandwidth and transmitting power. Computational resources generally refer to the CPU frequency of local mobile devices and edge servers. To fully leverage the advantages of the MEC paradigm, there is a need for joint optimization of offloading decisions, communication, and computational resources. This presents a major problem in wireless networks between user devices and MEC servers.

To address this challenge, several studies have inves-

Manuscript received July 5, 2023.
Manuscript revised October 23, 2023.
Manuscript publicized January 30, 2024.

[†]The authors are with the Key Laboratory of Information and Communication Systems, Ministry of Information Industry, Beijing Information Science and Technology University, Beijing 100101, China.

[††]The authors are with the Key Laboratory of Modern Measurement and Control Technology, Ministry of Education, Beijing Information Science and Technology University, Beijing 100101, China.

*This work is partially supported by Beijing Natural Science FoundationHaidian Original Innovation Joint Fund (No. L212026), R&D Program of Beijing Municipal Education Commission (KM202211232011), Key Project of Beijing Natural Science Foundation-Haidian Original Innovation Joint Fund (No.L222004).

a) E-mail: chunyupan@bistu.edu.cn

DOI: 10.23919/transcom.2023EBP3116

tigated the joint allocation of wireless and computational resources in MEC systems [7]–[11]. The Lyapunov optimization methods, online dynamic task scheduling, and game theory has been proposed to solve the problems of joint wireless resources, computational resources, and offloading decisions. The authors in Ref. [7] proposed a local compressed offload model to solve the resource allocation problem of multi-user mobile edge computing offload systems. In Ref. [8], the authors proposed a Lyapunov optimization based approach to study the task assignment scheduling scheme for maximum power consumption and execution delay in MEC systems with energy harvesting capability. The authors in Ref. [9] considered a heuristic algorithm for solving joint resource allocation decisions to minimize the time delay. The authors in Ref. [10] investigated a computational resource allocation scheme based on potential game theory to reduce the energy consumption of MEC networks and improve the efficiency of computational resources. In Ref. [11], the paper proposed a suboptimal resource allocation algorithm that generates priorities for users based on their channel gain and locally calculated energy consumption, and implements different offloading schemes for different priorities to minimize the weighted sum of delay and energy consumption. However, these algorithms are usually time-consuming and computationally intensive in complex MEC networks because they need to constantly resolve the problem in a time-varying MEC network environment.

Deep reinforcement learning has become a trend as an approach in solving optimization problems in MEC systems, in recent years [12]–[17]. DRL can adjust its strategy in unstable environments and can adapt to complex MEC scenarios by making different actions with its intelligences. The DRL agents can make adaptive offloading decisions and resource allocation through the different actions it makes. In Ref. [12], the authors proposed a distributed machine learning approach that makes it possible for DRL to perform online offloading in an MEC environment. The authors in Ref. [13] considered a DRL-based video offload scheme to maximize its long-term performance. The authors in Ref. [14] studied a temporal attentional deterministic policy gradient based on a deep reinforcement learning algorithm called Deep Deterministic Policy Gradient (DDPG) to solve the joint optimization problem of computational offloading and resource allocation in MEC. Ref. [15], this paper proposed a DRL-based offloading scheme to enhance the utility of multimedia devices in dynamic MEC. Simulation results demonstrate that the DRL scheme reduces energy consumption, computational experiments and task failure rate. The authors in Ref. [16] proposed a DRL-based offloading framework that can be adaptive to the common patterns behind various applications to infer the optimal offloading strategy for different scenarios. Ref. [17], the authors propose an advanced deep learning based computational offloading algorithm for multistage vehicle edge cloud computing networks to minimize the total time and energy cost of the whole system. Although DRL is very resilient in complex MEC networks, because most DRL learn in a centralized manner, the required action space and configuration of parameters explode when multimedia devices are added, which directly leads to less efficient training and easier privacy disclosure. To solve this problem, Federated learning (FL) is proposed to optimize MEC networks [18].

Federated learning is a distributed machine learning that enables distributed multiple device nodes to co-communicate and participate in the aggregation of global models. Different devices can perform local model training separately, communicate with each other through federated learning and upload model parameters from local model training for global model aggregation. Federated learning allows the exchange of model parameters without sharing raw data and enhances the collaboration capability of multiple distributed devices and protects the privacy and security of the devices.

Several studies have investigated the resource allocation and computational offloading problems involved in FL for two optimization objectives based on system latency and energy consumption minimization [19], [20]. The authors in Ref. [19] minimized the value of the FL loss function by optimizing the joint resource allocation and UE selection, and satisfied both the latency and energy consumption requirements for performing FL. The authors in Ref. [20] proposed an alternative directional algorithm formulating the joint optimization of CPU frequency and power control as a nonlinear programming (NLP) problem to solve the problem of minimizing the energy consumption of all multimedia devices subject to federated learning time requirements. References [21]–[23] focus on the combined learning of federation learning and deep reinforcement learning, i.e., training local DRL models and then integrating them together to develop a comprehensive global DRL model. The authors in Ref. [21] proposed a joint optimization scheme for optimal path selection and power allocation based on the federal deep Q-network learning algorithm, which maximizes network throughput while ensuring power constraints and mobility constraints, taking into account communication resources, but without considering a reasonable allocation of computational resources. In [22], this paper considered a multimodal deep reinforcement learning framework based on hybrid policies and proposes an online joint collaboration algorithm in combination with FL and validates the performance of the algorithm, however, the intelligent body agent in this work does not undertake some resource allocation operations such as allocation of power, computational offloading of tasks. The authors in [23] proposed a federate cooperative caching framework based on deep reinforcement learning but the work did not take into account task offloading.

We compare the objectives and resource optimization of ourstudy with some related work in the MEC systems, the resultsof which are shown in Table 1. It is obvious that our study can overcome the shortcomings of many previous works.

For mobile multimedia devices, their limited computing resources and battery capacity may hinder efficient task

Table 1 Comparing with some related work.

Reference	Delay	Energy	Communication	Computation	Offloading	DRL	FL
[7]	✓	×	✓	×	✓	×	×
[8]	✓	✓	✓	×	✓	×	×
[9]	✓	×	✓	✓	✓	×	×
[10]	×	×	×	✓	×	×	×
[11]	✓	✓	×	✓	✓	×	×
[12]	×	×	×	✓	✓	✓	×
[13]	✓	×	×	×	✓	✓	×
[14]	✓	✓	✓	×	✓	✓	×
[15]	×	×	×	×	✓	✓	×
[16]	✓	×	✓	×	✓	✓	×
[17]	✓	✓	✓	×	✓	✓	×
[19]	×	×	✓	×	×	×	✓
[20]	✓	✓	×	✓	×	×	✓
[21]	×	×	×	✓	×	✓	✓
[22]	✓	×	✓	×	×	✓	✓
[23]	✓	×	✓	×	×	✓	✓
Our Work	✓	✓	✓	✓	✓	✓	✓

completion. In such cases, offloading tasks to edge or cloud servers becomes necessary. The offloading decision made by the multimedia device plays a critical role in controlling the overall MEC system overhead and ensuring a good user experience. Additionally, task offloading consumes wireless channel resources, necessitating reasonable allocation of these resources in MEC systems. In this paper, we propose an adaptive offloading framework based on federated deep reinforcement learning to jointly optimize transmit power, computational resources, and offloading decisions, with the aim of minimizing delay and energy consumption for mobile multimedia devices in task completion. The contributions of this paper can be summarized as follows:

1. We transform the optimization problem into a multi-objective optimization problem with the objective of minimizing the weighted sum of delay and energy consumption required by the system to perform the task. To solve this complex problem, we jointly allocate computational and communication resources and transform the nonlinear planning problem into a federated deep reinforcement learning problem for multiple intelligent agents.

2. For multimedia devices, the changing location and different kinds of multimedia task of devices cause non-IID data. To reduce the impact of non-IID data. In this paper, we propose a mechanism for selection of participating federal learning devices. To ensure the communication overhead as well as convergence of FL learning.

3. We design an adaptive offloading algorithm based on FL and DRL, which jointly allocates computational resources and task offloading, which not only increases the overall scalability of the system but also accelerates the learning speed of deep reinforcement learning. It maintains relatively stable performance in the complex MEC network environment and outperforms other DRL algorithms.

The rest of this article is organized as follows: Section 2 describe the system model and the problem formulation is described in Sect. 2. In Sect. 4, we present the design of the FDRL-DDQN algorithm. Section 5 presents simulation results. Finally, Sect. 6 shows the conclusion of this paper.

2. System Model

In this paper, we consider a MEC network configuration that consists of a MEC server, an MCC server, a MEC base station (BS) and a set of $\mathcal{N} = \{1, 2, \ldots, N\}$ multimedia devices. As depicted in Fig. 1, when a task is generated, the user's task request is initially submitted to a multimedia device. Subsequently, the MEC BS receives offloading tasks from the multimedia devices. The computing tasks offloaded to the BS are processed by the BS server, and the results are then returned to the terminal. Meanwhile, the remaining multimedia tasks are executed locally. We make a diagram to explain the function of each layer in Fig. 2, and some key parameters are listed in Table 2.

We consider the time into consecutive time frames, which are divided into $\mathcal{T}$ time slots denoted as a set of $\mathcal{T} = \{1, 2, \ldots, T\}$. This article explores a ternary offloading strategy. Specifically, the local offloading decision of device i as x_i, where $x_i = 1$ signifies that the multimedia device executes the tasks locally, and $x_i = 0$ means the multimedia device offloads the multimedia tasks to the MEC server or MCC server. Moreover, we use y_i and z_i to represent the multimedia devices' offloading of computing tasks to the MCC server and MEC server, respectively. In this context, $z_i = 1$ denotes offloading to the MEC server, while $y_i = 1$ implies offloading to the MCC server. Therefore, we have the ternary offloading strategy as follows:

$$x_i + y_i + z_i = 1, \ \forall i \in \mathrm{N}. \tag{1}$$

2.1 Computing Model

This section focuses on modeling the delay and energy con-

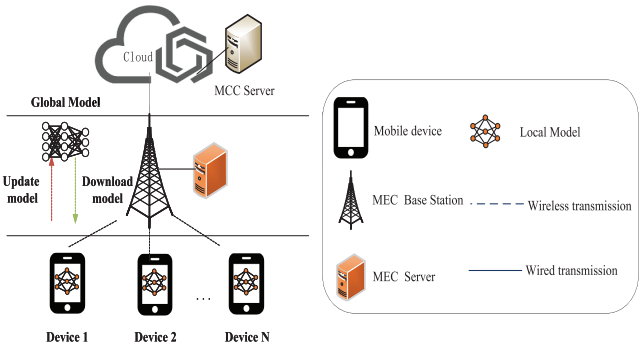

Fig. 1 A MEC system model.

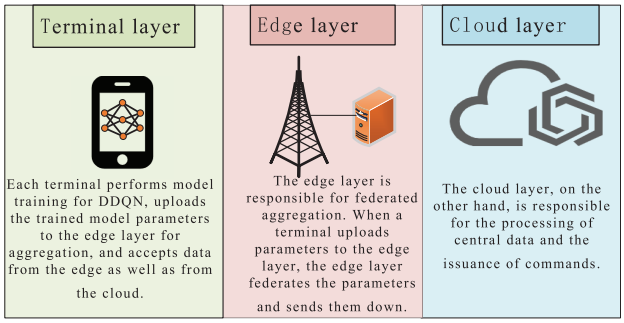

Fig. 2 The function of each layer.

Table 2 Key parameters.

Notation	Meaning
L_i	The size of tasks
x_i, y_i, z_i	binary offloading variables
r_i	the uplink transmission rate of multimedia device i
${T_i}^{bs}$	The communication delay of mobile task offloading
${E_i}^{bs}$	The energy consumption of mobile task offloading
${T_i}^{e}$	The computation delay of MEC server
${T_i}^{c}$	The computation delay of MCC server
${T_i}^{E}$	The delay of tasks offloading to the MEC server
${T_i}^{C}$	The delay of tasks offloading to the MCC server
${T_i}^{L}$	The processing dealy for task on device
${E_i}^{L}$	The processing energy consumption for task on device
θ^{eval}	The eval network parameter
θ^{global}	The global model parameter
$Q_i(s,a)$	The action state values of Bellman equation
$L(\theta)$	Loss function

sumption experienced by multimedia devices during the execution of multimedia tasks. When multimedia device i offloads its task to either the MEC server or MCC server, various factors come into play, including the size of the multimedia tasks, channel conditions, and transmitting power. It's important to note that the transmission of offloaded multimedia tasks occurs over wireless channels, involving both multimedia devices and base stations. Furthermore, the execution of multimedia tasks requires the allocation of uplink frequency resources for transmission. Therefore, the uplink transmission rate of multimedia device i is determined by

$$r_i = B\log_2(1 + \frac{p_i h_i}{\sigma^2}), \tag{2}$$

where B and p_i denote the operating bandwidth and transmit power of the multimedia device, respectively, h_i and σ^2 denote the transmission link gain and channel noise between the multimedia device and the base station, respectively.

The communication delay and the energy consumption of mobile task offloading are respectively given by

$${T_i}^{bs} = \frac{L_i}{r_i}, \tag{3}$$

$${E_i}^{bs} = p_i {T_i}^{bs}, \tag{4}$$

where L_i is the size of task (in bit). Since the computing power of edge servers and clouds is very resource-rich compared to local devices, this paper ignores the computing consumption at the edge servers or clouds, so only the energy consumed by their task transmission is calculated. From the device's point of view, when a task is offloaded to either server, the energy consumed to process the task is the energy spent on the task transfer. So both of the energy consumption of MCC server and MEC server would be equal to E_i^{bs}.

Due to limited computing power and battery capacity, multimedia devices offload tasks to edge servers or cloud services to meet QoS requirements. The computation delay of MEC server and MCC, while offloading is given, respectively, as follows:

$$T_i^e = \frac{C_i}{F^e}, \tag{5}$$

$$T_i^c = \frac{C_i}{F^c}, \tag{6}$$

where F^e and F^c denote the average computing power of the edge server and the cloud, respectively. C_i denotes the CPU cycle requirement of the task (in cycle/second). The delay of the multimedia tasks offloading to the MEC server and MCC server respectively as follows:

$$T_i^E = T_i^e + T_i^{bs}, \tag{7}$$

$$T_i^C = T_i^c + T_i^{bs}. \tag{8}$$

Assuming that user-submitted multimedia tasks are selected for execution on the local multimedia device and they do not need to be offloaded to the edge server for processing, the processing delay and energy consumption for task on device is defined as

$$T_i^L = \frac{C_i}{f_i^L}, \tag{9}$$

$$E_i^L = \kappa_i (f_i^L)^2, \tag{10}$$

where κ_i is the energy consumption factor related to the multimedia device, which depends on the CPU performance architecture of the terminal.

In this paper, we aim to optimize the computational resource allocation as well as the offloading policy, which minimizes the multimedia task execution cost. The long-term expected cost of each multimedia device is a weighted

sum of execution delay and energy consumption. Each of multimedia device cost is given by

$$T_i(p_i, f_i, x_i, y_i, z_i) = x_i T_i^L + y_i T_i^E + z_i T_i^C, \tag{11}$$

$$E_i(p_i, f_i, x_i, y_i, z_i) = x_i E_i^L + y_i E_i^e + z_i E_i^c, \tag{12}$$

where p_i, f_i, x_i, y_i and z_i represent the vectors of transmit powers, computation resource allocation, local computing, edge offloading, and cloud offloading decision of device i, respectively.

3. Problem Formulation

In this paper, the problem is formulated to solve the joint minimization of long-term delay and energy consumption of multimedia devices over time $\mathcal{T}$.

In solving the optimization problem of offloading decisions and computational resource allocation for MEC systems, the objective of this paper is to minimize the total cost of the combination of execution delay and energy consumption of the devices in the MEC system. Based on the above analysis, the optimization problem can be described as follows:

$$\begin{aligned} &\min_{p_i, f_i, x_i, y_i, z_i} \quad \omega T_i + \lambda E_i \\ &\text{subject to :} \\ &C1: f_i^L \le F_{\max}, \forall i \in N \\ &C2: x_i E_i^L + y_i E_i^e + z_i E_i^c \le E_{\max,i}, \forall i \in N \\ &C3: T_i \le T_{\max}, \forall i \in N \\ &C4: x_i + y_i + z_i = 1, \forall i \in N \\ &C5: x_i, y_i, z_i \in \{0, 1\}, \forall i \in N, \end{aligned} \tag{13}$$

where ω and λ in the above optimization problem are denoted as the delay and energy consumption weighting factors of device i in performing the multimedia task, respectively. Let $0 \le \omega \le 1$, and $0 \le \lambda \le 1$, $\omega + \lambda = 1$, the ratio of ω to λ is a constant, the value of the weight factor should be chosen according to the heterogeneity of the resources available on each multimedia device, if the device receives greater constraints in terms of energy resources than computational resources, the value should be larger, otherwise it should be smaller. The constraint C1 indicates that the computing resources allocated for the user should not exceed the total computing capacity of the MEC system F_{max}. C2 indicates a limit on the energy resources of the device, which should not exceed the maximum energy E_{max} that the MEC system can provide, and C3 expresses that the overall service time cost should not exceed the maximum allowable delay for the user T_{max}. C4 and C5 are the ternary offloading schemes used in this paper.

However, to satisfy the requirement of minimizing the total system cost under the multimedia task execution delay as well as energy consumption tolerance. With binary offloading variables (x_i, y_i, z_i) included of above formulated problem (13) makes the problem into a mixed integer nonlinear programming (MINLP) problem that cannot be solved in an acceptable time frame.

4. The Proposed FDRL-DDQN Algortihm

In this section, we present our solution to address the complex and non-convex optimization problem. We propose a deep reinforcement learning algorithm that combines federated learning, and for offloading actions, we adopt the Dueling DQN algorithm. This algorithm is referred to as FDRL-DDQN.

The framework of the FDRL-DDQN algorithm is illustrated in Fig. 3. The FDRL-DDQN algorithm contains three main components: the training of offloading decision and resource allocation, federated aggregation and update of local model parameters. In the first step, devices participating in federated learning are selected. Next, the local model is trained to learn multimedia task offloading decisions and resource allocation. Subsequently, the trained model parameters are federated and aggregated. Finally, the updated parameters are distributed to each multimedia device involved in federated learning. Algorithm 1 provides a detailed description of the proposed FDRL-DDQN algorithm in this paper. And We give a flow chart of the Federation framework in Fig. 4.

In a complex MEC network environment, mobile multimedia devices are faced with three options for computing multimedia tasks. This results in a total of 2^{3N} possible computation offloading options per device at each time slot. With an increasing number of multimedia devices, the complexity of the state and action spaces for intelligent agents also grows exponentially. Consequently, implementing centralized training becomes extremely challenging when dealing with large-scale datasets and expansive action spaces. Moreover, in mobile multimedia services, which involve extensive data transmission, centralized training leads to significant

Algorithm 1 The FDRL-DDQN algorithm

Input: wireless channel gains h_i, size of multimedia tasks L_i.
Output: offloading action α_i, resource allocation action f_i.
Set the total time frame t, maximum FLDDQN iteratons to M.
Initialize the networks parameters $\theta_i^{eval} = \theta_i^{target}$ of all device and global model parameters θ^{global}.
while $(M >= 0)$: **do**
  Set $\theta_i^{eval} = \theta_i^{target} = \theta^{global}$
  Select the set of participating training devices, $\mathcal{W}$,
  for $t = 1, 2, ..., T$ **do**
    Offloading action $\alpha_i = f_i(h_i, L_i)$
    Compute Q_i for all α_i
    Select the optimal offloading action $\hat{\alpha}_i = arg\ max Q_i$
    for each device i in I **do**
      Interact with environment and calculate the cost
      Train the model and get θ_i^{eval}
      Upload the network parameters θ_i^{eval} of the terminal device i to the MEC server
      Federated average use by (20) in MEC server and get the θ^{global}.
      Transmits θ^{global} to the devices to replace the original network parameters θ_i^{eval}.
    end for
  end for
end while

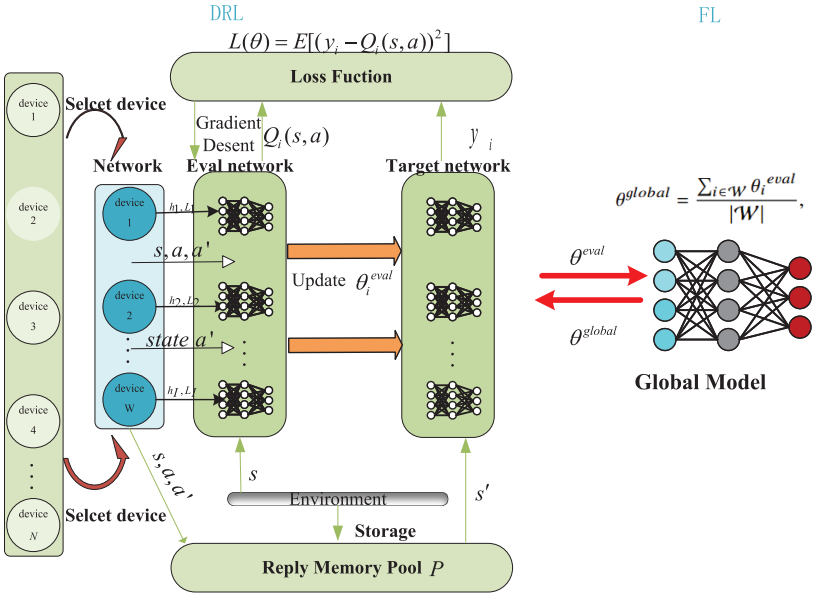

Fig. 3 Federated framework model.

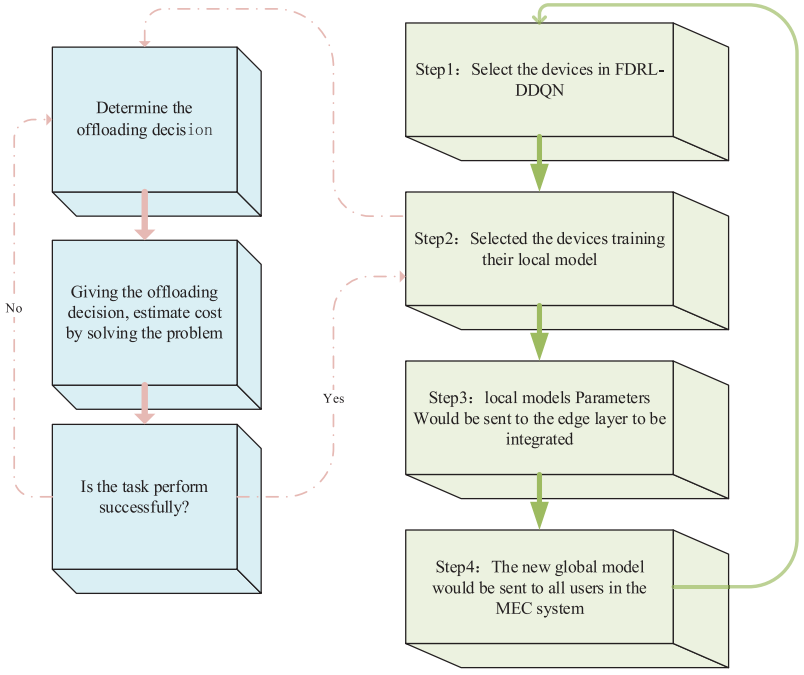

Fig. 4 Federated framework model.

communication overhead and raises privacy concerns. Federated learning, a distributed machine learning approach, offers several advantages in the MEC environment. Firstly, it enables individual training of intelligent agents, allowing them to cooperate and make independent decisions during multimedia task execution. This approach enhances learning efficiency, reduces communication overhead, and better adapts to large-scale MEC networks. Secondly, federated learning facilitates interactive updates of model parameters between distributed and central nodes, eliminating the need for sharing original data. This mechanism provides a robust guarantee for the security of local data.

4.1 Select Device

In cases where a large number of devices are involved in joint learning, it can result in increased drop rates and unnecessary communication overhead. To address this issue,

we introduce a device selection strategy in this paper. At the beginning of each iteration of the FDRL-DDQN algorithm, a specific set of multimedia device agents is carefully chosen to participate in the learning process. The principles of how to select devices in this article are as follows

$$\arg\max_{i\in N} \mathrm{MSE}(\frac{d_i P_{\max,i}}{F_{\max,i}}), \tag{14}$$

where d_i denotes the distance between the device and the BS and the function MSE denotes the mean squared error. This approach allows for the selection of the most cost effective method for different multimedia devices. Choosing the right device to participate in the learning process can also help with the overall learning speed.

4.2 Local Model Training

For the training of local agents in the FDRL-DDQN algorithm, each multimedia device employs the DDQN algorithm to train its own local model and learn offloading and resource allocation strategies. The Dueling DQN algorithm utilizes an experience pool to store data for each state at time t. This includes the action chosen based on the current state, the reward received for that action, the new state after performing the action, and whether the state terminates during the training process. As the amount of stored data in the experience pool reaches a sufficient size, small batches of data are randomly selected and fed into the neural network for training. This continuous training process optimizes the weight parameters in the neural network. By randomly selecting training data, a broad range of experiences can be learned, breaking the correlation between sample data and preventing overfitting issues resulting from local experiences. Based on the system model and optimization objectives presented in this chapter, the FDRL-DDQN algorithm defines three key elements: the state space, action space, and reward function, which can be defined as

1. State space:

$$S_i = \{L_i, h_i\}, \tag{15}$$

where S_i denotes the state space of each device, L_i denotes the amount of multimedia task, and h_i denotes the path gain of infinite transmission between the multimedia device and the base station.

2. Action space:

In the FDRL-DDQN model considered in this paper, the intelligence is responsible for making appropriate decisions based on the computational multimedia tasks. The decisions include, determining whether the computational multimedia tasks are offloaded to the edge server or the cloud server, and how much computational resources should be allocated when the multimedia tasks are executed locally. The action space consists of two parts, the multimedia device offloading decision $\{\alpha_i^L, \alpha_i^E, \alpha_i^C\}$, where α_i^L denotes that the task is executed locally, α_i^E denotes that the task is offloaded to the edge server, and α_i^C denotes that the task is offloaded to the cloud. The resource allocation strategy $f = \{f_1, f_2, \ldots, f_N\}$.

3. Reword function:
The cost of each agent is the weighted sum of the delay and energy consumption in the objective function. The optimization objective of this paper is to minimize the cost, so the reward function should be negatively correlated with the cost, so the reward function as shown

$$R_i = -\left(\frac{\omega(y_i T_i^E + z_i T_i^C) + \lambda(y_i E_i^e + z_i E_i^e)}{\omega x_i T_i^L + \lambda x_i E_i^L}\right). \tag{16}$$

The Dueling DQN algorithm is utilized to address complex decision control challenges in real-world multimedia environments. It combines Q-learning algorithms, empirical replay mechanisms, and target Q-values based on action value functions to approximate the Q-value of the optimal policy. Q-learning selects the action with the highest Q-value by consulting the Q-table, while dueling DQN uses a neural network to obtain the corresponding Q-value based on the input, resulting in improved operational speed and stability. As depicted in Fig. 5, the Dueling DQN architecture divides the fully connected layer of the network into two branches, each with its specific output. The upper branch represents the state value function, which quantifies the value of the static state environment itself, irrespective of actions taken. The lower branch represents the state-dependent action advantage function, which captures the average action payoff relative to states, indicating the additional value brought by decision-making behavior. These two branches are then combined to derive the Q-value for each action. This approach allows for mutual supervision, eliminates redundant degrees of freedom, mitigates the risk of inflated Q-value estimates, and enhances algorithm stability. Therefore, in this paper, the notation $u_i(s, a)$ is employed to represent the direct cost incurred by each device as determined through the aforementioned optimization process. Using the Bellman equation, the action state values are given by

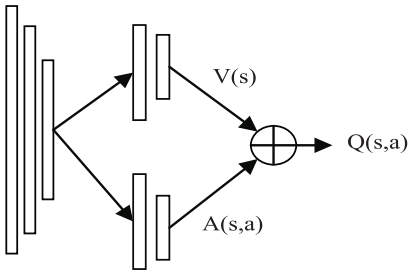

Fig. 5 The network of DDQN algorithm.

$$Q_i(s,a) = u_i(s,a) + \gamma \sum_{s' \in S} P_{ss'}(a) \max Q_i(s',a'), \quad (17)$$

where $Q_i(s,a)$ denotes the Q value corresponding to the action a generated according to the current state s. Similarly, $\max Q_i(s',a')$ is used to denote the action a' corresponding to the maximum Q value output by the stare s'. $P_{ss'}(a)$ stands for the transfer probability function, γ represent the discount factor. There are two network parameters in the dueling DQN network, one is the current Q network parameter, denoted by θ_i^{eval}, to evaluate the greedy strategy of the current network and the other is the target network parameter, denoted by θ_i^{target}, to evaluate the target value y_i. In each training iteration, the target value used to train the evaluation network in device i is calculated as

$$y_i = u_i(s,a) + \gamma Q_i(s', \arg\max_{a' \in A} Q_i(s',a';\theta_i^{eval}), \theta_i^{t\arg et}). \quad (18)$$

Meanwhile, to obtain the optimal strategy and minimize the gap between the target value and the evaluated value, we set the loss function as

$$L(\theta) = E[(y_i - Q_i(s,a))^2]. \quad (19)$$

4.3 Parameter Aggregation and Update

At the start of each learning round in FDRL-DDQN, the participating local devices upload their network parameter models to the MEC server for model aggregation. This aggregation process combines the models to create a global model within the MEC. Subsequently, the MEC server distributes the aggregated global model parameters to each multimedia device participating in FDRL-DDQN as the network parameters for the next round. In this paper, we employ FedAvg [24] as the model aggregation method.

$$\theta^{global} = \frac{\sum_{i \in \mathcal{W}} \theta_i{}^{eval}}{|\mathcal{W}|}, \quad (20)$$

where $|\mathcal{W}|$ denotes the total number of participating training devices. θ^{global} represents the global model parameters. Once the global model aggregation is complete, the model parameters are transmitted to the local device, which then updates its own model parameters. The local device utilizes its local data to train the network evaluation parameters during the offloading decision update. This update process continues iteratively for the local devices until the algorithm converges.

5. Simulation Results

In this section, we use tensorflow1.0 GPU version to implement the FDRL-DDQN framework in python and perform simulations to evaluate its performance. The main simulation parameter settings in this paper are shown in Table 3.

To simulate the proposed FDRL-DDQN algorithm, we constructe a network comprising 50 multimedia devices. However, only 10 devices were selected for each training round. Each device has a maximum computational capacity of 1 Gbps and a maximum energy consumption of 23 dBm. Due to the limitations of computing power on the multimedia devices in the MEC network, we utilize the smallest feasible neural network for our algorithm. Considering the computational constraints, our neural network consisted of an input layer, two hidden layers, and an output layer. The first and second layers consisted of 32 and 16 neurons, respectively. ReLU activation functions were used throughout the network.

Table 3 Simulation parameters.

Parameter	Value
Moblie device transmit power (p_i)	23 dBm
The number of CPU cycles required to process 1bit data (c)	500
Channel noise power (σ^2)	10^{-9}
Computing power of edge server and cloud server(F^e), (F^c)	6, 7, 8, 9, and 10 GHz
Computing power of the multimedia device (f_i^L)	500 KHz
Communication bandwidth (B)	180 KHz
Greedy coefficient (ε)	0.95
Discount factor (γ)	0.9
Learning rate	0.001
$T_{\max}$	0.5 s
Experience replay buffer	3000
Channel gain (h_i)	$[1,2] * 10^{-5}$ Uniform distribution

5.1 Convergence Performance

In this section, we evaluate the convergence performance of the FDRL-DDQN algorithm and compare it with the distributed DDQN. We examine the convergence of the two schemes using selected devices and all devices to participate in the federation to address the non-IID data issue of multimedia devices. Additionally, we analyze the impact of learning rate and batch size on the convergence of the FDRL-DDQN algorithm.

Firstly, we assess the convergence speed of the training loss in the FDRL-DDQN algorithm. In Fig. 6, the average training loss $L(\theta)$ of the FDRL-DDQN model is plotted. Initially, the algorithm exhibits significant fluctuations due to the lack of experience during the initial training, making it challenging for the intelligence to learn the optimum. However, as the experience pool accumulates sufficient data, the intelligence can make actions that lead to the optimal solution, resulting in maximum rewards. As the number of iterations increases, the loss function steadily decreases, indicating a smoother learning process. After approximately 200 iterations, the algorithm reaches the optimum value for the neural network.

In Fig. 7, we address the non-IID problem among mobile multimedia devices by selecting only a fraction of devices to participate in each round of federated learning, ensuring the convergence speed of the overall algorithm. To

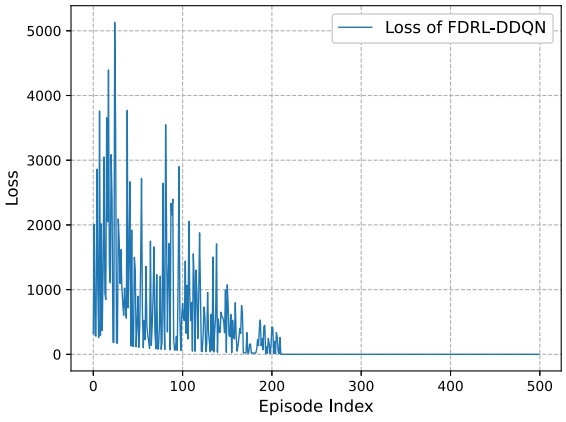

Fig. 6 Convergence process of loss function.

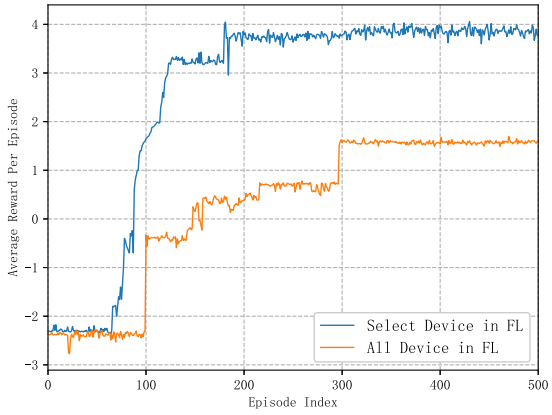

Fig. 7 Comparison of convergence between FDRL-DDQN and systems without selection mechanism.

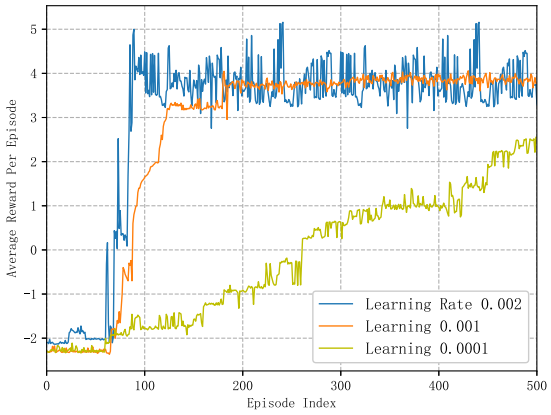

Fig. 8 Convergence of reward value under different learning rates.

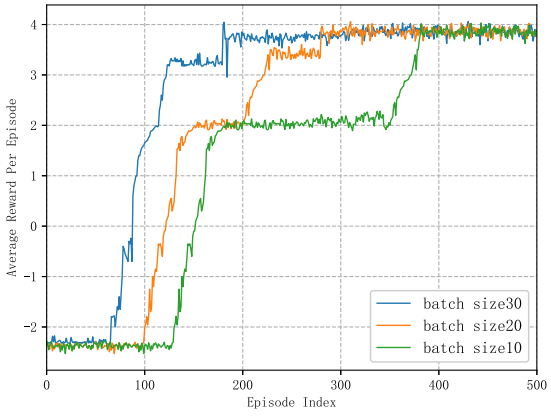

Fig. 9 Convergence of reward value under different batch size.

evaluate the effect of the device selection method on the convergence of the FDRL-DDQN algorithm, we compare the approach of adding all devices to federated learning with the partial device addition. The convergence performance of the FDRL-DDQN algorithm is validated using 5W randomly generated multimedia tasks, which are offloaded for optimal resource allocation decisions. In each iteration round, the intelligence derives a reward value based on its decision. The average reward of the FDRL-DDQN algorithm is plotted, showing an increasing trend with the number of iterations as the intelligence improves its decision-making ability. The algorithm converges after approximately 200 iterations. It can be observed from the figure that the average reward value of the overall algorithm, after utilizing the device selection mechanism, is significantly higher and converges faster compared to the approach of adding all devices to federated learning.

Furthermore, we examine the impact of learning rate and batch size on the convergence of the FDRL-DDQN algorithm. Figure 8 illustrates the effect of the intelligence's learning rate on the convergence performance of the FDRL-DDQN framework. We experiment with learning rates of 0.0001, 0.001, and 0.002. While a higher learning rate leads to faster learning, the figure shows that a learning rate of 0.0001 results in slow convergence due to the low learning rate. On the other hand, a learning rate of 0.002 increases learning efficiency, but it compromises algorithm stability, causing repeated oscillations that hinder convergence. Thus, in this paper, we set the learning rate to 0.001 in the simulation.

Another parameter of interest is the batch size for multimedia task processing. Figure 9 demonstrates that increasing the batch size improves the convergence of the FDRL-DDQN algorithm. With a small batch size of 10, convergence takes around 380 iterations. However, when the batch size is increased to 20, convergence occurs after 310 iterations, and with a batch size of 30, the algorithm converges quickly in only 180 iterations. Larger batch sizes enable training with more instances, providing the intelligence with faster experience accumulation. Consequently, the executed actions can reach optimal solutions more rapidly, resulting in faster algorithm convergence.

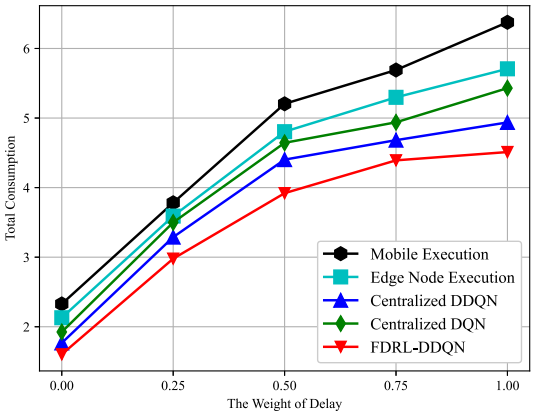

Fig. 10 Influence of delay weight on total system cost.

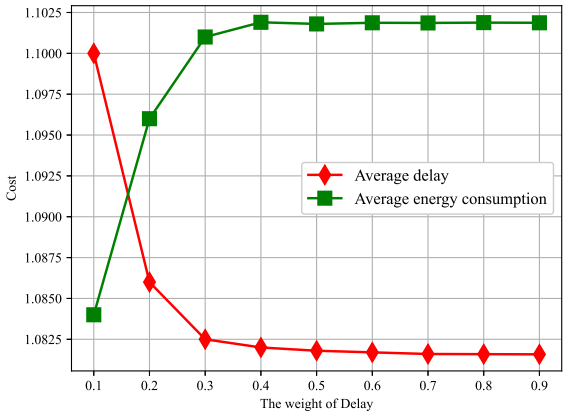

Fig. 11 The average delay and the average energy consumption curve.

5.2 Comparison of Total Cost

In this section, we first compare the proposed FDRL-DDQN algorithm with the distributed DDQN algorithm. To further evaluate the algorithm's performance, we also compare it with the centralized DDQN algorithm, centralized DQN algorithm, and two baseline computation offloading policies: mobile execution and edge node execution. Mobile execution refers to local device computation of multimedia tasks, while edge node execution involves offloading all tasks to the edge node. We investigate the effect of delay weights on the FDRL-DDQN algorithm in comparison to the four mentioned centralized algorithms. Additionally, we discusse the trade-off between delay and energy consumption.

In Fig. 10, we experiment with the weights of delay and energy consumption on algorithm performance. To handle different types of mobile multimedia tasks, we set the weights of delay ω to equal values of 0-1 and similarly γ to $1-\omega$. We set the delay weights of the centralized DQN algorithm, DDQN algorithm, to be consistent with FDRL-DDQN. As ω increases the system assembly also rises, the total cost of FDRL-DDQN is always lower than the other four algorithms. This is due to the fact that FDRL-DDQN is able to provide an optimal offloading strategy for the optimized target compared to the other four algorithms, thus achieving an overall cost reduction.

Figure 11 illustrates the equilibrium trend of the network's average delay and average energy consumption as the delay weight varies. In this simulation, the number of users is set to N=5. From Fig. 11, it is apparent that the network's average delay gradually decreases as the delay weight increases, while the average energy consumption of the network increases. Both the average delay and average energy consumption of the network stabilize when the delay weight reaches a certain threshold when the $\omega >= 0.4$. This happens because when the value of ω is small, increasing it reduces delay at the expense of energy performance. However, when the value of ω is large, due to limitations on user transmitting power, further increasing ω doesn't result in reduced average delay or increased average energy consumption.

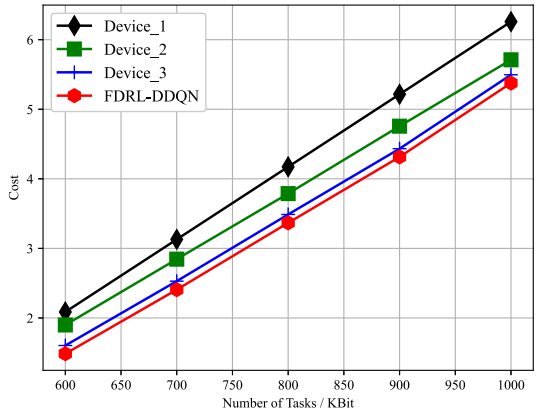

Fig. 12 The cost of FDRL-DDQN scheme is compared with distributed DDQN algorithm.

Figure 12 compares the average total cost of the proposed FDRL-DDQN algorithm with the distributed DDQN algorithm. We select three multimedia devices which are trained individually using the distributed DDQN algorithm without any parameter exchange between the three multimedia devices during the period. When the training is finished, we add the three multimedia devices to the FDRL-DDQN framework for retraining until convergence. The experimental results show that the cost of each device is reduced using the FDRL-DDQN algorithm, where the average consumption can be reduced by 20.3%. By combining federated learning with deep reinforcement learning, cooperative training between devices is achieved. It avoids the equipment alone training by environmental instability, action space, state space and other inexperienced impact, and provides a relatively stable intelligent body learning environment, combining different devices together intelligently. Because of the devices involved in training only upload the model parameters needed for learning, it can effectively protect the privacy and security of users. In addition, federated learning enables knowledge sharing between devices and

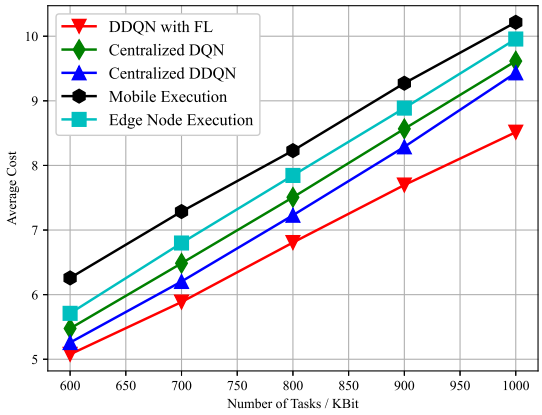

Fig. 13 Effect of the number of multimedia tasks on the total cost of the system.

enriches the data parameters they can collect.

To demonstrate the performance benefits of the FDRL-DDQN algorithm, in Fig. 13, it shows the impact of multimedia tasks data size on the average total cost of devices. We compared FDRL-DDQN algorithm with centralized DDQN algorithm, centralized DQN algorithm, mobile execution algorithm and edge note execution algorithm. It can be seen that the FDRL-DDQN has a faster learning speed than the basic centralized DDQN, and the cost rises more slowly with the number of tasks, with the smallest optimization performance at a total cost of 600 Kbits and the largest optimization performance at a total cost of 1000 Kbits. This is due to the fact that when the multimedia task data size increases, more and more information be interacted between devices, the advantage of federated learning can be fully reflected, and the learning speed of the intelligence will be faster and faster, while the centralized DDQN will decrease due to the increase of the multimedia task data size, which leads to the exponential growth of the action space received by the intelligence. FDRL-DDQN algorithm can reduce the total cost by 7.1% when the system multimedia task volume is 600 Kbit and 31.3% when the multimedia task volume is 1000 Kbit compared with centralized DDQN, while FDRL-DDQN algorithm can reduce up to 35.3% compared with mobile local offloading algorithm and 34.8% compared with edge node offloading algorithm, because Federated Deep Reinforcement Learning combines FL and DRL are organically combined to obtain the exact optimal policy by intelligent and effective learning for multiple device parameters. In addition, FDRL-DDQN can reduce the cost by up to 30.1% compared to the centralized DQN algorithm which has already widely used to offload multimedia task policies. The biggest advantage of the DDQN algorithm over the DQN algorithm is that it can ensure the stability of the target network, which helps the whole system to update the parameters and thus converge faster to get the optimal policy. When FL is combined with DDQN its effect is more obvious, FL makes DDQN algorithm more stable making the final result more accurate and faster convergence.

6. Conclusion

In this paper, we propose an adaptive offloading algorithm FDRL-DDQN that combines federation learning and deep reinforcement learning. For the computational offloading problem in the MEC scenario of mobile multimedia dynamic multimedia task arrival, we jointly allocate computational and communication resources with the goal of minimizing latency and energy consumption, and make reasonable offloading decisions. For the non-IID data problem with different multimedia devices, we design an adaptive device selection mechanism as a way to ensure the convergence of FL. In addition, we compared FDRL-DDQN with centralized Dueling DQN, distributed DDQN, mobile algorithm and edge algorithm with better results. Simulation results show that the algorithm has good latency and energy performance. In future work, we will consider resource coordination among multiple MEC servers, as well as investigate more flexible and generalized resource allocation and computational offloading strategies.

References

[1] A. Nauman, Y.A. Qadri, M. Amjad, Y.B. Zikria, M.K. Afzal, and S.W. Kim, "Multimedia internet of things: A comprehensive survey," IEEE Access, vol.8, pp.8202–8250, 2020.
[2] C.V.N. Index, "Global mobile data traffic forecast update," Cisco White Paper [Online]. Available: http://www.cisco.com/en/US/solutions/collateral/ns341/ns525/ns537/ns705/ns827/white_paper_c11-520862.pdf, 2014.
[3] D. Xu, Q. Li, and H. Zhu, "Energy-saving computation offloading by joint data compression and resource allocation for mobile-edge computing," IEEE Commun. Lett., vol.23, no.4, pp.704–707, 2019.
[4] J. Ren, G. Yu, Y. Cai, and Y. He, "Latency optimization for resource allocation in mobile-edge computation offloading," IEEE Trans. Wireless Commun., vol.17, no.8, pp.5506–5519, 2018.
[5] M. Muniswamaiah, T. Agerwala, and C.C. Tappert, "A survey on cloudlets, mobile edge, and fog computing," 2021 8th IEEE International Conference on Cyber Security and Cloud Computing (CSCloud)/2021 7th IEEE International Conference on Edge Computing and Scalable Cloud (EdgeCom), pp.139–142, 2021.
[6] X. Wang, Y. Han, C. Wang, Q. Zhao, X. Chen, and M. Chen, "In-edge AI: Intelligentizing mobile edge computing, caching and communication by federated learning," IEEE Netw., vol.33, no.5, pp.156–165, 2019.
[7] J. Ren, G. Yu, Y. Cai, Y. He, and F. Qu, "Partial offloading for latency minimization in mobile-edge computing," GLOBECOM 2017 - 2017 IEEE Global Communications Conference, pp.1–6, 2017.
[8] G. Zhang, W. Zhang, Y. Cao, D. Li, and L. Wang, "Energy-delay tradeoff for dynamic offloading in mobile-edge computing system with energy harvesting devices," IEEE Trans. Ind. Informat., vol.14, no.10, pp.4642–4655, 2018.
[9] Y. Deng, Z. Chen, and X. Chen, "Resource allocation for multi-user mobile-edge computing systems with delay constraints," GLOBECOM 2020 - 2020 IEEE Global Communications Conference, pp.1–6, 2020.
[10] H. Liu, H. Jia, J. Chen, X. Ge, Y. Li, L. Tian, and J. Shi, "Computing resource allocation of mobile edge computing networks based on potential game theory," 2018 IEEE 4th International Conference on Computer and Communications (ICCC), pp.693–699, 2018.
[11] C. You, K. Huang, H. Chae, and B.H. Kim, "Energy-efficient resource allocation for mobile-edge computation offloading," IEEE Trans.

Wireless Commun., vol.16, no.3, pp.1397–1411, 2017.
[12] L. Huang, S. Bi, and Y.J.A. Zhang, "Deep reinforcement learning for online computation offloading in wireless powered mobile-edge computing networks," IEEE Trans. Mobile Comput., vol.19, no.11, pp.2581–2593, 2020.
[13] T. Zhao, L. He, X. Huang, and F. Li, "DRL-based secure video offloading in MEC-enabled IoT networks," IEEE Internet Things J., vol.9, no.19, pp.18710–18724, 2022.
[14] J. Chen, H. Xing, Z. Xiao, L. Xu, and T. Tao, "A DRL agent for jointly optimizing computation offloading and resource allocation in MEC," IEEE Internet Things J., vol.8, no.24, pp.17508–17524, 2021.
[15] J. Wang, J. Hu, G. Min, W. Zhan, Q. Ni, and N. Georgalas, "Computation offloading in multi-access edge computing using a deep sequential model based on reinforcement learning," IEEE Commun. Mag., vol.57, no.5, pp.64–69, 2019.
[16] B. Guo, X. Zhang, Y. Wang, and H. Yang, "Deep-Q-network-based multimedia multi-service QoS optimization for mobile edge computing systems," IEEE Access, vol.7, pp.160961–160972, 2019.
[17] M. Khayyat, I.A. Elgendy, A. Muthanna, A.S. Alshahrani, S. Alharbi, and A. Koucheryavy, "Advanced deep learning-based computational offloading for multilevel vehicular edge-cloud computing networks," IEEE Access, vol.8, pp.137052–137062, 2020.
[18] J. Konen, H.B. Mcmahan, F.X. Yu, P. Richtárik, and D. Bacon, "Federated learning: Strategies for improving communication efficiency," arXiv preprint, arXiv:1610.05492v1, 2016.
[19] M. Chen, Z. Yang, W. Saad, C. Yin, H.V. Poor, and S. Cui, "A joint learning and communications framework for federated learning over wireless networks," IEEE Trans. Wireless Commun., vol.20, no.1, pp.269–283, 2020.
[20] J. Yao and N. Ansari, "Enhancing federated learning in fog-aided iot by CPU frequency and wireless power control," IEEE Internet Things J., vol.8, no.5, pp.3438–3445, 2020.
[21] R. Luo, H. Tian, and W. Ni, "Communication-aware path design for indoor robots exploiting federated deep reinforcement learning," 2021 IEEE 32nd Annual International Symposium on Personal, Indoor and Mobile Radio Communications (PIMRC), pp.1197–1202, IEEE, 2021.
[22] Z. Zhu, S. Wan, P. 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, 2021.
[23] X. Wang, C. Wang, X. Li, V.C. Leung, and T. Taleb, "Federated deep reinforcement learning for internet of things with decentralized cooperative edge caching," IEEE Internet Things J., vol.7, no.10, pp.9441–9455, 2020.
[24] H.B. McMahan, E. Moore, D. Ramage, and B.A. y Arcas, "Federated learning of deep networks using model averaging," arXiv preprint, arXiv:1602.05629, vol.2, 2016.

Rongqi Zhang is currently pursuing the M.Phill degree with the School of Information and Communication Engineering, Beijing Information Science and Technology University. He research interests include mobile edge computing, reinforcement learning and resource allocation.

Chunyu Pan received the Ph.D. degree with the School of Information and Communication Engineering, Beijing University of Posts and Telecommunications, Beijing, China. She is currently an Associate Professor in the School of Information and Communication Engineering at Information and Communication Engineering. Her research interests include cell free networks, heterogeneous wireless networks, resource allocation and mobile edge computing.

Yafei Wang received the Ph.D. degree in telecommunications engineering from the Beijing University of Posts and Telecommunications, Beijing, China, in 2013. She is currently a Professor in Beijing Information Science and Technology University. His main research interests include signal integrity analysis, and RF communication circuit design.

Yuanyuan Yao received the Ph.D. degree in information and communication engineering from Beijing University of Posts and Telecommunications, Beijing, China, in 2017. Since 2017, she has been with the School of Information and Communication Engineering, Beijing Information Science and Technology University, Beijing, China, as an associate professor. Her research interests include Cooperative communication of UAV; RIS assisted communication; Stochastic geometry and its applications in large-scale wireless networks; Intelligent Radio Resource Allocation in 6G etc.

Xuehua Li received the Ph.D. degree in telecommunications engineering from the Beijing University of Posts and Telecommunications, Beijing, China, in 2008. She is currently a Professor and the Deputy Dean of the School of Information and Communication Engineering with Beijing Information Science and Technology University, Beijing. She is a Senior Member of the Beijing Internet of Things Institute. Her research interests are in the broad areas of communications and information theory, particularly the Internet of Things, and coding for multimedia communications systems.