Computing and Informatics, Vol. 44, 2025, 245–271, doi: 10.31577/cai_2025_2_245

TASK OFFLOADING DECISION AND RESOURCE ALLOCATION STRATEGY BASED ON IMPROVED DDPG IN MOBILE EDGE COMPUTING

An Li, Yeqiang Zheng*

Center for Applied Mathematics of Guangxi, Yulin Normal University
Yulin 537000, China
e-mail: yqzheng@ylu.edu.cn

Wang Nong, Manyi Wei, Gaocai Wang

School of Computer and Electronic Information, Guangxi University
Nanning, 530004, China

Shuqiang Huang

College of Cyber Security, Jinan University
Guangzhou, 510632, China

Abstract. In mobile edge computing (MEC), the mobile device can offload tasks to the server at the edge of the mobile network for execution, thereby reducing the delay of task execution and the energy consumption of the mobile device. However, the limited resources of the edge server prevent the mobile device from offloading all tasks to the edge servers. To solve the problems, this paper constructs a multi-users and single edge server model for mobile edge computing. In order to minimize the weighted total cost combined energy consumption of mobile devices and task execution delay under the constraints of computing resources and storage resource of the edge server, we propose a task offloading decision and resource allocation algorithm based on improved deep deterministic policy gradient (DDPG) – PERD-DPG. In our algorithm, a special reward function is designed to get the reward

* Corresponding author

value for correlating negatively with the total cost. We can obtain the lowest total cost when the algorithm reaches the maximum reward value. Furthermore, we apply prioritized experience replay (PER) to improve DDPG. So, the PERDDPG has a more dynamic MEC scenario for making offloading decisions and computing resource allocation. Simulation results show that the proposed algorithm can get a better convergence speed and improve the cumulative reward compared to the existing algorithms, effectively reducing the weighted total cost of mobile devices and improving the success rate of task execution.

Keywords: Mobile edge computing, task offloading, resource allocation, improved DDPG, resource constraint

1 INTRODUCTION

With the development of mobile communication technology and the popularization of intelligent terminals, the mobile Internet has become an indispensable part of our lives. However, two technical problems derived from the mobile devices themselves seriously restrict the development of the mobile Internet. Firstly, mobile devices need to fully consider their size, weight and portability [1]. When the mobile device runs computationally intensive applications, it will cause a long delay and decrease the users' experience [2]. Secondly, battery technology, which has not made a breakthrough for a long time, also has a great impact on the users' experience. Mobile devices cannot operate efficiently due to the limitation of battery energy, and it further limits the computational performance of mobile devices. In addition, computing-intensive applications are often energy-intensive. When a user takes a longer time to run a computing-intensive task (such as online games, playing online videos, etc.), it will consume more energy from mobile devices, and decrease battery life [3].

For these disadvantages of mobile devices, the concept of Mobile Edge Computing (MEC) combined with the advantages of Mobile Cloud Computing (MCC) [4] and edge computing [5] has been proposed. As one of the cores of fifth-generation (5G) mobile communication technology, MEC sinks some services and functions located in cloud centers to the edges of the mobile network for handling computing intensive and delay-sensitive tasks by deploying computing, storage and communication resource on mobile network edge servers [6].

In MEC platform, task offloading and resource allocation are one of the key technologies of MEC. They aim to optimize the computing and allocate processes of computing-intensive or delay-sensitive tasks to reduce delay or save energy consumption of user equipment. In this paper, we propose a task offloading decision and resource allocation algorithm called PERDDPG, which efficiently addresses the challenges of task offloading and resource allocation in Multi-Access Edge Computing (MEC). Our main contributions are as follows.

1. We construct a task offloading model for a multi-users and single edge server with constraints on task execution delay, computing resource and storage resource of the edge server.
2. We apply prioritized experience replay (PER) to improve DDPG. So, the proposed algorithm PERDDPG has a more dynamic MEC scenario for making offloading decisions and computing resource allocation.
3. We combine MEC actual scenarios to optimize the task offloading decision and the edge server resource allocation scheme by using the PERDDPG algorithm proposed by this paper.
4. We verify the performance of the proposed algorithm PERDDPG in simulations. The algorithm can get a better convergence speed and improve the cumulative reward compared to the existing algorithms, effectively reduce the weighted total cost of mobile devices and improve the success rate of task execution.

2 RELATED WORK

In recent years, academic research on task offloading in MEC has primarily focused on two aspects: offloading decision and resource allocation. The first aspect, offloading decisions, involves the mobile device determining whether to offload tasks to the edge server based on the task attributes and associated costs. The second aspect, resource allocation, pertains to the edge server distributing its limited resources to enhance the efficiency of task transmission and processing after a task has been confirmed for offloading [7].

With the goal of reducing task execution delay and improving success rate, the authors propose a dynamic task offloading decision algorithm based on Lyapunov optimization theory, and the algorithm minimizes task execution delay and improves the success rate of task offloading in [8]. In [9], the authors propose a task offloading strategy for optimizing energy consumption with guaranteed delay in a specific Small Cell Networks (SCNS) MEC scenario. This strategy considers the impact of the link status of the upload network and the download network on the delay and energy consumption. In addition, the strategy uses the artificial fish group algorithm for global optimization and effectively reduces the energy consumption of mobile devices.

In [10], the authors construct a multi-user multi-task computation offloading framework using Energy Harvesting (EH) technology for Mobile Edge Cloud Computing (MECC). The framework determines the energy harvesting policy and the task offloading schedule based on Lyapunov optimization approach. In order to solve the scheduling problem which is NP-hard, centralized and distributed greedy maximal scheduling algorithm is proposed. In [11], considering trade-off energy consumption and execution time delay, the authors propose a dynamic scheduling mechanism that allows users to formulate offloading decisions based on the computational queue and wireless channel states of the task, and solve the optimization problem through convex optimization methods. In [12], the authors use queuing

theory to model the MEC task offloading system, then find the optimal offloading scheme and the optimal transmission power to calculate the execution time delay and energy consumption for each mobile device. To minimize the system energy efficiency with considering the completion time and energy, [13] proposes a distributed algorithm consisting of clock frequency configuration, transmission power allocation, and channel rate scheduling and offloading strategy selection.

In general, the above literature solves offloading decision based on some traditional methods, with poor adaptability for complex MEC environments. Driven by the development of machine learning, Deep Reinforcement Learning (DRL) has been widely applied in various fields. Therefore, using DRL to solve optimization problems in MEC is very promising. Among them, DRL has also obtained more applications in the study of MEC task offloading. In [14], the authors propose a delay-sensitive task offloading algorithm based on DRL. The algorithm combines timeout and deceleration signals to design new reward functions, and the algorithm can learn offloading decision from the environment. In order to maximize the long term performance of computing offloading policy, [15] proposes a model-free DRL-based online computing offloading algorithm for blockchain-empowered MEC. An adaptive genetic algorithm is introduced into the exploration of DRL to speed up the convergence and maintain the performance. For investigating the joint optimization of computing offloading and resource allocation in a dynamic multi-users MEC system, [16] proposes two computing offloading and resource allocation algorithms based on Q-learning and Double Deep Q Network (DDQN) methods to minimize the energy consumption for the whole MEC system. [17] constructs an offloading model with multi-service nodes and multi-dependence within mobile tasks in large-scale heterogeneous MEC. The authors propose an improved algorithm by combining Long Short-Term Memory (LSTM) network and candidate network technology. The authors focus on Software Defined Network (SDN) scenarios and propose a resource allocation algorithm to minimize the average service time based on Deep Q Network (DQN) in [18]. This algorithm can allocate computing and network resources adaptively and maintain performance in the changeable MEC environment. In [19], the authors propose a DQN based joint task offloading decision and communication bandwidth allocation algorithm. The algorithm weighs the total cost of task offloading in device energy consumption, edge server computational cost, and task execution delay.

Furthermore, an algorithm based on distributed DRL is proposed to effectively adjust computing offloading strategies in the constantly changing MEC environment of multiple unmanned aerial vehicles (UAVs) in [20]. In [21], an online offloading framework based on DRL is proposed to optimize task offloading decisions and wireless resource allocation in wireless power supply MEC networks. In [22], a joint determination of partial computing offloading destinations and computing resource allocation scheme is proposed between vehicles in a highly dynamic vehicle environment. In order to provide high quality, low latency, and low bit rate difference live streaming services for vehicles, the authors use the Soft Actor Critic (SAC) DRL algorithm for joint optimization of vehicle scheduling, bit rate selection, computing,

and spectrum resource allocation in [23]. In addition, the authors propose an optimized integrated scheme based on DRL, transformer, and model to solve end-to-end transmission problems in [24]. The authors investigate the problem of task offloading and resource allocation strategies in MEC system with heterogeneous tasks and propose a DRL-based solution in [25].

In conclusion, these methods all can effectively improve the performance of task offloading for MEC system, but do not consider the joint optimization offloading decision and the allocation of edge server computing resources. Following our current work in [26], in this paper, we construct a MEC scenario with multiple-user devices and a single-edge server to minimize the weighted total cost composed of mobile device energy consumption and task execution delay. We propose a task offloading decision and resource allocation algorithm to solve efficiently the task offloading problem in MEC system based on improved DDPG with prioritized experience replay.

3 SYSTEM MODEL

We consider a MEC scenario consisting of multiple users and a single-edge server. User Equipment (UE) generates a single task in each slot time and determines the execution site according to a certain offloading scheme. If the scheme determines that the task can be effectively executed locally, the task will be executed on the UE processing unit. If the scheme determines that the task needs to offload the edge server, the device will transfer the task to the edge server, and then the task will be executed on the edge server. The network model is shown in Figure 1.

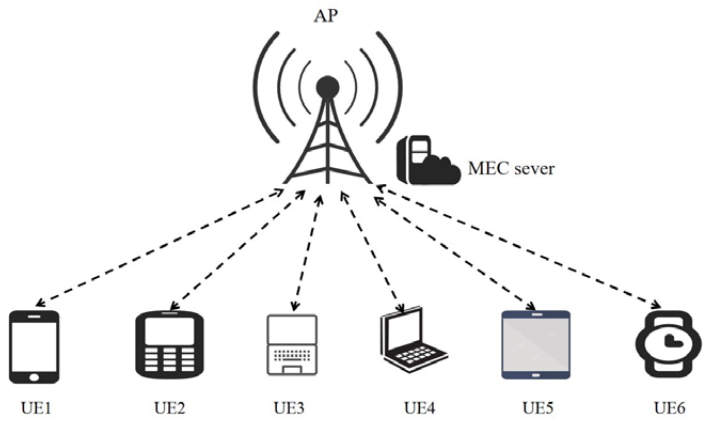


Figure 1. The network model

3.1 Scenario Description

In the above MEC model, we assume that the number of UE is $N = \{1, 2, \ldots, m\}$. Each UE has a computationally intensive task denoted by Tk_n within a same slot time, $n \in N$. The UE is connected to the Access Point (AP) through wireless channel and the MEC server is connected to the AP through the optical fiber, so the task transmission delay between the AP and the MEC server can be ignored. Each task submitted by UE can be indicated as $Tk_n = \{D_n, C_n, \tau_n\}$. Here, D_n is the size of the task, C_n is the CPU cycles required to compute per bit data, and τ_n is the maximum execution delay that task can tolerate.

In this paper, the task submitted by the UE with integrity processing request means that it cannot be split into several parts to be executed. So, each UE can only choose to execute the task locally or on the MEC server completely. We used the 0-1 variable (with $\alpha_n \in \{0, 1\}$) to represent the offloading decision of the UE. Here, if $\alpha_n = 0$, the UE chooses to perform execution locally, and if $\alpha_n = 1$, the UE chooses to offload the whole task to the MEC server for execution.

3.2 Communication Model

In our model, UE and AP are connected through wireless channel. Let g_n denote the channel gain between a UE and the AP. We assume that g_n is determined by the distance between UE and AP. Let d denote the communication distance between UE and AP. The channel gain can be calculated as follows.

$$g_n = (d)^{-\sigma}, \tag{1}$$

where σ is the path loss coefficient. Let p_n^t denote the transmit power of the task offloading from a UE to AP. The transmission rate R_t of the channel is defined as follows.

$$R_t = W \log_2 \left(1 + \frac{p_n^t g_n}{N_0 W}\right), \tag{2}$$

where W is the channel bandwidth between the UE and the AP and N_0 is the channel noise power spectral density.

3.3 Computational Model

1) Local Execution Model: UE has a certain computing capability at the local end which can handle the appropriate amount of task request. The submitted task is assigned to be executed on UE locally. Let f_n^l represent the local computing power of the UE. Thus, we can get the duration time T_n^l of local execution of Tk_n as follows.

$$T_n^l = \frac{C_n D_n}{f_n^l}. \tag{3}$$

So, the energy consumption E_n^l of local execution can be calculated as follows.

$$E_n^l = k(f_n^l)^2 C_n D_n, \tag{4}$$

where k is the energy factor which depends on chip technology of the UE. The weighted total cost C_n^l of local execution can be calculated as follows.

$$C_n^l = I_n^T T_n^l + I_n^E E_n^l. \tag{5}$$

In Equation (5), we define the weight parameters of the execution delay and the energy consumption of UE as $I_n^T + I_n^E = 1$ with $I_n^T, I_n^E \in [0, 1]$. For tasks with different requirement types, these weight parameters can be modified to highlight the optimization goal.

2) Edge Execution Model: To meet the quality of service requirements of the users, UE needs to offload some delay-sensitive tasks to the MEC server to execute them. The offloading time can be calculated as follows.

$$T_{n,t}^o = \frac{D_n}{R_t}. \tag{6}$$

The corresponding transmission energy consumption can be calculated as follows.

$$E_{n,t}^o = p_n^t T_{n,t}^o = \frac{p_n^t D_n}{R_t}. \tag{7}$$

Then, we can get the duration time $T_{n,p}^o$ of offloading execution of the task Tk_n.

$$T_{n,p}^o = \frac{C_n D_n}{f_n}, \tag{8}$$

where f_n is the computing resource assigned by the MEC server to execute the Tk_n. We define F as the amount of computing resource and M as the amount of storage resource of the MEC server. Then, we can know that the MEC server cannot allocate resource to execute tasks beyond its own limit. That is, $\sum_{n=1}^{M} \alpha_n f_n \leq F$ and $\sum_{n=1}^{M} \alpha_n D_n \leq M$. After the task is processed by the MEC server, the final data size is much smaller than the data size before executing. In addition, the download rate of the UE is much higher than the upload rate. Therefore, this model does not consider the delay that the UE downloads the executive results of the task. When the task is processed on the MEC server, the UE is waiting for acceptance. The waiting energy consumption can be calculated as follows.

$$E_{n,w}^o = p_n^w T_{n,p}^o = \frac{p_n^w C_n D_n}{f_n}. \tag{9}$$

So, the total time delay and total energy consumption of task offloading can be

calculated as follows respectively.

$$T_n^o = T_{n,t}^o + T_{n,p}^o = \frac{D_n}{R_t} + \frac{C_n D_n}{f_n}, \tag{10}$$

$$E_n^o = E_{n,t}^o + E_{n,w}^o = \frac{p_n^t D_n}{R_t} + \frac{p_n^w C_n D_n}{f_n}. \tag{11}$$

The weighted total cost of local execution can be calculated as follows.

$$C_n^o = I_n^T T_n^o + I_n^E E_n^o, \tag{12}$$

where I_n^T and I_n^E are the same parameters as the local computed model.

3.4 Problem Description

The optimization goal of this paper is to minimize the local cost which is weighted by time delay and energy consumption, while meeting the user's requirement for task delay. For a task to be successfully executed on MEC server, there are three requirements. Firstly, ensuring that the task requested storage and computing resources are less than the total resources provided by the MEC server. Secondly, the limited energy of UE needs to be satisfied. Finally, the execution time of the task on MEC server does not exceed the maximum of response delay.

The total cost of all UE is defined as follows.

$$C_{total} = \sum_{n=1}^{m} (1 - \alpha_n) C_n^l + \alpha_n C_n^0. \tag{13}$$

Assuming that the total task is x, the number of tasks successfully offloaded is y, and those successfully performed on the local end is z, then the rate of success for task execution is defined as:

$$ROS = \frac{y + z}{x}. \tag{14}$$

We must make offloading decision and resource allocation while satisfying the maximal delay and constraint of server resource. So, the optimization goal is to minimize the total cost of all UE. In summary, the problem can be constructed as

follows:

$$\begin{aligned} &\max \mathrm{ROS} \text{ and } \min C_{total} \\ &\qquad s.t.\ \ \alpha_n \in \{0,1\} \\ &\qquad 0 \le T_n^l, \quad T_n^o \le \tau_n \\ &\qquad 0 \le f_n \le F \\ &\sum_{n=1}^{M} \alpha_n f_n \le F \\ &\sum_{n=1}^{M} \alpha_n D_n \le M. \end{aligned} \tag{15}$$

In addition, tasks are executed by extracting from a task queue, and a single task is executed in a slot time on the MEC server. The default allocation of resource is always no bigger than the resource of the MEC server.

4 DESIGN OF OFFLOADING STRATEGY AND RESOURCE ALLOCATION ALGORITHM BASED ON IMPROVED DDPG

This paper presents a joint modeling of task offloading decision-making and server resource allocation solving problems in a multi-user and single-edge server MEC system. The goal is to maximize task execution success rate while minimizing the weighted total cost of users. This is a mixed integer nonlinear programming problem. To solve this problem, this section proposes an improved DDPG based offloading decision and resource allocation algorithm PERDDPG, which aims to minimize the weighted total cost composed of task execution delay and user's device energy consumption, as well as maximize the task execution success rate.

4.1 MDP Model

In general, using the DRL algorithm to solve problems requires constructing several key elements, namely state space, action space, and reward function, which are defined as follows.

1) State space S The MEC model considered in this paper uses the attribute information of arriving tasks and the remaining resource of edge server to simulate the state of the entire MEC system. The state space of the MEC environment can be represented as: $S_t = (Tk_1^t, \ldots, Tk_n^t, RC_t, RM_t)$, where $Tk_n^t = \{D_n^t, C_n^t, \tau_n^t\}$ indicates the task attribute in the current state, including the data size of the task D_n^t, the CPU cycles required to compute per bit data C_n^t, and the delay constraints of the task τ_n^t. $RC_t = RC_{t-1} - \sum_{i=0}^{n} \alpha_i^{t-1} f_i^{t-1}$

represents the remaining computing resource of the MEC server, and $RM_t = RM_{t-1} - \sum_{i=0}^{n} \alpha_i^{t-1} D_i^{t-1}$ represents the storage resource of the MEC server.

2) Action space α: Action space is the set of all action decisions that agent can make in the state. In the MEC system of this paper, the action space is composed of two parts: task offloading decision and computational resource allocation vector, and it can be represented as $\alpha_t = (\alpha_1^t, \ldots, \alpha_n^t, f_1^t, \ldots, f_n^t)$.

3) Reward function R: Rewards are used to evaluate the effectiveness of the action α_t selected by the algorithm in the current state S_t on the system, and the rationality of the reward function design greatly affects the performance of the algorithm. In general, the reward function needs to be designed based on optimization objectives. The goal of this paper is to maximize the success rate of task execution while minimizing the weighted total cost of UE. The goal of the PERDDPG algorithm is to achieve the maximum cumulative reward, so the reward function is negatively correlated with the weighted total cost, which can be set as:

$$r_t(S_t, \alpha_t) = \begin{cases} 1, & \text{violate constraints,} \\ 1 - \alpha_n \frac{C_n^0}{C_{max}} - (1 - \alpha_n) \frac{C_n^l}{C_{max}}, & \text{otherwise,} \end{cases} \tag{16}$$

where C_{max} is the maximal weighted total cost of the MEC system. From the above formula (16), when the task execution fails, that is, the delay of TK_n exceeds the delay, the amount of task data to be offloaded exceeds the remaining storage space of the edge computing node, or the computing resources allocated to the task exceed the remaining computing resources of the edge computing node, the minimal reward value is -1. When the task can be executed smoothly, the current reward value is calculated based on the weighted cost of the task. The smaller the weighted cost of the task, the greater the reward value.

4.2 DDPG Algorithm

DDPG algorithm consists of Actor network and Critic network [27]. This paper uses two fully connected neural networks with three-layer to fit. Using $\mu(s|\theta^\mu)$ parameterized represents Actor network, and $Q(s, \alpha|\theta^Q)$ parameterized represents Critic network. Here, θ^μ and θ^Q are network parameters. In order to increase the stability of network updates, DDPG continues the empirical reuse in the DQN algorithm to perform non-policy training on the network, and then minimize the correlation between samples. In order to provide consistent targets during time difference backup, DDPG also uses target network technology to train the network. The target Actor network and target Critic network can be parameterized as follows: $\mu'(s|\theta^{\mu'})$ and $Q'(s, \alpha|\theta^{Q'})$. Here, $\theta^{\mu'}$ and $\theta^{Q'}$ are network parameters. The initial parameters of the target network are the same as those of the online network. Online Actor network uses deep neural networks for deterministic behavior strategies μ simulate. To balance exploration during the network update process, the DDPG

algorithm introduces random noise to increase the randomness of action generation. The action selection formula is:

$$\alpha^t = \mu(S_t|\theta^\mu) + \eta. \tag{17}$$

Here, η is the exploration noise.

In the DDPG algorithm, the Critic network also uses deep neural networks to simulate the Q function. The Q function is known as the state-action value function, and it is defined as expected cumulative reward for the agent to continuously execute policy μ at in state S_t after taking action α_t. The expected cumulative reward is defined by the Bellman equation as

$$Q^\mu(S_t, \alpha_t) = E[r(S_t, \alpha_t) + \gamma Q^\mu(S_{t+1}, \mu(S_{t+1}))]. \tag{18}$$

Here, $r(S_t, \alpha_t)$ represents the immediate reward value returned by the agent after executing action α_t in state S_t. Using functions $J(\mu)$ to evaluate the current strategy μ. The evaluation function $J(\mu)$ is given as follows.

$$J(\mu) = \int_S \rho^\mu(S) Q^\mu(S, \mu_\theta(S))\,\mathrm{d}s = E_{S\sim\rho^\mu}[Q^\mu(S, \mu_\theta(S))]. \tag{19}$$

Where, $J(\mu)$ is the expectation of function $Q^\mu(S, \mu(S))$, when the state S obeys the distribution ρ^π and the strategy of action formulation is μ. In short, the Actor network is responsible for generating actions, and the Critic network for value evaluation.

To improve the generalization performance of the DDPG algorithm, this algorithm stores learning experiences through experiment replay technology, and then updates the parameters of deep neural networks through random sampling. The specific approach is to store the state transfer sequences $e_t = (S_t, \alpha_t, r_t, S_{t+1})$ obtained from agent interaction with the environment into the experience replay buffer $D = e_1, e_2, \ldots, e_t$ at each slot time t. During each training process, small batches of samples were randomly drawn from the experience replay buffer D, and the network parameters were updated using Adaptive Moment Estimation (Adam) algorithm. This approach disrupts the correlations between learning experience and reduces the variance generated during the updating of the network parameters while improving data utilization.

The algorithm training objective is to maximize function $J(\mu)$ and minimize the loss of the Critic network at the same time. In the update phase, the algorithm randomly selects Z groups of experience from the experience reuse pool D to calculate the target reward value y_i. The calculation steps of y_i are given as follows: Firstly, the next state vector S_{i+1} is input into the target Actor network, agent get the action α_{i+1}. Secondly, α_{i+1} is connected to S_{i+1} as the input of the target Critic network to obtain the target value Q'. The target reward value y_i can be represented as:

$$y_i = r_i + \gamma Q'(S_{i+1}, \alpha_{i+1}|\theta^{Q'}). \tag{20}$$

Thirdly, α_i is connected to S_i as the input of the online Critic network to get the actual target value Q. Finally, the error of the online Critic network is obtained according to the mean square error equation defined as Equation (21), and the network is updated by minimizing the modification error.

$$L(\theta^Q) = \frac{1}{Z}\sum_i (y_i - Q(S_i, \alpha_i|\theta^Q))^2. \tag{21}$$

For update of the online Actor network, the policy gradient was used to determine the update direction. The gradient formula is given as follows:

$$\nabla_{\theta^\mu} J = \frac{1}{Z}\sum_i \nabla_a Q(S_i, \alpha_i|\theta^Q)\nabla_{\theta^\mu}\mu(S_i|\theta^\mu). \tag{22}$$

Furthermore, in order to increase the stability of the learning process, the parameters $\theta^{\mu'}$ and $\theta^{Q'}$ of the target Actor network and the target Critic network are updated through iteration, and the update formula is given as follows:

$$\theta^{\mu'} = \lambda\theta^\mu + (1-\lambda)\theta^{\mu'}, \tag{23}$$

$$\theta^{Q'} = \lambda\theta^Q + (1-\lambda)\theta^{Q'}, \tag{24}$$

where λ is the soft update coefficient.

4.3 Improved Algorithm – PERDDPG

The uniform random sampling method used in the DDPG algorithm disrupts the correlation between learning experiences, improves data utilization, and reduces the variance generated when updating network parameters. However, the contribution of different empirical data to gradient learning varies. Ignoring the importance of empirical data can lead to low learning efficiency and even overfitting of neural networks. In actual MEC scenarios, there may be situations where rewards are sparse, making it difficult for algorithms to learn better strategies and thus unable to converge. The priority experience replay framework [28] can assign higher sampling weights to experience data with high learning efficiency, increasing its probability of being sampled. This paper uses the improved DDPG algorithm with priority experience replay to solve the problem of sparse rewards in MEC.

The core idea of priority experience replay is to sample experience data with extreme value more frequently. The standard for measuring the value of empirical data is a core issue. In most RL algorithms, TD error is often used to update the estimation of the Q function. And the larger the absolute TD error, the more positive the correction of the expected action evaluation value will be. Empirical data with high TD error contains higher value. Therefore, we choose the absolute TD error $|\delta|$ of experience as an indicator to evaluate the value of experience, and

the error δ_j of experience j can be expressed as:

$$\delta_j = \gamma(S_t, \alpha_t) + \gamma Q'(S_{t+1}, \alpha_{t+1}) - Q(S_t, \alpha_t), \tag{25}$$

where $Q'(S_{t+1}, \alpha_{t+1})$ is the target Q value. The larger the error δ_j of experience, the greater its contribution to gradient learning, and it should be selected first during sampling. Therefore, we define the priority indicator p_j of experience j as:

$$p_j = |\delta_j| + \psi, \tag{26}$$

where ψ is a very small normal number, and used to prevent a probability of 0.

The probability of experience j being sampled is:

$$\text{Prob}_j = \frac{p_j^{\varepsilon}}{\sum_K p_K^{\varepsilon}}, \tag{27}$$

where ε is the priority weight. When $\varepsilon = 0$, the sampling method is changed to random sampling. K is the total number of experiences.

To avoid the need to sort all empirical data based on the priority metric p_j for each sampling, we use the Sum-Tree data structure to store the empirical data, where the experience reuse pool D is a Sum-Tree. Sum-Tree is an efficient binary tree storage structure, where each leaf node stores a priority metric for a sample, namely the sampling probability Prob, and training data is sampled from the node. The intermediate node stores the sum of priority indicators for the child nodes, while the root node is the sum of priority indicators for all experience samples. During experience sampling, the Sum-Tree root node, i.e. the sum, is divided into Z segments according to the algorithm's set mini-batch Z size. A random number is selected for each segment, and then the root node of the Sum-Tree is traversed downwards according to a certain rule for each number. Finally, the priority index p_j searched is matched with the experience data to achieve more efficient experience replay. The Sum-Tree structure is shown in Figure 2.

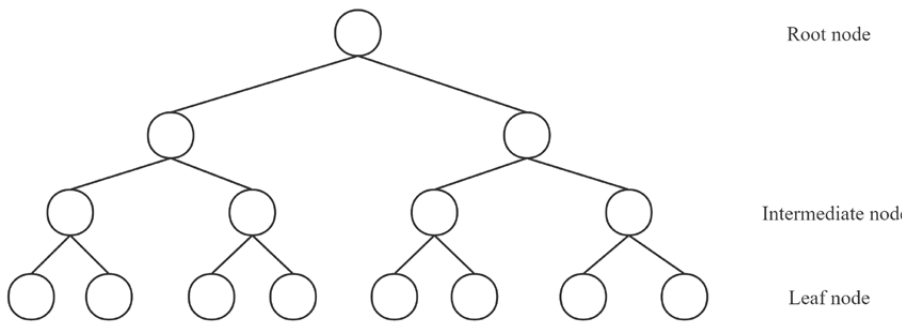


Figure 2. Sum-Tree structure

In the process of network updates, frequent replay of experiences with higher priority indicators changes the estimation of Q values in an uncontrolled manner, increasing the difficulty of training DNN and causing bias in the action generation strategy that obtains the highest cumulative return (even if the state distribution is fixed). We correct this bias by adding importance sampling weights during network updates:

$$w_i = \left(\frac{1}{K \cdot \text{Prob}_j}\right)^{\beta}. \tag{28}$$

Among them, β is the weight indicator, and the improved Critic network error function of the algorithm is updated by the following formula:

$$L\left(\theta^Q\right) = \frac{1}{Z}\sum_i w_i \left(y_i - Q\left(S_i, \alpha_i|\theta^Q\right)\right)^2. \tag{29}$$

The PERDDPG algorithm structure is shown in Figure 3.

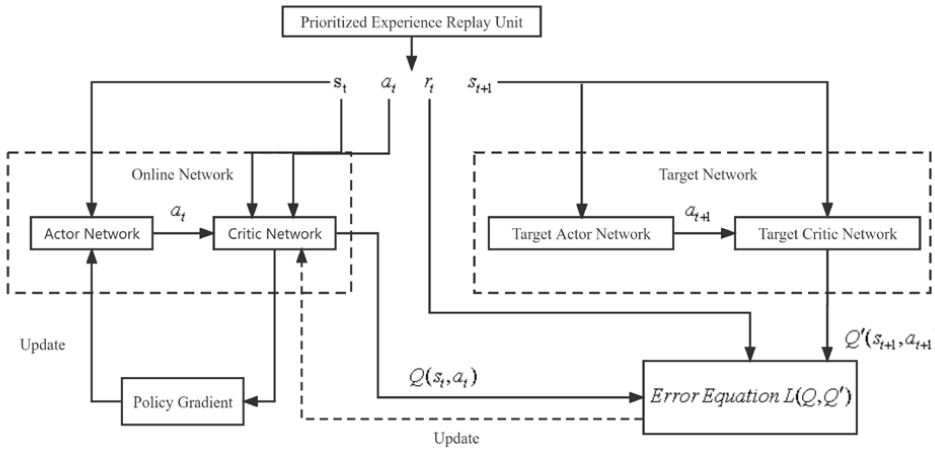


Figure 3. PERDDPG network structure

The PERDDPG algorithm steps are shown in the following.

5 SIMULATION RESULTS AND ANALYSIS

5.1 Experimental Environment and Parameters Setting

The simulation in this paper were performed on a PC with 16 GB memory, Intel Core i5 CPU (frequency is 2.3 GHz) and the Windows 10 home Chinese version OS. The Integrated Development Environment (IDE) of the simulation is Python 3.7. We call the Facebook's PyTorch library as the deep learning framework for

Algorithm 1 Task Offloading and Resource Allocation Algorithm Based on Improved DDPG (PERDDPG)

Input: The set of tasks generated by UE, the information transmission rate between UE and AP, and the remaining resources of edge servers.

Output: Decision on task offloading and allocation of computing resources.

Step 1: Initialize the MEC system environment;

Step 2: Initialize PERDDPG algorithm model;

Step 3: Initialize neural network parameters;

Step 4: Initialize Sum-Tree;

Step 5: For every episode;

Step 6: Initialize action exploration noise η, discount factor γ, soft update coefficient λ, target network update interval UF;

Step 7: Get the initial state S_1;

Step 8: For $t = 1$ to T;

1. Input the current state S_t and exploration noise into the online Actor network, and output the action α_t;
2. Execute action α_t, get reward r_t and the next state α_{t+1} from the environment;
3. Obtain Q reality value through online Critic network, and obtain Q target value through target Critic network;
4. According to Equation (18), calculate the priority indicators p_j corresponding to empirical data $(S_t, \alpha_t, r_t, S_{t+1})$, and store them in Sum-Tree;
5. According to rules, sample Z sampling priority indicators from Sum-Tree, and obtain corresponding empirical data;
6. Calculate the target return value using empirical data according to Equation (20);
7. Update online Critic network parameters by minimizing Equation (29);
8. Calculate the policy gradient according to Equation (22) and update the online policy network
9. If $t \%\% UF = 0$;
   Update the target network by using Equations (23) and (24);
   If $t + 1 = T$;
   End for
   Else
   Let $S_t = S_{t+1}$, and go to 6.

the PERDDPG algorithm. In addition, we use two DNNs with three-layer to fit the Actor network and the Critic network, and simulate a scenario of a multi-user device with a single edge server, where each UE can only send an offloading request for one task at a slot time. The experimental parameters referred to [29, 30] are shown in Table 1.

Parameters	**Parameter Value**
Data size of task D_n/MB	Uniform $[3, 5]$
CPU cycles required to process 1-bit data C_n/(Hz/bit)	500
Delay tolerance of task τ_n/s	13
Computing capability of UE f_n^l/GHz	$[1, 1.2, 1.4, 1.6, 1.8, 2]$
Transmission power of UE p_n^t/W	1
Waiting power of UE p_n^w/W	0.1
Coordinate range of UE d_n^x, d_n^y	0.1
Height for AP h/m	10
Channel Bandwidth W/MHz	2.0
Spectral density for noise power N_0/(dBm/Hz)	-174
Allocation range of computing resource f_n^o/GHz	$[10, 20]$
Path loss factor σ	4
Weight for energy consumption I_n^E	0.5
Discount factor γ	0.9
Soft update coefficient λ	0.001
Priority weight ε	0.6
Channel gain g_n	5×10^{-5}
Weight for importance level	0.4
Noise	Gaussian noise
Optimizer	Adam

Table 1. Experimental Parameters

5.2 Comparison Algorithms

To evaluate and validate the performance of the proposed PERDDPG algorithm in this paper, the following three algorithms are introduced for comparison.

1) **All Local Execution (All-local):** All tasks are selected to execute locally.
2) **Random Offloading Execution (Random):** Tasks are randomly offloaded to execute on local UE or MEC server. If the tasks are offloaded to the MEC server, and MEC server will randomly assign computing resource to execute them.
3) **Offloading decision and resource allocation algorithm based on DDPG:** A task offloading decision and resources allocation algorithm designed using DRL.

5.3 Experimental Results Analysis

1) Algorithm Convergence. Figure 4 compares the training performance of DDPG algorithm and PERDDPG algorithm under the same network parameters. As shown in Figure 4, under the same network parameters, both the DDPG algorithm and the PERDDPG algorithm can ultimately achieve relatively stable convergence. The DDPG algorithm converged at 900 episodes, with a cumulative reward value of 48. The PERDDPG algorithm converged at 400 episodes, with a cumulative reward value of 71. From the above data, it can be found that compared with the DDPG algorithm, the PERDDPG algorithm adopts PER technology, which enables the network to converge faster and achieve higher cumulative rewards in complex interactive environments.

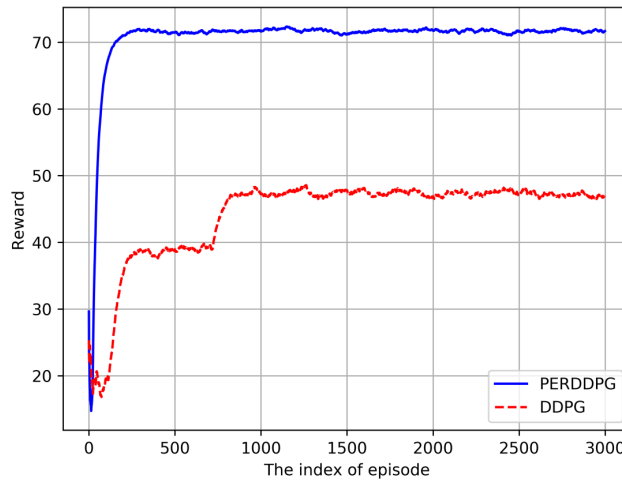


Figure 4. Comparison of the algorithm convergence performance

2) Effect of Various Factors on the Total Cost. In simulations, we compare and analyze the influence of the weighted total cost of mobile devices with different number of UE, different storage resource and computing resource of MEC server.

Figure 5 shows the experimental results of comparing algorithms with varying numbers of UE. As the number of UE increases, the total cost of the four algorithms also naturally increases. The PERDDPG algorithm proposed in this paper has always achieved the lowest total cost, saving 4.4 % to 15 % of the total cost compared to the DDPG algorithm. The performance of both deep reinforcement learning algorithms is better than that of the Random algorithm and All-local algorithm. When the number of mobile devices is 5, 10, and 15, the total cost saved by the DDPG and PERDDPG algorithms compared to the other two algorithms increases with the increase of the number of UEs. However,

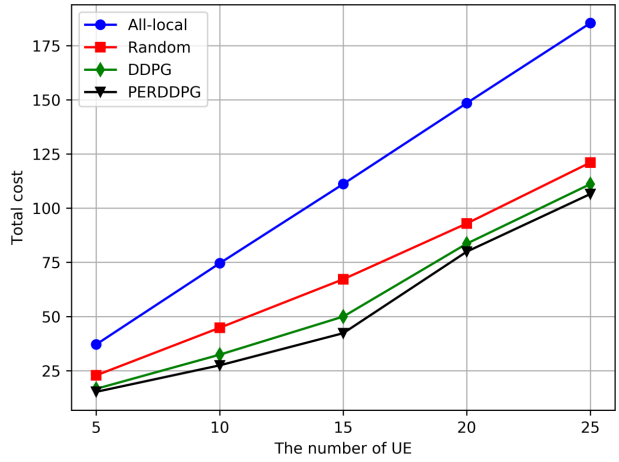


Figure 5. Effect of the UE quantity on the total cost

when the number of UEs is greater than 20, the total cost of these two algorithms is close to that of the Random algorithm. The reason is that server computing and storage resources are limited, and the best effect can only be achieved with less than 15 UEs. Having too many devices can lead to incomplete task offloading, and only tasks with high computational load can be offloaded for execution, while the remaining choices are executed locally.

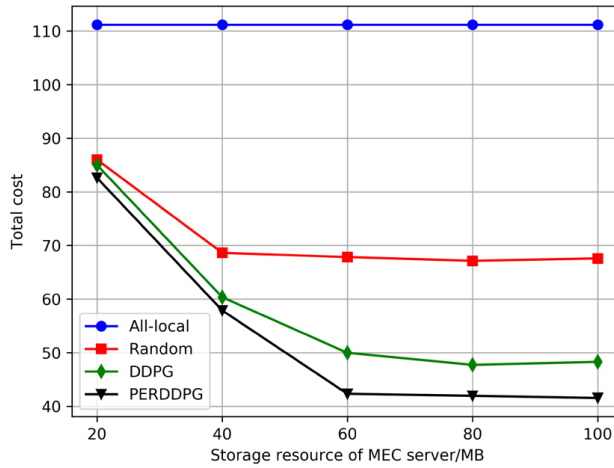


Figure 6. Effect of the MEC server storage resource on the total cost

Figure 6 shows the experimental results of comparing algorithms in an environment setting where edge servers store resources from small to large. Because

all tasks of the All-local algorithm are executed locally, the storage resources of the edge server have no impact on the total cost of the All-local algorithm. When the storage on the edge server is less than 60 MB, as the server storage resources increase, the Random, DDPG, and PERDDPG algorithms can offload more tasks to the edge server for execution, resulting in a decrease in total overhead. When the storage resources of the edge server exceed 60 MB, increasing the storage resources of the edge server can no longer significantly reduce the total cost of mobile devices. The reason is that the 60 MB storage space is already sufficient to meet the algorithm's optimal task offloading decision requirements in the current environment. The storage space of the edge server is no longer the main factor limiting the task offloading to the edge server for execution. The PERDDPG algorithm proposed in this article can reduce the total overhead by 2.8 % to 15.3 % compared to the DDPG algorithm.

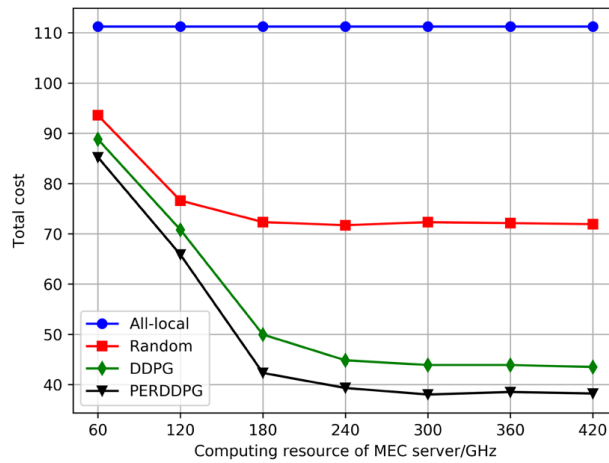


Figure 7. Effect of the MEC server computing resource on the total cost

Figure 7 shows the experimental results of comparing algorithms in an environment setting where computing resources on edge servers are increased from small to large. Because all tasks of the All-local algorithm are executed locally, the computing resources of the edge server have no impact on the total cost of the All-local algorithm. The Random algorithm offloads nearly 50 % of tasks to edge servers for execution. When the computing resources of edge servers are 180 GHz, the maximum resource requirements for these tasks can be met. Therefore, when the computing resources of edge servers increase to 180 GHz, the total cost of the Random algorithm no longer decreases. As the computing resources of edge servers increase, DDPG algorithm and PERDDPG algorithm will offload more tasks to edge servers for execution and allocate more computing resources to these tasks under limited storage resources, achieving the effect of reducing total overhead. The PERDDPG algorithm performs better than the

DDPG algorithm throughout the entire process, and it can save up to 15.3 % of the total cost compared to the DDPG algorithm.

3) Effect of Various Factors on the Success Rate of Task Execution. In these experiments, we compare and analyze the influence of the weighted total cost of mobile devices with different number of UE, storage resource and computing resource of MEC server.

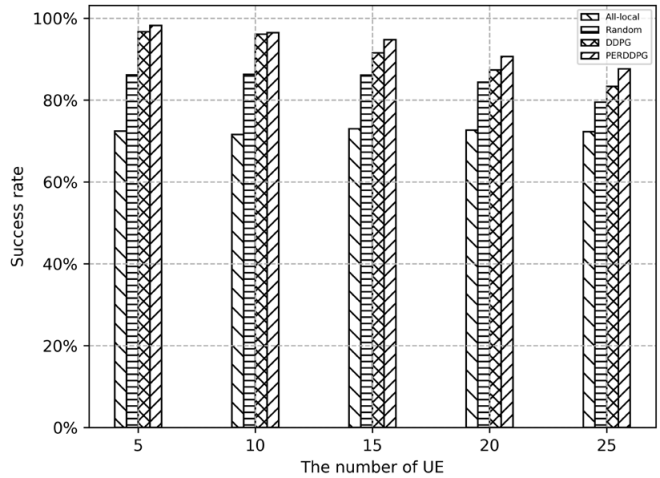


Figure 8. Effect of the UE quantity on the success rate of task execution

Figure 8 shows the experimental results of the comparison algorithm in environments with varying numbers of UE. As the number of UE increases, the task execution success rate of the PERDDPG algorithm proposed in this article has always been higher than the other three comparative algorithms. When the number of UE is less than 15, the DDPG algorithm can achieve a success rate similar to the PERDDPG algorithm, indicating that the effectiveness of these two algorithms is similar when the edge server resources are abundant. When the number of UEs is greater than 15, the task execution success rates of PERDDPG algorithm, DDPG algorithm, and Random algorithm will have a certain decrease with the increase of the number of UEs, with a significant difference. This is due to the limited resources of edge servers, which cannot meet the offloading needs of too many tasks, resulting in tasks only being able to be executed locally.

Figure 9 shows the experimental results of comparing algorithms in an environment setting where edge servers store resources from small to large. Because all tasks of the All-local algorithm are executed locally, the storage resources of the edge server have no impact on the success rate of the All-local algorithm's task execution. With the increase of edge server storage resources, the task execution success rate of the PERDDPG algorithm proposed in this article has always

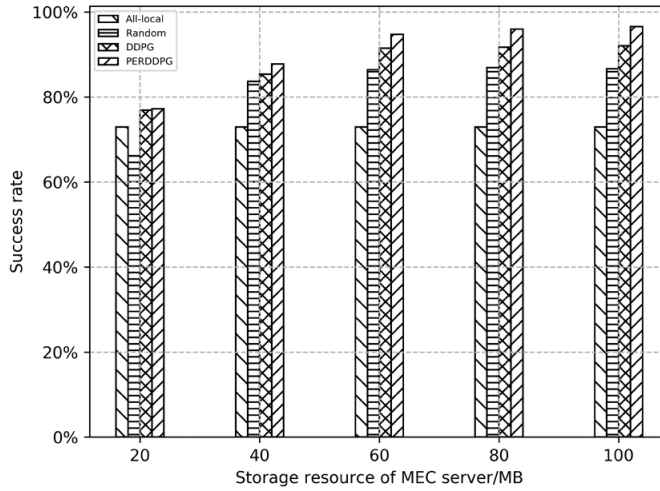


Figure 9. Effect of MEC server storage resource on the success rate of task execution

been higher than the other three comparative algorithms. When the storage resources is 20 MB, the Random algorithm will make more decisions that violate constraints, so the success rate of task execution is even lower than the All-local algorithm. When the storage on the edge server is less than 60 MB, as the server storage resources increase, the Random, DDPG, and PERDDPG algorithms can offload more tasks to the edge server for execution, resulting in an increasing success rate. When the storage resources of the edge server exceed 60 MB, the task execution success rate of the Random algorithm is no longer affected by the storage resources of the edge server, while the task execution success rate of the PERDDPG algorithm and DDPG algorithm will slightly increase.

Figure 10 shows the experimental results of comparing algorithms in an environment setting where computing resources on edge servers are increased from small to large. Because all tasks of the All-local algorithm are executed locally, the computing resources of the edge server have no impact on the success rate of the All-local algorithm's task execution. The Random algorithm offloads nearly 50 % of tasks to the edge server for execution. Insufficient computing resources on the edge server can result in the inability to meet the resource requirements of these tasks, causing some tasks to fail and resulting in a lower success rate than the All-local algorithm; When the computing resources of the edge server are greater than 180 GHz, the maximum resource requirement for this half of the task can be met. Therefore, when the computing resources of the edge server increase to 180 GHz, the success rate of the Random algorithm's task execution no longer increases. When the computing resources of the server are less than 300 GHz, as the computing resources of the edge server increase, the DDPG algorithm and PERDDPG algorithm will offload more tasks to the edge server

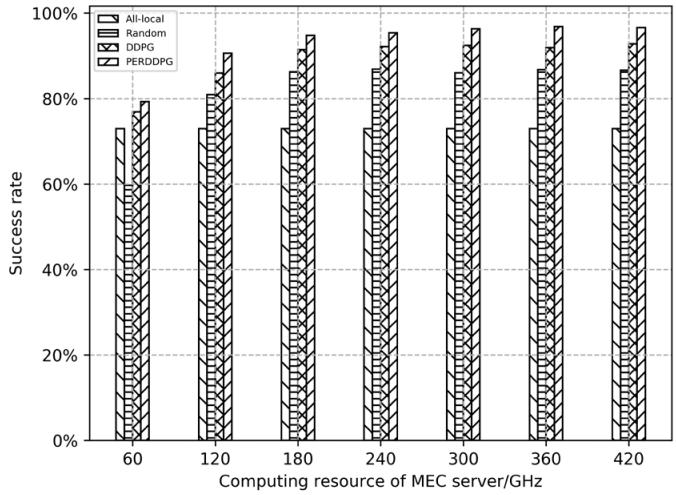


Figure 10. Effect of MEC server computing resource on the success rate of task execution

for execution under limited storage resources and allocate more computing resources to these tasks, achieving the effect of increasing the success rate of task execution. When the computing resources of the edge server reach 300 GHz, it can already allocate the maximum computing resources for all offloading tasks, and the computing resources of the edge server are no longer the main factor limiting the success rate of task execution.

6 CONCLUSIONS

In mobile edge computing, to reduce the delay of task execution and the energy consumption of mobile devices, a multi-users and single-edge server model is constructed in this paper. In order to minimize the weighted total cost combined energy consumption of mobile device and task execution delay under the constraints with computing resources and storage resources of the edge server, we propose a task offloading decision and resource allocation algorithm based on improved DDPG. A special reward function is designed to get the reward value for correlating negatively with the total cost. We can obtain the lowest total cost when the algorithm reaches the maximum reward value. Furthermore, we apply PER to improve DDPG. So, this improved algorithm has a more dynamic MEC scenario for making offloading decisions and computing resources allocation. Simulation results show that the proposed algorithm can get a better convergence speed and improve the cumulative reward compared to the existing algorithms, effectively reduce the weighted total cost of mobile devices and improve the success rate of task execution.

In future work, we can consider complicating the MEC model, such as the mobility of mobile devices and multi-edge servers offloading. We can also assume that

the edge servers have the ability to cache tasks to make more comprehensive task offloading decision. Combined with the cloud computing technology, the task offloading decision under side cloud cooperation can be considered.

Acknowledgments

This work was supported in part by the Natural Science Foundation of Guangxi (grant No. 2025GXNSFAA069236), National Natural Science Foundation of China (grant No. 62272198), Natural Science Foundation of Guangdong Province (grant No. 2024A1515010121).

Competing Interests

All authors disclosed no relevant relationships.

REFERENCES

[1] Flinn, J.: Cyber Foraging: Bridging Mobile and Cloud Computing. Springer, 2012, doi: 10.1007/978-3-031-02481-8.

[2] Dinh, H. T.—Lee, C.—Niyato, D.—Wang, P.: A Survey of Mobile Cloud Computing: Architecture, Applications, and Approaches. Wireless Communications and Mobile Computing, Vol. 13, 2013, No. 18, pp. 1587–1611, doi: 10.1002/wcm.1203.

[3] Satyanarayanan, M.: Mobile Computing: The Next Decade. ACM SIGMOBILE Mobile Computing and Communications Review, Vol. 15, 2011, No. 2, pp. 2–10, doi: 10.1145/2016598.2016600.

[4] Khan, A. U. R.—Othman, M.—Madani, S. A.—Khan, S. U.: A Survey of Mobile Cloud Computing Application Models. IEEE Communications Surveys & Tutorials, Vol. 16, 2014, No. 1, pp. 393–413, doi: 10.1109/SURV.2013.062613.00160.

[5] Shi, W.—Cao, J.—Zhang, Q.—Li, Y.—Xu, L.: Edge Computing: Vision and Challenges. IEEE Internet of Things Journal, Vol. 3, 2016, No. 5, pp. 637–646, doi: 10.1109/JIOT.2016.2579198.

[6] Premsankar, G.—Di Francesco, M.—Taleb, T.: Edge Computing for the Internet of Things: A Case Study. IEEE Internet of Things Journal, Vol. 5, 2018, No. 2, pp. 1275–1284, doi: 10.1109/JIOT.2018.2805263.

[7] Mach, P.—Becvar, Z.: Mobile Edge Computing: A Survey on Architecture and Computation Offloading. IEEE Communications Surveys & Tutorials, Vol. 19, 2017, No. 3, pp. 1628–1656, doi: 10.1109/COMST.2017.2682318.

[8] Mao, Y.—Zhang, J.—Letaief, K. B.: Dynamic Computation Offloading for Mobile-Edge Computing with Energy Harvesting Devices. IEEE Journal on Selected Areas in Communications, Vol. 34, 2016, No. 12, pp. 3590–3605, doi: 10.1109/JSAC.2016.2611964.

[9] Zhang, H.—Guo, J.—Yang, L.—Li, X.—Ji, H.: Computation Offloading Considering Fronthaul and Backhaul in Small-Cell Networks Integrated with MEC. 2017 IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS), 2017, pp. 115–120, doi: 10.1109/INFCOMW.2017.8116362.

[10] Chen, W.—Wang, D.—Li, K.: Multi-User Multi-Task Computation Offloading in Green Mobile Edge Cloud Computing. IEEE Transactions on Services Computing, Vol. 12, 2019, No. 5, pp. 726–738, doi: 10.1109/TSC.2018.2826544.

[11] Muñoz, O.—Pascual-Iserte, A.—Vidal, J.: Optimization of Radio and Computational Resources for Energy Efficiency in Latency-Constrained Application Offloading. IEEE Transactions on Vehicular Technology, Vol. 64, 2015, No. 10, pp. 4738–4755, doi: 10.1109/TVT.2014.2372852.

[12] Liu, L.—Chang, Z.—Guo, X.—Ristaniemi, T.: Multi-Objective Optimization for Computation Offloading in Mobile-Edge Computing. 2017 IEEE Symposium on Computers and Communications (ISCC), 2017, pp. 832–837, doi: 10.1109/ISCC.2017.8024630.

[13] Wang, Q.—Guo, S.—Liu, J.—Yang, Y.: Energy-Efficient Computation Offloading and Resource Allocation for Delay-Sensitive Mobile Edge Computing. Sustainable Computing: Informatics and Systems, Vol. 21, 2019, pp. 154–164, doi: 10.1016/j.suscom.2019.01.007.

[14] Meng, H.—Chao, D.—Guo, Q.—Li, X.: Delay-Sensitive Task Scheduling with Deep Reinforcement Learning in Mobile-Edge Computing Systems. Journal of Physics: Conference Series, Vol. 1229, 2019, No. 1, Art. No. 012059, doi: 10.1088/1742-6596/1229/1/012059.

[15] Qiu, X.—Liu, L.—Chen, W.—Hong, Z.—Zheng, Z.: Online Deep Reinforcement Learning for Computation Offloading in Blockchain-Empowered Mobile Edge Computing. IEEE Transactions on Vehicular Technology, Vol. 68, 2019, No. 8, pp. 8050–8062, doi: 10.1109/TVT.2019.2924015.

[16] Zhou, H.—Jiang, K.—Liu, X.—Li, X.—Leung, V. C.: Deep Reinforcement Learning for Energy-Efficient Computation Offloading in Mobile-Edge Computing. IEEE Internet of Things Journal, Vol. 9, 2022, No. 2, pp. 1517–1530, doi: 10.1109/JIOT.2021.3091142.

[17] Lu, H.—Gu, C.—Luo, F.—Ding, W.—Liu, X.: Optimization of Lightweight Task Offloading Strategy for Mobile Edge Computing Based on Deep Reinforcement Learning. Future Generation Computer Systems, Vol. 102, 2020, pp. 847–861, doi: 10.1016/j.future.2019.07.019.

[18] Wang, J.—Zhao, L.—Liu, J.—Kato, N.: Smart Resource Allocation for Mobile Edge Computing: A Deep Reinforcement Learning Approach. IEEE Transactions on Emerging Topics in Computing, Vol. 9, 2021, No. 3, pp. 1529–1541, doi: 10.1109/TETC.2019.2902661.

[19] Huang, L.—Feng, X.—Zhang, C.—Qian, L.—Wu, Y.: Deep Reinforcement Learning-Based Joint Task Offloading and Bandwidth Allocation for Multi-User Mobile Edge Computing. Digital Communications and Networks, Vol. 5, 2019, No. 1, pp. 10–17, doi: 10.1016/j.dcan.2018.10.003.

[20] Wang, Z.—Lv, T.—Chang, Z.: Computation Offloading and Resource Al-

location Based on Distributed Deep Learning and Software Defined Mobile Edge Computing. Computer Networks, Vol. 205, 2022, Art. No. 108732, doi: 10.1016/j.comnet.2021.108732.

[21] Huang, L.—Bi, S.—Zhang, Y. J. A.: Deep Reinforcement Learning for Online Computation Offloading in Wireless Powered Mobile-Edge Computing Networks. IEEE Transactions on Mobile Computing, Vol. 19, 2020, No. 11, pp. 2581–2593, doi: 10.1109/TMC.2019.2928811.

[22] Shi, J.—Du, J.—Wang, J.—Yuan, J.: Deep Reinforcement Learning-Based V2V Partial Computation Offloading in Vehicular Fog Computing. 2021 IEEE Wireless Communications and Networking Conference (WCNC), 2021, pp. 1–6, doi: 10.1109/WCNC49053.2021.9417450.

[23] Fu, F.—Kang, Y.—Zhang, Z.—Yu, F. R.—Wu, T.: Soft Actor-Critic DRL for Live Transcoding and Streaming in Vehicular Fog-Computing-Enabled IoV. IEEE Internet of Things Journal, Vol. 8, 2021, No. 3, pp. 1308–1321, doi: 10.1109/JIOT.2020.3003398.

[24] Wang, S.—Bi, S.—Zhang, Y. J. A.: Deep Reinforcement Learning with Communication Transformer for Adaptive Live Streaming in Wireless Edge Networks. IEEE Journal on Selected Areas in Communications, Vol. 40, 2022, No. 1, pp. 308–322, doi: 10.1109/JSAC.2021.3126062.

[25] Jiang, T.—Chen, Z.—Zhao, Z.—Feng, M.—Zhou, J.: Deep-Reinforcement-Learning-Based Task Offloading and Resource Allocation in Mobile Edge Computing Network with Heterogeneous Tasks. IEEE Internet of Things Journal, Vol. 12, 2025, No. 8, pp. 10899–10906, doi: 10.1109/JIOT.2024.3514108.

[26] Peng, Z.—Wang, G.—Nong, W.—Qiu, Y.—Huang, S.: Task Offloading in Multiple-Services Mobile Edge Computing: A Deep Reinforcement Learning Algorithm. Computer Communications, Vol. 202, 2023, pp. 1–12, doi: 10.1016/j.comcom.2023.02.001.

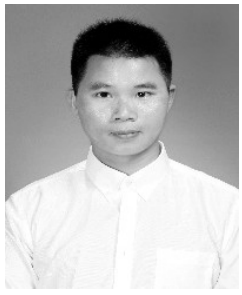
[27] Lillicrap, T. P.—Hunt, J. J.—Pritzel, A.—Heess, N.—Erez, T.—Tassa, Y.—Silver, D.—Wierstra, D.: Continuous Control with Deep Reinforcement Learning. CoRR, 2019, doi: 10.48550/arXiv.1509.02971.

[28] Schaul, T.—Quan, J.—Antonoglou, I.—Silver, D.: Prioritized Experience Replay. CoRR, 2015, doi: 10.48550/arXiv.1511.05952.

[29] Li, J.—Gao, H.—Lv, T.—Lu, Y.: Deep Reinforcement Learning Based Computation Offloading and Resource Allocation for MEC. 2018 IEEE Wireless Communications and Networking Conference (WCNC), 2018, pp. 1–6, doi: 10.1109/WCNC.2018.8377343.

[30] Tong, Z.—Deng, X.—Ye, F.—Basodi, S.—Xiao, X.—Pan, Y.: Adaptive Computation Offloading and Resource Allocation Strategy in a Mobile Edge Computing Environment. Information Sciences, Vol. 537, 2020, pp. 116–131, doi: 10.1016/j.ins.2020.05.057.

An Li received his Master's degree from the Guangxi University. Now, he is a Lecturer in the Center for Applied Mathematics of Guangxi at the Yulin Normal University. He mainly focuses on network energy consumption optimization and network space security.



Yeqiang Zheng received his Master's degree from the Guangxi University. Now, he is a Senior Engineer in the Center for Applied Mathematics of Guangxi at the Yulin Normal University. He mainly focuses on mobile edge computing and network optimization.

Wang Nong is currently pursuing the Master's degree in the School of Computer and Electronics Information at the Guangxi University, Nanning, China. His research interests include mobile edge computing and wireless network.

Manyi Wei is currently pursuing her Master's degree in the School of Computer and Electronics Information at the Guangxi University, Nanning, China. Her research interests include mobile edge computing and UAVs technology.

Gaocai Wang is Professor at the School of Computer, Electronics and Information, Guangxi University. He received his Ph.D. in computer application technology from the Central South University, China, in 2004. Currently, he is the supervisor of Master's degree candidates in computer science and technology and doctoral supervisor of electrical engineering in the Guangxi University. His research interests include computer network technology, wireless network technology, algorithm optimization.

Shuqiang Huang received his Ph.D. degree from the South China University of Technology, Guangzhou, China, in 2010. Currently, he is Professor at the Jinan University. He has published more than 60 academic papers. His main research interests include communication technology and modeling, optical networks, wireless networks, artificial intelligence, and machine learning.