IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received October 15, 2020, accepted October 30, 2020, date of publication November 6, 2020, date of current version November 18, 2020.

Digital Object Identifier 10.1109/ACCESS.2020.3036416

Optimization of Task Offloading Strategy for Mobile Edge Computing Based on Multi-Agent Deep Reinforcement Learning

HAIFENG LU, CHUNHUA GU, FEI LUO, WEICHAO DING, SHUAI ZHENG, AND YIFAN SHEN

School of Information Science and Engineering, East China University of Science and Technology, Shanghai 200237, China

Corresponding author: Chunhua Gu (chgu@ecust.edu.cn)

This work was supported in part by the National Natural Science Foundation of China under Grant 61472139, in part by the Shanghai Automobile Industry Science and Technology Development Foundation under Grant H100-2-19160, and in part by the Shanghai Sailing Program under Grant 20YF1410900.

ABSTRACT Combined with wireless power transfer (WPT) technology, mobile edge computing can provide continuous energy supply and computing resources for mobile devices, and improve their battery life and business application scenarios. This article first designs the mobile edge computing (MEC) model of mobile devices with random mobility and hybrid access point (HAP) with data transmission and energy transmission. On this basis, the selection of target server and the amount of data offloading are taken as the learning objectives, and the task offloading strategy based on multi-agent deep reinforcement learning is constructed. Then combined with MADDPG algorithm and SAC algorithm, the problems of multi-agent environment instability and the difficulty of convergence are solved. The final experimental results show that the improved algorithm based on MADDPG and SAC has good stability and convergence. Compared with other algorithms, it has achieved good results in energy consumption, delay and task failure rate.

INDEX TERMS Mobile edge computing, task offloading, wireless power transfer, multi-agent, deep reinforcement learning.

I. INTRODUCTION

With the rapid development and widespread popularity of the Internet of Things (IoT) technology, cloud computing has been unable to meet the demand in business scenarios where the amount of data collection is too large, immediate and continuity interaction is required, such as online games, real-time streaming media, and augmented reality [1]. To solve above problems, the MEC builds an open platform for data collection, data processing and data analyzing at the edge of the network, so that mobile devices can actively offload computing tasks to edge servers, thereby reducing service response time, improving device battery life, ensuring data security and user privacy [2]. In addition, with the large-scale deployment of 5G and the continuous development of mobile communication system, most energy-consuming applications, including video streaming services, AR/VR transmissions, etc., are now running on battery-powered mobile devices, which leads to huge energy consumption and interruption of user services. Therefore, in order to meet the continuous service needs of mobile devices, the WPT technology realizes wireless charging of mobile devices and IoT devices by using the principle that radio frequency (RF) signals can transmit energy in the far field. It is a flexible, controllable, on-demand and low-cost solution, which has the characteristics of stable power supply, short response time, simple installation and environment-friendly [3]. In order to satisfy the information download request and wireless charging request of mobile devices, HAP can realize wireless information transmission (WIT) and WPT in the same frequency spectrum based on the broadcast characteristics of RF signal and wireless channel. The key difference between HAP and traditional access point (AP) is that the former enables WPT and WIT services at the same time, which fundamentally solves the problem of low computing ability and short battery life of mobile devices, makes the business scenarios more diversified [4].

Although edge servers can relieve computing pressure and battery life pressure for mobile devices through HAP,

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

since WIT and WPT are performed in the same frequency spectrum, only one operation can be completed at the same time. If all the mobile tasks are offloaded to the edge server for processing, the amount of data transmission will be too large, the service time of WIT is too long and the service time of WPT is too short, which will eventually cause the mobile device to run out of power and interrupt user services. At the same time, if a large number of mobile tasks are collectively offloaded to a small number of edge servers for processing, considering the limited computing performance and network bandwidth of the edge servers, it will cause serious congestion of computing tasks and significant delays in user services [5]. Therefore, in order to make full use of MEC's computing resources and ensure that mobile devices can provide continuous services, it needs to design a strategy which can determine the amount of offloaded data and the target server for mobile tasks, so that the edge server and mobile devices can perform collaborative processing to effectively improve the user's service quality. In this article, we propose a strategy that uses multi-agent deep reinforcement learning to solve the problem of how much and where to offload the tasks in the MEC by comprehensively considering the computing performance, the signal range, the geographic location of the edge server, and the computing performance, remaining capacity of battery, energy transmission, location information, application data amount of the mobile device. And it effectively reduces the energy consumption, delay and task failure rate of mobile devices and edge servers, and improves the service quality of the entire MEC platform. The main contributions of this article include three aspects:

1) The MEC model of mobile devices with random mobility and HAP nodes with data transmission and energy transmission is constructed. Since the HAP node can only perform WIP or WPT at the same time, it is necessary to reasonably design the running time of both in unit time to ensure the quality of user service. In this article, the time required for data transmission is calculated by considering the amount of data offloading and the transmission rate based on the location of mobile devices. At the same time, the remaining time is used as the energy transmission time to charge the device, so as to ensure that the total power of the device can complete the computing task and transmission task in unit time.
2) Combined with the actual application scenario of MEC, the target server selection and data amount to be offloaded are taken as the learning objectives, and the task offloading strategy based on multi-agent deep reinforcement learning is constructed. This article combines the MADDPG algorithm and the SAC algorithm to solve the problem of instability and convergence difficulty in the multi-agent environment. Among them, the MADDPG algorithm optimizes the strategy of each agent through the idea of centralized training and distributed execution to reduce the variance of the algorithm; the SAC algorithm introduces maximum entropy into the reward function to ensure that it can explore more action possibilities, and enhance its exploration ability and robustness. At the same time, the MADDPG algorithm and the SAC algorithm are used to solve the continuous action space problem. Considering that the target server selection is a discrete problem, this article uses the reparameterization trick of Gumbel Softmax to solve discrete problems without losing gradient information.
3) The energy consumption, cost, delay and task failure rate of task offloading strategies are comprehensively compared to analyze their advantages and disadvantages. Experiments show that the improved algorithm based on MADDPG and SAC has good stability and convergence. Compared with other algorithms, it achieves good results in energy consumption, delay and task failure rate when the number of mobile devices is large.

The remainder of this article is organized as follows. In Section 2, the scope of related works is discussed. The MEC model and evaluation metrics such as energy consumption, delay, cost and task failure rate are described in Section 3. The proposed task offloading algorithm is presented in Section 4. The experimental setup and performance evaluation are described in Section 5. Finally, Section 6 concludes the paper and gives directions for future work.

II. RELATED WORK

With the popularization of 5G technology, the task offloading problem of MEC has received extensive attention, and there are a lot of researches in recent years. For instance, the reference [6] proposed an Orthogonal Frequency-Division Multiplexing Access (OFDMA) based multi-user and multi-MEC-server system, which is used to investigate the task offloading strategies and wireless resources allocation for latency-critical applications. The reference [7] mathematically modeled the MEC architecture, it optimized the MEC calculation offload strategy to decide when to offload the user 's computing tasks to the MEC server for processing, and verified the effectiveness of the strategy through the face recognition application by measuring the round-trip time. Compared with the local execution of mobile devices, the strategy greatly reducing the service delay and saving the energy consumption of the device. The reference [8] studied the multi-user service delay problem in the MEC offloading scenario and proposed a new type of partial computing offloading model, which optimizes the allocation of communication and computing resources through strategies such as optimal data segmentation. Compared with local execution of devices and edge cloud execution, the proposed partial offloading strategy can minimize the delay of all devices, thereby improving the user's service experience quality. In the above research, when heuristic algorithms are used to deal with large-scale task offloading problems, algorithms take too long to generate decision-making due to the high dimension of the problem. At the same time, such algorithms can only

find approximately optimal solutions. Therefore, it cannot meet the expected requirements in actual use. In addition, reinforcement learning is also widely used in the problem of offloading MEC tasks. The reference [9] proposed that mobile tasks can select multiple base stations in the MEC for offloading according to demand. And on this basis, it studied the task offloading strategy based on a deep reinforcement learning algorithm to maximize the long-term performance. The reference [10] used RL-based resource management algorithms to optimize the data processing volume of cloud servers and edge servers. And the strategy can reduce service delay and operating costs. The reference [11] chose whether to offload tasks to the edge server which serves for multiple users by deep reinforcement learning algorithm, so as to reduce energy consumption and average computing delay. At the same time, it further optimized the selection strategy of the edge server by adding wireless transmission rate, charging power and battery power to the learning process, so that the entire computing cluster provides better service performance.

Based on the above researches, compared with heuristic algorithms, deep reinforcement learning has the characteristics of self-learning and self-adaption by combining the advantages of deep learning and reinforcement learning. It needs fewer parameters and has better global search capabilities, which can solve the more complex, high-dimensional and more realistic task [12]. However, the researchers of current MEC task offloading, which based on deep reinforcement learning, are all about single agents. They mainly focus on one learning target such as the amount of data offloading or the selection of edge servers. The amount of data offloading only pays attention to the amount of data to be processed by the local device and the remote server cluster respectively, but the impacts of performance and location between different devices are ignored. The selection problem of the target server only focuses on whether the data is offloaded and where it is offloaded, which ignores the fact that the application data can be split for collaborative computing, to save computing time and transmission time. Therefore, this article takes the target server selection and the data amount to be offloaded as comprehensive learning goals with the multi-agent deep reinforcement learning algorithm, so as to improve the resource utilization rate of the entire MEC platform [13]. In addition, in order to ensure that the multi-agent algorithm has better convergence and generalization performance in the MEC partial offloading problem, this article combines the advantages of the SAC algorithm and the MADDPG algorithm to form a partial offloading strategy:

1) In order to solve the problem that deep reinforcement learning algorithm is influenced by super parameters and easy to fall into local optimal solution, the SAC algorithm adds the maximum entropy to the reward function, so that the actor can explore as many actions as possible on the premise of completing the task, to achieve approximate optimal multiple trajectory selection. Therefore, the SAC algorithm is conducive to learning new tasks and as the initialization of more complex tasks. At the same time, the algorithm has stronger exploration ability and robustness, and can solve the problem of unstable convergence.
2) In multi-agent environment, since the strategy of each agent is constantly learning and changing, which causes an unstable environment for a single agent, so the variance value of traditional deep reinforcement learning algorithm will become larger with the increase of the number of agents. Therefore, the MADDPG algorithm solves the problem of instability in the multi-agent environment by centralized training and distributed execution. When the MADDPG algorithm is updated, the overall optimization can be performed according to the training strategy of each agent, so as to improve the stability and robustness of the algorithm. In addition, the MADDPG algorithm allows each agent to design its own reward function, which can be used to solve the problem of cooperation or confrontation.

III. SYSTEM MODEL

A. MEC MODEL

In order to marginalize and localize computing resources and cache resources, edge servers are usually deployed at HAPs to ensure that mobile devices can obtain WIT and WPT services [14]. As shown in Figure 1, the entire MEC system is composed of HAP nodes, edge servers and mobile devices. Each HAP has a certain signal coverage area, and mobile devices in this range can offload tasks to an edge server for calculation and get a certain amount of power supplement. However, as the location of mobile devices changes, the connection between the mobile devices and the edge server will become extremely unstable due to the long relative distance, which will further exceed the signal range and cause service interruption.

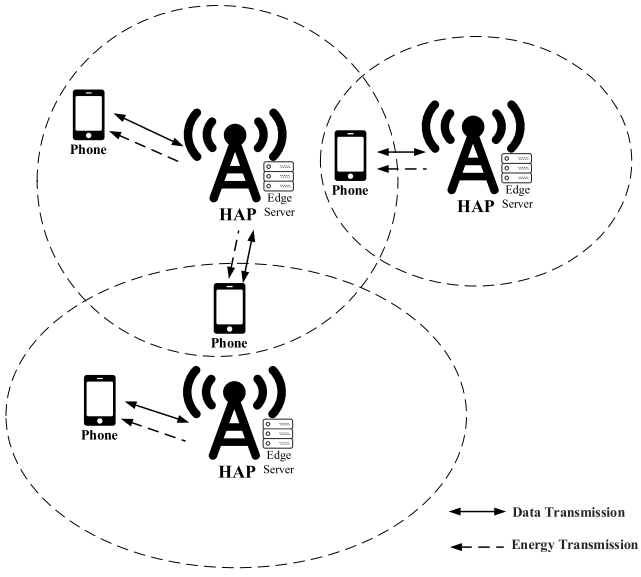

FIGURE 1. MEC structure based on HAP.

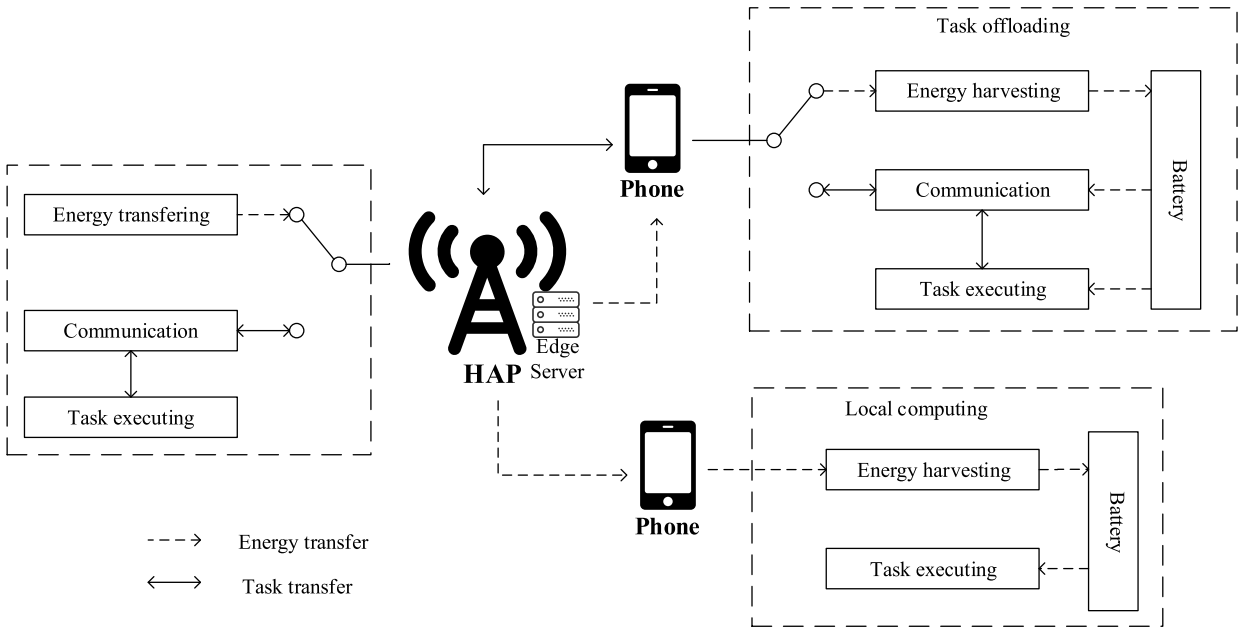

FIGURE 2. Resource transfer of task offloading and local computing.

In the entire MEC system, the HAP node has the functions of energy transmission and data transmission. It gives priority to energy transmission per unit time to ensure that the mobile device has a certain processing capability, and then provides data upload and download services to realize task offloading of the mobile device. As shown in Figure 2, the mobile device can decide whether to perform the task offloading operation according to the strategy. If this operation is performed, the mobile device first converts the HAP radio frequency signal into electrical energy and stores it in the battery at each time step. Then the battery provides energy for data transmission and task calculation. In this operation, one part of data is transmitted from the mobile device to the edge server for remote calculation and the edge server returns the result, another part is calculated directly on the mobile device, which make full use of the computing resources of edge servers and mobile devices. But it mainly needs to consider the delay caused by data transmission and the energy consumption generated by computing and communication of mobile devices. If task offloading is not performed, the mobile device will store the converted electrical energy in the battery and directly provide energy for local computing. Local computing only uses the computing resources of the mobile device, while the delay and energy consumption of its computing must be considered [15].

B. PROBLEM MODEL

In this article, all computing devices in MEC are represented by (ES, MD), where ES represents a set of m edge servers $\{es_1, es_2, \ldots, es_i, \ldots, es_m\}$, and the signal range SR_i and processing performance EC_i of each edge server are set separately; MD represents a set of n mobile devices $\{md_1, md_2, \ldots, md_j, \ldots, md_n\}$, the processing performance of each mobile device is set to MC_j, the corresponding application is denoted by $\{data_j, L_j^{max}\}$, $data_j$ and L_j^{max} refer to the total amount of data to be processed by the application and the maximum allowable completion time, where $data_j$ is directly proportional to the complexity of the application. Suppose that the data partition of each application is full granularity, that is to say, the application can be divided into subprograms of any size. It is assumed that the operation process of all computing devices will be in accordance with the set of time steps $T = \{1, 2, \ldots, t, \ldots\}$, and the amount of data to be processed by the application in each time step is C. The MEC needs to determine the target server TS_t and the amount of offloading data $\lambda_t C$ based on the location of each mobile device, energy supply power, remaining power, remaining amount of application data, and connectable edge servers in the current time step t [16]. At the same time, the remaining data $(1 - \lambda_t)C$ needs to consume power while processing on the mobile device, which cannot exceed the remaining power of the device, otherwise the application processing fails. The delay, energy consumption, cost, and charging capacity of mobile devices and edge servers in the current time step t are modeled as:

1) DELAY MODEL

Assuming that the CPU frequencies of the mobile device and the target server are η_t^{local} and $\eta_t^{offload}$, respectively, and the data amount of local processing and remote processing are $(1-\lambda_t)C$ and $\lambda_t C$ in the unit time, respectively, the time required for mobile device j to process data is:

$$H_{j,t}^{local} = \frac{(1 - \lambda_t)C}{\eta_t^{local}}. \quad (1)$$

The delay caused by data offloading mainly includes data uploading delay and data processing delay. The data uploading delay is determined by the transmission rate TR between the edge server and the mobile device. Its calculation formula at time step t is:

$$TR_t = w \log_2(1 + \frac{p_{tran} d_t^{-\theta} |h_t|^2}{N}). \tag{2}$$

where w is the upload bandwidth, p_{tran} is the transmission power of the mobile device, d_t is the distance between the edge server and the mobile device, $\theta \geq 2$ is the path loss, h_t is the channel attenuation coefficient, and N is the Gaussian distribution of noise. Therefore, the calculation formula for uploading delay is:

$$\tau_t^{up} = \frac{\lambda_t C}{TR_t}. \tag{3}$$

The data processing delay formula of the edge server is:

$$\tau_t^{exec} = \frac{\lambda_t C}{\eta_t^{offload}}. \tag{4}$$

The time required for mobile device j to offload data is:

$$H_{j,t}^{offload} = \tau_t^{up} + \tau_t^{exec}. \tag{5}$$

Since data processing and offloading are performed simultaneously, the time required by mobile device j to process unit data in time step t is:

$$H_{j,t} = \max\{H_{j,t}^{local}, H_{j,t}^{offload}\}. \tag{6}$$

2) ENERGY CONSUMPTION MODEL

Because the mobility of mobile devices is not convenient for replenishing power in time, and the edge server is deployed near the base station to facilitate power supply management, this article takes the energy consumption of mobile devices as the main research object, which is composed of computing energy consumption and transmission energy consumption. The calculated energy consumption of mobile device j at time step t is:

$$E_{j,t}^{exec} = k(\eta_t^{offload})^3. \tag{7}$$

where k is the energy consumption coefficient based on the type of CPU. The data transmission energy consumption of mobile device j at time step t is:

$$E_{j,t}^{tran} = (p_0 + \alpha p_{tran})\tau_t^{up}. \tag{8}$$

where p_0 is the fixed energy consumption of the mobile device for communication, α is the signal power amplifier coefficient, p_{tran} is the transmission power consumption of the mobile device, and τ_t^{up} is the data transmission time. Therefore, the total offload energy consumption of mobile device j at time step t is:

$$E_{j,t}^{offload} = E_{j,t}^{exec} + E_{j,t}^{tran}. \tag{9}$$

3) COST MODEL

Users need to pay the corresponding fees to obtain the computing resources that provided by the edge server. In this article, a dynamic price model based on the remaining amount of computing resources is used. When the remaining amount of resources is less, the resource price is higher. At this time, users tend to choose service nodes with lower prices as an offloading server, thereby reducing user expenses while increasing resource utilization. At the same time, because computing resources of the mobile device belong to the user, which does not need to pay the operator for computing, the cost model only needs to calculate the used resource by the edge server, and the dynamic price model of the i-th edge server based on the remaining amount of computing resources is:

$$\mathrm{Cos}\, t_{i,t} = \gamma \cdot \tau_{i,t}^{exec} \cdot pc_i. \tag{10}$$

where $\tau_{i,t}^{exec}$ is the computing time of the edge server, pc_i is the unit price of the computing resources, and γ is the ratio of computing resources currently occupied by the edge server.

4) ENERGY SUPPLY MODEL

Mobile devices in MEC can use RF-DC converter to convert RF signal into electric energy and store it in battery. The electrical energy collected per unit time is inversely proportional to the relative distance between the mobile device and the edge server. The energy conversion calculation formula of mobile device j is as follows:

$$E_{j,t}^{harvest} = \upsilon p_{tran} d_t^{-\theta} G. \tag{11}$$

where $\upsilon \in (0, 1)$ is the energy conversion efficiency, p_{tran} is the transmission power consumption of the mobile device, d_t is the distance between the edge server and the mobile device, $\theta \geq 2$ is the path loss, and G is the integrated channel gain between the edge server and the mobile device.

IV. ALGORITHM DESIGN

Reinforcement learning is a sequential decision-making method that continuously conducts trial-and-error learning in the target environment and modifies strategies through feedback results to maximize rewards. Although it has many advantages, it also lacks scalability and is essentially limited to fairly low-dimensional problems. This is mainly because reinforcement learning algorithms have the same memory complexity, computational complexity, and sample complexity as other algorithms. Therefore, in order to solve the high-dimensional decision-making problem that reinforcement learning is difficult to deal with, deep reinforcement learning combines the perceptual ability of deep learning with the decision-making ability of reinforcement learning, and solves the problem with high-dimensional state space and action space by strong function approximation and deep neural network. In this article, the MDP model based on the MEC environment is built, which combines with the multi-agent deep reinforcement learning algorithm, to solve the problem

of edge server decision-making and data offloading decision-making [10].

A. MDP MODEL

1) STATE SPACE

In order to comprehensively consider the characteristics of mobile tasks and edge server resources in MEC, this article defines the state space of the j-th mobile device at time step t as $S_t^j = (TR_{1,t}, \ldots, TR_{I,t}, \ldots, TR_{m,t}, U_{1,t}, \ldots, U_{I,t}, \ldots, U_{m,t}, RD_{j,t}, RB_{j,t}, HB_{j,t})$, where $TR_{i,t}$ represents the transmission rate between the mobile device and the i-th edge server, $U_{i,t}$ represents the CPU utilization of the i-th edge server, $RD_{j,t}$ represents the amount of remaining data that the mobile device needs to process, $RB_{j,t}$ represents the remaining power of the mobile device, and $HB_{j,t}$ represents the power generated by the mobile device using RF-DC conversion.

2) ACTION SPACE

In order to offload part of the mobile task to the target edge server for collaborative computing, this article designs two agents with the same state space but different action spaces to determine the target server and the data amount to be offloaded. In the decision-making problem of target server, the action space is defined to be corresponding to the set of edge servers, so the discrete action space is $A^{edge} = (es_1, es_2, \ldots, es_m)$, using $(0/1)_i^j$ to indicate whether the task of the j-th mobile device is offloaded to the i-th edge server. For example, action space $A_j^{edge} = (0, 0, 1, \ldots, 0)$ indicates that the target server of the j-th mobile device is the 3rd edge server; For the problem of offloading data amount, the continuous action space $A^{percent} = \lambda_t$ is the offloading percentage of the data amount C in the time step t, and the precision is kept to two decimal places.

3) REWARD FUNCTION

The multi-agent deep reinforcement learning based on the target server and the offloading amount of data is a completely cooperative game problem. The objective of both agents is to reduce the delay, energy consumption, cost, and task failure rate as much as possible. Therefore, the reward function of the agent is consistent, and its calculation formula is:

$$\begin{cases} F = -\sigma \sum_{j=1}^{n} H_{j,t} - \xi \sum_{j=1}^{n} E_{j,t}^{offload} - \delta \sum_{i=1}^{m} Cost_{i,t} \\ \qquad - \omega \sum_{j=1}^{n} I(RB_{j,t} = 0), \\ \sigma + \xi + \delta = 1. \end{cases} \tag{12}$$

where $\sum_{j=1}^{n} H_{j,t}$ and $\sum_{j=1}^{n} E_{j,t}^{offload}$ represent the total delay and total energy consumption of all mobile devices; $\sum_{i=1}^{m} Cost_{i,t}$ represents the total cost of all edge servers; The σ, ξ, δ represent the weights of the above three indicators; $I(RB_{j,t} = 0)$ represents that the remaining power of mobile device j is empty when the value is 1, otherwise it is 0. This value is used to measure whether the task is processed successfully; ω represents the penalty value for the task processing failure.

B. METHODOLOGY

1) SAC

The SAC is an improved actor-critic algorithm based on maximized entropy reinforcement learning, which maximizes the entropy to enable the actor to explore action possibilities as many as possible under the premise of completing the task, so as to achieve several approximately optimal trajectory choices [17]. Therefore, the SAC algorithm has stronger exploration ability and robustness, and is not easy to fall into the local optimal solution. Its optimal strategy calculation formula is:

$$\begin{cases} \pi^* = \arg\max_{\pi} \sum_{t=0}^{T} E_{(s_t,a_t)\sim\rho_\pi}[\gamma^t (r(s_t, a_t) + \alpha H(\pi(\cdot|s_t)))], \\ H(\pi(\cdot|s_t)) = -\log \pi(\cdot|s_t). \end{cases} \tag{13}$$

where π^* represents the optimal decision, T represents the time series, ρ_π represents the trajectory distribution probability under the decision π, $\gamma \in [0, 1]$ represents the discount coefficient, $r : S \times A \to R$ represents the reward function, $s_t \in S$ represents the environmental state at time step t, $a_t \in A$ represents the action taken at time step t, $\alpha > 0$ is a weighting coefficient used to control the entropy, it is more inclined to explore when the value is larger, and $H(\pi(\cdot|s_t))$ represents the entropy of the strategy π in the state s_t.

In order to solve the high-dimensional continuous control problem, the SAC algorithm approximately calculates the state value function $V_\varphi(s_t)$, soft Q function $Q_\theta(s_t, a_t)$ and strategy function $\pi_\phi(a_t|s_t)$ by a neural network. Besides, it updates each parameter alternately by stochastic gradient descent (SGD), where the strategy function follows the Gaussian distribution, and its mean vector and covariance matrix are all obtained by neural network fitting. The objective function of the state value function $V_\varphi(s_t)$ is:

$$J_V(\varphi) = E_{s_t:D}[\frac{1}{2}(V_\varphi(s_t) - E_{a_t:\pi_f}[Q_\theta(s_t, a_t) - \log \pi_f(a_t|s_t)])^2]. \tag{14}$$

where D represents the replay buffer of past experience, φ, θ, ϕ represent the parameters of each neural network.

The update method of soft Q function is similar to other Q-learning algorithms, and it updates Bellman residuals. The difference is that its value function contains entropy, and its objective function is:

$$\begin{cases} J_Q(\theta) = E_{(s_t,a_t,s_{t+1}):D}[\frac{1}{2}(Q_\theta(s_t, a_t) - Q_{t\arg et}(s_t, a_t))^2], \\ Q_{t\arg et}(s_t, a_t) = r(s_t, a_t) + \gamma(Q_{\bar{\theta}}(s_{t+1}, a_{t+1}) \\ \qquad\qquad - \alpha \log(\pi_f(a_{t+1}|s_{t+1}))). \end{cases} \tag{15}$$

where $\bar{\theta}$ represents the parameter of the target Q network, and the parameter θ of the Q network will be updated

every a certain time, so as to calculate the loss function by the differences between the two Q network parameters, which improves the stability and convergence of training.

The strategy function is updated through the soft Q function. It expects that the action probability distribution of the strategy in the state s_t can confirm to the distribution of Q value, so it updates the strategy network parameters by minimizing the KL divergence:

$$\begin{cases} J_\pi(\phi) = D_{KL}(\pi_\phi(\cdot|s_t)||\exp(\frac{1}{\alpha}Q_\theta(s_t,\cdot) - \log Z(s_t))) \\ = E_{s_t\sim D, a_t\sim\pi_\phi}[\log \pi_\phi(a_t|s_t) - \frac{1}{\alpha}Q_\theta(s_t,a_t) + \log Z(s_t)], \\ a_t = f_\phi(\varepsilon_t; s_t) = f_\phi^\mu(s_t) + \varepsilon_t \odot f_\phi^\sigma(s_t). \end{cases} \tag{16}$$

where $Z(s_t)$ is the sum of the all action Q values' expectation with the current strategy in the state s_t, the function f is used to calculate the average and variance of the Gaussian distribution, and ε is the noise of Gaussian sampling. Therefore, the strategy function update formula and gradient formula based on action sampling are:

$$\begin{cases} J_\pi(\phi) = E_{s_t\sim D,\varepsilon\sim N}[\log \pi_\phi(f_\phi(\varepsilon_t;s_t)|s_t) - Q_\theta(s_t, f_\phi(\varepsilon_t;s_t))], \\ \nabla_\phi J_\pi(\phi) = \nabla_\phi \log \pi_\phi(a_t|s_t) + (\nabla_{a_t}\log \pi_\phi(a_t|s_t) \\ \qquad - \nabla_{a_t} Q(s_t,a_t))\nabla_\phi f_\phi(\varepsilon_t; s_t). \end{cases} \tag{17}$$

2) MADDPG

Traditional reinforcement learning is difficult to apply directly to a multi-agent environment. The main reason is that each agent is constantly learning and improving strategies. Therefore, changes in the strategies of the other agents cause the instability of the dynamic environment for a single agent, then the agent's own state transition probability will be different in different situations, there is $P(s'|s,a,\pi_1,\ldots,\pi_n) \neq P(s'|s,a,\pi_1',\ldots,\pi_n')$ for any $\pi_i \neq \pi_i'$. Thereby, the multi-agent reinforcement learning cannot directly use the experience replay method for training [18]. At the same time, the complexity of the environment will increase with the increasing number of agents, and the optimization method of estimating the gradient by sampling will also cause great variance, so the strategy gradient algorithm cannot be trained in a multi-agent environment. In view of the above problems, it is mainly because there is no interaction between the various agents, which leads to the neglect of the whole. Therefore, this article mainly solves the problems of target server selection and the data amount of task offloading by the MADDPG algorithm.

The MADDPG algorithm is an extension of the DDPG algorithm. It solves the problem of perception among multiple agents by centralized training and decentralized execution. The Actor of the DDPG algorithm will choose action a_t according to the current state s_t during training, then the Critic utilizes the state action function to calculate the Q value as feedback to the action taken by the Actor, and then it calculates the difference between the estimated Q value and the actual Q value to update the network parameters, and the Actor improves strategies based on the Critic 's feedback. In addition, the Critic of the MADDPG algorithm can obtain the state and action of other agents during training to calculate a more accurate Q value. That is, each agent not only based on its own state but also based on the behavior of other agents to evaluate the value of current actions for achieving centralized training; At the same time, after the training, the Actor of each agent only needs to take appropriate actions according to its state rather than obtain the information of other agents to assist calculation, so as to achieve decentralized execution [19].

This article assumes that $\phi = (\phi_1,\ldots,\phi_k)$ represents the strategy parameters of k agents, $\pi = (\pi_1,\ldots,\pi_k)$ is the corresponding strategy, and the strategy gradient formula of the i-th agent is:

$$\nabla_{\phi_i} J(\phi_i) = E_{s\sim\rho_\pi, a_i\sim\pi_i}[\nabla_{\phi_i}\log \pi_i(a_i|s_i)\cdot Q_i^\pi(x,a_1,\ldots,a_k)]. \tag{18}$$

where s_i represents the observation value of the i-th agent, $x = (s_1,\ldots,s_k)$ represents the state vector containing the observation values of all agents, and $Q_i^\pi(x,a_1,\ldots,a_k)$ represents the Q value evaluated by centralized Critic for the i-th agent. Because each agent learns a different Q_i^π function, it can have different reward values to complete cooperation or competition tasks. For the agent's deterministic strategy μ_{ϕ_i} (abbreviated as μ_i), the gradient formula is:

$$\nabla_{\phi_i} J(\mu_i) = E_{x,a\sim D}[\nabla_{\phi_i}\mu_i(a_i|s_i) \cdot\nabla_{a_i} Q_i^\mu(x,a_1,\ldots,a_k)|_{a_i=\mu_i(s_i)}]. \tag{19}$$

The element composition of the experience replay buffer D is $(x,a_1,\ldots,a_k,r_1,\ldots,r_k,x')$, which records the observation values, actions and rewards at the current moment and observation values at the next moment of all agents. The update formula of the centralized Critic's action value function Q_i^μ is:

$$\begin{cases} L(\phi_i) = E_{x,a,r,x'}[(Q_i^u(x,a_1,\ldots,a_k) - y)^2], \\ y = r_i + \gamma Q_i^{\bar{\mu}}(x',a_1',\ldots,a_k')|_{a_j'=\bar{\mu}_j(s_j)} \end{cases} \tag{20}$$

where $Q_i^{\bar{u}}$ represents the target network and $\bar{\mu} = (\mu_{\bar{\phi}_1},\ldots,\mu_{\bar{\phi}_k})$ is the set of target strategies with delay parameters $\bar{\phi}_i$. At the same time, since the MADDPG algorithm can only solve the problem of continuous action space, and the problem of edge server selection is a discrete problem, this article utilizes re-parameterization of Gumbel Softmax to perform category sampling without losing gradient information, so as to realize the mapping relationship between continuous actions and discrete actions. The calculation formula is:

$$x = soft\max(\frac{\log p_i - \log(-\log \varepsilon_i)}{\tau})_{i=1}^k,\ \varepsilon_i \sim U[0,1]. \tag{21}$$

where p represents the probability vector of k-dimensional, and the parameter $\tau > 0$ is used to control the smoothness of the softmax function. The larger the value is, the smoother the

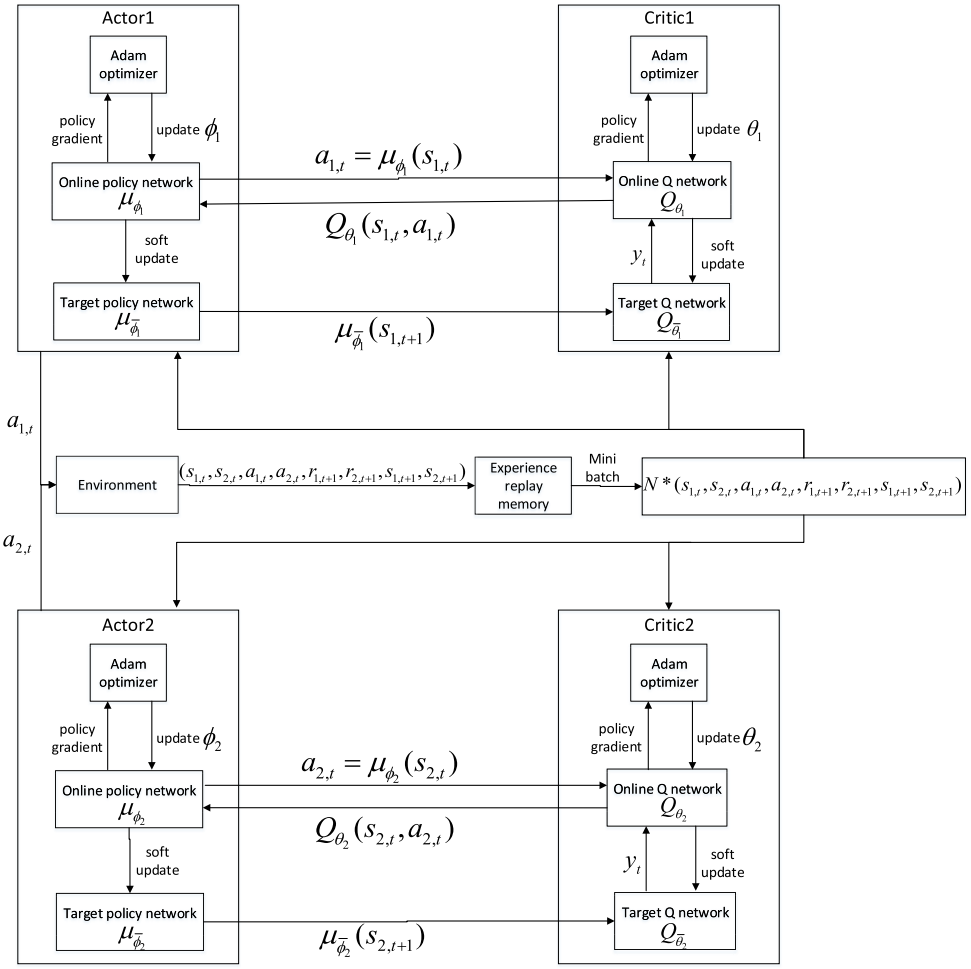

FIGURE 3. Multi-agent algorithm flow chart.

distribution generates, and the smaller the value is, the closer the distribution is to the discrete one-hot distribution [20]. Therefore, it can obtain a discrete distribution that is closer to the reality in training by reducing τ gradually.

Although the Actor in the MADDPG algorithm uses a random strategy to ensure sufficient exploration, but the Critic's deterministic strategy only considers one optimal action for a state and cannot explore all possible optimal actions. Therefore, the algorithm is easy to fall into the local optimal solution in this case. In order to solve this problem, this article combines the MADDPG algorithm and the SAC algorithm to enable it to explore optimal paths as many as possible in a multi-agent environment, thereby enhancing the robustness and generalization of the algorithm. The improved algorithm flow is as follows:

According to the above algorithm flow, Figure 3 is the flow chart of the improved algorithm in two agents.

V. EXPERIMENT

A. SIMULATION ENVIRONMENT

This article builds the task offloading model of MEC by comprehensively considering the computing performance, signal range and geographic location of the edge server; the computing performance, remaining power, charging power, location information of the mobile device, and the data amount of different application services. The initial location information of the edge server and mobile device is simulated based on the Melbourne CBD area in the EUA data set, and the location of mobile devices changes with time following the Truncated Levy Walk mobility model to ensure that it moves in the area covered by the signal [21]. The signal coverage radius of the HAP is randomly distributed between [100,400], the uploading bandwidth of the mobile device $w = 10$MHz, the fixed communication power $p_0 = 0.4$W, the data transmission power $p_{tran} = 0.1$W, the signal power amplifier coefficient $\alpha = 40$, the energy conversion efficiency $\upsilon = 0.8$, the integrated channel gain $G = 20$, the initial power is 4000mah, and it is assumed that each mobile device can only send one application request at the same time [22]–[25]. In order to consider the computing performance and power consumption of different edge servers and mobile devices, this article refers to Standard Performance Evaluation Corporation (SPEC) to set the device configuration and average performance power consumption ratio. A larger value indicates that the device consumes less energy at the same performance and the energy consumption coefficient k is

Algorithm 1 MADDPG+SAC

Input: reward function r

Output: Online policy parameter ϕ_i of each actor after training

1. initialize each Actor's online policy parameter ϕ_i, target policy parameter $\bar{\phi}_i$, online Q function parameter θ of centralized Critic, target Q function parameter $\bar{\theta}$ and memory replay buffer D
2. **for** episode $= 1$ to episode$_{max}$ **do**
3. reset environments and get state $s_{i,1}^{episode}$ of agent i
4. **for** t=1 to T_{done} **do**
5. select action $a_{i,t}^{episode} \sim \pi_{\phi_i}(\cdot|s_{i,t}^{episode})$ of agent i
6. get observation $s_{i,t+1}^{episode}$ and immediate reward $r_{i,t+1}^{episode}$ of agent i after execute action $a_{i,t}^{episode}$ in environment
7. store the transition $(s_t^{episode}, a_t^{episode}, r_{t+1}^{episode}, s_{t+1}^{episode})$ in D
8. **for** each agent i **do**
9. calculate Q_{Target}^{i} by (15)
10. calculate critic loss $L_i(\theta) = \frac{1}{2}(Q_{Target}^{i} - Q_{\theta}^{i}(s_{I,t}^{episode}, a_{1,t}^{episode}, \ldots, a_{N,t}^{episode}))^2$
11. **end for**
12. calculate overall critic loss $L(\theta) = \frac{1}{N}\sum_{i=1}^{N} L_i(\theta)$ and update online Q network parameter θ
13. **for** each agent i **do**
14. update online policy network parameter ϕ_i by calculating the gradient value $\nabla_{\phi} J_{\pi}(\phi)$ according to (17)
15. **end for**
16. update target Q network parameter $\bar{\theta}$ of critic by $\bar{\theta} \leftarrow \tau\theta + (1-\tau)\bar{\theta}, \tau \in [0, 1]$
17. update target policy network parameter $\bar{\phi}_i$ of each agent i by $\bar{\phi}_i \leftarrow \tau\phi_i + (1-\tau)\bar{\phi}_i, \tau \in [0, 1]$
18. **end for**
19. **end for**

smaller [26], [27]. At the same time, models of edge servers and mobile devices follow a uniform distribution respectively, and the detailed information is shown in Table 1.

Due to the different amounts of data calculation and popularity requested by different types of applications, this article sets the application of each mobile device to sample according to its popularity value, and its data amount will also follow a uniform distribution within the setting interval. Detailed settings for different applications are shown in Table 2.

B. RESULT ANALYSIS

In order to ensure that the strategies generated by deep reinforcement learning algorithms are efficient and usable, this article first selects 126 edge servers' location information and a certain number of mobile devices' location information from the EUA data set as the initial starting point of each device. Then, the simulation environment is trained by fixing the movement trajectory of each mobile device and

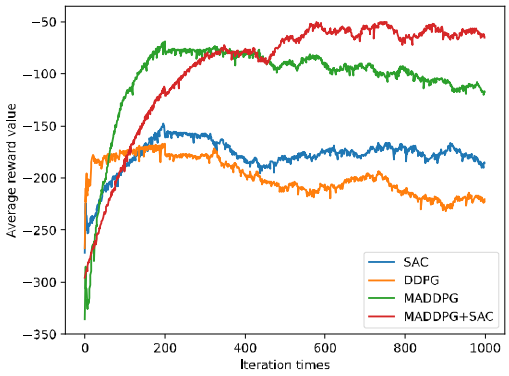

FIGURE 4. The average reward value of each algorithm.

the requested application data. Finally, the random motion path data and application data of each device are used to test the trained decision model, so as to compare the universality and efficiency of each strategy. Figure 4 shows the results of the average reward value of each episode obtained by each deep reinforcement learning algorithm in the training process. The larger the value is, the better the result of the decision model is. It can be seen from the figure that the DDPG algorithm and the SAC algorithm have poor convergence results in a multi-agent environment. Compared with DDPG algorithm, the reward value of SAC algorithm after convergence is higher but the convergence speed is slower. This is mainly because SAC algorithm needs more iterations to explore more decision paths, and it is easier to obtain better solutions. In addition, the MADDPG algorithm and the improved MADDPG + SAC algorithm perform better in a multi-agent environment, and the improved MADDPG + SAC algorithm has a higher reward value after convergence.

Figure 5 is a graph of resource consumption generated by each algorithm during task offloading. The number of mobile devices will increase in this experiment, and the total data amount of all applications will account for 50% -150% of the processing capacity of the entire edge server cluster. Among them, the offloading strategy based on the Mobile algorithm can achieve good results in terms of cost, but performs poorly in terms of energy consumption and task failure rate. This is mainly because Mobile algorithm takes priority in processing application data on local devices, and then gradually offloads to edge servers when resources are insufficient. In this article, mobile devices are only considered for energy consumption and edge servers are only considered for the cost, so this strategy consumes the least cost but consumes the most energy. But at the same time, according to the data in Table 1, the processing capacity of the mobile device is much poorer than that of the edge server, so the processing of application data by the mobile device will have a higher delay and failure rate. In addition, the offloading strategy based on the Edge algorithm performs best in terms of energy consumption, while it performs generally in the rest. The main reason is that the Edge algorithm preferentially offloads subtasks to the edge

TABLE 1. Detailed Configuration Table of Computing Devices.

Model	Type	CPU Frequency/MHz	Cores	Memory/GB	Average Performance to Power Ratio	Unit Price/CNY
RX350 S7	Edge Server	2200	16	24	5035	0.05
DL325 Gen10	Edge Server	2000	32	128	8083	0.03
DL360 Gen10	Edge Server	2500	28	48	11550	0.01
TX120	Mobile Device	2666	2	4	454	0
TX150 S5	Mobile Device	2666	2	4	356	0
TX150 S6	Mobile Device	2400	4	8	667	0

TABLE 2. Detailed Information Table of Mobile Applications.

Application	Popularity	Min amount of data/bit	Max amount of data/bit
translation_language	0.1	3000	40000
face_recognition	0.2	300000	30000000
natural_language_processing	0.4	10000	100000
speech_recognition	0.2	80000	800000
virtual_reality	0.1	100000	3000000

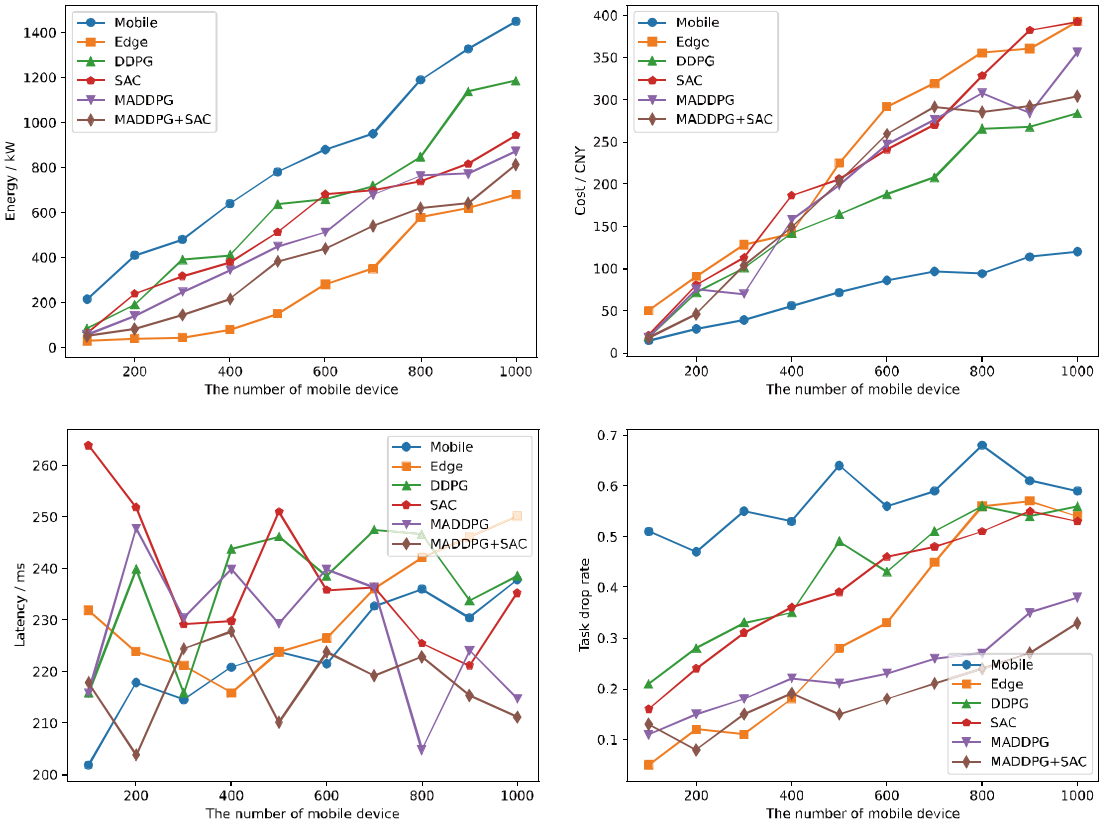

FIGURE 5. Performance of each algorithm in task offloading.

server cluster for processing, which results in the resource utilization of all edge servers can be maintained at a high level and the cost is high, and the corresponding mobile devices consume less energy. Because the processing performance of the edge server can meet the processing requirements of more tasks, so it performs better than the Mobile algorithm in terms of task failure rate.

DDPG algorithm, SAC algorithm, MADDPG algorithm and MADDPG + SAC algorithm all use deep reinforcement learning to automatically generate corresponding offloading strategies from data. As shown in Figure 5, it can be seen that with the growth of the number of mobile devices, the offloading strategy generated by DDPG algorithm performs well in terms of cost, while the performance of SAC algorithm is better than DDPG algorithm in terms of energy consumption and task failure rate. The strategies generated by the above two deep reinforcement learning algorithms perform generally in various indicators, which is mainly because

the training results of the two algorithms are unstable in multi-agent environment, and it is difficult to converge to the optimal solution. In contrast, MADDPG algorithm can effectively learn stable strategies by centralized training and distributed execution, which is better than DDPG algorithm and SAC algorithm in comprehensive performance, and the improved MADDPG + SAC algorithm performs best in all deep reinforcement learning algorithms in terms of energy consumption, delay, and task failure rate when the number of mobile devices is the largest.

VI. CONCLUSION

In order to solve the task offloading problem of mobile devices in large-scale heterogeneous MEC clusters, this article first proposes to use multi-agent deep reinforcement learning to solve the problem of how much and where to offload. Then, according to the EUA data set, the offloading strategies generated by each algorithm are simulated. Finally, the advantages and disadvantages of each algorithm strategy are verified by comparing energy consumption, cost, delay and task failure rate. According to the results of comparing various algorithms, the improved MADDPG + SAC algorithm has good performance in comprehensive results.

In future work, we intend to improve the multi-agent reinforcement learning algorithm by transfer learning, reusing knowledge that comes from previous experience or other agents can learning a more complex MEC task, and it makes the task offloading strategy more practical.

REFERENCES

[1] B. Li, Z. Fei, J. Shen, X. Jiang, and X. Zhong, "Dynamic offloading for energy harvesting mobile edge computing: Architecture, case studies, and future directions," *IEEE Access*, vol. 7, pp. 79877–79886, 2019.

[2] 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, Oct. 2018.

[3] F. Wang, J. Xu, X. Wang, and S. Cui, "Joint offloading and computing optimization in wireless powered mobile-edge computing systems," *IEEE Trans. Wireless Commun.*, vol. 17, no. 3, pp. 1784–1797, Mar. 2018.

[4] Z. Wei, B. Zhao, J. Su, and X. Lu, "Dynamic edge computation offloading for Internet of Things with energy harvesting: A learning method," *IEEE Internet Things J.*, vol. 6, no. 3, pp. 4436–4447, Jun. 2019.

[5] Z. Yang, J. Hou, and M. Shikh-Bahaei, "Resource allocation in full-duplex mobile-edge computing systems with NOMA and energy harvesting," 2018, *arXiv:1807.11846*. [Online]. Available: http://arxiv.org/abs/1807.11846

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

[7] Z. Ding, J. Xu, O. A. Dobre, and H. V. Poor, "Joint power and time allocation for NOMA-MEC offloading," *IEEE Trans. Veh. Technol.*, vol. 68, no. 6, pp. 6207–6211, Jun. 2019.

[8] F. Messaoudi, A. Ksentini, and P. Bertin, "On using edge computing for computation offloading in mobile network," in *Proc. GLOBECOM IEEE Global Commun. Conf.*, Singapore, Dec. 2017, pp. 1–7.

[9] Y. He, Z. Zhang, F. R. Yu, N. Zhao, H. Yin, V. C. M. Leung, and Y. Zhang, "Deep-reinforcement-learning-based optimization for cache-enabled opportunistic interference alignment wireless networks," *IEEE Trans. Veh. Technol.*, vol. 66, no. 11, pp. 10433–10445, Nov. 2017.

[10] M. Min, L. Xiao, Y. Chen, P. Cheng, D. Wu, and W. Zhuang, "Learning-based computation offloading for IoT devices with energy harvesting," *IEEE Trans. Veh. Technol.*, vol. 68, no. 2, pp. 1930–1941, Feb. 2019.

[11] J. Xu, L. Chen, and S. Ren, "Online learning for offloading and autoscaling in energy harvesting mobile edge computing," *IEEE Trans. Cognit. Commun. Netw.*, vol. 3, no. 3, pp. 361–373, Sep. 2017.

[12] J. Li, H. Gao, T. Lv, and Y. Lu, "Deep reinforcement learning based computation offloading and resource allocation for MEC," in *Proc. IEEE Wireless Commun. Netw. Conf. (WCNC)*, Barcelona, Spain, 2018, pp. 1–6.

[13] D. Zeng, S. Pan, Z. Chen, and L. Gu, "An MDP-based wireless energy harvesting decision strategy for mobile device in edge computing," *IEEE Netw.*, vol. 33, no. 6, pp. 109–115, Nov./Dec. 2019.

[14] C. Li, "Energy efficient computation offloading for nonorthogonal multiple access assisted mobile edge computing with energy harvesting devices," *Comput. Netw.*, vol. 164, pp. 1–12, Dec. 2019.

[15] H. Lin, Z. Chen, and L. Wang, "Offloading for edge computing in low power wide area networks with energy harvesting," *IEEE Access*, vol. 7, pp. 78919–78929, 2019.

[16] S. Mao, S. Leng, S. Maharjan, and Y. Zhang, "Energy efficiency and delay tradeoff for wireless powered mobile-edge computing systems with multi-access schemes," *IEEE Trans. Wireless Commun.*, vol. 19, no. 3, pp. 1855–1867, Mar. 2020.

[17] T. Chen, "Soft actor-critic-based continuous control optimization for moving target tracking," in *Proc. Int. Conf. Image Graph.* Cham, Switzerland: Springer, 2019, pp. 630–641.

[18] W. Li, B. Jin, and X. Wang, "SparseMAAC: Sparse attention for multi-agent reinforcement learning," in *Database Systems for Advanced Applications*, vol. 11448. Cham, Switzerland: Springer, 2019, pp. 96–110.

[19] R. Lowe, "Multi-agent actor-critic for mixed cooperative-competitive environments," in *Proc. Adv. Neural Inf. Process. Syst.*, 2017, pp. 6379–6390.

[20] P. Christodoulou, "Soft actor-critic for discrete action settings," 2019, *arXiv:1910.07207*. [Online]. Available: http://arxiv.org/abs/1910.07207

[21] P. Lai, Q. He, M. Abdelrazek, F. Chen, J. Hosking, J. Grundy, and Y. Yang, "Optimal edge user allocation in edge computing with variable sized vector bin packing," in *Proc. 16th Int. Conf. Service-Oriented Comput. (ICSOC)*, Hangzhou, China, 2018, pp. 230–245.

[22] Q. Peng, Y. Xia, Z. Feng, J. Lee, C. Wu, X. Luo, W. Zheng, S. Pang, H. Liu, Y. Qin, and P. Chen, "Mobility-aware and migration-enabled online edge user allocation in mobile edge computing," in *Proc. IEEE Int. Conf. Web Services (ICWS)*, Milan, Italy, Jul. 2019, pp. 91–98.

[23] C. Guerrero, I. Lera, and C. Juiz, "A lightweight decentralized service placement policy for performance optimization in fog computing," *J. Ambient Intell. Humanized Comput.*, vol. 10, no. 6, pp. 2435–2452, 2019.

[24] P. Lai, Q. He, M. Abdelrazek, F. Chen, J. Hosking, J. Grundy, and Y. Yang, "Optimal edge user allocation in edge computing with variable sized vector bin packing," in *Proc. Int. Conf. Service-Oriented Comput.*, vol. 11236, Nov. 2018, pp. 230–245.

[25] J. Feng, Q. Pei, F. R. Yu, X. Chu, and B. Shang, "Computation offloading and resource allocation for wireless powered mobile edge computing with latency constraint," *IEEE Wireless Commun. Lett.*, vol. 8, no. 5, pp. 1320–1323, Oct. 2019.

[26] 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, Nov. 2020.

[27] Z. Kuang, L. Li, J. Gao, L. Zhao, and A. Liu, "Partial offloading scheduling and power allocation for mobile edge computing systems," *IEEE Internet Things J.*, vol. 6, no. 4, pp. 6774–6785, Aug. 2019.

HAIFENG LU was born in 1993. He received the master's degree from the Computer Science Department, Donghua University, Shanghai, in 2017, and the Ph.D. degree from the School of Information Science and Engineering, East China University of Science and Technology. His current research interests include edge computing and reinforcement learning.

CHUNHUA GU was born in 1970. He is currently a Professor and a Ph.D. Supervisor with the School of Information Science and Engineering, East China University of Science and Technology. His current research interests include cloud computing and the Internet of Things. He was a Senior Member of China Computer Federation.

FEI LUO was born in 1978. He received the B.S., M.S., and Ph.D. degrees in computer science from the Huazhong University of Science and Technology, in 1997, 2004, and 2008, respectively. Since 2015, he has been an Associate Professor with the Huazhong University of Science and Technology. He has authored more than 30 articles and more than ten inventions. His research interests include distributed computing and cloud computing.

WEICHAO DING was born in 1989. He received the B.S. degree in computer science and technology from Northeast Forestry University, Harbin, China, in 2013. He is currently pursuing the M.S. and Ph.D. degrees in computer applications with the East China University of Science and Technology, Shanghai, China. His current research interests include cloud computing, cloud resource management and optimization, and big data applications.

SHUAI ZHENG was born in 1994. He received the B.S. degree in information science and technology from Donghua University, Shanghai, China, in 2018. He is currently pursuing the degree with the East China University of Science and Technology, Shanghai. His current research interests include edge computing and reinforcement learning.

YIFAN SHEN was born in 1997. He received the B.S. degree in computer science and technology from Nanjing Forestry University, Nanjing, China, in 2019. He is currently pursuing the M.S. degree in computer technology with the East China University of Science and Technology, Shanghai, China. His research interests include mobile edge computing, deep reinforcement learning, and convex optimization.