Task Offloading based on Deep Reinforcement Learning with LSTM for Mobile Edge Computing

1st Bin Xu
School of Internet of Things
Nanjing University of Posts and Telecommunication
Nanjing, China
xubin2013@njupt.edu.cn

2nd Jinming Chai
School of Internet of Things
Nanjing University of Posts and Telecommunication
Nanjing, China

3rd Dan Liu
School of Internet of Things
Nanjing University of Posts and Telecommunication
Nanjing, China

4th Zhichao Zhuang
School of Internet of Things
Nanjing University of Posts and Telecommunication
Nanjing, China

5th Yunkai Zhao
School of Internet of Things
Nanjing University of Posts and Telecommunication
Nanjing, China

6th Xingjian Xu
School of Internet of Things
Nanjing University of Posts and Telecommunication
Nanjing, China

7th Jianming Zhu
School of Internet of Things
Nanjing University of Posts and Telecommunication
Nanjing, China

8th Jin Qi
School of Internet of Things
Nanjing University of Posts and Telecommunication
Nanjing, China

2022 China Automation Congress (CAC) | 978-1-6654-6533-5/22/$31.00 ©2022 IEEE | DOI: 10.1109/CAC57257.2022.10055708

***Abstract*—Mobile devices may offload computationally intensive tasks to edge servers for processing in Mobile Edge Computing (MEC), thereby improving the quality of experience. Heuristic algorithms are feasible for MEC offloading decisions and resource allocation, but they are not suitable for high real-time MEC systems, ignoring the impact of channel dynamic changes on the computational offloading problem. In this paper, we construct a MEC system in a time-varying fading channel scenario and propose a deep reinforcement learning algorithm based on LSTM (DR-LSTM) to solve the joint optimization problem of task offloading decision and resource allocation. The DR-LSTM is combined with an order-preserving quantization algorithm to generate offloading decision, and a linear relaxation method is used to solve the resource allocation problem. Finally, it is verified through simulations that the DR-LSTM can effectively solve the task offloading and resource allocation problem under this model.**

Keywords—Mobile edge computing, Computation offloading, Deep reinforcement learning, LSTM

I. INTRODUCTION

With the growth of IoT technology and the widespread use of new applications, more and more computationally intensive tasks have increased demands on the processing capacity and task processing delay of mobile devices. However, limited by their own computing power, battery capacity and other attributes, mobile devices can hardly meet the demand for computing resources and latency for tasks of increasing scale. In the mobile edge computing (MEC) , requests from mobile devices can be executed nearby rather than being satisfied by backhauling from remote clouds. By spreading the burden of cloud servers to the base station, the quality of service is improved [1]. Due to the ongoing advancements in wireless charging technologies, the mobile devices can be charged through wireless channels. This technology significantly reduces deployment costs and effectively increases the computing power of mobile devices [2]. The resources of channel transmission are limited, and redundant offloading transmission will lead to a waste of computing resources, so it is very important to build a reasonable MEC system model and formulate corresponding task offloading and resource allocation strategies to improve the effectiveness of resource utilization in MEC.

There has been a lot of studies conducted to address task offloading and resource allocation problem. The problem of jointly optimizing task offloading and resource allocation to achieve maximum computational performance in multi-user scenarios is the main challenge. The computational offloading problem in edge computing can usually be expressed as mixed integer programming (MIP). When the number of devices rises, the search space for the problem become larger and the computational complexity becomes greater. The quality of user experience is difficult to be satisfied by using heuristic algorithm, such as genetic algorithm [3], particle swarm algorithm [4]. In [5], Zhang et al. jointly optimize the task offloading decision, computing resource and wireless resource allocation problem. This issue is decomposed into several subproblems using Lyapunov optimization theory. Then these subproblems are solved by convex decomposition methods and matching game algorithm. In [6], Ning et al. suggested an iterative heuristic resource allocation approach for dynamic task offloading for serially dependent workloads in a cloud-side collaboration situation. In [7], Yan et al. constructed the MEC model with task dependency, and a dichotomous search technique and Gibbs sampling algorithm are provided to get the best task offloading policy and resource allocation. As in [5-7] use convex optimization and heuristic algorithm, which usually have high time complexity. And once the wireless channel conditions change, task offloading decision and resource allocation need to be re-solved. Therefore, they are not suitable for real-time processing in fast fading channel. Deep reinforcement learning is an algorithm that combines deep learning with reinforcement learning, which is widely used in

This work was supported in part by the Project funded by China Postdoctoral Science Foundation under Grant 2020M671552, in part by the Postgraduate Research and Practice Innovation Program of Jiangsu Province under Grant SJCX21_0301.

various complex decision solving fields, such as combinatorial optimization and multi-party games [8]. Deep reinforcement learning techniques have been suggested in a number of studies as a solution to the task offloading issue in mobile edge computing. In [9], Amine Abouaomar et al. studied the problem of service migration in a MEC-enabled vehicular network. It is modeled as a multi-agent Markov decision process and solved by leveraging deep Q learning (DQL) algorithm. In [10], Qiu et al. proposed a new model-free deep reinforcement learning-based online offloading method. In [11], Wang et al. proposed an optimized caching and computation offloading decision in MEC systems using federated learning (FL) combined with DRL. As in [9-11], these studies did not consider the dynamics and timing of the channel in the scene. Although there has been some progress in the use of deep reinforcement learning to address the issues of task offloading and resource allocation, there is still a pressing need for research into how to propose a deep reinforcement learning algorithm that is appropriate for the model in light of the model's characteristics.

In time-varying channel MEC scenarios, the instability of the wireless channel state leads to difficulties in guaranteeing the data transmission quality as well as the wireless charging quality. In this paper, a mobile edge computing offloading system in time-varying fading channel scenario is constructed. A deep reinforcement learning with LSTM algorithm (DR-LSTM) is proposed to handle the task offloading and resource allocation problem in this scenario.

II. System Model

In this section, a computational model of wireless supply edge under time-varying fading channel is constructed, as shown in Fig. 1.

A. Scenario Model

The MEC server not only handles task data from the devices, but also serves as a stable wireless power supply for each mobile device.

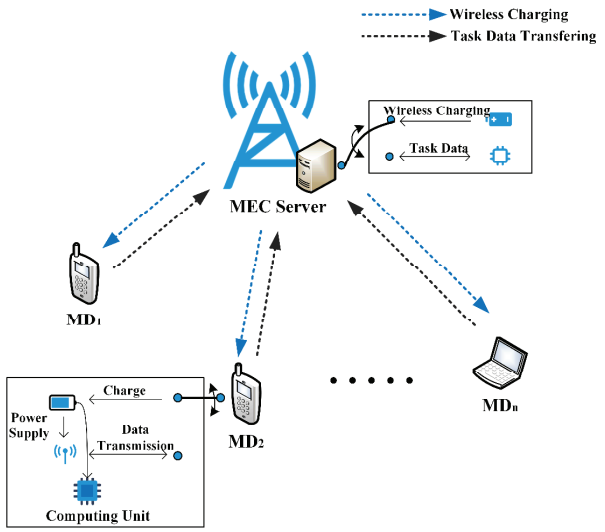


Fig. 1. System model

In Fig. 1, There are n devices, one MEC server, and the set of devices is represented as $N = \{1,2,\cdots,n\}$. The vector $X = \{x_1, x_2, \cdots, x_n\}$ is used to represent the offloading decision of each device. If x_i is 1, it signifies that device i offloads the task to the MEC server for execution. If x_i is 0, it offloads the task to local execution. h_i represents the channel condition between the MEC server and the device i .The set $H = \{h_1, h_2, \cdots, h_n\}$ denotes the channel conditions between n devices to the MEC server.

In this model, the wireless power supply of the devices uses the same frequency band as the task data transmission. So in order to avoid mutual interference between wireless power supply and data transmission, we construct the channel as a time division multiplexing.

Each time slice is LT in length, and the device generates a task and calculates it within each time slice. Each device divides the time slice into two parts, one for receiving and sending task data with the MEC server, and the other for wireless power supply. In each time slice, the device will use the obtained energy to complete the computation of the task. $\varepsilon \in [0,1]$ is the split ratio of time slices, εLT is the wireless power supply duration and $1-\varepsilon LT$ is the data transmission duration. It is assumed that the channel in the model is in a time-varying fading state, so h_i won't change in each time slice. According to the wireless channel time-varying fading formula $\bar{h}_i = A_d \left(\frac{3\cdot 10^8}{4\pi f_c d_i}\right)^{d_e}$, $\bar{h}_i$ represents the average channel gain. $h_i = \bar{h}_i \alpha_i^t$ is generated based on the Rayleigh fading channel model. $A_d = 4.11$ represents the antenna gain. $f_c = 915$MHz denotes the carrier frequency, and $d_e = 2.8$ denotes the path loss index. d_i represents the distance between the mobile device and the MEC server, which is uniformly distributed between (2.5,5.2) meters, and α_i is an independent random channel fading factor that follows a unit-mean exponential distribution. Thus, the energy obtained by device i through wireless power supply during the εLT time can be given [12]:

$$\xi_i = \varpi P h_i \varepsilon LT, \tag{1}$$

where $\varpi \in (0,1)$ is the efficiency in collecting energy and P denotes the wireless transmitting power of the MEC server.

In the local computing mode, the device does not need to transmit task data and receives wireless power supply in the whole time period. v_i is the computing speed of the processor, $t_i \in [0, LT]$ is the task computing time, Λ represents the number of cycles needed to process one bit of task data. $\frac{v_i t_i}{\Lambda}$ denotes the amount of data computed by device i in the time t_i. The device will consume the power energy and complete the computation of the task in the entire time. The energy consumption of the device during computing the task as follows [12]:

$$E_i = \kappa_i v_i^3 t_i, \tag{2}$$

κ_i denotes the computational energy efficiency factor, and in the local computation mode, the device does not need to consider the task data transmission time. So $E_i = \xi_i$, $t_i = LT$, it is known that $v_i = \left(\frac{\xi_i}{\kappa_i t_i}\right)^{\frac{1}{3}}$ and the fixed parameter $\frac{(\varpi P)^{\frac{1}{3}}}{\Lambda}$ is

replaced by Pa. The computational rate of the task in the local mode is simplified as follows [12]:

$$rl_i(\varepsilon) = \frac{v_i t_i}{\Lambda LT} = \frac{v_i}{\Lambda} = \frac{(\varpi P)^{\frac{1}{3}}}{\Lambda}\left(\frac{h_i}{\kappa_i}\right)^{\frac{1}{3}}\varepsilon^{\frac{1}{3}} = Pa\left(\frac{h_i}{\kappa_i}\right)^{\frac{1}{3}}\varepsilon^{\frac{1}{3}}. \quad (3)$$

In MEC server computing mode, the device offloads the task to MEC server after collecting energy. We denote the offloading time of the i-th device as $to_i LT, to_i \in [0,1)$. The MEC server's computing speed and transmission power are substantially higher than the device's, so the time spent for data downlink as well as computational tasks during task offloading to the MEC server is neglected in the scenario, which leads to equation as follows [12]:

$$\sum_{i=1}^{n}(to_i + \varepsilon) \le 1. \quad (4)$$

All the power obtained by the device through wireless charging is used for task offloading, and the task offloading rate is considered as its computation rate.

The device's current computational rate, as determined by Shannon's formula [12], is as follows:

$$rm_i(\varepsilon, to_i) = Bto_i log\left(1 + \frac{\pi P(h_i)^2 \varepsilon}{to_i \sigma}\right), \quad (5)$$

where B is the channel bandwidth and σ is the noise power.

B. Problem Formulation

The task data computing rate of all devices within each time slice as follows:

$$V(H, X, To, \varepsilon) = \left(\sum_{i=1}^{n}\left((1 - x_i)rl_i(\varepsilon) + x_i rm_i(\varepsilon, to_i)\right)\right), \quad (6)$$

where $To \in \{to_1, to_2, \cdots, to_n\}$.

In each time slice, the optimization goal is to maximize the computing rate of the tasks created by all devices, as follows:

$$\begin{aligned} &Pr1: \max_{X,To,\varepsilon,} V(X, To, \varepsilon) \\ &subject\ to \\ &C1: \sum_{i=1}^{n}(to_i + \varepsilon) \le 1, \varepsilon \ge 0, to \ge 0, \\ &C2: x_i \in \{0,1\}. \end{aligned} \quad (7)$$

III. LSTM-based Deep Reinforment Learning Algorithm

In reality, because the channel is in a time-varying fading state, it is need to take the change of the current channel conditions into account when solving equation (7). To solve this complexity problem and take full advantage of the regularity of time-varying channel fading, a DR-LSTM algorithm is used in this paper, which is outlined below.

A. The DR-LSTM Framework

The DR-LSTM utilizes the concept of deep reinforcement learning, which continuously explores the environment and receives feedback for adjustment after making a response. The DR-LSTM framework is shown in Fig. 2.

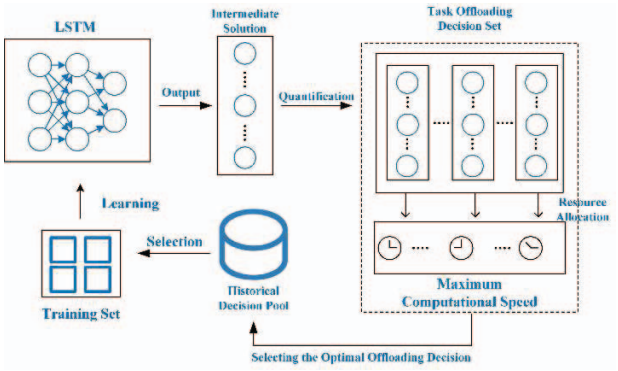


Fig. 2. DR-LSTM framework

As shown in Fig. 2, the DR-LSTM consists of handling task offloading and resource allocation problem, where the development of task offloading decision relies on the LSTM neural network. The DR-LSTM designed a task offloading policy function π, which can rapidly generate an optimal offloading decision when channel conditions are inputted. The task offloading decision is generated by using the LSTM neural network, which is characterized by its embedded parameters k, such as the weights that connect the hidden neurons.

In each time slice, the DR-LSTM takes the current channel set $H = \{h_1, h_2 \cdots, h_n\}$ as input and outputs a set of intermediate solutions X_temp based on the current policy π_{θ_t}. The DR-LSTM parameterizes the output intermediate solutions as $X\_temp = f_{\pi_{\theta_t}}(h_i)$, and obtains a set of offloading decisions by quantization. Then the DR-LSTM put the current channel conditions H and the task offloading decision into the historical decision pool. The DR-LSTM will choose a group of training samples from the historical decision pool after every certain time slice to train the LSTM neural network and change its parameters. During the continuous iteration, the parameters of the LSTM network are continuously adjusted and gradually improved. The following section describes the task offloading decision and the part of resource allocation strategy.

B. Offloading Decision and Resource Allocation

In each time slice, the LSTM first outputs an intermediate solution $X\_temp = \{xt_1, xt_2, \cdots, xt_n\}$, where $\forall xt_i \in [0,1]$. Then it is quantized into β offloading schemes by an order-preserving quantization algorithm [13], where β is fixed parameter. The quantization algorithm is described in the following. For the intermediate solution, it is first ordered by the distance between each dimension and 0.5. Then form the vector $X\_d = \{xd_1, xd_2, \cdots, xd_n\}$, where $|xd_1 - 0.5| \le |xd_2 - 0.5| \le \cdots \le |xd_n - 0.5|$.

$XS = \{X_1, X_2, \cdots X_\beta\}$ denote the set of quantized solutions. Then for any $X_\varphi = \{x_{\varphi 1}, x_{\varphi 2}, \cdots, x_{\varphi n}\} \in XS$, $\varphi = 1,2, \cdots, \beta$, each of its dimensions as follows[12]:

$$x_{\varphi i}=\begin{cases}1, xt_i > xd_\varphi, \varphi \in \{2,3,\cdots \beta\},\\ 1, xt_i = xd_\varphi, xd_\varphi \le 0.5, \varphi \in \{2,3,\cdots \beta\},\\ 0, xt_i = xd_\varphi, xd_\varphi > 0.5, \varphi \in \{2,3,\cdots \beta\},\\ 0, xt_i < xd_\varphi, \varphi \in \{2,3,\cdots \beta\},\\ 1, xt_i > 0.5, \varphi = 1,\\ 0, xt_i \le 0.5, \varphi = 1.\end{cases} \quad (8)$$

Bring the set of quantized solutions XS into the equation (7) to get the offloading decision X_best that maximizes the objective function of the equation (7). Compared to the traditional KNN quantization algorithm [14], the order-preserving quantization algorithm produces a greater distance between offloading decisions, creating a higher diversity in the set of candidate decisions.

Equation (7) is decoupled into two problems, i.e. solving for resource allocation given the task offloading decision, as follows:

$$\begin{aligned} &Pr2: \max_{\varepsilon, To} V(To, \varepsilon)\\ &subject\ to\\ &C1: \sum_{i=1}^{n} (to_i + \varepsilon) \le 1, \varepsilon \ge 0,\ to_i \ge 0. \end{aligned} \quad (9)$$

According to the concavity determination theorem of multivariate functions, the objective function is a convex function with respect to the variables ε and To, which is solved by using the Lagrangian relaxation.

C. Deep Reinforcement Learning Mechanism

Using the empirical replay technique, the offloading decisions are stored in a history decision pool for updating the LSTM neural network within each time slice. When the task offloading decision in the *j-th* time slice is made, the channel conditions H in the *j-th* time slice and the offloading decision X in that time slice are added to the decision pool as a collection of samples. When the decision pool reaches the capacity limit, the earliest samples putted into the decision pool are removed to accommodate the latest samples. Because of the advantages of LSTM in processing temporal data, the samples in the decision pool are not directly used for training. A continuous time slice is taken, its channel and offloading decisions are composed as training sample inputs. The Adam algorithm [15] is used to adjust the LSTM neural network parameters θ_t and the offloading decision of the next time slice is used as the output. Fig. 3 below shows the historical decision pool as well as the model for the training samples.

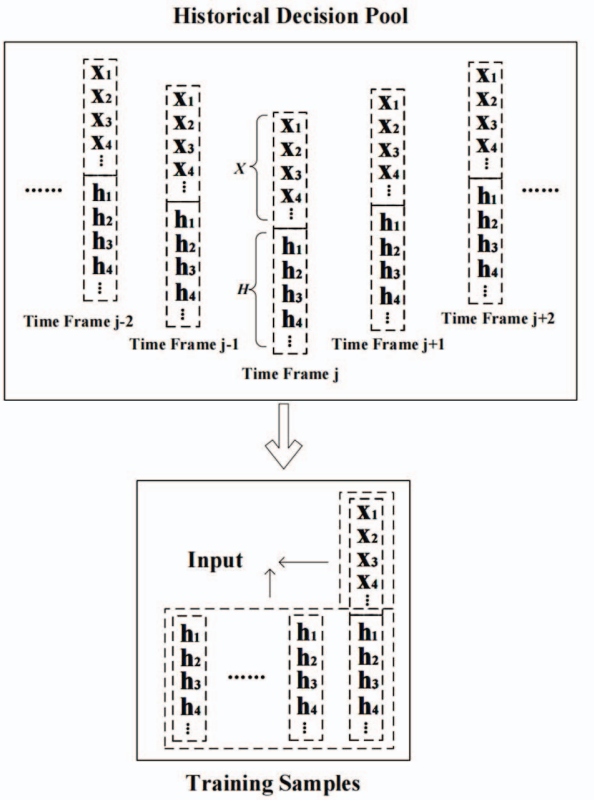


Fig. 3. Historical decision pool and training samples

At the beginning of the algorithm run, there are not enough samples in the decision pool when the LSTM uses random initialization of the parameters. The LSTM network is used because it can track the channel state $H = \{h_1, h_2, \cdots, h_n\}$ over time. Specifically, the LSTM network will take the historical channel conditions as matrix inputs to learn the time-varying fading channel dynamics.

IV. Numerical Simulations

In this part, we will use experiments to verify the performance of the DR-LSTM algorithm. The experimental parameters are set as shown in Tab. 1.

TABLE I. The specific parameters of the simulation

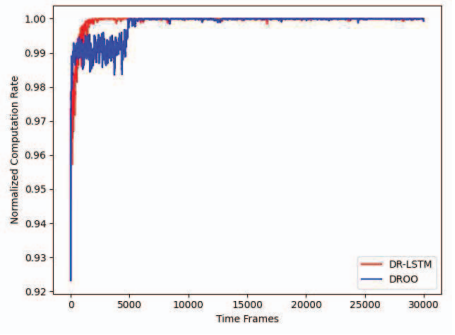
P	ϖ	κ_i	B	β
3w	0.5	10^{-26}	2MHz	9
σ	n	**Training interval**	**Training samples number**	**Historical decision pool memory size**
10^{-10}	10	10	100	1024

In order to verify the advantages of the DR-LSTM algorithm in time-varying fading channel, this paper compares DR-LSTM with DROO [12], and compares the normalized computation rate of the two algorithms in each time slice. The normalized computation rate VS is calculated as follows [12]:

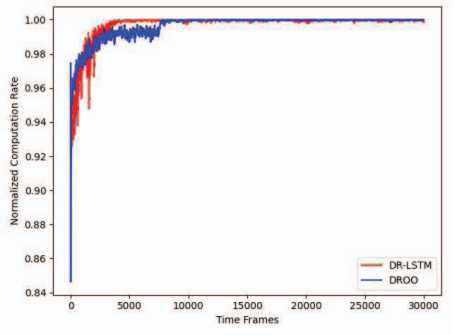
$$VS = \frac{V(X, To, \varepsilon)}{\max\limits_{x^* \in \{0,1\}^n} V(x^*, To, \varepsilon)}. \quad (10)$$

The maximum calculated rate on the denominator is obtained by enumerating all offloading actions, x^* is the optimal solution among all offloading decisions.

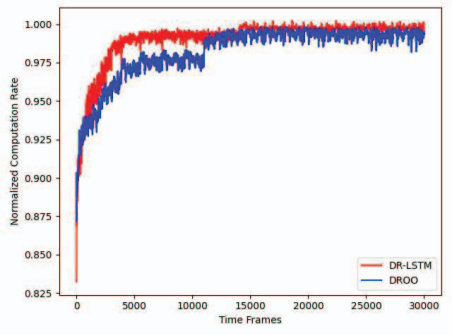
Fig. 4 shows the statistics of the normalized computation rate for each of the two algorithms with different number of devices over 30,000 time slices in a time-varying fading channel.



(a) Normalized computation rates when $n = 5$



(b) Normalized computation rates when $n = 10$



(c) Normalized computation rates when $n = 20$

Fig. 4. Normalized computation rates of different number of devices

As shown in Fig. 4, the normalized computation rate obtained by DR-LSTM is slightly lower than DROO in the initial time slices at the number of devices $n = 5$ and $n = 10$. With the increase of time slices, the effect of DR-LSTM gradually exceeds that of DROO after 2000 time slices. At $n = 20$, the effect of DR-LSTM is significantly better than that of DROO, and at 1000 time slices although there are fluctuations after convergence, the fluctuation of DR-LSTM is between 0.98 and 1.00, while the fluctuation of DROO is between 0.975 and 0.98. The reason for this effect is that LSTM can learn more effectively in time-varying fading channel, and given that the training samples of DNN networks do not need to consider timing effects and only one sample is needed as training sample for each training, it is easier to obtain training samples and adjust their own parameters in the early iterations of the algorithm. As the number of iterations of the algorithm increases, the number of samples in the historical decision pool increases, and the LSTM obtains more training samples, and uses the temporal laws embedded in the samples to better adjust its parameters and gradually obtain better solution results.

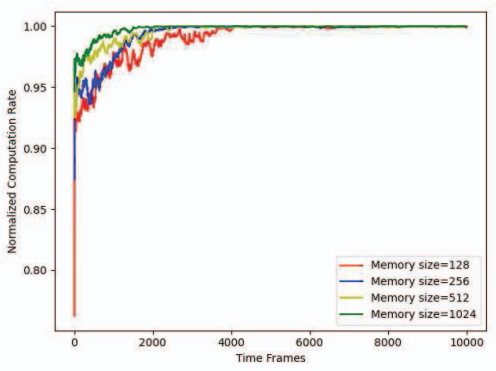


Fig. 5. Normalized computation rates of different memory sizes

As shown in Fig. 5, the effect of different historical decision pool memory size on the convergence performance of DR-LSTM is further investigated. From the above figure, it can be seen that when the memory size is small, there are large fluctuations on the convergence performance and the convergence speed is slower. At larger memory size, the convergence speed is faster and the fluctuation is less.

V. CONCLUSION

In this paper, a task offloading and resource allocation model for channel fading scenarios is constructed for the dynamic change of channel state in edge environment, and a deep reinforcement learning algorithm based on LSTM (DR-LSTM) is proposed for the model. The DR-LSTM can learn from past offloading decision experience, use LSTM neural network and an order-preserving quantization algorithm to generate task offloading decision within each time slice. The resource allocation problem is solved by linear relaxation method. Finally, it is verified through simulation experiments that DR-LSTM can effectively solve the task offloading and resource allocation problems under this model, and it can continuously adjust the parameters of LSTM according to the environmental feedback to quickly find the optimal solution, reduce the training loss, and improve the algorithm solving effect. In future research, we plan to apply the DR-LSTM algorithm to more complex scenarios, such as scenarios where multiple servers and devices work together. In order to further improve the performance of the algorithm in actual edge computing scenarios, the use of different neural networks combined with reinforcement learning algorithms to solve the problem will be deeply considered.

REFERENCES

[1] Liyanage, Madhusanka, et al. "Driving forces for multi-access edge computing (MEC) IoT integration in 5G." ICT Express 7.2 (2021): 127-137.

[2] Park, Junhee, et al. "Latency minimization for wireless powered mobile edge computing networks with nonlinear rectifiers." IEEE Transactions on Vehicular Technology 70.8 (2021): 8320-8324.

[3] Yala, Louiza, Pantelis A. Frangoudis, and Adlen Ksentini. "Latency and availability driven VNF placement in a MEC-NFV environment." 2018 IEEE Global Communications Conference (GLOBECOM). IEEE, 2018.

[4] Jabri, Issam, et al. "Vehicular fog gateways selection on the internet of vehicles: A fuzzy logic with ant colony optimization based approach." Ad Hoc Networks 91 (2019): 101879.

[5] Zhang, Qi, et al. "Dynamic task offloading and resource allocation for mobile-edge computing in dense cloud RAN." IEEE Internet of Things Journal 7.4 (2020): 3282-3299.

[6] Ning, Zhaolong, et al. "A cooperative partial computation offloading scheme for mobile edge computing enabled Internet of Things." IEEE Internet of Things Journal 6.3 (2018): 4804-4814.

[7] Yan, Jia, et al. "Optimal task offloading and resource allocation in mobile-edge computing with inter-user task dependency." IEEE Transactions on Wireless Communications 19.1 (2019): 235-250.

[8] J. B. Liang, H. H. Zhang C. Jiang and T. S. Wang,"Research Progress of Task Offloading Based on Deep Reinforcement Learning in Mobile Edge Computing, "Computer Science, vol. 48, no.7, pp: 316-323, 2021,(in Chinese).

[9] Abouaomar, Amine, et al. "A deep reinforcement learning approach for service migration in mec-enabled vehicular networks." 2021 IEEE 46th conference on local computer networks (LCN). IEEE, 2021.

[10] Qiu, Xiaoyu, et al. "Online deep reinforcement learning for computation offloading in blockchain-empowered mobile edge computing." IEEE Transactions on Vehicular Technology 68.8 (2019): 8050-8062.

[11] Wang, Xiaofei, et al. "In-edge ai: Intelligentizing mobile edge computing, caching and communication by federated learning." IEEE Network 33.5 (2019): 156-165.

[12] Huang, Liang, Suzhi Bi, and Ying-Jun Angela Zhang. "Deep reinforcement learning for online computation offloading in wireless powered mobile-edge computing networks." *IEEE Transactions on Mobile Computing* 19.11 (2019): 2581-2593.

[13] Wang, Feng, et al. "Joint offloading and computing optimization in wireless powered mobile-edge computing systems." IEEE Transactions on Wireless Communications 17.3 (2017): 1784-1797.

[14] Subramaniam, Prashanth, and Maninder Jeet Kaur. "Review of security in mobile edge computing with deep learning." 2019 Advances in Science and Engineering Technology International Conferences (ASET). IEEE, 2019.

[15] Kingma, Diederik P., and Jimmy Ba. "Adam: A method for stochastic optimization." arXiv preprint arXiv:1412.6980 (2014).