Computing Resource Allocation Based on Multi-base Station and Multi-user Scenario in Mobile Edge Computing

Yaozhang Zhong[1,2] , Yingkui Du [1,2], Jing Zhao[1,2*], Qinghang Gao[1,2], Yujuan Zou [1,2], Yong Luo[1,2], Kailin Chao[1,2] and Ziyu Yin[1,2]

[1] School of Software, Jiangxi Normal University, Nanchang 330022, China.
[2]Jiangxi Provincial Engineering Research Center of Blockchain Data Security and Governance, Nanchang 330022, China
{202140100846, gaoqinghang, jxnu_dyk, luoyong1020, chaokailin}@jxnu.edu.cn
{zhaojinghaze, zyj_jxnu, yinziyu0121}@163.com

Abstract

With the rapid development of 5G communication technology and the Internet of Things, a large number of new services have emerged on mobile terminal devices, which require low latency and high bandwidth, resulting in an explosive growth of mobile communication traffic. Traditional cloud computing is not a good solution to this situation. Therefore, mobile edge computing comes into being. Mobile Edge Computing (MEC) is a new computing mode. The edge server is deployed at the edge. The computing task generated by the mobile terminal device can be unloaded to the edge server that is physically closer to the mobile terminal device for processing. The problem of limited computing power of mobile terminal users is solved. At present, many researches on mobile edge computing are based on multi-user single-server scenarios. Based on this, this paper will study and analyze the resource allocation of the edge end in the multi-base station and multi-user scenario in mobile edge computing. Firstly, the multi-base station and multi-user application scenario model is described, and then the concept of edge computing resource allocation is proposed. Finally, the effectiveness of the two resource allocation algorithms is verified through simulation experiments. The results show that the proposed resource allocation scheme can effectively reduce the task processing delay of mobile users, and the dynamic minimum connection algorithm can allocate the computing resources at the edge more rationally.

Keywords: Internet of Things, Mobile Edge Computing, Edge Servers, Resource Allocation

1 Introduction

In recent years, with the rapid development of the Internet of Things and 5G communication technology, mobile terminal devices (smartphones, smart watches, smart bracelets, pads, etc.) are gradually becoming an important tool for our sslife, study, work, and entertainment, and their number

The 7th International Conference on Mobile Internet Security (MobiSec’23), Dec. 19-21, 2023, Okinawa, Japan, Article No. 40

*Corresponding author: School of Software, Jiangxi Normal University, Nanchang 330022, China, Tel: + 86-571-88165512 is also increasing. With the increasing number of these mobile terminal devices, many new applications are also emerging, which has brought explosive growth in mobile data traffic. Many emerging applications often need to consume a lot of computing resources, such as AR, VR, and large-scale games, to achieve low latency requirements. However, the computing resources of most mobile terminal devices are very limited. For some delay-sensitive and computation-intensive tasks, it is difficult to meet the needs of low delay only by mobile terminals themselves.

Traditional cloud computing supports powerful computing and storage capabilities. In cloud computing mode, compute-intensive tasks and delay-sensitive tasks generated by mobile terminal devices can be offloaded to cloud servers far away from mobile terminal devices to solve the problem of insufficient computing power of mobile terminal devices. However, the cloud server located in the core network is usually very far away from the mobile terminal device. If all end users unload the computing tasks generated by the mobile terminal device to the cloud server, the network load will be increased, resulting in insufficient bandwidth, and the transmission delay of the task will be greatly increased, which still cannot meet the service demand of low delay.

To solve this situation, the European Telecommunications Standards Institute (ETSI) in 2014, was the first to put forward the edge mobile computing (MEC) [1]. Mobile edge computing technology is usually deployed at the edge base station. Compared with the cloud server of the core network, the mobile edge server is much closer to the mobile terminal device. Therefore, if the computing task of the mobile terminal device is uninstalled, The computing task can be offloaded to the mobile edge server that is closer to the location and within the coverage of the base station signal, to meet the needs of mobile end users to achieve low latency, which not only alleviates the phenomenon of core network congestion, but also greatly reduces the network delay, thus improving the quality of user experience.

2 Related Work

As a new computing paradigm, mobile edge computing can better solve the problem of insufficient computing power, storage capacity, and battery capacity of mobile terminal devices, which has attracted a large number of scholars to study it. At present, most scholars in the field of mobile edge computing focus on computing offload, and the optimization goals are delay optimization, energy consumption optimization, or weighted and optimized between delay and energy consumption.

(1) Energy consumption optimization. Reference [2] studies the computational offloading problem in small cellular networks. A hierarchical unloading algorithm based on genetic algorithm (GA) and PSO algorithm is designed. Minimize power consumption across all terminals by jointly optimizing compute offload decisions, spectrum, power, and server compute resource allocation. Reference [3] studies the computational unloading problem of a single mobile device and different types of random arrival tasks in a dynamic MCC environment. Using Lyapunov optimization theory, a DREAM algorithm is designed. By jointly optimizing compute offload decisions, mobile device compute resource allocation, and network interface selection, mobile device power consumption is minimized while maintaining task queue stability.

(2) Delay optimization. Literature [4] studies the computational unloading problem in industrial scenarios and designs a computational unloading algorithm PSAO based on the PSO algorithm. By introducing a penalty function, PSAO can effectively reduce the computing delay of mobile devices under the limitation of energy consumption of mobile devices. Literature [5] studies the computational unloading problem of random arrival tasks in a single mobile device. Using the Markov decision process, a one-dimensional search algorithm is designed to determine the unloading strategy of a task. Reference [6] studies the computational unloading problem of multi-type random

arrival tasks between edge computation-enabled base stations and MEC-enabled base stations (MEC-BSs). To balance the task computation delay of each MEC-BS, the problem is modeled as a non-cooperative game and a distributed iterative algorithm is designed. Simulation results show that the algorithm has fast convergence, and the total computation delay can be reduced by 45% ~ 50% on average in multiple scenarios.

(3) Weighting and optimization of energy consumption in time delay. Reference [7] studies the problem of multi-user and multi-task computing offloading in mobile edge computing networks. A computational unloading algorithm based on reinforcement learning is designed to minimize the system cost, that is the weighted sum of energy consumption and delay. Literature [8] also studies the problem of multi-user and multi-task computing offloading in MEC. A hierarchical unloading algorithm based on a flow shop scheduling algorithm and reinforcement learning algorithm is designed. Minimize the weighted sum of system power consumption and delay by jointly optimizing compute offload decisions, task scheduling, and server compute resource allocation. Reference [9] studies the multi-user computing offloading problem in mobile edge cloud computing. In the multi-channel wireless interference environment, the problem is modeled as a multi-user computing offload game, and a distributed computing offload algorithm is designed. The simulation results prove the convergence of the algorithm and its good unloading performance. Reference [10] studies the multi-mobile device computing offloading problem in MCC. In the multi-channel wireless competition environment, the problem is modeled as a non-cooperative game, and a fully distributed computing offloading algorithm based on machine learning is designed, that is, each mobile device only needs to make offloading decisions based on local information, without information exchange. Simulation results also prove the convergence of the proposed algorithm and its advantages over the comparison algorithm.

At present, in the field of mobile edge computing, there are many research articles on the research direction of mobile client computing offloading, but there are relatively few studies on the allocation of computing resources at the edge, and the following problems exist. when computing tasks are offloading to the edge server: The service scope of the mobile edge server overlaps, and mobile terminal devices in the overlapped part can select multiple mobile edge servers for uninstallation. How to allocate mobile edge servers to edge users reasonably remains to be solved.

This paper focuses on the allocation of edge computing resources, and how to make reasonable use of edge computing resources. The main contributions of this article are as follows:

1. proposing and simplifying a multi-user and multi-server system model.
2. using a real dataset.
3. using two algorithms to schedule edge resources, and verifying the effectiveness of the algorithm.

3 System Model

In this section, the MEC compute offload scenario is described and then modeled.

In real life, MEC applications can be seen everywhere, such as automatic driving, license plate number recognition, AR, VR, and face recognition, which are currently very popular. For example, when a MEC system is deployed at an intersection, as shown in Figure 1, compute-intensive tasks such as face recognition applications of mobile terminal devices can be offloaded to the edge of the MEC system, thereby reducing the tas processing delay and improving the user's quality experience. However, in rush hour, due to the increase in the number of people, the unloading of terminal tasks to the edge of the terminal at the same time also increases sharply, resulting in the problem of robbing the edge of resources, resulting in a significantly longer response time of mobile terminal applications. Therefore, our work is to design a MEC system, that can realize the rational utilization of edge resources, improve the utilization rate of resources, and reduce the delay of task processing. Then this paper designs a multi-base station multi-user system model to simplify this application scenario. Finally, the communication model and calculation model are introduced.

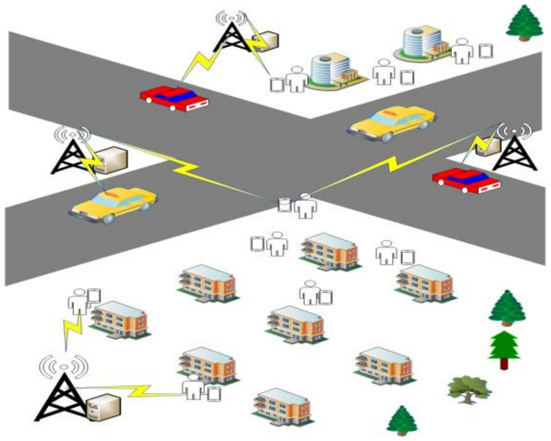

Figure 1: The scenario with multiple mobile devices in mobile edge computing.

To simplify the system and solve the problems in resource allocation at the edge, this paper proposes a mobile edge computing system with multiple mobile devices is presented. As shown in Figure 2, the mobile edge computing system model diagram consists of n base stations, and each base station is equipped with a mobile edge server, so the total number of mobile edge servers and m mobile terminal devices is n.

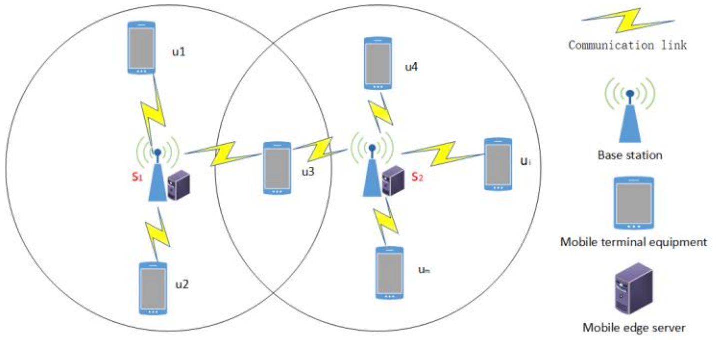

Figure 2: The mobile edge computing system with multiple mobile devices

U_i is used to represent the i th mobile terminal device, $Task_i$ is the task to be calculated in the device U_i, and the attribute of the task can be represented in two dimensions as $Task_i=\{D_i, C_i\}$, where D_i represents the data size of the task to be calculated, and C_i represents the total number of CPU cycles required to process the task. S_i is used to represent the i th mobile edge server.$S_i=\{1,2, \bullet\bullet,n\}$ indicates a collection of mobile edge servers. The mobile terminal device set is represented by $U_i=\{1,2, \bullet\bullet,m\}$.

According to the requirements of the computing task to be processed, the mobile terminal device can choose to perform the computing task locally, or it can choose to transmit the computing task wirelessly to the base station, and then transmit it to the mobile edge server for execution. Finally, the base station returns the result processed by the mobile edge server to the mobile terminal device.

This paper assumes that computing tasks generated by all mobile end-user devices are indivisible, either executed locally or offloaded to mobile edge servers, and the computing resources of each mobile end-user device are the same.

3.1 Local Computing Model

Local computing refers to the task of computing on a mobile terminal device, which is completed independently by the mobile terminal device. The local computing model refers to that the computing tasks generated by the mobile terminal devices are directly executed in the mobile terminal devices. If the mobile terminal user's device U_i chooses to execute the computing task $Task_i$ locally, then the local computing delay T_i^L of the mobile terminal user's device U_i is defined as:

$$T_i^L = \frac{C_i}{f_i} \tag{1}$$

Where C_i is the number of CPU cycles required to complete $Task_i$, and f_i is the computing power of user device i. Local computing energy consumption E_i^{local} can be expressed as:

$$E_i^{local} = \mathrm{k} f_i^2 C_i \tag{2}$$

Where k represents the energy consumption constant [11].

3.2 Unloading Calculation Model

The unloading computing model refers to the unloading of computing tasks generated by mobile terminal devices to the edge server for execution. In this mode, computing tasks generated by mobile terminal devices are first transmitted from the mobile terminal devices to the base station, and then the base station sends the task data to the mobile edge server, and then the edge server is moved for data processing. Finally, the base station feeds the results back to the mobile end-user device. Therefore, the delay of unloading the calculation model consists of three parts: first, the upstream transmission delay of transferring tasks to the edge server; second, the computing delay of processing tasks by the edge server; and finally, the downstream transmission delay of mobile the edge server to feed the calculation results back to the user. Since the amount of data returned by the server after calculation is usually small, the delay in unloading the calculation model is not enough. Therefore, the downstream transmission delay is negligible [12].

Assuming that the initial position of the mobile device remains unchanged in the scene, the transmission rate r_i from the device U_i to the base station can be expressed as [13] :

$$r_i = B_i \log_2(1 + \frac{q_1 h_1}{N_0}) \tag{3}$$

Where: B_i is the bandwidth of terminal device i; q_1 is the transmission power of the terminal equipment;h_1 indicates path loss; N_0 is the channel noise power.

Define the transmission delay T_i^{up} on U_i as:

$$T_i^{up} = \frac{D_i}{r_i} \tag{4}$$

Where D_i is the size of the input data required for the calculation task $Task_i$, r_i is the uplink rate of user device i in the wireless channel of the system model, then the energy consumed by the mobile terminal device in the transmission process E_i^{up} can be expressed as:

$$E_i^{up} = \mathrm{p} * T_i^{up} \tag{5}$$

The computing delay T_i^{com} of a mobile edge server is defined as:

$$T_i^{com} = \frac{C_i}{F_i} \tag{6}$$

F_i is used by the mobile edge server to calculate the computing resources allocated by $Task_i$. During task processing, the mobile terminal device needs to maintain a certain receiving power p_w and wait for the return of the calculation result. In this case, the energy consumption W_i^{wait} of the device can be expressed as：

$$E_i^{wait} = P_w * T_i^{com} \tag{7}$$

Then the total delay Tidownload of uninstalling the calculation model is:

$$T_i^{download} = T_i^{up} + T_i^{com} \tag{8}$$

4 Simulation Results and Analysis

In this section, this article will introduce our experimental preparation work, experimental results, and result analysis.

Firstly, this article uses a real dataset from Melbourne, Australia. The description is a real dataset of the geographical location relationship between user terminal devices and edge-side base stations near CBD. This dataset contains the latitude and longitude position relationships of 816 user terminal devices and 125 edge-side base stations. With the latitude and longitude of user terminal devices and edge servers (assuming one edge server is deployed on each base station side), the distance between each user terminal device and edge server can be calculated by writing a program using relevant formulas. Then, based on the distance between them and the coverage radius of the base station signal, the set of edge servers that can be uninstalled by the user terminal equipment can be filtered out, and the computing tasks generated by the end-user equipment can be offloaded to the optional server set for processing and calculation.

This article assumes that the effective signal coverage radius of each base station on the edge side is 200m, and the position of mobile terminal equipment remains unchanged during task offloading. The distribution and relationship between the end user equipment and edge servers in this dataset are calculated as shown in Figure 3.

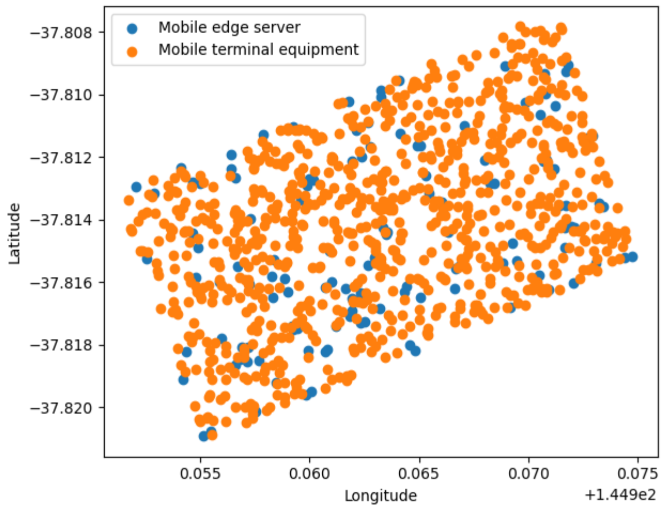

Figure 3: User and server location map

4.1 Experimental Parameter Setting

The experimental platform of this paper is Python3.8.12 based on Windows 11. All the important parameters involved in the experiment are shown in Table 1 [15].

Argument	Significance	Value
m	The total number of mobile terminal devices	816
n	The total number of mobile edge servers	125
D_i/KB	Task data size	Unif(500,1000)
C_i/cycles	Number of CPU cycles required to process the task	D_i×500×1024
f_i/GHz	Computing capability of the local device	1

B_i/MHz	bandwidth	5
q_1/mW	Mobile terminal devices transmit power	100
h_1	Path loss index	1.6×10^{-7}
N_0/Watts	Channel noise power	10^{-7}
F_i/GHz	Server computing resources	Unif(5， 10)

Table 1: Experimental parameters

4.2 Experimental Result

To verify the effect of load balancing, this paper will compare the two situations of the local processing of tasks generated by mobile terminal devices and unloading to the edge server:

Figure 4 shows the effect diagram of the local calculation delay of tasks generated by mobile terminal devices, and the average delay is about 0.38144 seconds.

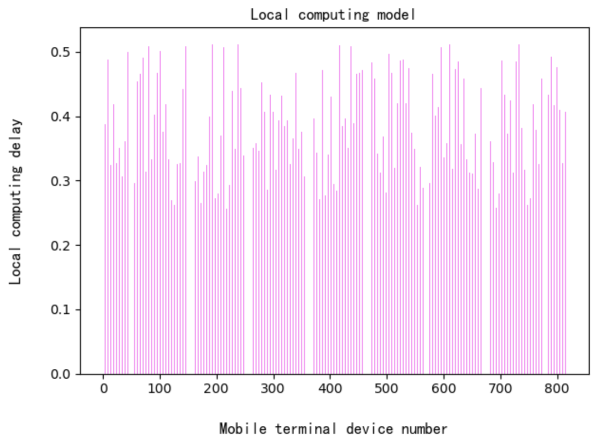

Figure 4: Local computing delay

Under the uninstallation calculation model, if the static random algorithm is used for load balancing unloading, task computation delay = task upload delay + edge server computation delay. The average computing delay of the task uninstalled to the edge server for processing is about 0.12782 seconds, which is much smaller than the average local computing delay of 0.38144 seconds.

If the dynamic minimum connection algorithm is used for load balancing unloading, task computation delay = task upload delay + edge server computation delay. The average computing delay of the task uninstalled to the edge server for processing is about 0.12894 seconds, which is much smaller than the average local computing delay of 0.38144 seconds.

The experimental results of two resource scheduling algorithms are shown in Figure 5. By comparing Figure 4 and Figure 5, it is evident that offloading computing tasks from mobile terminal devices to edge computing can significantly reduce the latency.

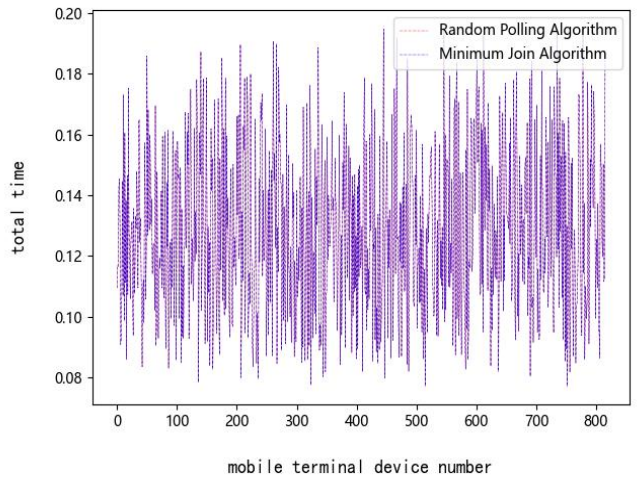

Figure 5: Unloading Calculation Model-task computation delay

By simply comparing the latency of local computing and offloading of terminal device tasks to mobile edge computing, the advantages and disadvantages of the two algorithms cannot be seen. Therefore this article also verified the superiority of the two resource scheduling algorithms mentioned earlier, and the results are shown in Figure 6:

The load effect of the random polling edge server shows that the average number of calls to the edge server is 6.528 times, and the variance is about 11.577.

The load-balancing effect of the edge server of the minimum connection algorithm shows that the average number of calls to the edge server is 6.528 times, and the variance is about 0.889.

By comparing these two resource scheduling algorithms, it can be seen that even if the average number of calls to the server is the same, there is a significant difference in the variance between the two. The variance of the dynamic minimum connection algorithm is much smaller than that of the static polling algorithm.

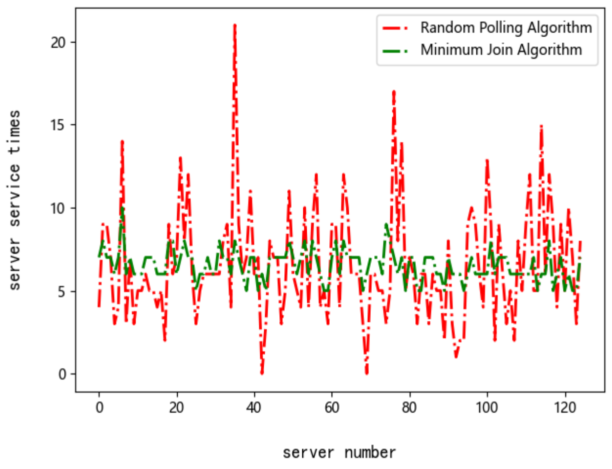

Figure 6: Server load condition

4.3 Experimental Analysis

The simulation results show that unloading the computing tasks of mobile terminal devices to the edge server for processing can greatly reduce the computing delay and improve the user's quality experience. At the same time, to achieve a reasonable allocation of computing resources at the edge and achieve a better load-balancing effect, this paper adopted the dynamic minimum join algorithm and the static random polling algorithm to allocate computing resources at the edge. By comparing the variance of the two methods, this paper can see that the volatility of the dynamic minimum connection algorithm is significantly lower than that of the static random polling algorithm. So the dynamic minimum join algorithm is significantly better than the static random polling algorithm in resource allocation. Therefore, if a dynamic minimum connection algorithm is adopted in the application scenario proposed in this article, it can better achieve load balancing of edge computing resources.

5 Conclusions

This article investigates the resource allocation problem at the edge in a multi-mobile terminal device and multi-edge server environment, to reduce latency and improve edge resource utilization.

The dynamic minimum connection algorithm and static polling algorithm were tested separately, and both showed significant effects in reducing mobile device latency. However, this article also considered the utilization of edge resources. In scenarios with multiple users and servers, the coverage of base station signals usually overlaps, leading to multiple choices for users. If the selection is reasonable, it can greatly improve the utilization of edge resources.

In future research, our team will consider the issues of multi-user mobility and multi-edge server collaboration. The user's movement will cause the mobile terminal device to reselect the target server for uninstallation, which in turn requires further consideration of edge resource allocation issues. The collaboration between edge servers will further improve the utilization of edge resources. In addition, our team will also add an optimization goal, which is the issue of energy consumption. Combined with the current environmental protection theme, while making reasonable use of computing resources, our team can achieve the optimization goal of saving energy consumption.

6 Acknowledgments

This work is supported by the National Natural Science Key Foundation of China grant No. 62067003, No.62262030, and No.62363015, the Jiangxi Provincial Natural Science Foundation Under Grant No.20224BAB212015, the Jiangxi Provincial 03 Special Project and 5G Project (20224ABC03A13, 20232ABC03A26), and the Foundation of Jiangxi Educational Committee Under Grant No. GJJ210338.

References

[1] Y. C. Hu, M. Patel, D. Sabella, et al. (2015). Mobile edge computing—A key technology towards 5G. E TSI white paper, vol. 11, no. 11, pp. 1-16.

[2] F. Guo, H. Zhang, H. Ji, X. Li and V. C. M. Leung. (2018, Dec). An Efficient Computation Offloading Management Scheme in the Densely Deployed Small Cell Networks With Mobile Edge Computing. in I

EEE/ACM Transactions on Networking, vol. 26, no. 6, pp. 2651-2664, doi: 10.1109/TNET.2018.2873002.

[3] J. Kwak, Y. Kim, J. Lee and S. Chong. (2015, Dec). DREAM: Dynamic Resource and Task Allocation for Energy Minimization in Mobile Cloud Systems. in IEEE Journal on Selected Areas in Communications, vol. 33, no. 12, pp. 2510-2523, Dec. 2015, doi: 10.1109/JSAC.2015.2478718.

[4] B. Luo, B. Yu. (2020). Computational offloading Strategy based on particle swarm Optimization in mobile edge Computing, Journal of Computer Application, vol. 40, no. 08, pp. 2293-2298.

[5] J. Liu, Y. Mao, J. Zhang, and K. B. Letaief. (2016). Delay-optimal computation task scheduling for mobile-edge computing systems. 2016 IEEE International Symposium on Information Theory (ISIT), Barcelona, Spain, pp. 1451-1455, doi: 10.1109/ISIT.2016.7541539.

[6] W. H. Fan, L. Yao, J. T. Han, et al. (2021). Game-based multi-type task offloading among mobile-edgecomputing-enabled base stations [J]. IEEE Internet of Things Journal, vol. 8, no. 24, pp. 17691-17704.

[7] T. Alfakih, M. M. Hassan, A. Gumaei, C. Savaglio and G. Fortino. (2020). Task Offloading and Resource Allocation for Mobile Edge Computing by Deep Reinforcement Learning Based on SARSA. in IEEE Access, vol. 8, pp. 54074-54084, doi: 10.1109/ACCESS.2020.2981434.

[8] Z. F. Kuang, Q. L. Chen, L. F. Li, X. H. Deng, Z. G. Chen. (2022). Multi-user Edge Computing Task Offloading Scheduling and Resource Allocation Based on Deep Reinforcement Learning [J]. Chinese Journal of Computers, vol. 45, no. 04, pp. 812-824.

[9] X. Chen, L. Jiao, W. Li and X. Fu. (2016, Oct). Efficient Multi-User Computation Offloading for Mobile-Edge Cloud Computing," in IEEE/ACM Transactions on Networking, vol. 24, no. 5, pp. 2795-2808, doi: 10.1109/TNET.2015.2487344.

[10] H. Cao, J. Cai. (2018, Jan). Distributed Multiuser Computation Offloading for Cloudlet-Based Mobile Cloud Computing: A Game-Theoretic Machine Learning Approach. in IEEE Transactions on Vehicular Technology, vol. 67, no. 1, pp. 752-764, doi: 10.1109/TVT.2017.2740724.

[11] Z. Tong, X. M. Deng, F. Ye, et al. (2020). Adaptive computation offloading and resource allocation strategy in a mobile edgecomputing environment [J]. Information sciences，537：116-131.

[12] P. Zhao, H. Tian, C. Qin and G. Nie. (2017). Energy-Saving Offloading by Jointly Allocating Radio and Computational Resources for Mobile Edge Computing. in IEEE Access, vol. 5, pp. 11255-11268, doi: 10.1109/ACCESS.2017.2710056.

[13] X. H. Wang, L. C. Wang, X. B. Zhang. (2022). Computing resource allocation and Partial task Unloading Algorithm based on Edge computing [J]. Information Recording Materials, vol. 23, no.10, pp. 223-226,doi:10.16009/j.cnki.cn13-1295/tq.2022.10.072.

[14] P. Lai, Q. He, Abdelrazek M, et al. (2019). Optimal Edge User Allocation in Edge Computing with Variable Sized Vector Bin Packing[J]. doi:10.1007/978-3-030-03596-9_15.

[15] H. Gu, M. J. Zhang, Y Pan. (2023). Optimization of task offloading and resource allocation algorithms used in mobile edge computing. Modern Electronics Technique, vol.46, no.07, pp. 67-72, 2023, doi:10.16652/j.issn.1004-373x.2023.07.014.