International Journal of Latest Engineering Research and Applications (IJLERA) ISSN: 2455-7137

Volume – 07, Issue – 01, January 2022, PP – 06-12

Research on Resource Allocation and Task Offloading Algorithms in Mobile Edge Computing Based on Reinforcement Learning

Xue Tang[1], Zhan Wen[1*], Yihan Shi[1], Xinyuan Zhang

[1](College of Communication Engineer, Chengdu University of Information Technology, Chengdu, China)

Xue Tang xuetangshocker@foxmail.com; Zhan Wen[*]wenzhan@cuit.edu.cn; Xinyuan Zhang xinyuanzhangputao@foxmail.com

Abstract: Mobile Edge Computing (MEC) will sink the services located in the cloud computing center to the edge of the mobile network, thereby reducing power consumption and latency. How to allocate services to the server and mobiles to maximize system efficiency is an important issue in MEC. This paper proposes a Resource Allocation and Task Offloading algorithm based on Reinforcement Learning (RATORL) in a multi-user and single server application scenario. It takes the total cost of the user as the final optimization goal. According to the user's task characteristics and server status, the reinforcement learning method was used to optimize the allocation of computing resources and task offloading. The simulation results show the proposed RATORL algorithm can effectively reduce the user's overhead in different system states compared with the traditional algorithms.

Keywords: Mobile edge computing, Resource allocation, Task offloading, Machine Learning, Reinforcement learning

I. INTRODUCTION

With the rapid development and popularization of mobile terminal equipment, its processing and storage capabilities continue to grow, giving birth to many new applications [1], such as face recognition, augmented reality, and mobile medical care [2]. The traditional cloud computing service model is difficult to meet the low latency requirements of these applications. In 2014, the European Telecommunications Standards Association merged edge computing with mobile networks from the perspective of network architecture, committed to building a more intelligent network, and proposed Mobile Edge Computing (MEC) for the first time, defining it as a mobile network The edge provides IT service environment and cloud computing capabilities [3]. MEC is a brand-new distributed computing method based on mobile communication networks. Through the cloud service environment built on the Radio Access Network (RAN) side, certain network services and network functions can be separated from the core network and reduce Time delay, energy consumption, improve user experience, and enhance privacy protection.

At present, a large number of scholars and research institutions at home and abroad have conducted in-depth research on mobile edge computing. Yang T et al. [4] considered an edge computing network composed of MEC node and N mobile users and used deep reinforcement learning to optimize the allocation of computing resources. The experimental results show that it has a significant effect in reducing latency. According to the different delay requirements, Liu J et al. [5] considered the task offloading problem under the conditions of sufficient time and time shortage and adopted the Markov decision process method to find the optimal task scheduling strategy. Ning Z et al. [6] studied the delay problem of delay-sensitive applications in mobile edge computing. By formulating the user computing offload problem as a mixed integer linear programming problem, they designed an iterative heuristic mobile edge computing resource allocation algorithm To dynamically make unloading decisions.

The development of mobile terminals has always been limited by battery technology, unable to process extremely complex tasks with high power for a long time. MEC offloads the computing tasks of mobile terminals to a nearby edge server for execution to alleviate the power problem of mobile terminals. Cao S et al. [7] proposed an energy-optimized mobile computing offloading algorithm for the problem of mobile computing offloading in heterogeneous networks, which can achieve the maximum energy saving of mobile terminals under a given application execution time requirement. Labidi W et al. [8] proposed a joint optimization framework for wireless resource scheduling and computing offloading based on the long-term evolution of small cells. The ultimate optimization goal is to minimize the average energy consumption of the user's local terminal. They studied offline dynamic planning methods. Two solutions, deterministic and random, are designed to find the best radio scheduling offloading strategy, and it is proved that the dynamic offline strategy can achieve the best energy efficiency on the mobile terminal. Zhang K et al. [9] studied the energy-saving

computing offloading mechanism of mobile edge computing in 5G heterogeneous networks and proposed a joint optimization offloading and radio resource allocation scheme to obtain the minimum offloading system energy consumption under delay constraints. It also considers the energy cost of task calculation and file transfer.Chen MH et al. [10] considered a network scenario consisting of an edge server and a user with N independent subtasks, to optimize the user's offloading decision, and minimize the total cost of energy, computing, and delay. This problem can be expressed as a non-convex quadratic constrained quadratic programming NP-hard problem. Through semi-definite relaxation and a new randomized mapping method, an effective unloading decision algorithm is proposed to improve the moving edge Calculated performance. Compared with the literature [10], Chen MH et al. [11] studied a multi-user mobile cloud computing system consisting of a computing access node and a remote cloud server. The communication resources and computing resources of all users and computing access nodes are used as offloading. To minimize the overall energy consumption and time delay of the system, through the analysis of the problem and express it as a non-convex quadratic constrained NP-hard problem, the semi-deterministic relaxation and a novel random mapping are finally proposed. Effective solution. Simulation results show that the proposed algorithm can provide near-optimal performance with only a few random iterations. In the literature [12], the author studied the compromise solution of the two key but conflicting problems of the power consumption of mobile devices and the execution delay of computing tasks in the multi-user mobile edge computing system and proposed a solution based on Lyapunov. Online algorithms to implement local task execution and offloading strategies.

Aiming at the multi-user and single-edge server network model, this paper takes the total cost of user equipment as the final optimization goal, and proposes a resource allocation and task offloading algorithm based on reinforcement learning (RATORL), and sets the total cost as the reinforcement learning algorithm. State, resource allocation, and task offloading are actions of reinforcement learning, through iterative learning, to find out the resource allocation and task offloading schemes suitable for the current system environment. Experimental results show that under the system model of this article, compared with other traditional algorithms, the proposed RATORL algorithm can reduce system overhead under different server resources and task data volumes.

II. SYSTEM MODEL

This article proposes an application scenario that includes a Base Station (BS) and M Mobile Devices (MD), and the edge server is directly embedded in the base station. As shown in Figure 1.

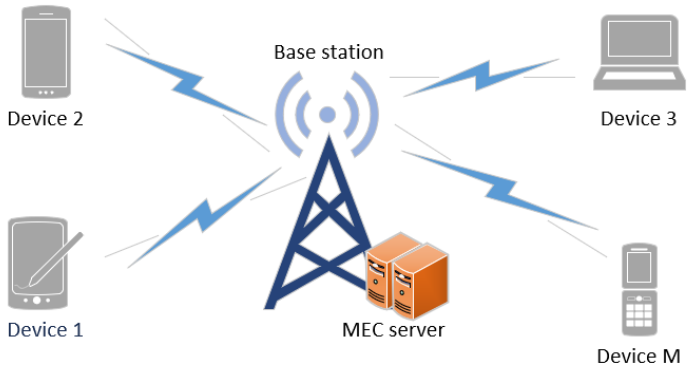

Fig. 1 Network Model

In this paper, $\mathcal{M} = \{1,2,3,\dots,M\}$ is expressed as a set of MD. We assume that each MD has only one computationally complex task to be processed, and the task of the mobile device MD_i can be expressed as $A_i \triangleq (D_i, C_i)$. Among them, D_i represents the size of input data related to the task A_i (such as code or input parameters); C_i represents the total number of calculation cycles required to complete the task A_i, C_i and D_i are positively correlated.

MD tasks can be processed in two ways, either locally or offloaded to the edge server for processing. We define the task processing decision vector of the entire MEC system$\partial = \{\partial_1, \partial_2, \dots \partial_N\}$, where $\partial \in \{0,1\}$. When $\partial_i = 0$, it means that MD_i selects to process the task locally, and $\partial_i = 1$ means that MD_i offloads the task to the

edge server for processing.

If MD_i chooses to process locally and submit the task to the processing center of the device for processing. We define f_i^l as the computing power of MD_i, and the unit is the number of CPU cycles. The time delay T_i^l cost in local processing of task A_i can be obtained as:

$$T_i^l = \frac{C_i}{f_i^l} \ , \ \forall \mathrm{i} \in \mathcal{M} \quad (1)$$

We definep_l as the computational power of MD_i, and according to the local processing delay, the energy consumption E_i^l generated by the local processing of task A_i can be obtained as:

$$E_i^l = p_l \cdot T_i^l \ , \quad \forall \mathrm{i} \in \mathcal{M} \quad (2)$$

If MD_i selects offloading, the task needs to be uploaded to the server and the server allocates certain computing resources. We define f_i^s as the computing resources allocated by the server to MD_i, and the unit is CPU cycles. We define the signal-to-noise ratio of the communication between MD_i and the server as σ_i, and the bandwidth allocated by the server to MD_i is W_i. According to Shannon's theorem, the upload rate r_i of MD_i is:

$$\mathrm{r}_i = W_i \log_2(1 + \sigma_i) \ , \quad \forall \mathrm{i} \in \mathcal{M} \quad (3)$$

From this, it can be calculated that the time T_i^{up} taken by MD_i to upload the task is:

$$T_i^{up} = \frac{D_i}{\mathrm{r}_i} \ , \ \forall \mathrm{i} \in \mathcal{M} \quad (4)$$

The time delay T_i^s spent by the server processing task A_i is:

$$T_i^s = \frac{C_i}{f_i^s} \ , \ \forall \mathrm{i} \in \mathcal{M} \quad (5)$$

During the unloading process, the MD_i upload task consumes energy. After the task is processed, MD_i receives and returns the results also consumes energy. But the amount of returned result data is much smaller than the upload time, so MD_i is not considered Delay and energy consumption of receiving results [13]. We define p_{up} as the upload power of MD_i, and the energy consumption E_i^{up} generated by the MD_i upload task is:

$$E_i^{up} = p_{up} \cdot T_i^{up} \ , \quad \forall \mathrm{i} \in \mathcal{M} \quad (6)$$

Therefore, we can get the total time delay T_i^{off} and energy consumption E_i^{off} required for MD_i to select unloading processing as:

$$T_i^{off} = T_i^{up} + T_i^s = \frac{D_i}{\mathrm{r}_i} + \frac{C_i}{f_i^s} \quad (7)$$

$$E_i^{off} = E_i^{up} = p_{up} \cdot \frac{D_i}{\mathrm{r}_i} \ , \quad \forall \mathrm{i} \in \mathcal{M} \quad (8)$$

Finally, we define the weighted sum of delay and energy consumption as the total cost of the user and set the optimization goal to minimize the total cost of the user under the condition of limited computing resources and communication resources. ω_t and ω_e respectively represent the delay weight and energy consumption weight. When ω_t is larger, it means that the current user is delay-sensitive, otherwise, it means that the current user is more concerned about energy consumption. The final optimization problem is as follows:

$$min_{\partial,F}\, G_{all} = \sum_{i=1}^{M} (1 - \partial_i)\left(\omega_t T_i^l + \omega_e E_i^l\right) + \partial_i\left(\omega_t T_i^{off} + \omega_e E_i^{off}\right) \quad (9)$$

III. ALGORITHM MODELING AND DESIGN

Reinforcement learning is a type of machine learning, among which Q-Learning is a classic value-based reinforcement learning algorithm. This paper analyzes the proposed edge computing network model, takes time delay and energy consumption weighting as the final optimization goal, and uses reinforcement learning methods to optimize the offloading decision and the server's computing resource allocation. Reinforcement learning has three core elements: state, action, and reward. The agent (MEC system), through repeated learning, will select actions based on the current state and record the rewards obtained. When in this state again, it can select better actions to execute to get more rewards. Combined with the reinforcement learning of this article, the definition is as follows:

Status: The total cost of all current users, represented by G_{all}.

Action: The action in this article consists of two parts, namely the user's offloading decision vector $\partial = \{\partial_1, \partial_2, \dots \partial_N\}$ and the server's computing resource allocation vector $F = \{f_1^s, f_2^s, \dots f_N^s\}$. Therefore, we can combine them to get the action$\{\partial_1, \partial_2, \dots \partial_N, f_1^s, f_2^s, \dots f_N^s\}$.

Reward: To minimize the total cost G_{all} of the users used, we should associate the reward with the optimization goal. The ultimate goal of reinforcement learning is to maximize the reward, so the reward is negatively related to the optimization goal, so we define the reward as $\mathrm{R}(s,a) = \frac{\mathrm{G}^l - \mathrm{G}(s,a)}{\mathrm{G}^l}$, where G^l represents the total system overhead that all tasks are executed locally, $\mathrm{G}(s,a)$ represents the total system delay energy

consumption caused by the execution of action a in the current state s, the larger $R(s,a)$, It shows that the current system overhead is smaller.

The key of Q-Learning is value learning, that is, different actions are scored and recorded as Q_table. The rows of Q_table represent the action space A, the columns represent the state space S, and each specific value $Q(\mathrm{s_t}, \mathrm{a_t})$ Represents the expectation that the Agent can obtain rewards when performing the action $a_t(a_t \in A)$ in the state of $s_t(s_t \in \mathrm{S})$. After the agent executes the action, it will reach the next state s_{t+1} and perform a new round of iteration. The update of Q_table is through formula (10):

$$\mathrm{Q}(s_t, a_t) \leftarrow (1-\alpha)\mathrm{Q}(s_t, a_t) + \alpha\left[\mathrm{R}(s_t, a_t) + \gamma \max_{a_{t+1}} \mathrm{Q}(s_{t+1}, a_{t+1})\right] \quad (10)$$

Among them, s_t represents the current state of the Agent; a_t represents the current action of the Agent; $R(s_t, a_t)$ represents the reward obtained by the Agent performing the action a_t in the state s_t; s_{t+1} represents the next state of the Agent; a_{t+1} represents the next action of the Agent; α represents the learning rate and $\alpha \in (0,1)$, its size determines how much Q value is learned this time, γ represents the discount factor, and $\gamma \in (0,1)$, The larger the value of γ, the more long-term rewards the Agent values, on the contrary, the more attention it is to immediate rewards. If the agent chooses the largest Q each time, the algorithm will easily fall into the local optimum. This paper uses the Q-Learning method based on $\varepsilon - greedy$ search strategy to solve the optimization problem. That is, every time the agent chooses Q, there is a probability of $(1-\varepsilon)$ to choose the action with the largest Q value, and the probability of ε to randomly choose other actions.

IV. SIMULATION RESULTS AND DISCUSSIONS

To ensure a high-quality product, diagrams and lettering MUST be either computer-drafted or drawn using India ink.

In this section, we use the Python programming simulation method to build a simulation scenario containing multiple mobile devices and an edge server as shown in Figure 1. The simulation parameters are listed in TableI.

Table II. Simulation parameters.

Parameter	Value
The total bandwidth of the server	10 MHz
The computing resource of the server	5 GHz
The local computing resource of MD	1 GHz
SNR	[5,25] dB
Transmit power	0.1 W
$\boldsymbol{\omega_t}$	0.5

The total bandwidth of the server is 10 MHz, the computing resource of the server is 5 GHz, the local computing resource of the mobile device is 1 GHz, the signal-to-noise ratio of the communication between the mobile device and the server is [5,25] dB, and the mobile device's emission The power is 0.1 W, the learning rate and discount factor of reinforcement learning are 0.01 and 0.9, and both the delay weight ω_t and the energy consumption weight ω_e are set to 0.5.

To verify the effectiveness of the proposed algorithm, we have compared with the following three algorithms: Random selection algorithm (Random), that is, mobile devices randomly select an offloading method to process tasks; All local algorithm (all_local), all mobile devices select Keep tasks on the local device for execution; the full server algorithm (all_ser) means that all mobile devices will offload all tasks to the edge server for processing regardless of network conditions and computing resources. At this time, the computing resources of the server will be shared equally Every device.

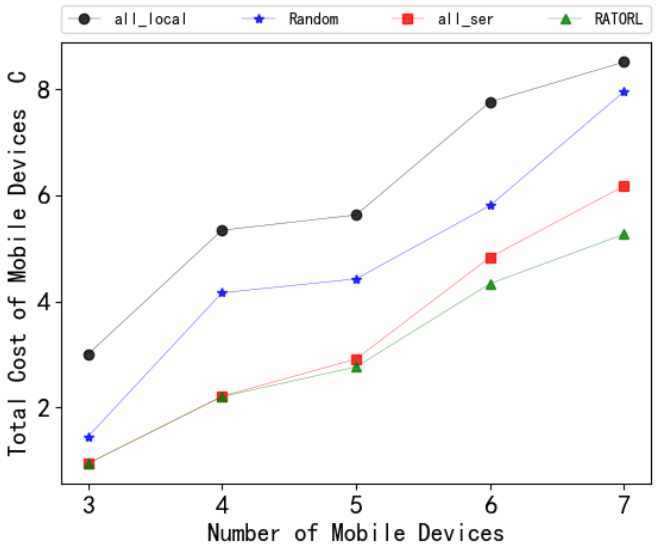

Fig. 2 The relationship between the number of mobile devices and the total cost

As shown in Figure 2, the abscissa represents the number of mobile devices, and the ordinate represents the total overhead of all mobile devices. The user's data size ranges from 300Kbits to 500Kbits. With the increase in the number of users, the total overhead of the four algorithms is showing an increasing trend. Among them, the all-local algorithm retains all tasks for processing locally, and the computing resources of the local device are less than the computing resources of the server, resulting in the worst performance when the number of users increases. The performance of the full server algorithm is better than the random algorithm and the full local algorithm. Because the energy consumption of the local device when processing tasks is higher than the energy consumption of the device to send data, and the computing power of the server is stronger than that of the local device, when the number of users is 3 to 5 , The performance gap between all-local and random is still relatively large, but when the number of users exceeds 5, the computing resources allocated to each device decreases, resulting in a reduction in the gap between all-local and random. When the number of users is 3 to 5, the performance of the RATORL algorithm is similar to that of the full offload algorithm, because when the number of users is small, each user can be allocated more computing resources; when the number of users exceeds 5, although the local device calculates The power consumption is higher than the power consumption of sending data, but uploading all user tasks to the server for processing results in longer delays, and ultimately still leads to high system overhead. The RATORL algorithm selectively retains some user tasks when there are many users. Local processing effectively reduces the overall overhead of the system.

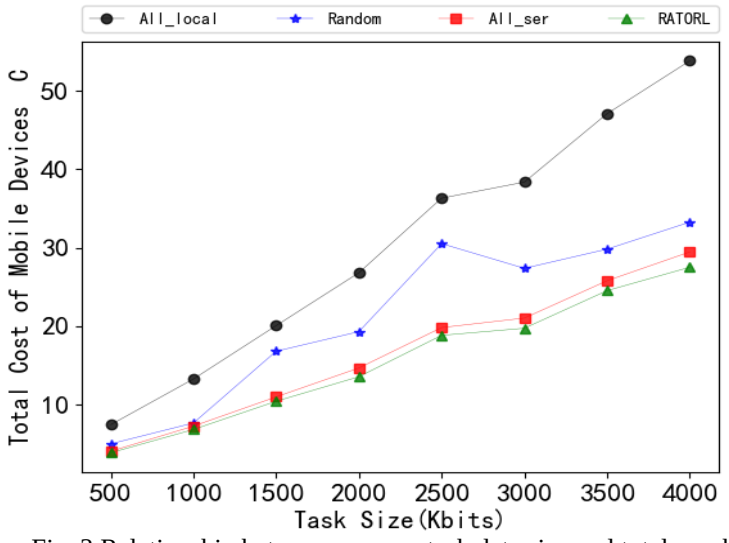

Fig. 3 Relationship between average task data size and total overhead

As shown in Figure 3, the abscissa represents the number of mobile devices, and the ordinate represents the total overhead of all mobile devices, where the number of user devices is 5. As the amount of task data gradually increases, the total equipment overhead is showing an upward trend. Among them, the performance of the all-local algorithm is the worst, because the amount of data and the amount of calculation are positively correlated, the local device has low computing power and the calculation power is higher than the transmission power; the random algorithm is the processing method of the random decision-making device, and the device is processed locally or offloaded. All are possible, so the performance is slightly better than the all-local algorithm. When the average task data size is 500Kbits to 1500Kbits, the performance of the full offloading algorithm is similar to that of the RATORL algorithm; as the amount of data increases, the edge server load increases. Because the RATORL algorithm combines the local device and the edge server, the total overhead of the mobile device is required. Lower than the full uninstall algorithm.

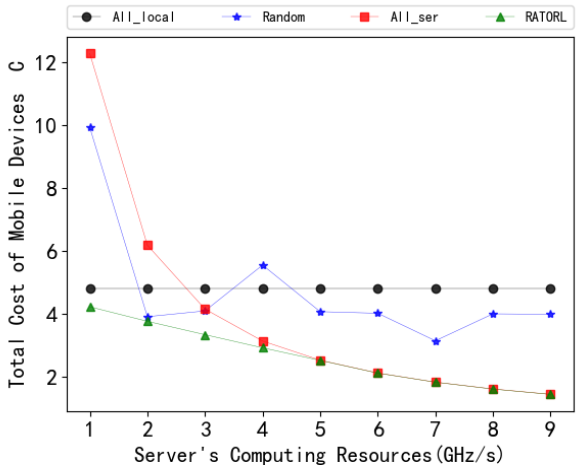

Fig. 4 The relationship between server computing resources and total cost

The abscissa in Figure 4 represents the total amount of server computing resources, and the ordinate represents the total overhead of mobile devices, and the number of user devices is fixed at 5. The all-local algorithm retains all tasks for local processing, so the computing resources of the server do not affect the total overhead. When the server computing resources are between 1Ghz/s and 3Ghz/s, the overall performance of the full uninstall algorithm is the worst, because when the server computing resources are small, each user who is uninstalled to the server can get fewer computing resources, resulting in total It's expensive. With the increase of server computing resources, when the server computing resources range from 3Ghz/s to 9Ghz/s, the full offloading algorithm has a better performance in reducing the total equipment overhead, but in general, the RATORL algorithm performs best overall, Because he selectively offloads some users to the server for processing according to the capacity changes of the server's computing resources, making better use of the computing resources of local equipment and edge servers.

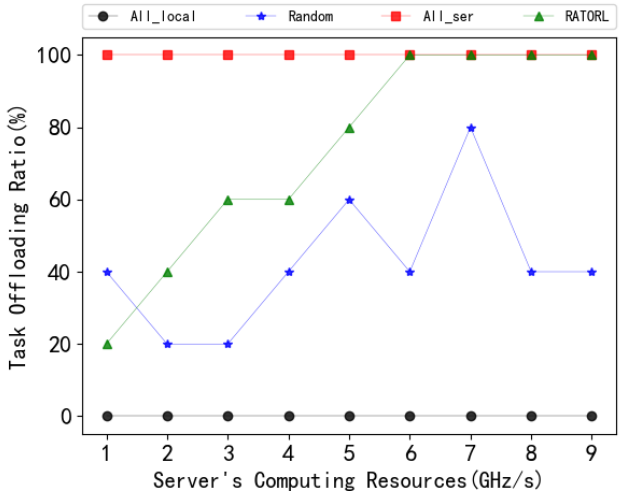

Fig. 5 The relationship between server computing resources and offload ratio

As shown in Figure 5, the abscissa represents the total amount of server computing resources, and the ordinate represents the unloading ratio of mobile device tasks. The number of user devices is fixed at 5. It can be seen that the four algorithms are different in the case of different server computing resources. Task handling method. When the server computing resources are low, the RATORL algorithm tends to keep the tasks on the local device for processing. However, as the server computing resources increase, the RATORL algorithm offloads more tasks to the edge server for processing.

V. CONCLUSION

For mobile edge computing, computing resources and communication resources are very precious. Effective task offloading and resource allocation strategies can not only improve resource utilization, but also improve user experience. Therefore, based on the scenario of multi-user equipment and single mobile edge server, this paper proposes a resource allocation and task offloading algorithm (RATORL) based on reinforcement learning. Through the comparison of experimental results, the RATORL algorithm is compared with other traditional algorithms in terms of reducing user equipment overhead. For better performance.

REFERENCES

[1] Quality, Reliability, Security and Robustness in Heterogeneous Systems: 15th EAI International Conference, QShine 2019, Shenzhen, China, Nvember 22–23, 2019, Proceedings[M]. Springer Nature, 2020.

[2] T. Taleb, K. Samdanis, B. Mada, H. Flinck, S. Dutta, and D. Sabella, "On multi-access edge computing: A survey of the emerging 5G network edge cloud architecture and orchestration," IEEE Commun. Surveys Tuts., vol. 19, no. 3, pp. 1657_1681, 3rd Quart., 2017.

[3] Porambage P, Okwuibe J, Liyanage M, et al. Survey on multi-access edge computing for internet of things realization[J]. IEEE Communicaions Surveys & Tutorials, 2018, 20(4): 2961-2991.

[4] Yang T, Hu Y, Gursoy M C, et al. Deep reinforcement learning based resource allocation in low latency edge computing networks[C]//2018 15th International Symposium on Wireless Communication Systems (ISWCS). IEEE, 2018: 1-5.

[5] Liu J, Mao Y, Zhang J, et al. Delay-optimal computation task scheduling for mobile-edge computing systems[C]//2016 IEEE international symposium on information theory (ISIT). IEEE, 2016: 1451-1455.

[6] Ning Z, Dong P, Kong X, et al. A cooperative partial computation offloading scheme for mobile edge computing enabled Internet of Things[J]. IEEE Internet of Things Journal, 2018, 6(3): 4804-4814.

[7] Cao S, Tao X, Hou Y, et al. An energy-optimal offloading algorithm of mobile computing based on HetNets[C]//2015 International Conference on Connected Vehicles and Expo (ICCVE). IEEE, 2015: 254-258.1

[8] Labidi W, Sarkiss M, Kamoun M. Energy-optimal resource scheduling and computation offloading in small cell networks[C]//2015 22nd international conference on telecommunications (ICT). IEEE, 2015: 313-318. 7

[9] Zhang K, Mao Y, Leng S, et al. Energy-efficient offloading for mobile edge computing in 5G heterogeneous networks[J]. IEEE access, 2016, 4: 5896-5907. 1

[10] Chen M H, Liang B, Dong M. A semidefinite relaxation approach to mobile cloud offloading with computing access point[C]//2015 IEEE 16th International Workshop on Signal Processing Advances in Wireless Communications (SPAWC). IEEE, 2015: 186-190.12

[11] Chen M H, Dong M, Liang B. Joint offloading decision and resource allocation for mobile cloud with computing access point[C]//2016 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP). IEEE, 2016: 3516-3520.

[12] Mao Y, Zhang J, Song S H, et al. Power-delay tradeoff in multi-user mobile-edge computing systems[C]//2016 IEEE Global Communications Conference (GLOBECOM). IEEE, 2016: 1-6.

[13] Cheng K, Teng Y, Sun W, et al. Energy-efficient joint offloading and wireless resource allocation strategy in multi-MEC server systems[C]//2018 IEEE international conference on communications (ICC). IEEE, 2018: 1-6.

