International Journal of Innovative
Computing, Information and Control
Volume **17**, Number **4**, August **2021**

 ISSN 1349-4198
pp. **1153–1170**

RESOURCE SCHEDULING AND COMPUTING OFFLOADING STRATEGY FOR INTERNET OF THINGS IN MOBILE EDGE COMPUTING ENVIRONMENT

Weijun Lei

School of Information Engineering
Xi'an University
No. 168, South Taibai Road, Xi'an 710065, P. R. China
leiweijun123@126.com

Received January 2021; revised May 2021

Abstract. *In order to solve the problem that the computing resources and battery energy of existing mobile devices cannot meet the rapid development of computing-intensive and delay-sensitive applications, this paper proposes a resource scheduling and computing offloading strategy for Internet of Things (IoT) in a mobile edge computing environment. Firstly, the paper designs a computing offloading and resource allocation system model. The model includes Macrocell Base Station (MBS) with Mobile Edge Computing (MEC) servers deployed and Smallcell Base Station (SBS) that can be used as a relay to achieve flexible resource management and control. Then, the computing model and communication model are built according to system model. Besides, the optimization problem is established considering minimizing the system delay and energy consumption as optimization goal. Finally, introducing game theory, the potential game model is used to solve resource allocation and computing offloading problem. Mobile devices can select MEC nodes for computing offloading according to the game result. The simulation experiment of our proposed strategy is carried out based on MATLAB platform. Experimental results show that the proposed potential game equation can converge to Nash equilibrium. Moreover, when the number of mobile users is 1000, the energy consumption and delay of the proposed strategy are 240 J and 13 ms respectively, which are better than other comparative strategies.*

Keywords: MEC, Internet of Things, Resource scheduling, Computing offload, Game theory, Delay and energy consumption

1. **Introduction.** In recent years, with the development of technologies such as sensors, big data, cloud computing and intelligent information processing, IoT has ushered in a golden period of development [1]. The development of IoT has brought unprecedented changes to information technology industry, and also posed new challenges to the traditional computing model. The popularization and rapid development of IoT are inseparable from the support of computing technology. Cloud computing has become a new integration point for the integration of IoT and Internet with its super-large scale, virtualization, high reliability and convenience [2,3]. However, the explosive growth of computing data has become a key bottleneck restricting the development of cloud computing. Therefore, MEC came into being. MEC technologies directly deploy MEC Server (MECS) on base station. Thus, the mobile user equipment that needs to compute offload does not have to choose a remote cloud server, but can adopt a nearby strategy. That is, the computing load is offloaded to edge servers nearby, so that mobile users can receive short-distance

DOI: 10.24507/ijicic.17.04.1153

cloud computing services within the coverage of wireless access network, and users can enjoy high-quality network experience [4,5]. In this way, MEC extends computing resources and storage resources from the core wide area network to edge network [6].

Under MEC framework, mobile terminals can access the computing and storage resources of remote Centralized Cloud (CC) through mobile operators [7,8]. However, mobile terminals need to send data to a server far away from terminals. MEC imposes a large amount of additional load on the radio and backhaul of mobile network, resulting in a large amount of data transmission, network load and spectrum occupation. This leads to higher latency, which seriously affects the demand for large bandwidth and delay-sensitive business applications [9]. In order to solve the problem of high latency in MEC, the idea of moving cloud computing capabilities to the edge of mobile terminal users came into being. In the case of edge computing, computing power is located near mobile terminal user network topology. Compared to MEC, edge computing can provide lower latency [10]. In addition, MEC is a centralized deployment and edge computing can be deployed in a distributed manner. However, the computing power that edge computing can provide is limited, which is lower than that of MEC [11,12].

Resource scheduling and computing offloading refers to the process of uploading tasks that cannot be effectively processed on the terminal device with limited resources to the edge server in the specified wireless area for execution, and then returning the calculation results to the original terminal device after the task is executed on the edge server. Effective task offloading and resource allocation, is not only to ensure the user's QoS request, but also to minimize the energy consumption of the system, to ensure the interests of service providers, which is also a major difficulty. In the mobile wireless network environment where MEC is deployed, there are two thorny problems: whether users unload computing tasks to MEC server, and how to effectively allocate computing resources to meet the user experience. Aiming at these two problems, a resource scheduling and computing offloading strategy for IoT in an MEC environment is proposed. The innovative points for the proposed method are

1) Due to the problem of resource limitation among multiple mobile terminal users and MEC service nodes, our proposed strategy constructs a resource allocation and computing offloading model. It deploys MBS of MEC servers and the small SBS that can be used as a relay. A formulaic optimization problem is established to minimize the delay and energy consumption of MEC system as optimization goal.

2) In order to improve the performance of task offloading, the proposed strategy introduces game theory. It uses a latent game model to solve the problem of distributed task offloading, and transforms the energy optimization objective function based on time delay limitation into a latent game equation. Mobile devices select MEC nodes for task offloading according to the game result.

This paper is divided into six chapters. Chapter 1 introduces the significance of task offloading and resource allocation algorithm in the Internet of Things, and the motivation and contribution of this research. The second chapter introduces the development status of related research. Chapter 3 describes the system model and problem description. In Chapter 4, the proposed resource allocation algorithm of edge computing based on game theory is introduced in detail. Chapter 5 discusses the experimental results and evaluates the performance of the proposed method by using relevant standards. Chapter 6 shows the conclusion and prospect.

2. **Related Research.**

The basic idea of edge computing is to offload computing tasks to computing resources close to data source, so computing offloading is the focus of many researchers [13]. On the one hand, from the user's point of view, computing offloading can

be seen as a key use case. It can run new and complex applications on user devices while reducing its energy consumption [14]. On the other hand, computing offloading brings some challenges, such as selecting an appropriate task offloading model, accurately estimating energy consumption and effectively managing simultaneous offloading of multiple users [15]. Scholars on the task offloading of MEC have conducted research from many aspects, such as delay, energy efficiency and caching. Computing offloading usually considers three aspects: immediate delay optimization, energy efficiency optimization and joint cache optimization [16,17].

1) Task offloading based on time delay optimization

Aiming at the delay research of MEC, [18] aimed at the task scheduling problem with heterogeneous delay sensitivity in shared fog network, and constructed a mathematical model to capture the main characteristics of U-free network. An integer nonlinear model of task offloading is established to minimize the average response time for mobile users to offload application load to the cloud. However, the overall computing resource consumption is relatively high. [19] proposed an online offloading strategy based on Lyapunov optimization to minimize the overall completion time of application in order to solve the optimal offloading strategy problem of Internet of Vehicles in fog computing. However, communication energy consumption is relatively expensive [20]. Considering the processing time of different computing nodes and transmission cost of heterogeneous networks, [21] proposed a context-aware mixed integer programming model. This model makes decisions about offloading and scheduling offloading tasks among shared computing resources in heterogeneous mobile clouds by formulating the optimal offline optimization solution. It can minimize the overall task completion time and task energy consumption, but the transmission cost still needs to be optimized [22].

2) Task offloading based on energy efficiency optimization

Mobile devices generally have limited battery power. The emergence of new services poses new challenges to the endurance of mobile devices. MEC architecture aims to reduce the energy consumption of user equipment by migrating computationally intensive tasks to the edge of network [23,24]. In order to minimize the energy consumption of mobile devices, offload selection, wireless resource allocation and computing resource allocation are jointly optimized [25]. Aiming at more fine-grained task offloading, [26] used non-convex technology to solve the three mixed integer nonlinear programming problems of offloading decision-making, resource allocation and overclocking decision in the process of computing task offloading. It minimized system-wide computing overhead and other risks. [27] established a collaborative architecture for MEC. A node-based task scheduling optimization algorithm was proposed under this architecture. The algorithm realized the full utilization of computing resources in network by dynamically redistributing tasks between task overload nodes and nearby nodes. Due to the limited computing resources of an SBS, the user experience needs to be further optimized under the condition of a large number of users in order to better ensure service quality.

3) Task offloading based on joint cache optimization

The limited storage resources of terminal devices often affect user experience. Users can use the resources of MEC to overcome the storage limitations of their devices [28]. [29] introduced Lyapunov optimization theory to solve the problem of flexible computing resource scheduling and radio resource allocation. It combined convex decomposition method and matching game to realize trade-off between the energy efficiency of edge computing and service delay. [30] mainly studied the dynamic migration of users in computing offloading process. A computing offloading strategy based on Markov chain is proposed to predict the possible migration and access locations of users in the future. However, for

uneven computing workload in the network space, it is difficult to ensure low computing delay and high-quality service for end users [31].

3. Computing Offloading and Resource Allocation System Model.

3.1. **Network scenario analysis.** The computing offloading and resource allocation system in IoT is shown in Figure 1. There is an MBS equipped with an MEC server and an SBS equipped with an MEC server. Both MEC servers can process computing tasks in parallel. Moreover, the computing power of MEC servers equipped with MBS is stronger than that of MEC servers equipped with SBS, and the service range of MBS and SBS is the internal area of ellipse.

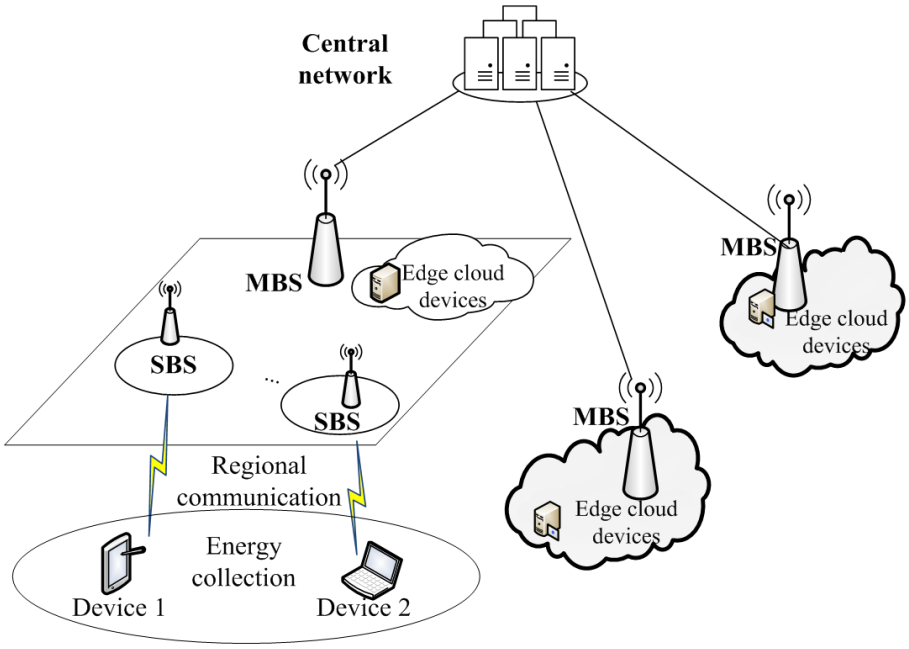

FIGURE 1. Computing offloading and resource allocation system model of IoT

Each mobile terminal user has a computing task to be processed. Mobile terminal users have three ways to process computing tasks: perform computing tasks locally; offload the computing tasks to MEC servers at MBS to complete the calculation; offload computing tasks to MEC server at SBS to complete the calculation.

There are n different mobile terminal users in the system, represented by $N = \{1, 2, \ldots, n\}$. Each mobile terminal user i has a delay-sensitive or computationally intensive task to be processed, expressed as:

$$T_i = \{D_i, C_i\}, \quad i \in N \tag{1}$$

where D_i represents the data size of task T_i to be processed, and C_i represents the number of CPU cycles required to calculate task C_i to be processed.

In the system, some mobile terminal users are running delay-sensitive applications. And another part of mobile terminal users need to save power due to insufficient power. In order to meet the different personalized needs of different mobile terminal users, the delay-energy consumption trade-off mechanism coefficients γ_i^T and γ_i^E are introduced, and the two meet following conditions:

$$\gamma_i^T + \gamma_i^E = 1 \tag{2}$$

Let $x_{i,j} = \{0,1\}$, $i \in N$ denote the decision-making mechanism variables of mobile terminal user i, where $j = \{1,2\}$ denotes the calculation method of decision. In $j = 1$, mobile terminal users choose to offload computing tasks to MEC servers at MBS. In $j = 2$, the mobile terminal user chooses to offload computing tasks to MEC servers at SBS.

Therefore, when the mobile terminal user chooses to offload computing tasks to MEC server at MBS or SBS, $x_{i,j} = 1$. Conversely, when the mobile terminal user chooses to complete the calculation locally, $x_{i,j} = 0$.

According to the above content, computing offloading decision vector can be obtained

$$\mathbf{x} = [x_{1,j}, x_{2,j}, \ldots, x_{n,j}], \quad x_{i,j} \in \{0,1\}, \ \ j \in \{1,2\} \tag{3}$$

3.2. **Computing model.** The computing model includes local computing consumption and computing offloading consumption.

3.2.1. *Local computing consumption.* Let f_i^l denote the processor computing capability of mobile terminal user i, and different mobile terminal users have different computing capabilities. Then the delay consumption of mobile terminal user i to execute computing task T_i locally is calculated as follows:

$$T_{i,e}^L = \frac{C_i}{f_i^l} \tag{4}$$

Let δ_l denote the energy consumption of one CPU cycle of mobile terminal user i, and the energy consumption of mobile terminal user i to execute computing task T_i locally is calculated as follows:

$$E_{i,e}^L = C_i \cdot \delta_l \tag{5}$$

When mobile terminal user i performs a computing task locally, since there is no other form of delay or energy consumption, the total delay consumption and energy consumption of local computing can be calculated as follows:

$$T^L(\mathbf{x}) = \sum_{i \in N, j \in \{1,2\}} (1 - x_{i,j}) \cdot \left(\frac{C_i}{f_i^l} \right) \tag{6}$$

$$E^L(\mathbf{x}) = \sum_{i \in N, j \in \{1,2\}} (1 - x_{i,j}) \cdot (C_i \cdot \delta_l) \tag{7}$$

3.2.2. *Computing offloading consumption.* When the mobile terminal user i chooses to offload computing tasks to MEC servers, the delay consumption includes two parts, namely the delay in uploading data and completing computing tasks by MEC servers [32].

The upload delay for mobile terminal user i who chooses to offload computing tasks to MEC servers is calculated as follows:

$$T_i^{up} = \frac{D_i}{r_i^u}, \quad x_{i,j} \neq 0 \tag{8}$$

where r_i^u calculates any upload rate for offloading.

Let f_{MBS}^{MEC} denote the computing power of MEC server at MBS, and f_{SBS}^{MEC} denote the computing power of MEC servers at SBS, so we can get

$$f^{MEC} = \begin{cases} f_{MBS}^{MEC}, & \text{if } j = 1 \\ f_{SBS}^{MEC}, & \text{if } j = 2 \end{cases} \tag{9}$$

Then the execution delay of mobile terminal user i choosing to offload computing tasks to MEC servers is calculated as follows:

$$T_{i,e}^{MEC} = \frac{C_i}{f^{MEC}} \tag{10}$$

Similarly, when mobile terminal user i chooses to offload computing tasks to MEC servers, the energy consumption should include two parts, namely the energy consumption of uploading data and the energy consumption of MEC server computing tasks [33].

Because the energy consumption of MEC servers in standby is much higher than the energy consumption brought about by the completion of computing tasks, and for the mobile terminal user experience, the user does not perceive non-local energy consumption. The energy consumption at MEC is irrelevant, so the energy consumption of MEC server computing tasks is not considered.

Thus, mobile terminal user i chooses to offload computing tasks to MEC servers to complete the calculation of energy consumption for computing tasks as follows:

$$\begin{aligned} E_i^{up} &= \frac{p_i D_i}{r_i^u}, \quad x_{i,j} \neq 0 \\ p_i &= \begin{cases} p_i^M, & \text{if } j = 1 \\ p_i^S, & \text{if } j = 2 \end{cases} \end{aligned} \tag{11}$$

where p_i^M and p_i^S respectively represent the transmission power between mobile terminal user i and MBS, and the transmission power between mobile terminal user i and SBS.

In summary, we can get the total delay consumption of offloading method:

$$T^{MEC}(\mathbf{x}) = \sum_{i \in N, j \in \{1,2\}} x_{i,j} \cdot \left(\frac{D_i}{r_i^u} + \frac{C_i}{f^{MEC}} \right) \tag{12}$$

The total energy consumption of offloading method is calculated as:

$$E^{MEC}(\mathbf{x}) = \sum_{i \in N, j \in \{1,2\}} x_{i,j} \cdot \left(\frac{p_i D_i}{r_i^u} \right) \tag{13}$$

3.3. **Communication model.** Using OFDMA communication model of 5G cellular network, the frequency spectrum of mobile terminal users accessing the same base station is orthogonal to each other. Therefore, there is interference $I_{M,S}$ between the signal transmitted to MBS and the signal transmitted to SBS. Although interference is affected by many factors, good coordinated control can be achieved through different technologies. In order to simplify analysis, it is assumed that mobile terminal users are subject to static interference [34,35].

g_i^M and g_i^S represent the gain between mobile terminal user i and MBS, and the gain between mobile terminal user i and SBS. ρ_0 stands for background noise. According to Shannon's theorem, the spectral efficiency of data uploaded by mobile terminal user i is

$$\begin{aligned} e_{i,j}^M &= \log_2 \left(1 + \frac{p_i^M g_i^M}{I_{M,S} + \rho_0} \right), \quad j = 1 \\ e_{i,j}^S &= \log_2 \left(1 + \frac{p_i^S g_i^S}{I_{M,S} + \rho_0} \right), \quad j = 2 \end{aligned} \tag{14}$$

Organize the above formula as:

$$e_{i,j} = \begin{cases} e_{i,j}^M, & \text{if } j = 1 \\ e_{i,j}^S, & \text{if } j = 2 \end{cases} \tag{15}$$

B represents the bandwidth available to each mobile terminal user i. Thus, the upload rate at which mobile terminal user i chooses to offloading computing tasks can be expressed as:

$$r_i^u(\mathbf{x}) = x_{i,j} \cdot B \cdot e_{i,j} \tag{16}$$

3.4. **Objective function.** The unit is unified in energy consumption model and delay model, that is, the running cost and time cost are converted into corresponding costs. It can be obtained that the total cost of MEC_i in the process of completing offloading computing is the sum of total delay consumption and energy consumption:

$$K_i = T^{MEC}(\mathbf{x}_i) + E^{MEC}(\mathbf{x}_i) + T^L(\mathbf{x}_i) + E^L(\mathbf{x}_i) \tag{17}$$

The research goal is to optimize long-term performance of each MEC, that is, to minimize the cost of a single MEC in long-term energy consumption constraint optimization process:

$$\min_{i \in N} K_i \tag{18}$$

The network composed of MEC is a distributed system, so any SBS or MBS does not have complete information about the overall system. This hinders the derivation of optimal solution. In response to the above situation, using non-cooperative game theory, an edge computing resource allocation algorithm based on game theory is proposed. First define a non-cooperative game $(M, \{x_i\}_{i \in M}, \{K_i\}_{i \in M})$, where M represents the set of MEC, x_i represents all possible offloading schemes of MEC_i, and K_i represents the cost function of MEC_i. In this scenario, the goal of each MEC is to minimize its own cost under the constraints of energy consumption by adjusting its own peer-to-peer offloading scheme in each time slot t. When the game reaches Nash equilibrium, that is, when the offloading schemes of other MECs remain unchanged, any MEC cannot unilaterally choose different peer-to-peer offloading schemes to further reduce its own costs.

Suppose that when the game between MECs reaches Nash equilibrium, the offloading scheme in network is $\beta^{NE} = \left\{\beta_i^{NE}\right\}_{i \in N}$. This solution can minimize the cost of each MEC in network. The running cost and risk cost in cost function are both related to the amount of peer-to-peer offloading tasks. In order to facilitate analysis, from the perspective of peer-to-peer offloading scheme β_i, objective function K_i is reorganized and expanded into $K_i'(\beta_i)$, which is expressed as follows:

$$\begin{aligned} K_i'(\beta_i) = & \underbrace{w_c \left[\beta_{ii} \left(\frac{1}{r_i^u - \omega_i(\beta_i, \beta_{-i})} + \kappa e_i(t)\right)\right]}_{(1)} \\ & + \underbrace{\sum_{j \in N \setminus \{i\}} \beta_{ij} \left[w_c \left(\frac{1}{r_j^u - \omega_j(\beta_i, \beta_{-i})} + \frac{\upsilon}{1 - \upsilon\lambda(\beta_i, \beta_{-i})}\right)\right]}_{(2)} \end{aligned} \tag{19}$$

where β_{-i} represents the offloading scheme of other MEC except MEC_i. Since the offloading scheme of other MECs will affect the cost of MEC_i, it is necessary to consider both its own scheme and the impact of other MEC options. The first part of Equation (19) is the cost of tasks when it is local (including the cost of calculation delay and energy consumption). The second part selects the cost of peer-to-peer offloading calculation for tasks (including the delay cost of communication and calculation).

Therefore, in MEC peer-to-peer offloading, the optimal load balancing problem for each MEC_i is

$$\begin{cases} \min_{\beta_i \in \mathbf{x}_i} K_i'(\beta_i) \\ \beta_i \in \mathbf{x}_i \end{cases} \tag{20}$$

By analyzing the objective function, a resource allocation algorithm based on game theory is proposed. This algorithm enables each MEC in local area network to minimize its own cost by adjusting the plan in each time interval t.

4. Edge Computing Resource Allocation Algorithm Based on Game Theory.

4.1. **Nash equilibrium.** Based on the description of communication model, when a mobile device performs task offloading, it usually needs to transmit task data by channel. When multiple devices are offloaded at the same time, channel competition is large, and transmission interference is serious, and channel congestion may occur at the same time, which not only causes each device. The reduction of data rate between mobile devices and base stations may also lead to tasks that cannot be completed in time, reducing system performance and service quality. At this time, the mobile device should choose to offload locally.

Given a multi-user task offloading strategy result vector a, for decision result β_i ($\beta_i \geq 0$) of user i who has selected offloading tasks, the task scheduling decision is $\beta_{-i} = \{\beta_1, \beta_2, \ldots, \beta_{i-1}, \beta_{i+1}, \ldots, \beta_N\}$, where β_{-i} represents the offloading decision of other users except user i. When the offloading decision of user i is given, the multi-user task offloading decision can be expressed as $\Gamma = (N, \{\beta_i\}_{i\in N}, \{K_i\}_{i\in N})$. The purpose of decision is to minimize the total overhead of task scheduling and offloading process. The objective function can be expressed as:

$$\begin{aligned} &\min_{\beta_i\in\{0,1\}} K_i(\beta_i, \beta_{-i}), \quad \forall i \in N \\ &K_i\left(\beta_i^*, \beta_{-i}^*\right) \leq K_i\left(\beta_i, \beta_{-i}^*\right), \quad \forall \beta_i \in A_i, \ \ i \in N \end{aligned} \tag{21}$$

The node cluster is regarded as N participants, and each participant schedules tasks according to a certain strategy. When other participants achieve optimal strategy, this participant no longer changes his strategy. When all participants no longer change their strategies, the system as a whole reaches a state of equilibrium. The entire system is constantly breaking the old balance and achieving a new balance as the work arrives and is completed. This is a dynamic process.

Each cluster node determines a_i, b_i and c_i according to its own scheduling strategy. If the scheduling strategy of each cluster is feasible for system as a whole, then maintain the status quo. Following the execution of tasks and changes in system conditions, when these strategies are not feasible for the whole, the strategies of some node clusters are adjusted through designed adjustment function, so that the system transitions to a new equilibrium state. When the following conditions are met, system can support the scheduling strategy of each node cluster.

$$\Delta_i(t) \cong \left\{ \begin{array}{l} \tilde{\beta} : \tilde{\beta} = \arg\min\limits_{\beta\in A_i} K_i(\beta_i, \beta_{-i}) \text{ and} \\ K_i(\beta^*, \beta_{-i}(t)) < K_i(\beta_i(t), \beta_{-i}(t)) \end{array} \right\} \tag{22}$$

When $\Delta_n(t)$ is not empty, mobile devices have not reached Nash equilibrium state. The system continues to iterate and update decisions. After a limited number of iterations, all mobile devices will reach Nash equilibrium state, that is, each device has reached its own optimal state. When the decision of a certain device is updated, other devices in system are updated at the same time, and the overall load of system remains balanced and stable.

A threshold is defined for the opportunity consumption of mobile devices, and only mobile devices whose opportunity consumption is less than the threshold can provide resources. The purpose is to avoid resource consumption when there are not enough resources in mobile devices.

Calculating the total cost of offloading and offloading decisions needs to meet the following constraints:

$$K_i(\beta_i, \beta_{-i}) = \left\{ \begin{array}{ll} K_i^L, & \beta_i = 0 \\ K_i^{MEC}, & \beta_i = 1 \end{array} \right. \tag{23}$$

1) Each mobile device gets its own parameter value, and each mobile device makes a calculation decision. And each mobile device in decision time slot receives the parameters of its neighboring mobile devices. At each time t, each device makes a customized decision.

2) Equation (22) is used to determine whether Nash equilibrium has been reached.

3) If all costs reach the minimum at this time, iterative process is terminated. Otherwise, in the next time, namely $t+1$, return to 1) and continue the iterative process.

4.2. **Optimal load balancing.** Aiming at MEC environment, a task allocation and computing offloading algorithm is proposed based on the above system modeling and the establishment of game equations. First of all, the algorithm can realize that multiple mobile devices coordinate with each other before offloading computing tasks, and jointly determine the purpose of computing offloading decisions with each other. The algorithm can achieve Nash equilibrium in the game model of multiple mobile devices and multiple MEC base stations. Furthermore, a potential game equation for task offloading is constructed. The latent equation has a limited strategy set, and can achieve Nash equilibrium within a limited number of iterations. In the proposed algorithm, mobile devices need to broadcast their own communication related parameters to MEC base stations before updating its own decision. According to calculation request and channel conditions, the algorithm allocates an MEC base station that can minimize energy consumption for mobile users to access. Now consider the computing and offloading decision update situation in a time slot, and introduce the concept of better response and optimal response.

Definition 4.1. *The participant changes offloading decision from a_i to a_i', if energy consumption function satisfies*

$$E_i(a_i', a_{-i}) < E_i(a_i, a_{-i}) \tag{24}$$

Then offloading decision a_i' is called the better response decision.

Definition 4.2. *For a given other participant's strategy a_{-i}, when there is no better strategy than strategy a_i^* to change, at the same time strategy a_i^* satisfies*

$$E_i\left(a_i^*, a_{-i}\right) < E_i(a_i, a_{-i}) \tag{25}$$

then offloading decision a_i^* is called the optimal response decision.

Based on the concept of constructing a better response to potential game, in each given time slot, mobile devices can choose a better strategy according to its own preferences. According to the optimal response concept, each participant is playing the role of optimal participant compared to other participants.

The process of mobile devices updating its own decision and completing the computing offloading is shown in Figure 2.

First, each mobile device initializes computing offloading strategy in a time slot. When a mobile device has a computing task to perform, according to the potential game algorithm of multi-user and multi-MEC, each time slot will have a mobile device to choose to update its own strategy. In the time slot t, if the changeable policy set space is not empty, mobile devices will broadcast the relevant parameters of its computing task and send Request-to-Update (RTU) to MEC base stations to request to update its own offloading strategy. The policy space set in time slot t needs to meet: $\Theta_i(t) = \{a_i' : K_i\left(a_i', a_{-i}\right) < K_i(a_i, a_{-i})\}$. Assuming that in time slot t, mobile device L receives the information and can update the strategy in next time slot $a_i(t+1) \in \Theta_i^*(t)$. Then mobile devices will send information to other devices to notify them that they have obtained this update opportunity. For those mobile devices that have not obtained updated information, they will keep existing decision $a_i(t+1) \in a_i(t)$ as the decision for next time slot.

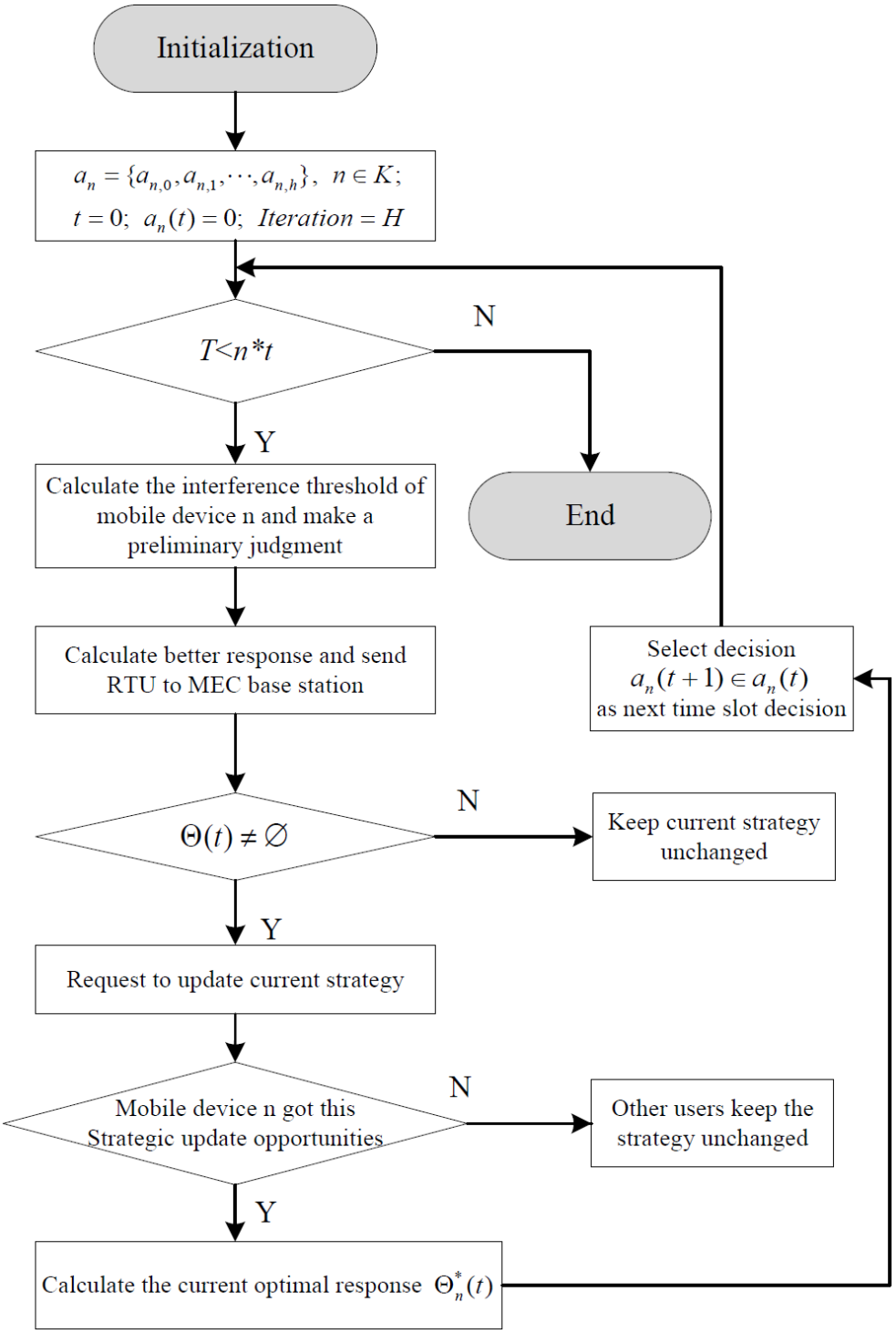

FIGURE 2. Algorithm flow of task allocation and computing offloading

After a limited number of time slot iterations, each mobile device's decision will reach a relatively better value. When there is no RTU information transmission in the entire computing offloading system, it indicates that entire system has reached Nash equilibrium. And the algorithm has converged to global optimal situation.

According to Definition 4.1 and Definition 4.2, the distributed computing offloading algorithm has convergence. For the potential game mentioned, it can always have a Nash equilibrium.

5. **Experimental Program and Analysis.** Set up a simulation environment in MATLAB, which runs on a computer with Intel® Core2 Duo CPU E8200 @ 2.66GHz processor. The service range of SBS and MBS is a circular area with a radius of 1000 meters. Multiple mobile terminal users are randomly allocated in the area, and one MEC server is configured at MBS and SBS. The simulation parameters are shown in Table 1. All the simulation data refer to the relevant literature, and take a more conventional value.

TABLE 1. Simulation parameter setting

Parameter	Value	Parameter	Value
N	100	ρ_0	-100 dBm/Hz
g_0	-40 dB	δ_l	3 W/GHz
d_0	1 m	B	2 MHz
η	4	f_i^l	100 MHz – 1 GHz
f_{MBS}^{MEC}	5 GHz	f_{SBS}^{MEC}	3 GHz
D_i	100 KB – 1 MB	γ_i^T	0.1-0.9

5.1. **Simulation environment.** In the simulation experiment, it is assumed that 10 SBS, 5 MBS and 300 terminals are deployed. And 300 terminals were randomly allocated to 10 SBS and 5 MBS. In this way, the attribution relationship between terminals and SBS and MBS in the real environment is simulated. At the same time, it is assumed that the generation of terminal tasks obeys Poisson distribution, and the value range of task generation speed is $[0, 4]$ per second. The expected number of CPU cycles required for each task is 40 M, and the energy consumption of a single CPU cycle is set to 8.2 nJ. Thus, the expected energy consumption of each task in SBS and MBS is 9×10^{-5} Wh and 7×10^{-4} Wh respectively. In addition, assume that the expected input data size of each task is 0.1 Mb. Therefore, the expected transmission delay for a task is 100 ms for a typical 100 Mb Fast Ethernet LAN.

5.2. **Convergence analysis.** In order to prove that proposed algorithm can reach Nash equilibrium point, it is necessary to analyze the algorithm convergence. The result is shown in Figure 3.

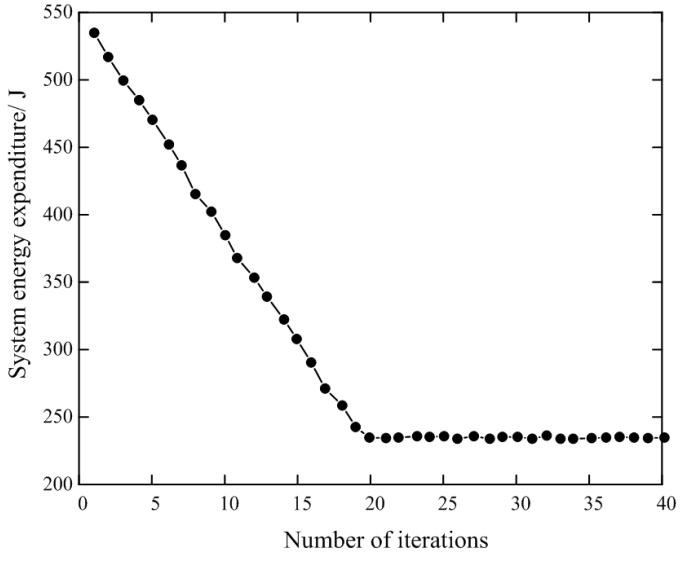

FIGURE 3. Convergence analysis of the algorithm

It can be seen from Figure 3 that the energy consumption of system decreases linearly as the number of iterations increases. When the number of iterations exceeds 20, distributed task offloading algorithm reaches Nash equilibrium. However, when the number of iterations is less than 20 times, the algorithm has been searching for Nash equilibrium point. It can be seen from the overall curve that the proposed algorithm can achieve convergence within a limited number of iterations.

5.3. **Cost analysis of reaching Nash equilibrium.** It is assumed that there are 300 mobile devices in network, and each device has a computing task to be processed. And divide devices into 10 clusters to play game, and finally the cost of reaching Nash equilibrium is shown in Figure 4.

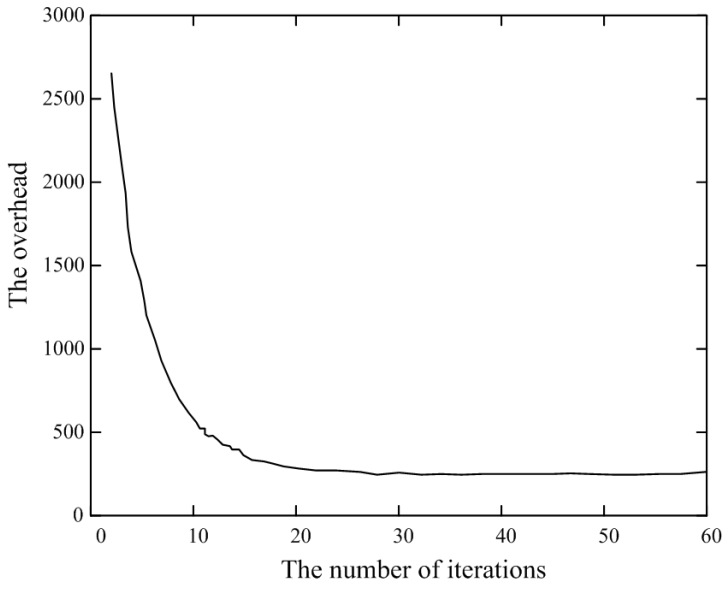

FIGURE 4. The cost of reaching Nash equilibrium

It can be seen from Figure 4 that by increasing the number of iterations, a stable state can be achieved when it exceeds 20 times. Therefore, proposed algorithm can reach Nash equilibrium state in a limited time.

5.4. **Parameter analysis.** In order to analyze the effect of delay-energy consumption trade-off mechanism coefficient on the performance of offloading strategy, the values of γ_i^T and γ_i^E are set in simulation, and other conditions remain unchanged. Then, the results of influence for different weighing mechanism coefficients on calculating the average delay of offloading strategy are shown in Figure 5.

It can be seen from Figure 5 that under the same number of mobile users and the number of MECs, increasing delay coefficient γ_i^T will help reduce the average offload delay. The effect will tend to be obvious when the number of mobile users is large. When the number of mobile users is 1000, system offloading delay when $\gamma_i^T = 0.8$ is reduced by 12.3% compared with offloading delay when $\gamma_i^T = 0.6$. Similarly, when $\gamma_i^T = 0.8$, system offloading delay is reduced by 16.8% compared with offloading delay when $\gamma_i^T = 0.5$. In addition, the influence of different trade-off mechanism coefficients on the balance of offloading strategy resource allocation is shown in Figure 6.

It can be seen from Figure 6 that the change of energy trade-off coefficient γ_i^E affects the degree of resource allocation balance. In the case of the same number for mobile users and the number of MECs, reducing energy trade-off coefficient γ_i^E is not conducive to the

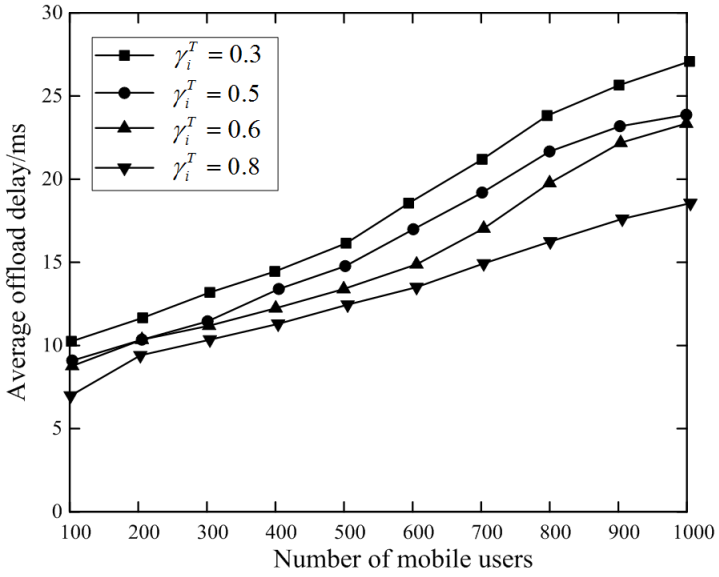

FIGURE 5. Influence of different trade-off mechanism coefficients on average offloading delay

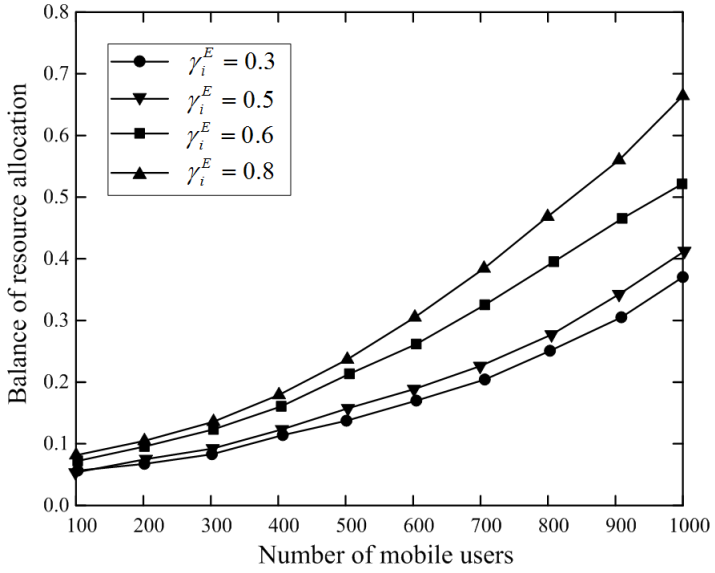

FIGURE 6. Influence of different trade-off mechanism coefficients on balance of resource allocation

balanced allocation of resources among MECs. When the number of mobile users is less than 500, the resources required for computing offloading account for a small proportion of total MEC resources. When γ_i^E value is reduced from 0.5 to 0.3, the balance of system resource allocation does not change much. With the increase in the number of mobile users, the difference in the degree of resource allocation balance has become increasingly obvious. When the number of mobile users is 1000, the balance of $\gamma_i^E = 0.3$ is 8.4% lower than that of 0.5. When $\gamma_i^E = 0.8$, the balance is 11.9% higher than when $\gamma_i^E = 0.5$.

As γ_i^T is larger, the average offloading delay decreases less obviously. However, the larger the γ_i^E, the higher the balance of system resource allocation. Thus, considering the experimental results of delay-energy consumption trade-off mechanism coefficients γ_i^T and γ_i^E, γ_i^T is set to 0.8 and γ_i^E is set to 0.6.

5.5. **System delay analysis.** In order to demonstrate the delay performance of the proposed strategy, it is compared with the strategies in [21], [27], and [30]. The experimental results under different numbers of mobile users are shown in Figure 7.

It can be seen from Figure 7 that the average delay of system increases as number of users increases. However, the proposed strategy has lower system latency compared to other offloading strategies. This is because it uses game theory for task offloading, taking account of the load conditions of MEC and base stations, and performing task offloading with optimal offloading decision and offloading rate. It makes full use of local computing resources while balancing the business load at MEC-BS. This avoids the additional delay overhead under congestion conditions and effectively reduces system delay. When the number of mobile users is 1000, the average delay is only 13 ms.

At the same time, the computing power of MBS and SBS will directly affect the task processing capacity of system, and the average delay change is shown in Figure 8.

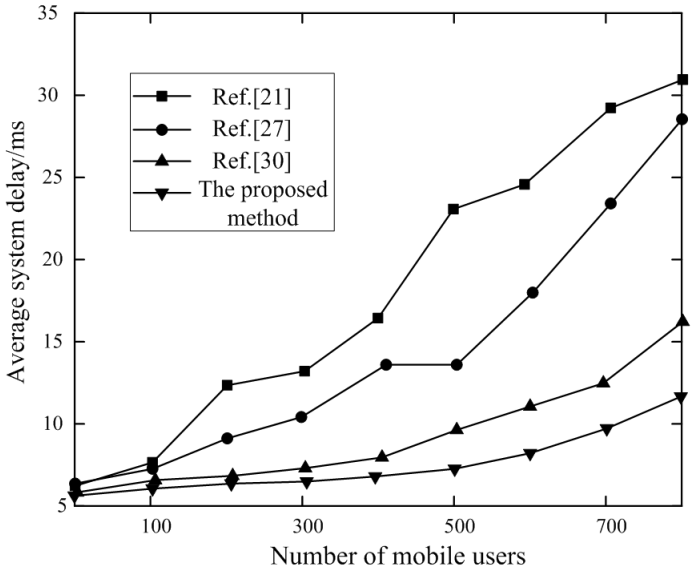

FIGURE 7. Average delay of system with different number of users

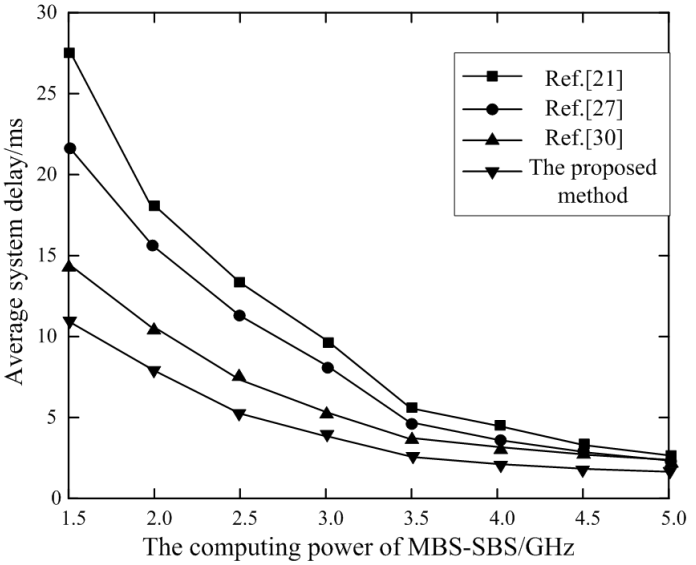

FIGURE 8. Average delay of system under different MBS-SBS computing capabilities

It can be seen from Figure 8 that as the computing power of MBS-SBS increases, the average delay of system is decreasing. And under the same circumstances, the proposed algorithm obviously has lower system delay than other strategies. And as the computing power continues to increase, the average delay of system eventually stabilizes. When the computing power of MBS-SBS continues to increase, the computing load of each MBS-SBS is relatively light. At this time, there is little difference in calculation delay, which mainly depends on communication delay. [21] used a context-aware model to implement offloading and scheduling offloading task decisions among shared computing resources. However, shared resources can easily cause congestion, and it is difficult to reduce system delay. [27] realized the problems of computing offloading and resource allocation based on newly introduced task causality and completion constraints. Among them, it focused on task switching decision-making. The increase of the computing power of MBS-SBS will reduce delay, but the effect is mediocre. [30] improved the user offloading rate and reduces delay to a certain extent by jointly optimizing the energy transmission beamforming at multiple APs. However, too many AP nodes increase the selection time of offloading path, so the delay reduction effect is limited. Our proposed strategy considers communication and calculation delays at the same time, and minimizes the system delay by adjusting user's offloading options and offloading rate. This effectively balances the business load while reducing system's delay overhead.

5.6. **System energy cost analysis.** In order to compare proposed strategy with the performance of strategy resource scheduling and computing offloading in [21], [27] and [30], a comparative experiment was carried out in terms of energy cost, and the results are shown in Figure 9.

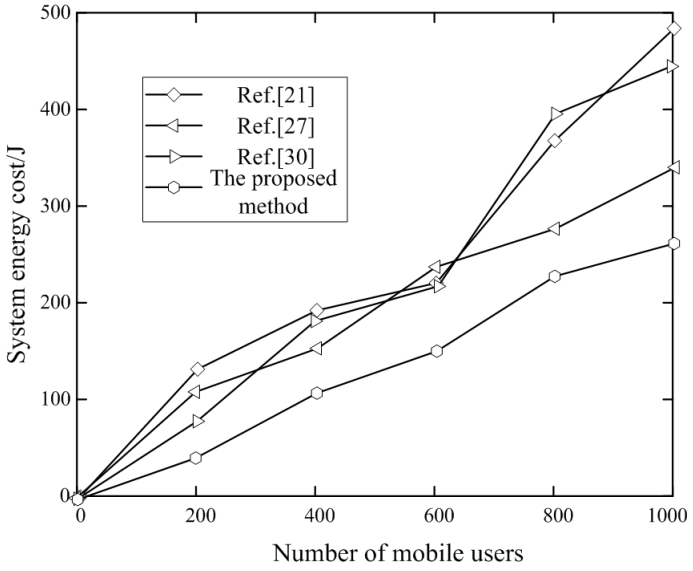

FIGURE 9. System energy cost of different algorithms

It can be seen from Figure 9 that the proposed strategy uses game theory to implement resource allocation and computing offloading, with the least energy consumption and the best performance. In fact, the only equilibrium point of the proposed potential game has been proved. When the latent equation reaches Nash equilibrium, the objective function of distributed computing offload reaches global optimum. So the game finally got a satisfactory solution. [21] realized the allocation of shared resources by a context-aware model. Due to the limited capacity of shared resources, the energy consumption per unit

of CPU is relatively large, resulting in significantly higher energy costs. [27] randomly exchanged computing tasks for each mobile device. Due to the random allocation of access, more mobile devices may not be able to match the server. Increase the interference between devices and increase the transmission energy consumption of computing tasks. [30] optimized energy transmission beam forming at multiple APs jointly. Thus, when there are fewer mobile users, the advantage of energy consumption is not obvious. As the number of users increases, multiple APs can provide users with more offloading decisions. Therefore, when the number of mobile users is 1000, its energy consumption is 320 J. Our proposed strategy uses game theory to realize the reasonable allocation of MBS and SBS tasks and reduce energy expenditure. When the number of mobile users is 1000, energy consumption is only 240 J.

6. **Conclusion.** With the increasing popularity of mobile terminal users and emerging computing-intensive and demanding latency applications, the data traffic in mobile communication system has increased dramatically. A resource scheduling and computing offloading strategy for IoT in an MEC environment is proposed in order to meet the various personalized needs of them and realize green communication concept. Based on the constructed computing offloading and resource allocation system model with MBS and SBS, computational model and communication model of this system are designed, the optimization goal of minimizing system delay and energy consumption is established, and game theory is introduced to solve the problem. Furthermore, potential game model is used to solve resource allocation and computing offloading problem. Mobile devices can select MEC nodes for computing offloading according to the game result. The simulation experiment of the proposed strategy is carried out based on MATLAB platform. Experimental results show that our proposed potential game equation can converge to Nash equilibrium when the number of iterations exceeds 20. Besides, the energy consumption and delay of the proposed strategy are better than other comparison strategies. Taking 1000 mobile users as an example, the energy consumption and time delay of the proposed strategy are only 240 J and 13 ms respectively, which has certain practical value.

Under the considered communication model, non-static interference can be considered for the interference problem between mobile terminal users choosing to access SBS and MBS. Subsequent research work can consider the impact of interference caused by the decision of mobile terminal users.

Acknowledgment. This work was supported by the Science and Technology Plan Project of Xi'an (No. 2019KJWL26), the Science and Technology Plan Project of Xi'an (No. 2019218014GXRC016CG017-GXYD16.1), the Internet of Things Application Engineering Laboratory of Xi'an, Shaanxi Key Laboratory of Surface Engineering and Remanufacturing.

REFERENCES

[1] R. Gupta, Resource provisioning and scheduling techniques of IoT based applications in fog computing, *International Journal of Fog Computing*, vol.2, no.2, pp.57-70, 2019.

[2] J. Kim and H. Chang, Development of the revised security management model for small and medium size healthcare organizations, *ICIC Express Letters, Part B: Applications*, vol.12, no.2, pp.161-168, 2021.

[3] X. Xu, Q. Liu, Y. Luo et al., A computation offloading method over big data for IoT-enabled cloud-edge computing, *Future Generation Computer Systems*, vol.9, pp.522-533, 2019.

[4] M. S. Mekala and P. Viswanathan, A survey: Energy-efficient sensor and VM selection approaches in green computing for X-IoT applications, *International Journal of Computers & Applications*, vol.42, no.3, pp.290-305, 2020.

[5] C. Zhang, H. Zhao and S. Deng, A density-based offloading strategy for IoT devices in edge computing systems, *IEEE Access*, vol.6, no.4, pp.73520-73530, 2018.
[6] S. Hu and G. Li, Dynamic request scheduling optimization in mobile edge computing for IoT applications, *IEEE Internet of Things Journal*, vol.7, no.2, pp.1426-1437, 2020.
[7] W. Sun, J. Liu and Y. Yue, AI-enhanced offloading in edge computing: When machine learning meets industrial IoT, *IEEE Network*, vol.33, no.5, pp.68-74, 2019.
[8] M. Aazam, S. Zeadally and K. A. Harras, Offloading in fog computing for IoT: Review, enabling technologies, and research opportunities, *Future Generation Computer Systems*, vol.87, pp.278-289, 2018.
[9] L. Lei, H. Xu, X. Xiong et al., Multi-user resource control with deep reinforcement learning in IoT edge computing, *IEEE Internet of Things Journal*, vol.6, no.6, pp.10119-10133, 2019.
[10] W. Zhan, C. Luo, J. Wang et al., Deep-reinforcement-learning-based offloading scheduling for vehicular edge computing, *IEEE Internet of Things Journal*, vol.7, no.6, pp.5449-5465, 2020.
[11] J. Fan, X. Wei, T. Wang et al., Churn-resilient task scheduling in a tiered IoT infrastructure, *China Communications*, vol.16, no.8, pp.162-175, 2019.
[12] A. Ashok, P. Steenkiste and F. Bai, Vehicular cloud computing through dynamic computation offloading, *Computer Communications*, vol.120, pp.125-137, 2018.
[13] S. Wan, J. Lu, P. Fan et al., Toward big data processing in IoT: Path planning and resource management of UAV base stations in mobile-edge computing system, *IEEE Internet of Things Journal*, vol.7, no.7, pp.5995-6009, 2020.
[14] C. Shu, Z. Zhao, Y. Han et al., Multi-user offloading for edge computing networks: A dependency-aware and latency-optimal approach, *IEEE Internet of Things Journal*, vol.7, no.3, pp.1678-1689, 2020.
[15] F. Jiang, K. Wang, L. Dong et al., Deep-learning-based joint resource scheduling algorithms for hybrid MEC networks, *IEEE Internet of Things Journal*, vol.7, no.7, pp.6252-6265, 2020.
[16] Q. D. La, M. V. Ngo, T. Q. Dinh et al., Enabling intelligence in fog computing to achieve energy and latency reduction, *Digital Communications and Networks*, vol.5, pp.3-9, 2019.
[17] Y. Liu, Y. Li, Y. Niu et al., Joint optimization of path planning and resource allocation in mobile edge computing, *IEEE Transactions on Mobile Computing*, vol.19, no.9, pp.2129-2144, 2020.
[18] Y. Jiang and D. H. K. Tsang, Delay-aware task offloading in shared fog networks, *IEEE Internet of Things Journal*, vol.5, no.6, pp.4945-4956, 2018.
[19] P. Cai, F. Yang, J. Wang et al., JOTE: Joint offloading of tasks and energy in fog-enabled IoT networks, *IEEE Internet of Things Journal*, vol.7, no.4, pp.3067-3082, 2020.
[20] D. Sivaganesan, Design and development AI-enabled edge computing for intelligent-IoT applications, *Journal of Trends in Computer Science and Smart Technology*, vol.2019, no.2, pp.84-94, 2019.
[21] B. Zhou, A. V. Dastjerdi, R. N. Calheiros et al., An online algorithm for task offloading in heterogeneous mobile clouds, *ACM Transactions on Internet Technology*, vol.18, no.2, pp.1-25, 2018.
[22] B. Mallikarjuna, Feedback-based fuzzy resource management in IoT-based-cloud, *International Journal of Fog Computing*, vol.3, no.1, pp.1-21, 2020.
[23] J. Kokila, N. Ramasubramanian and N. Naganathan, Resource efficient metering scheme for protecting SoC FPGA device and IPs in IoT applications, *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol.27, no.10, pp.2284-2295, 2019.
[24] X. Chen, L. Jiao, W. Li et al., Efficient multi-user computation offloading for mobile-edge cloud computing, *IEEE/ACM Transactions on Networking*, vol.24, no.5, pp.2795-2808, 2015.
[25] H. Wu, X. Lyu and H. Tian, Online optimization of wireless powered mobile-edge computing for heterogeneous industrial Internet of Things, *IEEE Internet of Things Journal*, vol.6, no.6, pp.9880-9892, 2019.
[26] K. Wang, Z. Xiong et al., Joint time delay and energy optimization with intelligent overclocking in edge computing, *Science China (Information Sciences)*, vol.63, no.4, pp.154-169, 2020.
[27] H. Tang, C. Li, J. Bai et al., Dynamic resource allocation strategy for latency-critical and computation-intensive applications in cloud-edge environment, *Computer Communications*, vol.134, no.1, pp.70-82, 2019.
[28] H. Qin, S. Zawad, Y. Zhou et al., Reinforcement-learning-empowered MLaaS scheduling for serving intelligent Internet of Things, *IEEE Internet of Things Journal*, vol.7, no.7, pp.6325-6337, 2020.
[29] Q. Zhang, L. Gui, F. Hou et al., Dynamic task offloading and resource allocation for mobile edge computing in dense cloud RAN, *IEEE Internet of Things Journal*, vol.7, no.4, pp.3282-3299, 2020.

[30] S. S. Gill, P. Garraghan and R. Buyya, ROUTER: Fog enabled cloud based intelligent resource management approach for smart home IoT devices, *The Journal of Systems and Software*, vol.154, no.8, pp.125-138, 2019.
[31] X. Xu, C. He, Z. Xu et al., Joint optimization of offloading utility and privacy for edge computing enabled IoT, *IEEE Internet of Things Journal*, vol.7, no.4, pp.2622-2629, 2020.
[32] H. Ke, J. Wang, H. Wang et al., Joint optimization of data offloading and resource allocation with renewable energy aware for IoT devices: A deep reinforcement learning approach, *IEEE Access*, vol.7, no.8, pp.179349-179363, 2019.
[33] T. Wang, J. Zhou, A. Liu et al., Fog-based computing and storage offloading for data synchronization in IoT, *IEEE Internet of Things Journal*, vol.6, no.3, pp.4272-4282, 2019.
[34] S. Shen, Y. Han, X. Wang et al., Computation offloading with multiple agents in edge-computing-supported IoT, *ACM Transactions on Sensor Networks*, vol.16, no.1, pp.8.1-8.27, 2020.
[35] Y. Dai, D. Xu et al., Joint load balancing and offloading in vehicular edge computing and networks, *IEEE Internet of Things Journal*, vol.6, no.3, pp.4377-4387, 2019.