International Journal of Innovative Computing, Information and Control
Volume **16**, Number **6**, December **2020**

ICIC International ©2020 ISSN 1349-4198
pp. 1895–1914

COMPUTING OFFLOADING AND RESOURCE ALLOCATION ALGORITHM BASED ON GAME THEORY FOR IOT DEVICES IN MOBILE EDGE COMPUTING

Jianqiang Xu[1], Zhujiao Hu[2] and Junzhong Zou[1,*]

[1]School of Information Science and Engineering
East China University of Science and Technology
No. 130, Meilong Road, Shanghai 200237, P. R. China
xujianqiang2019@126.com; *Corresponding author: zoujunzhongecust@126.com

[2]School of Microelectronics
Fudan University
No. 825, Zhangheng Road, Shanghai 201203, P. R. China
huzhujiao@163.com

Received June 2020; revised October 2020

Abstract. *With the explosive growth of Internet of Things (IoT) devices and the popularization of intelligent terminals, the computing resources of various new applications are intensive, and existing work offloads them to mobile edge centers for computing. However, due to limitations in the coverage and resource capabilities of edge devices, there is still a problem of unbalanced load in the network. Therefore, in order to balance the load reasonably, this paper proposes a computing offloading and resource allocation algorithm for IoT devices using game theory in Mobile Edge Computing (MEC). Firstly, a multi-user multi-MEC computing offloading and resource allocation model is designed, which manages a large number of devices by multiple base stations and multiple MEC partitions, and models the system including computing model and task model. Then, a prediction method of MEC server waiting time is proposed according to the task queue in base stations. The task processing sequence, time and frequency are arranged to improve the calculation efficiency of this algorithm, which makes rational use of computing resources for devices and MEC. Finally, the problem of multi-user and multi-MEC computing offloading and resource allocation is transformed into a multi-user game problem, and it is concluded that there is always a Nash equilibrium in this game problem. Furthermore, the optimal solution of this problem, that is, the optimal device computing offloading and resource allocation scheme, is obtained when the system is in Nash equilibrium state. The simulation results based on MATLAB platform show that our proposed algorithm is effective compared with other algorithms. It can converge reliably and has obvious optimization in energy consumption and time delay; besides the performance of this algorithm is improved.*

Keywords: Mobile edge computing, Game theory, Computing offloading, Resource allocation, IoT device, Nash equilibrium

1. **Introduction.**

With the development of mobile Internet and the popularization of intelligent terminals, various new applications, such as Augmented Reality (AR), virtual reality and natural language processing, are constantly emerging. These applications usually have resource-intensive characteristics, and they need to consume a lot of computing resources and storage resources when they run. Besides, the requirements for Quality of Service (QoS) are extremely high [1]. Although the performance of intelligent terminal processors continues to increase, it is still difficult to meet the needs of processing

DOI: 10.24507/ijicic.16.06.1895

high-performance applications in a short time, which seriously affects users' Quality of Experience (QoE) [2]. Therefore, how to expand intelligent terminal resources to meet the high-performance task execution demands is currently an urgent problem to be solved [3].

MEC provides a feasible solution for the effective resolution of above problems. The MEC technology allows terminal devices to offload computationally intensive tasks to MEC servers for execution. With the help of high computing performance MEC servers, the task execution delay is reduced [4]. At the same time, since the terminal offloads its own computing tasks, the energy consumption required to perform tasks can be significantly reduced. Therefore, MEC can effectively alleviate the outstanding contradiction between the limited resource of intelligent terminals and the demand for high-performance task processing.

In order to solve the contradiction between terminal resource limitation and application task processing high-performance requirements, a solution based on Mobile Cloud Computing (MCC) technology can be adopted [5]. In MCC system, intelligent terminals can access remote cloud servers located in the core network. They offload applications or tasks to cloud servers and use their powerful computing and storage resources to efficiently execute tasks or services. However, traditional MCC models that rely on centralized cloud servers have certain limitations [6]. For example, the deployment location of cloud servers is usually far from smart terminals. Transmitting a large amount of task data to cloud servers located in Core Network (CN) not only consumes a lot of network resources, but also increases the backhaul link load, which in turn leads to network congestion and other problems [7].

In response to the shortcomings of MCC, researchers proposed an edge cloud solution to move the powerful cloud computing capabilities closer to the source of task data [8]. As a computing model with great potential, the edge cloud implements the marginalization of user task processing by configuring computing and storage capabilities for network edge nodes. This effectively reduces the cloud computing load and reduces the network bandwidth overhead while providing users with faster response times for computing tasks. In recent years, with the in-depth development of cloud computing and mobile Internet industries, researchers have proposed a variety of similar solutions, such as MEC, micro-cloud and fog computing, combining cloud computing models and edge cloud concepts [9].

Since the nodes performing edge computing are located close to end users, the peak traffic is alleviated, which not only reduces the load pressure of core network, but also reduces user's transmission delay and delay jitter during the task offloading process. Therefore, the distributed computing nodes deployed on the edge realize the distribution of cloud traffic and computing pressure. At the same time, the response time of IoT applications is also faster than the corresponding cloud computing services. In addition, edge computing migrates computing and communication overhead from user terminal nodes with limited battery or power supply to edge nodes with relatively rich resources [10]. In this way, the energy consumption of nodes is reduced, and its survival time is extended, thereby increasing the life of entire IoT network.

However, the computing and storage resources of edge computing servers are extremely limited compared to cloud data centers, and the arrival of computing workload in edge networks may be highly dynamic and heterogeneous [11]. Therefore, it is difficult for a single edge service to provide users with satisfactory computing services in all time periods. That is, within a certain period of time, there may be a surge in the amount of user upload tasks received by some edge servers (the increase in the number of connected terminal devices or the occurrence of large amounts of data in the area under jurisdiction), resulting in energy consumption of edge servers too fast or tasks wait too long. This will cause a load imbalance in the network and reduce the overall performance of network [12]. Thus, how to effectively allocate computing resources in the "edge" to "support" servers with too many tasks has become a new challenge.

2. **Related Research.**

For the task offloading and resource allocation in edge computing, [13] proposed a virtual cloud solution. By interconnecting terminals, they can accomplish computing tasks collaboratively. This solved the problem that mobile terminals cannot communicate with cloud data center steadily or transfer tasks to cloud data center when the network conditions are not good. [14] designed a verification platform for facial recognition, which migrated the facial recognition application originally deployed in cloud data center to the edge of network, which greatly reduced the original response time. [15] used MEC or fog computing infrastructure components to create mobile clouds. It provided users with necessary computing and storage resources at the edge of network, and on this basis, a safe, reliable and adaptable programming model and framework are designed. Due to the rapid growth of terminal devices in IoT, [16] proposed a distributed architecture whose scope extends from the edge of network to edge servers, and is mainly used to process massive amounts of IoT data. In view of the service development of heterogeneous devices in IoT and the deployment of edge computing services, [17] proposed a scalable dynamic intermediate infrastructure SECS (Scalable Edge Computing Services), which facility meets the network's future needs for scalability and reliability. By combining the above documents, we can find that "the edge of network" is not clearly defined, and the devices expected to participate in edge computing may also differ depending on the application scenario. In terms of users offloading tasks to edge devices, algorithms are divided into two categories based on different optimization goals.

(1) Algorithms based on objective function optimization. [18] studied the offloading problem of small cellular network computing based on MEC. In this paper, considering the forward link and backward link, an offloading model for computing tasks based on energy efficiency optimization is designed. Moreover, an optimization objective function is proposed according to the mathematical models of communication resources and computing resources. [19] optimized the application's selection of local, edge and cloud when task offloading, time delay and energy consumption were modeled and normalized to the system loss respectively. Based on this, a heuristic algorithm with minimum delay and energy consumption as the goal was proposed. This algorithm was an approximate optimal algorithm, which effectively reduces the loss in the system. [20] designed a mobile offloading computing framework based on MEC and UDN (COMED) based on ultra-dense network. The framework implemented efficient computing offloading, base station sleep scheduling and user-base station association. On this basis, authors took the minimization of overall energy consumption of devices and base stations as the optimization goal, and proposed a task offloading algorithm based on Lyapunov optimization theory. [21] took the offloading decision, resource allocation and content caching strategy in MEC as optimization problems, and transformed the original non-convex problems into convex problems. In order to effectively solve this problem, a distributed solution based on alternating direction multipliers was proposed. [22] innovatively introduced the deep learning of IoT into the edge computing environment. Through the training of deep learning network on cloud server, they divided the learning network into a lower layer deployed on edge server close to the input data and a higher layer deployed in the cloud for offloading processing. Besides, they also designed an effective algorithm to optimize the service capability of edge computing model.

However, in many cases, the offloading schemes among users affect each other, and it is impossible to individually optimize individual users. Therefore, an algorithm based on game theory was proposed. In this type of algorithm, it can be divided into two categories due to different optimization goals. One of them is to maximize utility. [23] constructed an MEC-based offloading framework for the problem of searching for service nodes when vehicle offloading request tasks are performed in vehicle-mounted network. The framework took account of the problems of limited resources, heterogeneity and diversity of tasks, and the framework was modeled as a matching mode for combinatorial auctions. On this basis, a multi-round sequential combined auction mechanism was proposed, which equates the matching problem to a multi-dimensional grouping knapsack problem and uses dynamic programming to obtain the optimal matching. [24] studied the problem of maximizing the overall network throughput based on the fairness of user resource allocation in MEC. In this paper, Nash bargaining game was used to analyze this problem, and a user priority determination algorithm considering delay constraints was proposed. [25] studied edge computing services supporting mobile Blockchain. In order to effectively allocate computing resources on the edge, authors proposed a market auction model composed of Blockchain owners, edge computing service providers, miners and maximized social welfare.

(2) Algorithms based on overall performance optimization. [26] designed an offloading algorithm based on potential games in a small cellular network with multiple users and multiple MEC nodes. In this algorithm, users selected access conditions based on the interference generated when they are connected to each other, and finally optimized the energy consumption and delay of each mobile device. [27] studied the algorithm of cloud and wireless resource joint allocation based on evolutionary game in MEC environment. This algorithm combined the choice of service provider with the allocation of resources to minimize energy consumption, time delay and economic cost of mobile terminals, and used replication factor dynamics method to verify the balance of evolutionary games. [28] researched resource allocation of multi-user MEC offloading system based on Time-Division Multiple Access (TDMA) and Orthogonal Frequency-Division Multiple Access (OFDMA). For TDMA MEC offloading systems with unlimited or limited cloud computing capabilities, they transformed the optimal resource allocation into a convex optimization problem to minimize the weighted sum of mobile energy consumption under the constraints of computing latency. Experimental results proved that the optimal strategy has better overall performance than the derived offloading priority function.

Secondly, since the resources and coverage of edge devices are limited, edge devices in the network will have an uneven load among devices due to the distribution of users or the amount of offloaded tasks. In response to this problem, [29] established an MEC collaborative architecture. Under this architecture, a node-based task scheduling optimization algorithm was proposed. The algorithm achieved the full utilization of computing resources in the network by the scheme of dynamic task redistribution between task overload nodes and nearby nodes. Due to the limited computing resources of single Small Base Stations (SBSs), in order to better guarantee the quality of service, [30] proposed an "SBSs alliance" formation algorithm based on game theory. And on this basis, an incentive mechanism based on proportional payment was proposed and a security risk management based on social trust was constructed, which enables collaborative computing among SBSs and improves resource utilization. In addition, [31] constructed a new peer-to-peer offloading framework OPEN and described the optimal peer-to-peer offloading scheme. Then, in this framework, two ways of coordinated peer-to-peer offloading were considered, and a new peer-to-peer offloading game was proposed using variational inequality to prove Nash equilibrium.

Since small base stations are purely distributed management, and purpose is to minimize the cost of each small base station. Therefore, the process of resource allocation is a process of game between edge devices. Thus, a game resource allocation algorithm based on game theory for MEC is proposed. The innovation of our proposed method is as follows.

1) A multi-user multi-MEC computing offloading and resource allocation model is designed according to the characteristics of IoT devices. This model manages a large number of devices by multiple base stations and multiple MEC partitions, and models the system including computing model and task model to comprehensively manage the computing resources of devices and MEC.

2) Since existing methods do not consider the service waiting time of MEC, our proposed algorithm proposes a prediction method for the waiting time of MEC server, which arranges the task processing sequence, time and processing frequency according to the task queue in base stations. In addition, it improves the calculation efficiency of this algorithm and reasonably uses the computing resources of devices and MEC.

3) In order to minimize the energy consumption of computing offloading and resource allocation algorithms and to shorten time delay, the problem of multi-user multi-MEC computing offloading and resource allocation is transformed into a multi-user game problem. It is concluded that there is always a Nash equilibrium in this game problem. When the system is in Nash equilibrium state, the optimal solution of this problem is obtained, that is, the optimal devices computing offloading and resource allocation scheme.

3. **Problem Description and System Modeling.**

3.1. **Network scenario analysis.**

The layered edge computing network architecture is shown in Figure 1. The following assumptions are made about the network scenario: N Base Stations (BS) ($BS_i \in \{BS_1, BS_2, \ldots, BS_N\}$) are deployed within a certain range. These devices have a certain amount of computing power, which is presented by CPU frequency (that is, the number of machine cycles completed in 1 second), that is, the computing power of each BS can be expressed by frequency set $f = \{f_i\}_{i \in N}$. In the edge computing layer, BSs communicate with each other through optical fibers to form a Local Area Network (LAN). In addition, there are M user terminals ($m = \{1, \ldots, M\}$) in this scenario. They are randomly deployed around BS, and each BS has a certain number of user terminals $M_i \subseteq M$, $i \in N$. These user terminals can upload their computing tasks to the corresponding BS for processing by wireless communication.

Since the number of user terminals under the jurisdiction of BSs is different or the task volume uploaded by user terminals under the jurisdiction is different, the total task volume received by each BS may be different in the same time period. In order to make full use of resources in the network, this paper analyzes the cost of completing tasks from the perspective of energy consumption, time delay and transmission risk required by BSs to complete tasks. In the case where each BS has limited long-term energy consumption, its own cost is minimized.

In order to facilitate the introduction of subsequent algorithm design steps, the necessary basic concepts are explained.

Definition 3.1. *If small base station BS_i can only handle part of workload locally, the rest needs to be offloaded to other BSs for processing. Such small base stations are called "hot spots" (BS_1 in the figure), and "hot spots" are represented by set H.*

Definition 3.2. *If small base station BS_i can complete all workload locally, the small base station is called "autonomous point" (BS_2, BS_3 in the figure). The "autonomous point" is represented by set I.*

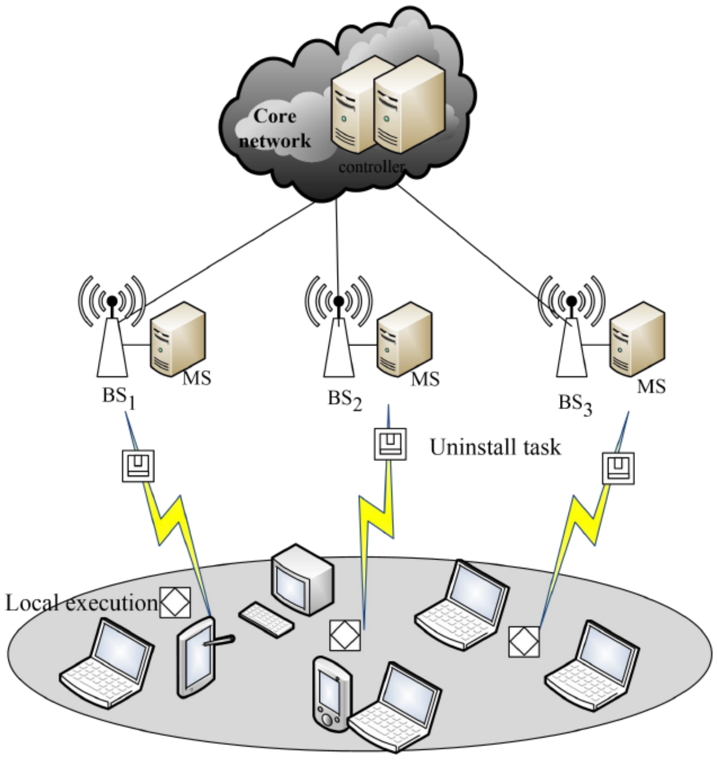


Figure 1. Multi-user and multi-MEC server system model

Definition 3.3. *If small base station BS_j does not communicate with small base station BS_i during the entire resource allocation process. This type of small base station is called an "unrelated point" with small base station BS_i (as shown in the figure, BS_3 is the "unrelated point" of BS_1). The "don't care" of BS_i is represented by set C_i.*

Definition 3.4. *If small base station BS_j provides computing resources for small base station BS_i in the entire resource allocation project. The small base station BS_j is called the "assist point" of small base station BS_i (BS_2 is the "assist point" of BS_1 in the figure), and the "assist point" of BS_i is represented by set A_i.*

3.2. **System modeling.** In order to facilitate the research of this problem, IoT device computing offloading and resource allocation models in the MEC are constructed, including the task model and computing model.

3.2.1. *Task model.*

(1) Upload task

In order to facilitate the peer-to-peer offloading decision, the continuous time is divided into multiple equal time slots (for example, 5 minutes). In each unit time slot, in order to be closer to the actual situation, it is assumed that user terminal m task uploads follow Poisson distribution. And the average task upload rate of user terminal m in time slot t is π_m^t (that is, the number of task uploads in a unit time slot, randomly selected within the $[0, \pi_{\max}]$ range). Therefore, in time slot t, the average arrival rate of tasks on BS_i is $\phi_i^t = \sum_{m \in M_i} \pi_m^t$. In addition, because the user terminal may generate different types of service requests, in order to simplify model, only the amount of tasks required to complete tasks and the number of CPU clock cycles required to complete tasks are considered. Thus, a single task is defined as $a_n = (L, K)$, where L represents the data size of tasks (in bits) and K represents the number of CPU clock cycles required to complete tasks. Since the proposed algorithm only studies the resource allocation between BSs, it only considers the interaction of task data between user terminals and BSs, and does not consider energy consumption and delay generated during the specific upload and reception processes.

(2) Peer-to-peer offload task

Due to the different amount of tasks reaching each BS, the workload between BSs is often uneven. In order to make full use of the resources in network, BS adopts a peer-to-peer way to perform computing offloading to improve the efficiency of entire system and balance energy consumption [32]. In the process of performing peer-to-peer offloading, it is assumed that tasks can only be offloaded once. That is, when tasks are offloaded from BS_i to BS_j, in order to avoid repeated offloading affecting transmission delay, it will only be processed on BS_j. Since BS_i may offload multiple tasks to one or more small base stations, the offloading scheme of BS_i in time slot t is defined as $\beta^t_{i\bullet} = \left\{\beta^t_{ij}\right\}_{j \in N}$. Among them, β^t_{ij} represents the average offload tasks from BS_i to BS_j in time slot t (that is, the amount of tasks that are offloaded peer-to-peer per second), and β^t_{ii} can represent the amount of tasks that are still being executed in BS_i. Therefore, the offloading scheme of each BS in the time slot t can be represented as $\beta^t = \{\beta^t_{i\bullet}\}_{i \in N}$, where $\{\beta^t_{i\bullet}\}_{i \in N}$ can be used to represent all feasible schemes of BS_i. In addition, tasks received by BS_i from other BSs are defined as $\beta^t_{\bullet i} = \left\{\beta^t_{ji}\right\}_{j \in N}$. From this, the total amount of tasks that need to be processed on BS_i is $\omega^t_i\left(\beta^t\right) \cong \sum_{j \in N} \beta^t_{ji}$. To sum up, the offloading program needs to meet the following conditions:

1) Non-negative: $\beta^t_{ji} \geq 0$, $\forall i, j \in N$; that is, the offloading amount of BS_i cannot be less than zero.

2) Conservation: $\sum_{j=1}^{N} \beta^t_{ij} = \phi^t_i$, $\forall i \in N$; that is, all tasks of BS_i should be equal to the amount of tasks reached.

3) Stability: $\omega^t_i\left(\beta^t\right) \leq f_i/K$, $\forall i \in N$; that is, the amount of tasks that BS_i needs to process cannot exceed the speed of services it can provide.

3.2.2. *Computing model.* It can be seen from the task model that BS task computing methods can be divided into two types. The first is that tasks are calculated locally in BS_i. In this case, only the current state of BS_i is considered, and there is no need to consider the communication between BSs [33]. The second is to distribute tasks to other suitable BSs for calculation. At this time, the time delay caused by the inter-BS communication needs to be considered. The two cases are modeled separately below.

(1) Local computing

1) Computing energy consumption: It is assumed that the BS_i energy consumption is linearly related to workload, so the local energy consumption can be expressed as:

$$E^{lt}_i = e \cdot \omega^t_i\left(\beta^t\right), \quad e > 0 \tag{1}$$

where e is the energy consumption of k CPU cycles.

2) Computational delay: Due to the limited computing power of BSs, the workload needs to be processed in order. Thus, in addition to considering CPU processing time, server scheduling queuing delay should also be considered. At the same time, the task generation time is independent of each other, and the number of task generations in disjoint interval is independent of each other. Therefore, in order to facilitate the calculation, the average waiting delay required to complete the computing task is modeled based on $M/M/1$ queuing system. Since the number of CPU cycles required to complete a task varies depending on the type of tasks, and in order to better apply to $M/M/1$ queuing system, it is assumed that the number of CPU cycles required by the task follows an exponential distribution. Since the processing speed is constant, the time to complete tasks also follows an exponential distribution. Since the arrival rate of computing tasks follows Poisson distribution, BS can model the calculation delay by $M/M/1$ queuing system.

Therefore, the average calculation delay of a task completed on BS_i is

$$d_i^t\left(\beta^t\right)=\frac{1}{u_i-\omega_i^t\left(\beta^t\right)} \tag{2}$$

where $\omega_i^t\left(\beta^t\right)$ is the amount of tasks that need to be processed on BS_i per unit time when the system offloading scheme is β^t. u_i is the expected task completion rate, that is, the amount of tasks that can be completed per second.

(2) Offloading computing

Due to the limited bandwidth of local area network, peer-to-peer offloading between BSs can also cause additional delays due to network congestion. At the same time, due to the low energy consumption of wired transmission, the transmission and reception energy consumption of BS is not considered, and only the transmission delay is considered. The transmission delay is only related to total traffic in LAN. Therefore, first define the total traffic as: $\lambda^t\left(\beta^t\right)=\sum_{i\in N}\lambda_i^t\left(\beta^t\right)$, where $\lambda_i^t\left(\beta^t\right)=\sum_{j\in N\setminus\{i\}}\beta_{ji}=\phi_i^t-\beta_{ii}^t$ represents the amount of tasks offloaded from BS_i to other BSs. In order to facilitate calculation, it is assumed that the data volume of the computing task is exponentially distributed. Due to the constant bandwidth transmission speed, the transmission service time also follows an exponential distribution. The average transmission delay can be obtained by modeling the $M/M/1$ queuing system:

$$d^{Tt}\left(\beta^t\right)=\frac{1}{\frac{1}{\tau}-\lambda^t\left(\beta^t\right)}=\frac{\tau}{1-\tau\lambda^t\left(\beta^t\right)},\quad \lambda^t<\frac{1}{\tau} \tag{3}$$

where τ is the service time of a task under non-congested conditions for network broadband. The reciprocal $\frac{1}{\tau}$ represents the system service rate.

In summary, in each time slot t, the delay generated by tasks on BS_i performing offload computing is the sum of delay transmitted by BS_i to BS_j and the delay that BS_j completes tasks:

$$D_i^t\left(\beta^t\right)=\lambda_i^t d^{Tt}\left(\beta^t\right)+\sum\nolimits_{j\in N}\beta_{ij}^t d_j^t\left(\beta^t\right) \tag{4}$$

The total energy consumption of BS_i is

$$E_i^l\left(\beta^t\right)=E_i^{lt} \tag{5}$$

In order to be more practical, it is assumed that the energy consumption of BSs is limited. However, in order to make the optimization of system more flexible, long-term online optimization of energy consumption of BSs was selected. That is, BS does not need to strictly abide by energy consumption constraints in each time slot, but only needs to make its average energy consumption meet energy consumption constraints in the long-term online optimization process [34]. The peer-to-peer offloading scheme will affect each other between different time slots due to energy consumption. If the current time slot consumes too much energy, the energy available in the future will decrease. In addition to this, BS summarizes and analyzes the lack of computing and storage resources in the past. At the same time, there is no effective model to predict future situations. Thus, Lyapunov drift plus penalty theory is used to optimize the energy consumption constraint problem. First, an energy consumption limit queue is defined, in which the initialization state is $Q(t)=\{q_i(t)\}_{i\in N}$ and the initialization state is $q_i(1)=0$, $\forall i\in N$. On this basis, the evolution process of energy consumption limit queue of each BS between time slots is

$$q_i(t+1)=\max\left\{q_i(t)+E_i^t\left(\beta^t\right)-\bar{E}_i,0\right\} \tag{6}$$

where $q_i(t)$ is the deviation of current energy consumption and long-term energy consumption constraints; $\bar{E}_i$ is the amount of long-term energy consumption constraints.

At the same time, because energy consumption and delay are important factors that affect whether the task chooses local computing or offloading computing, the energy consumption and delay are normalized, and BS_i running overhead is

$$C_i\left(\beta^t\right) = D_i^t\left(\beta^t\right) + q_i(t)E_i^t\left(\beta^t\right) \tag{7}$$

In order to facilitate analysis, the unit is unified, that is, the operating cost $C_i\left(\beta^t\right)$ multiplied by the parameter is converted into operating cost, as shown below:

$$\Psi_i' = \zeta_c \Psi_i\left(\beta^t\right) = \zeta_c\left[D_i^t\left(\beta^t\right) + q_i(t)E_i^t\left(\beta^t\right)\right] \tag{8}$$

where ζ_c is the parameter corresponding to the conversion of delay and energy cost into value cost.

3.3. **Waiting time prediction of MEC server.** When base stations receive task fragments from other base stations, if there are other task fragments being processed on base stations, the base station puts task fragments just received into the task queue. After the current task processing on base station ends, if a new task segment is assigned to base stations, base stations will rearrange the task processing order, processing time and processing frequency in the task queue [35].

As shown in Figure 2, assume that there are m task fragments in task queue of base station BS_1, and these m task fragments have different processing time requirements. The task shards in task queue are sorted from front to back according to task processing time. The task shard is denoted as $\bar{\gamma}_1, \bar{\gamma}_2, \ldots, \bar{\gamma}_m$, and the corresponding data volume is denoted as $\bar{D}_1, \bar{D}_2, \ldots, \bar{D}_m$. Suppose that base station BS_1 rearranges the task sharding processing in base stations at time T_r, and the cut-off time points of these task slicing processes are $\{T_1, T_2, \ldots, T_m\}$ respectively.

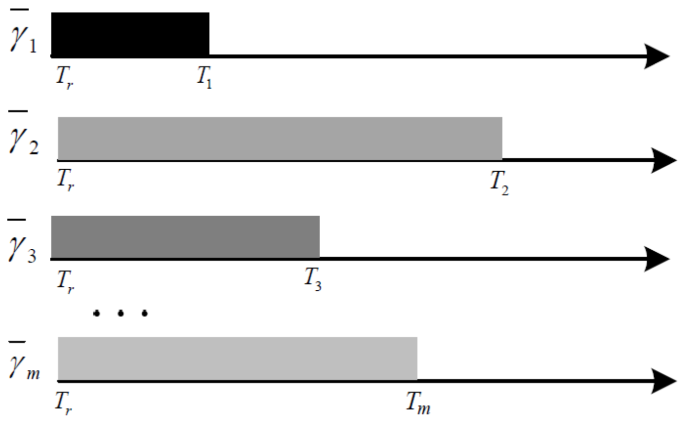


Figure 2. Task queue

Let $F = \left\{\bar{f}_1, \bar{f}_2, \ldots, \bar{f}_n\right\}$ denote the processing frequency adopted by base station BS_1 when processing m task fragments. The processing frequency F of base stations processing task fragment satisfies:

$$f_{\min} \le \bar{f}_i \le f_{\max} \tag{9}$$

Let $\delta = \{\bar{t}_1, \bar{t}_2, \ldots, \bar{t}_m\}$ be the actual length of processing time for task slice $\{\bar{\gamma}_1, \bar{\gamma}_2, \ldots, \bar{\gamma}_m\}$. The processing time and processing frequency of the same task shard satisfy:

$$\frac{\alpha \bar{D}_i}{\bar{t}_i} = \bar{f}_i \tag{10}$$

where α is the number of CPU operation rounds required to calculate each byte of data. Suppose task slice $\{\bar{\gamma}_1, \bar{\gamma}_2, \ldots, \bar{\gamma}_m\}$ starts to be processed at time point $\{B_1, B_2, \ldots, B_m\}$, so the processing time of tasks should satisfy

$$B_i + \bar{t}_i < T_i \quad (i \in \{1, 2, \ldots, M\}) \tag{11}$$

The energy consumption of all tasks in the task queue of base station processing is expressed as:

$$E_q(F) = k\sigma^2\alpha \sum_{i=1}^{n} \bar{D}_i \bar{f}_i^2 \tag{12}$$

where σ^2 is the background noise density.

4. Computing Offloading Algorithm Based on Game Theory.

4.1. Energy consumption constraint optimization.

Out of the elastic constraints on BS's long-term energy consumption, according to the Lyapunov drift penalty technique, an energy deficit queue is established for each BS. The energy consumption of BS is coupled with time, so that BS can meet the energy consumption constraint in the long-term optimization process. The specific content is shown in Figure 3.

In the long-term optimization process, each BS uses clock synchronization to trigger this part. The optimal offloading scheme is solved in each time slot. According to offloading

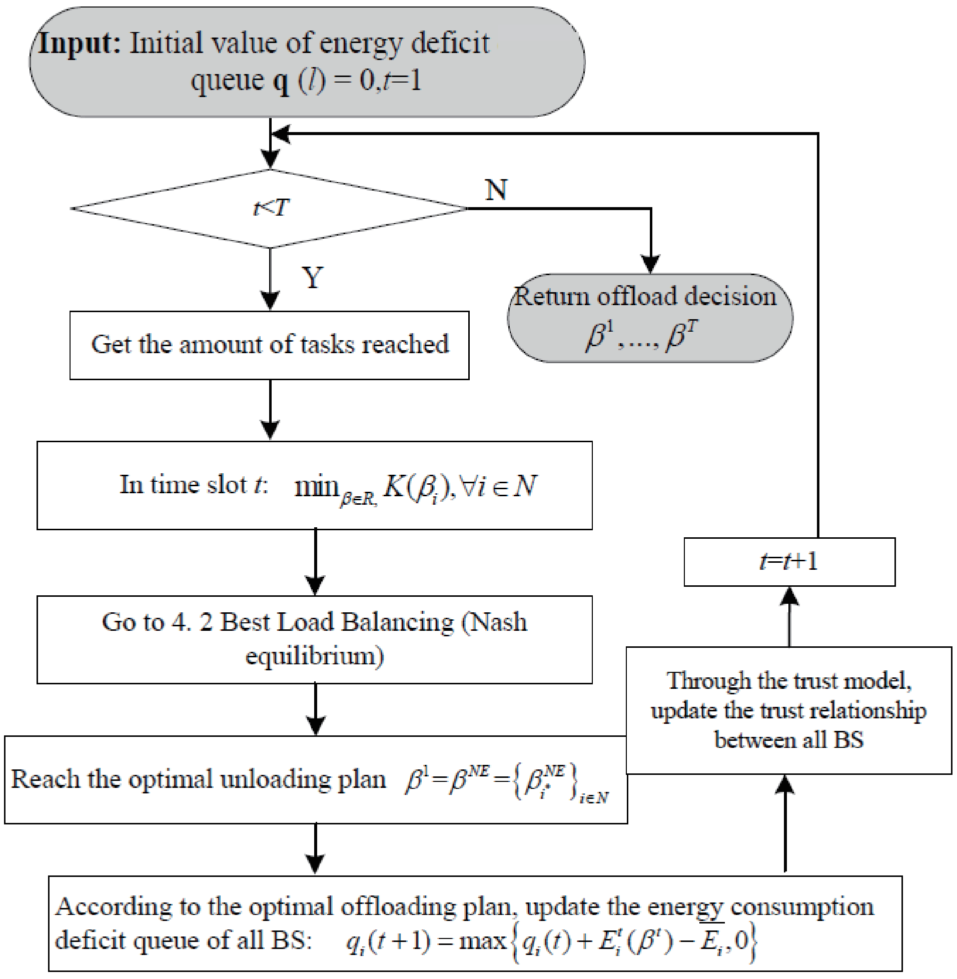

Input: Initial value of energy deficit queue $\mathbf{q}$ $(l) = 0, t=1$
$t<T$
N
Y
Return offload decision $\beta^1, \ldots, \beta^T$
Get the amount of tasks reached
In time slot t: $\min_{\beta \in R_i} K(\beta_i), \forall i \in N$
$t=t+1$
Go to 4. 2 Best Load Balancing (Nash equilibrium)
Through the trust model, update the trust relationship between all BS
Reach the optimal unloading plan $\beta^1 = \beta^{NE} = \{\beta_i^{NE}\}_{i \in N}$
According to the optimal offloading plan, update the energy consumption deficit queue of all BS: $q_i(t+1) = \max\{q_i(t) + E_i^t(\beta^t) - \underline{E}_i, 0\}$

FIGURE 3. Process of energy consumption constraint optimization

scheme, the energy consumed by each BS in current time slot for local computing can be obtained. Substitute it into Equation (12) to update respective energy consumption queues in order to control their own energy consumption in the next time slot [37]. In addition, because the offloading scheme of BSs will affect the trust relationship between BSs in the next time slot, the trust relationship between BSs will be updated after the energy consumption deficit queue is updated.

4.2. **Construction of multi-user game model.** Next, we build a multi-user game model. Let $\beta_{-n} = (\beta_1, \ldots, \beta_{n-1}, \beta_{n+1}, \ldots, \beta_N)$ denote the offloading decision of all users except user n. If user n gets decision result β_{-n} of everyone else, user n needs to make an offloading decision based on existing information. Or execute it on the local CPU ($\beta_n = -1$), then select a channel for task offloading ($\beta_n \geq 0$). The basis for decisions is as follows:

$$\min_{\beta_n \in A_n\{-1,0,1,\ldots,M\}} Z_n(\beta_n, \beta_{-n}), \quad \forall n \in N \tag{13}$$

where $Z_n(\beta_n, \beta_{-n})$ represents the load function of user n, which is defined as follows:

$$Z_n(\beta_n, \beta_{-n}) = \begin{cases} K_n^m, & \beta_n = -1, \\ K_n^s(\beta), & \beta_n \geq 0. \end{cases} \tag{14}$$

In this way, the offloading decision game of multi-user computing tasks can be constructed. $\Gamma = (N, \{\mathrm{A}_n\}_{n\in N}, \{Z_n\}_{n\in N})$, where N is the user set, A_n is the policy set, and Z_n is the minimum computing load for each user. Next, we introduce Nash equilibrium in this game.

Definition 4.1. *Decision set* $\mathrm{A}^* = (\beta_1^*, \ldots, \beta_n^*)$ *is a Nash equilibrium in the offloading decision game of multi-user computing tasks. If in the case of decision result set* A^*, *no user can reduce his computing load by changing his decision results, which is*

$$Z_n\left(\beta_n^*, \beta_{-n}^*\right) \leq Z_n(\beta_n, \beta_{-n}^*), \quad \forall \beta_n \in \mathrm{A}_n,\ n \in N \tag{15}$$

According to the existing research, it can be concluded that there is a Nash equilibrium in offloading decision game of multi-user computing tasks. At the same time, Nash equilibrium state can be reached by a limited number of iterations.

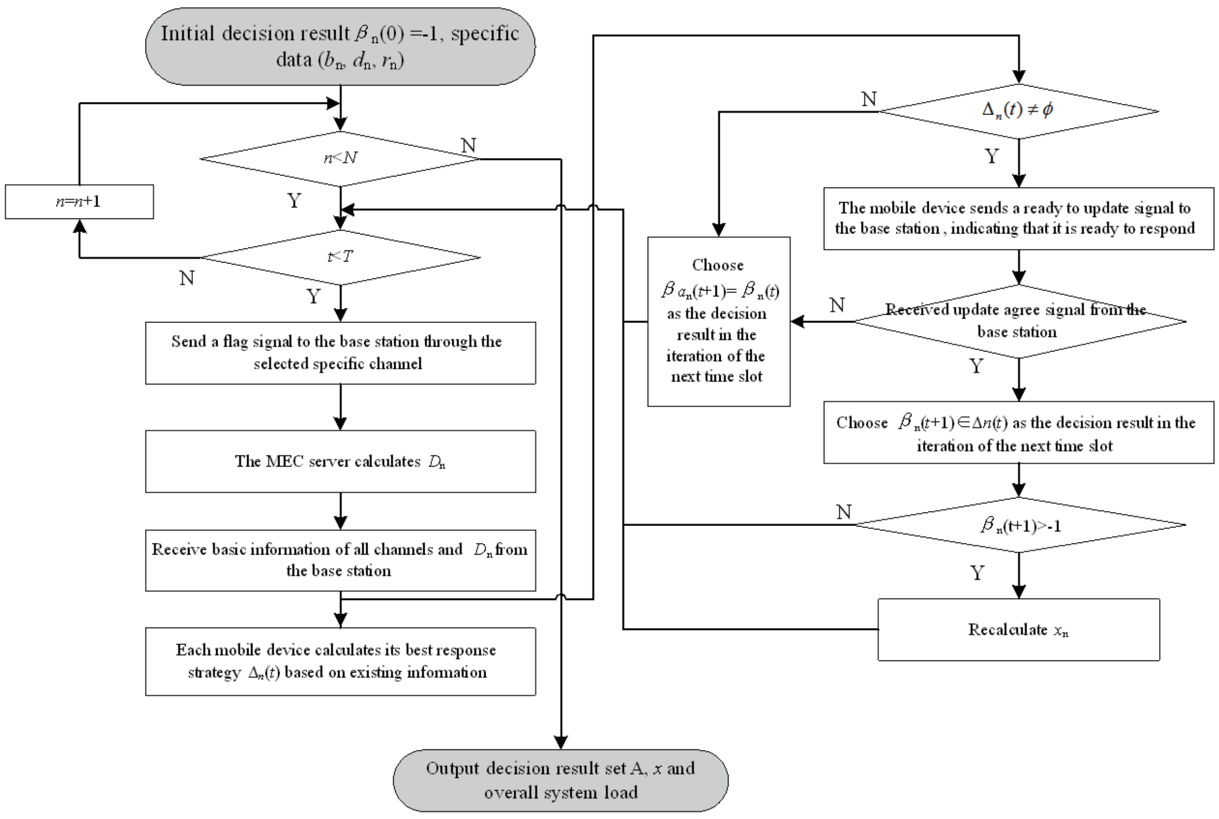
4.3. **Task offloading algorithm.** Next, a two-stage task offloading algorithm for solving Nash equilibrium is given.

The design of this algorithm is equivalent to a centralized distributed task. First, the base station in the system center has a time synchronization function. Therefore, the operation of all mobile devices can be synchronized by the time of base stations [38,39]. In each time slot, all mobile devices try to update their decision results to reduce the computational load. However, not all update requests can obtain the permission of central base station, so that in each time slot, there will be three steps to update the decision results.

Step 1: wireless interference measurement

In this stage, each mobile device will get the basic information of all channels from base stations. The mobile device can use this information to calculate channel interference. All mobile devices (that is $a_n(t) \geq 0$) that choose to offload at this time will send a flag signal to base stations. The flag signal may be channel ID selected by mobile devices. After receiving all sign signals, base stations can calculate the received power of each channel by the following formula:

$$p_m(\boldsymbol{a}(t)) \cong \sum_{i \in N\{n\}: a_i = a_n} q_i g_{i,s},\ m \in M \tag{16}$$



Initial decision result $\beta_n(0)$ =-1, specific data (b_n, d_n, r_n)
$n<N$
N
Y
$n=n+1$
$t<T$
N
Y
Send a flag signal to the base station through the selected specific channel
The MEC server calculates D_n
Receive basic information of all channels and D_n from the base station
Each mobile device calculates its best response strategy $\Delta_n(t)$ based on existing information
Output decision result set A, x and overall system load
$\Delta_n(t) \neq \phi$
N
Y
The mobile device sends a ready to update signal to the base station, indicating that it is ready to respond
Choose $\beta a_n(t+1) = \beta_n(t)$ as the decision result in the iteration of the next time slot
Received update agree signal from the base station
N
Y
Choose $\beta_n(t+1) \in \Delta n(t)$ as the decision result in the iteration of the next time slot
$\beta_n(t+1)>-1$
N
Y
Recalculate x_n

FIGURE 4. The process of task offloading algorithm

The base station then sends this information to all mobile devices. In this way, each mobile device n can calculate interference by the following formula:

$$\mu_n(m, a_{-n}(t)) = \begin{cases} p_m(\boldsymbol{a}(t)) - q_n g_{n,s}, & a_n(t) = m, \\ p_m(\boldsymbol{a}(t)), & a_n(t) \neq m. \end{cases} \tag{17}$$

In other words, for channel $a_n(t)$ selected by mobile device n, the resulting interference is equal to the total received power of channel minus the power of mobile device n. For other channels, interference is the received power of that channel.

Step 2: waiting time prediction of MEC

At this stage, MEC server needs to predict average waiting time. When service requirements are met, waiting time $D_n = 0$. When service requirements are not met, it is necessary to make predictions according to Formula (10) and send this data to the mobile device along with wireless interference in step one.

Step 3: offloading decision update

Each mobile device obtained the interference of each channel in phase one, and the waiting delay at MEC server in phase two. At this stage, each mobile device uses the above data to calculate optimal response set according to the following formula:

$$\Delta_n(t) \cong \left\{ \tilde{a} : \tilde{a} = \arg\min_{a \in \mathrm{A}_n} Z_n(a_n, a_{-n}) \text{ and } Z_n\left(\tilde{a}, a_{-n}(t)\right) < Z_n(a_n(t), a_{-n}(t)) \right\} \tag{18}$$

If calculated $\Delta_n(t)$ is not empty, it means that the mobile device has not reached Nash equilibrium state, and computing load can be reduced by updating decisions. Then the mobile device will select a decision result to send RTU signal to base stations. After receiving all RTU signals, the base station randomly selects one or more mobile devices that do not affect each other to allow decision updates. Other mobile devices that have not received UA signal will not update their decision in the next time slot. At the same time, for the users who need to wait, the second stage decision is made. Then after a limited number of iterations, all mobile devices will reach Nash equilibrium state. In other words, no mobile device can reduce its computing load by updating its own decisions. In this way, the algorithm solves the offloading decision problem of computing tasks in a multi-user scenario.

5. **Experimental Scheme Verification and Analysis.**

In order to verify the edge computing resource allocation algorithm based on game theory, this paper uses MATLAB to simulate it, where the size of simulation area is 1000m∗1000m, users and small base stations are randomly distributed in it. The relevant parameters used in the simulation are shown in Table 1.

Table 1. Simulation parameters

Parameter	Value	Parameter	Value
number of base stations	10	number of users	20
maximum number of users that base stations can access	3∼5	base station broadband	10∼15MHz
noise power	−75dBm	computing power of MEC	10∼15Gcycles/s
maximum number of users that can be served by MEC	3∼5	computing power of users	1∼2Gcycles/s
task input data volume	1∼2Mbits	task generation speed	$[0, 4]$/s
energy consumption per CPU cycle	8.2nJ	task expected transmission delay	100ms

5.1. **Iterative analysis.**

The number of iterations required for this system to reach Nash equilibrium state is shown in Figure 5 under the condition of different number of users.

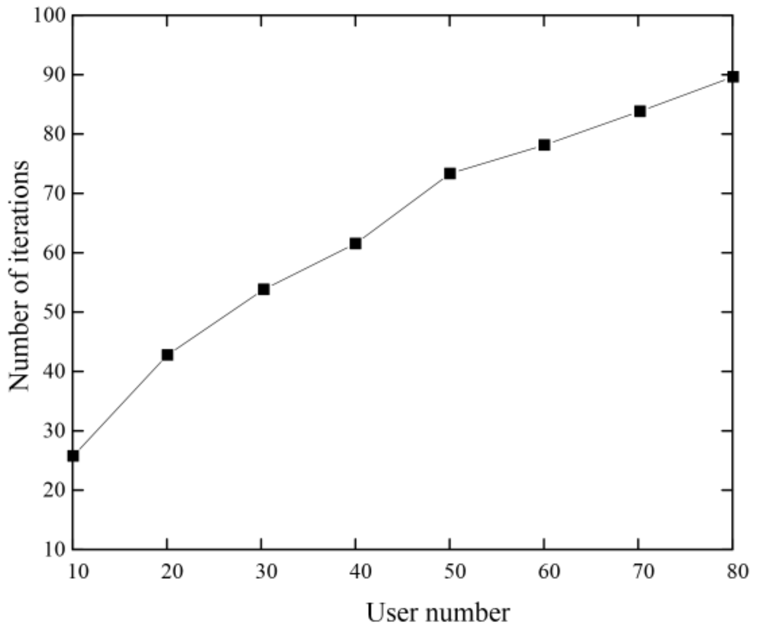


Figure 5. The number of iterations needed for the system to reach Nash equilibrium

It can be seen from the figure that as the number of users increases, the number of iterations required for this system to reach Nash equilibrium also increases with a linear trend. This shows that our proposed computing offloading and resource allocation algorithm for IoT device has very good performance.

At the same time, the relationship between task execution energy consumption and algorithm iteration times of the proposed algorithm and the two baseline algorithms is shown in Figure 6.

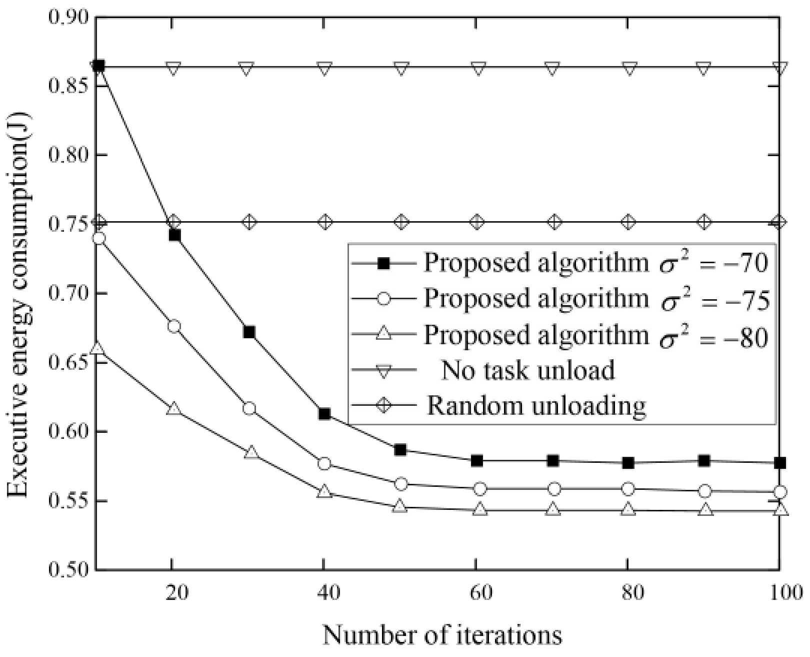


FIGURE 6. The relationship between task execution energy consumption and iteration times of algorithm

It can be seen from the figure that as the number of iterations of this algorithm increases, the energy consumption of task execution tends to converge in a smaller number of times. Comparing the task execution energy consumption under different channel noises, it can be seen that the task execution energy consumption decreases as channel noise decreases. The reason is that reduction in channel noise leads to an increase in user's transmission rate, which in turn leads to a delay in task execution and a reduction in task execution energy consumption. Both the non-computing offloading algorithm and random offloading algorithm are non-iterative algorithms, their task execution energy consumption does not change with the number of iterations.

5.2. **Parameter discussion.** In order to analyze how parameter ζ_c for the conversion of time delay and energy cost into value cost affects the energy consumption of distributed computing offloading algorithm, energy cost is compared by changing the value of conversion parameter. The results are shown in Figure 7 comparing the situation of 10 users to 50 users respectively, and assuming that the amount of computing task data for each mobile device is the same. The external environment is consistent, and this paper only changes the conversion parameter value.

It can be seen from the figure that as conversion parameter ζ_c increases, the energy consumption of distributed task offloading algorithm also increases accordingly. The larger conversion parameter ζ_c, the greater time delay and energy consumption. In actual

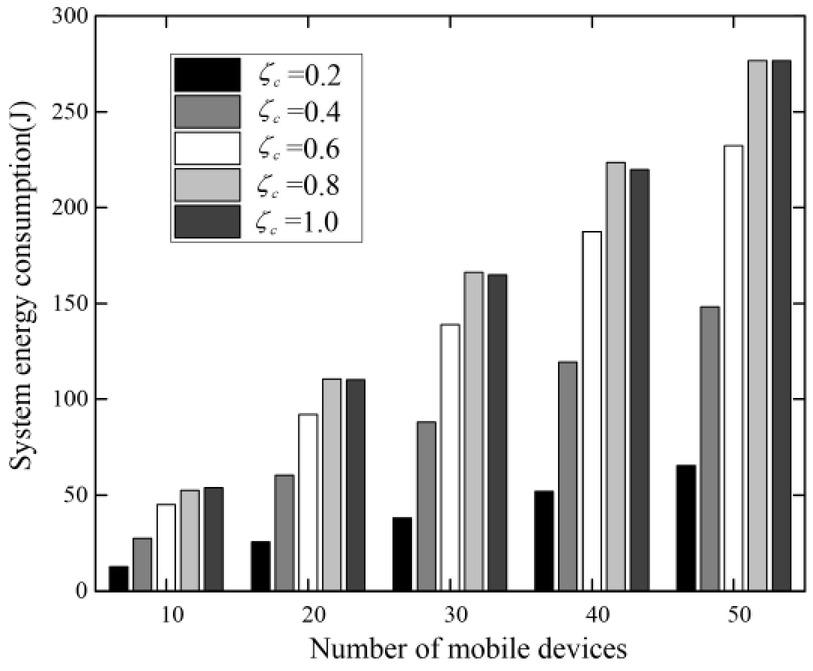


FIGURE 7. The relationship between execution time of recommendation algorithm and the size of datasets

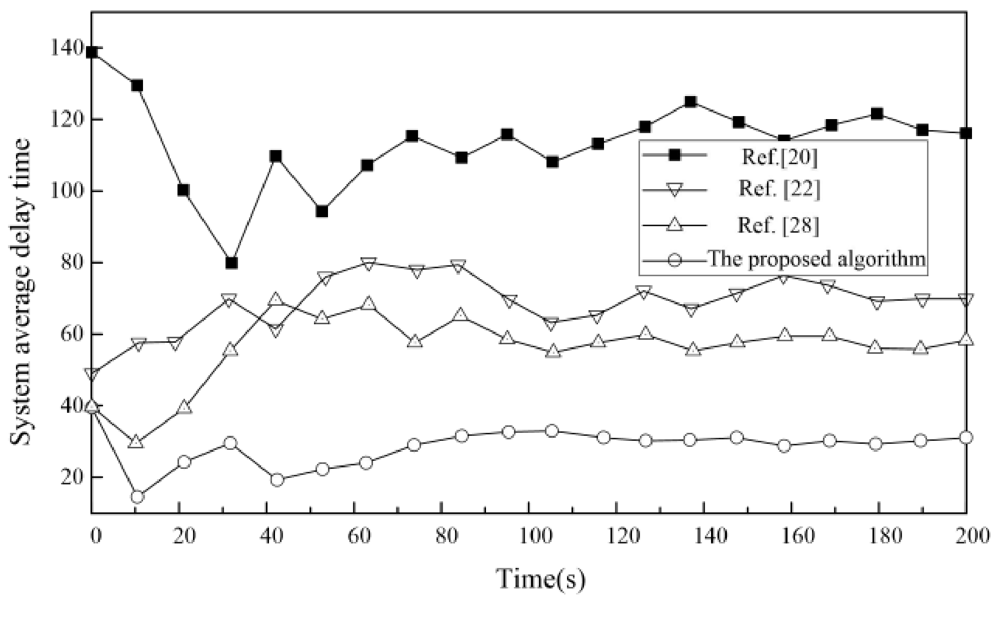


FIGURE 8. The average value of system delay

applications, it is necessary to adjust ζ_c according to the user's different computing tolerances for delay and energy consumption. In the case of meeting user's minimum delay requirements, it reduces the system energy consumption as much as possible.

5.3. **Average value of time delay.** In the time period of 0-200s, the average value of this system delay was compared and analyzed, and the results are shown in Figure 8.

[20] adopted a non-equivalent offloading algorithm, and each BS handles all tasks offloaded by end users under its jurisdiction. The amount of task arrivals shows spatiotemporal variability, which makes some BSs need to process more tasks, so it will take longer

to complete all tasks. Therefore, the average delay time of this system in each time period is longer. [22] and [28] used the energy consumption limit algorithm per unit time, which does not limit peer-to-peer offloading between BSs. However, [22] had strict energy consumption control in each time period, which is extremely inflexible. [28] needed to offload all remaining tasks to other BSs after calculating energy consumption to reach the limit. This will cause a large congestion delay in the network, making offloading computing delay likely to be greater than local computing delay. Our proposed algorithm comprehensively considers the effects of energy consumption and time delay; thus the system delay time of this algorithm is smaller compared to other algorithms. At the same time, it can be seen from the figure that the offloading scheme of each BS reaches Nash equilibrium by non-cooperative game.

5.4. **The relationship between task execution cost and the number of requested users.** Comparing our proposed algorithm with [28], [22] and [20], the relationship between task execution cost and the number of requested users is shown in Figure 9.

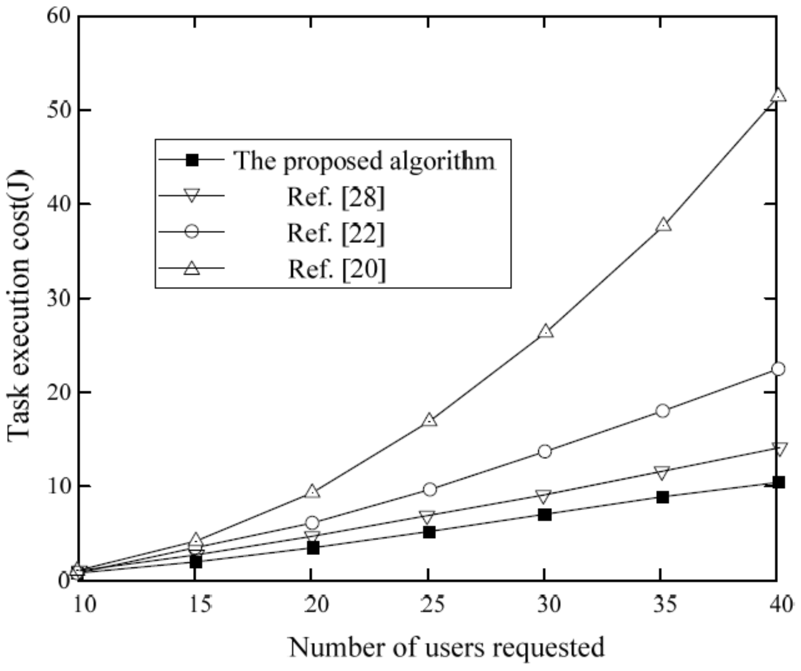


Figure 9. The relationship between task execution cost and the number of requested users

As can be seen from the figure, as the number of users increases, the task execution overhead increases. Compared with other algorithms, the task execution cost of our proposed algorithm is the lowest, and the performance gap of each algorithm increases as the number of users increases. The reason is that when the number of users increases, resource competition at MEC server will cause the system performance to degrade.

5.5. **Comparison of task execution energy consumption obtained by different algorithms in different situations.** In different situations, the energy consumption of task execution by different algorithms is shown in Figure 10. Two extreme cases are considered, the best case is that MEC and devices have sufficient computing resources and the channel is not congested. The worst case is insufficient computing resources and a lot of communication data.

As can be seen from the figure, the task execution energy consumption of algorithms in [20] under the best and worst cases is the highest among the four algorithms. The optimal

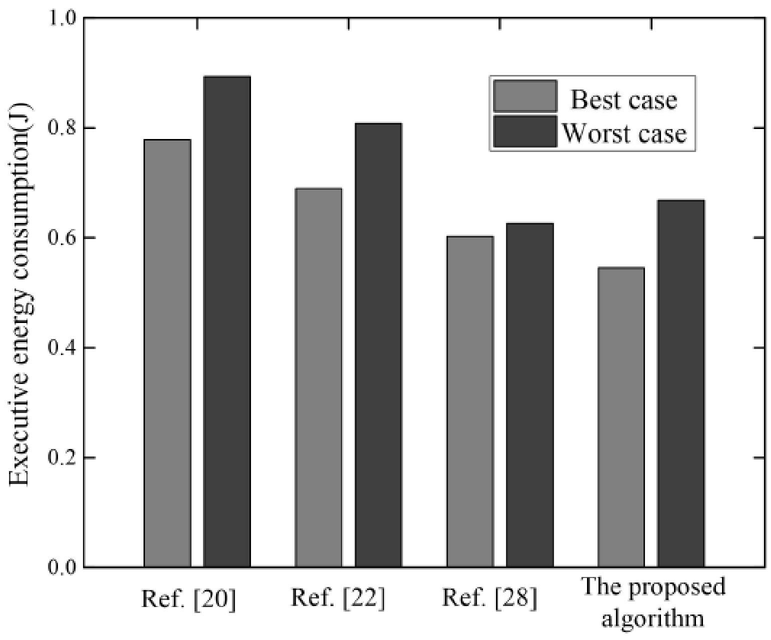


FIGURE 10. Comparison of the energy consumption of each algorithm in different situations

energy consumption of random task offloading algorithm in [22] is lower than that in [20], which can reflect to a certain extent that task offloading has the advantage of saving devices energy consumption. However, due to its failure to determine a joint strategy based on the channel gain between users and small cells and the load of MEC server, the worst case task execution energy consumption is close to the offloading algorithm in [20]. In addition, the task execution energy consumption obtained in [28] is lower than the energy consumption obtained by our proposed algorithm in the best case. However, the energy consumption difference of the proposed algorithm in the two cases is small, and its fairness is good. This is because [28] optimizes task offloading and resource allocation strategies for system task execution energy consumption and goal, and fails to take account of user fairness, which leads to higher energy consumption for user task execution in a worse situation than our proposed algorithm.

6. **Conclusion.** With the continuous maturity of technologies such as 5G communications and new storage systems, the widespread popularity of terminal devices that carry smart chips, the era of smart IoT has arrived. As an emerging technology, MEC reduces the pressure on core network by shifting computing load from core cloud data center to edge devices. This makes the network transmission cost lower and more efficient, and application complexity is not limited by terminals. Thus, this paper proposes a computing offloading and resource allocation algorithm using game theory for IoT device in MEC to solve the problem in offloading strategy and network resource allocation optimization in MEC system. Based on our proposed multi-user multi-MEC computing offloading and resource allocation system, the problem is modeled and a prediction method for waiting time of MEC server is designed to arrange the task processing sequence, time and processing frequency according to task queue in base stations. It improves the calculation efficiency of this algorithm and rationally uses the computing resources of devices and MEC. Furthermore, multi-user multi-MEC computing offloading and resource allocation problem is transformed into a multi-user game problem. When the system is in Nash equilibrium

state, the optimal solution to this problem is obtained, that is, the optimal device computing offloading and resource allocation scheme. Finally, simulation experiments based on MATLAB platform show that our proposed algorithm can reliably converge compared with other algorithms. Besides, the system energy consumption is small and time delay is short, and it can maintain good system performance in more extreme situations.

In a multi-user scenario, only one MEC server is considered. When multiple MEC servers are connected to a central cloud server at the same time, task offloading strategies may change differently. In addition, considering users' mobility, the task offloading strategy needs to involve communication switching and MEC server transfer. These are issues that need in-depth study in the future.

Acknowledgment. This work is supported by the fund of Minhang District Human Resources and Social Security Bureau, wireless intelligent handheld terminal based on RFID Technology (No. 11C26213100798) and RFID intelligent handheld mobile terminal and solution for food and drug traceability system (No. 1401H122500), Jiangsu University Natural Science Research Project (No. 18kjb5200001), also supported by the project fund of Shanghai Economic and Information Commission and the application of artificial intelligence in new retail.

REFERENCES

[1] P. Wang, C. Yao, Z. Zheng et al., Joint task assignment, transmission, and computing resource allocation in multilayer mobile edge computing systems, *IEEE IoT Journal*, vol.6, no.2, pp.2872-2884, 2019.

[2] 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.95, pp.522-533, 2019.

[3] Z. Liang, Y. Liu, T. M. Lok et al., Multiuser computation offloading and downloading for edge computing with virtualization, *IEEE Transactions on Wireless Communications*, vol.18, no.9, pp.4298-4311, 2019.

[4] R. Vijayalakshmi, V. Vasudevan, S. Kadry et al., Optimization of makespan and resource utilization in the fog computing environment through task scheduling algorithm, *International Journal of Wavelets Multiresolution & Information Processing*, vol.18, no.01, pp.37-42, 2019.

[5] A. Khalili, S. Zarandi and M. Rasti, Joint resource allocation and offloading decision in mobile edge computing, *IEEE Communications Letters*, vol.23, no.4, pp.684-687, 2019.

[6] M. Liu, F. R. Yu, Y. Teng et al., Distributed resource allocation in blockchain-based video streaming systems with mobile edge computing, *IEEE Transactions on Wireless Communications*, vol.18, no.1, pp.695-708, 2019.

[7] W. Fang, S. Ding, Y. Li et al., OKRA: Optimal task and resource allocation for energy minimization in mobile edge computing systems, *Wireless Networks*, vol.25, no.5, pp.2851-2867, 2019.

[8] Z. Yan, C. Pan, K. Wang et al., Energy efficient resource allocation in UAV-enabled mobile edge computing networks, *IEEE Transactions on Wireless Communications*, vol.18, no.9, pp.4576-4589, 2019.

[9] P. Paymard, S. Rezvani and N. Mokari, Joint task scheduling and uplink/downlink radio resource allocation in PD-NOMA based mobile edge computing networks, *Physical Communication*, vol.32, no.03, pp.160-171, 2019.

[10] A. Alnoman, S. Erkucuk and A. Anpalagan, Sparse code multiple access-based edge computing for IoT systems, *IEEE IoT Journal*, vol.6, no.4, pp.7152-7161, 2019.

[11] S. A. Alqahtani, Analysis of an adaptive priority-based resource sharing scheme for multiservice IoT communications over LTE-A networks, *Arabian Journal for Science and Engineering*, vol.44, no.4, pp.3457-3472, 2019.

[12] Y. Dai, D. Xu, S. Maharjan et al., Joint load balancing and offloading in vehicular edge computing and networks, *IEEE IoT Journal*, vol.6, no.3, pp.4377-4387, 2019.

[13] J. Du, F. R. Yu, X. Chu et al., Computation offloading and resource allocation in vehicular networks based on dual-side cost minimization, *IEEE Transactions on Vehicular Technology*, vol.68, no.2, pp.1079-1092, 2019.

[14] N. Cheng, F. Lyu, W. Quan et al., Space/aerial-assisted computing offloading for IoT applications: A learning-based approach, *IEEE Journal on Selected Areas in Communications*, vol.37, no.5, pp.1117-1129, 2019.

[15] X. Wu, W. Jiang, Y. Zhang et al., Online combinatorial based mechanism for MEC network resource allocation, *International Journal of Communication Systems*, vol.32, no.7, e3928, 2019.

[16] S. Mao, S. Leng, S. Maharjan et al., Energy efficiency and delay tradeoff for wireless powered mobile-edge computing systems with multi-access schemes, *IEEE Transactions on Wireless Communications*, vol.19, no.3, pp.1855-1867, 2020.

[17] Y. Y. Shih, H. P. Lin, A. C. Pang et al., An NFV-based service framework for IoT applications in edge computing environments, *IEEE Transactions on Network and Service Management*, vol.16, no.4, pp.1419-1434, 2019.

[18] Y. He, J. Ren, G. Yu et al., D2D communications meet mobile edge computing for enhanced computation capacity in cellular networks, *IEEE Transactions on Wireless Communications*, vol.18, no.3, pp.1750-1763, 2019.

[19] H. A. Alameddine, S. Sharafeddine, S. Sebbah et al., Dynamic task offloading and scheduling for low-latency IoT services in multi-access edge computing, *IEEE Journal on Selected Areas in Communications*, vol.37, no.3, pp.668-682, 2019.

[20] C. Li, W. Chen, J. Tang et al., Radio and computing resource allocation with energy harvesting devices in mobile edge computing environment, *Computer Communications*, vol.145, no.09, pp.193-202, 2019.

[21] C. Wu and Y. Zhang, Toward efficient transparent computing for IoT apps by on-chip kernel offload, *IEEE IoT Journal*, vol.6, no.3, pp.4085-4097, 2019.

[22] B. Yang, X. Cao, X. Li et al., Mobile-edge-computing-based hierarchical machine learning tasks distribution for IIoT, *IEEE IoT Journal*, vol.7, no.3, pp.2169-2180, 2020.

[23] Y. Chen, N. Zhang, Y. Zhang et al., Dynamic computation offloading in edge computing for IoT, *IEEE IoT Journal*, vol.6, no.3, pp.4242-4251, 2019.

[24] C. C. Chang, W. K. Lee, Y. Liu et al., Signature gateway: Offloading signature generation to IoT gateway accelerated by GPU, *IEEE IoT Journal*, vol.6, no.3, pp.4448-4461, 2019.

[25] L. Zhao, J. Wang, J. Liu et al., Optimal edge resource allocation in IoT-based smart cities, *IEEE Network*, vol.33, no.2, pp.30-35, 2019.

[26] H. Karami, S. F. Mousavi, A. Zarei et al., Optimal reservoir operation using bat and paper swarm algorithm and game theory based on optimal water allocation among consumers, *Water Resources Management*, vol.33, no.9, pp.3071-3093, 2019.

[27] R. Sun, Z. Wei, Z. Lyu et al., Power control algorithm based on non-cooperative game theory in successive interference cancellation, *Wireless Networks*, vol.25, no.6, pp.3297-3305, 2019.

[28] C. You, K. Huang, H. Chae et al., Energy-efficient resource allocation for mobile-edge computation offloading, *IEEE Transactions on Wireless Communications*, vol.16, no.3, pp.1397-1411, 2017.

[29] 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.01, pp.70-82, 2019.

[30] C. Shu, Z. Zhao, Y. Han et al., Multi-user offloading for edge computing networks: A dependency-aware and latency-optimal approach, *IEEE IoT Journal*, vol.7, no.3, pp.1678-1689, 2020.

[31] Z. Xiao, X. Dai, H. Jiang et al., Vehicular task offloading via heat-aware MEC cooperation using game-theoretic method, *IEEE IoT Journal*, vol.7, no.3, pp.2038-2052, 2020.

[32] S. Meng, Y. Wang, L. Jiao et al., Hierarchical evolutionary game based dynamic cloudlet selection and bandwidth allocation for mobile cloud computing environment, *IET Communications*, vol.13, no.1, pp.16-25, 2019.

[33] E. Abdelhalim, M. Obayya and S. Kishk, Distributed fog-to-cloud computing system: A minority game approach, *Concurrency Practice & Experience*, vol.31, no.15, e5162, 2019.

[34] X. Cai, X. Liu and Z. Qu, Game theory-based device-to-device network access algorithm for heterogeneous networks, *Journal of Supercomputing*, vol.75, no.5, pp.2423-2435, 2019.

[35] J. Mei, C. Chen, J. Wang et al., Coalitional game theory based local power exchange algorithm for networked microgrids, *Applied Energy*, vol.239, no.03, pp.133-141, 2019.

[36] A. Nagurney, M. Salarpour and P. Daniele, An integrated financial and logistical game theory model for humanitarian organizations with purchasing costs, multiple freight service providers, and budget, capacity, and demand constraints, *International Journal of Production Economics*, vol.212, no.06, pp.212-226, 2019.

[37] Y. Pang, N. Wang and H. Xia, A game theory approach for secure control of cyber-physical systems, *Zidonghua Xuebao/Acta Automatica Sinica*, vol.45, no.1, pp.185-195, 2019.
[38] S. Riahi and A. Riahi, Game theory for resource sharing in large distributed systems, *International Journal of Electrical and Computer Engineering*, vol.9, no.2, pp.1249-1257, 2019.
[39] C. Li, W. Chen, H. Tang et al., Stochastic computation resource allocation for mobile edge computing powered by wireless energy transfer, *Ad Hoc Networks*, vol.93, 101897, 2019.