Hindawi
Wireless Communications and Mobile Computing
Volume 2022, Article ID 1117597, 14 pages
https://doi.org/10.1155/2022/1117597

Research Article

A Two-Way Update Resource Allocation Strategy in Mobile Edge Computing

Zhe Yang,[1,2,3] Yi-peng Chen,[1,2,3] Fei Gu,[1,2] Jin Wang,[1,2] and Lei Zhao[1,2,3]

[1]*School of Computer Science and Technology, Soochow University, Suzhou 215006, China*
[2]*Provincial Key Laboratory for Computer Information Processing Technology, Soochow University, Suzhou 215006, China*
[3]*Provincial Laboratory for Big Data Intelligent Engineering, Soochow University, Suzhou 215006, China*

Correspondence should be addressed to Zhe Yang; yangzhe@suda.edu.cn

Received 25 April 2022; Revised 2 June 2022; Accepted 21 June 2022; Published 13 July 2022

Academic Editor: Kalidoss Rajakani

Existing studies of mobile edge computing resource allocation strategy problem merely optimize delay and energy cost, seldom considering the benefit of edge servers. So, a two-way update strategy based on game theory (TUSGT) was proposed. TUSGT converts the task competition relationship among edge servers into a noncollaborative game issue and adopts a potential game-based joint optimization strategy, allowing edge servers to determine task selection preference by maximizing their own benefit as the objective. At mobile device side, the EWA algorithm of online learning was used to update parameters, exerting impact on edge server's task selection preference from a global perspective and improving overall deadline hit rate. The simulation test results show that, compared to BGTA, MILP, greedy strategy, random strategy, and ideal strategy, TUSGT promotes deadline hit rate by up to 30% and increases edge server's average benefit by up to 65%.

1. Introduction

With the increased network bandwidth and popularized mobile smart devices, the number of computation intensive applications based on mobile devices is also growing rapidly [1]. These applications have large computational work and high requirement for real time [2, 3]. But restricted by computation capability and energy cost, it is difficult to complete computing tasks solely on mobile devices. So the mobile cloud computing architecture emerges, aimed at using cloud resources to carry out computation-intensive tasks, so as to optimize user experience [4, 5] and relieve insufficient computation capability of mobile device. However, due to long distance between cloud resources and mobile devices, the transmission of large amount of data leads to time delay requirement hard to be met. Thus, in 2014, the European Telecommunication Standards Institute put forward a Mobile Edge Computing (MEC) architecture [6], providing nearby computing resource-edge server for mobile devices. These edge servers own more computing resources compared to mobile devices, so mobile devices download computation-intensive tasks to nearby edge server, greatly reducing task completion delay and mobile devices' energy cost [7].

Nevertheless, MEC's computing resources are not unlimited, and an edge server is also hard to meet the computing need of all mobile devices surrounding it; thereby, the studies on task offloading and resource allocation strategy are of great significance to promote overall MEC performance [8–10] and it is also one of the hot spots in MEC research field [11]. Most of existing task offloading and resource allocation strategies take task delay and mobile devices' energy cost as the primary optimization objective, rarely considering the benefit of edge servers, i.e., the cost benefit of edge servers. It is an idealized hypothesis to let edge server complete mobile devices' computing tasks regardless of benefit, and this is impossible to realize in real application scenarios. As letting edge server to perform a task will inevitably produce energy cost overhead, if an edge server is made to perform a task without any benefit, it will choose to not perform any task so as to reduce overhead.

By introducing benefit factor into MEC system, edge servers will only make the best choice for themselves, since their goal is to maximize their own benefit. But for mobile devices, MEC system is expected to ensure that more tasks can be completed on time by task offloading and resource allocation. Therefore, how to guarantee edge server benefit while getting more mobile device tasks to be completed on time as possible is a crucial target.

In this paper, a two-way update strategy based on game theory (TUSGT) was proposed to solve the above problem. The main work of this paper is as follows:

(1) In a multiuser and multiedge server MEC system, a new TUSGT resource allocation strategy was raised. It converts task competition relationship between edge servers into a noncollaborative game problem and designs a corresponding strategy according to the game characteristics to solve the problem, in which every edge server takes maximizing its own benefit as objective to determine task selection preference. Meanwhile, the system uses exponent weight algorithm (EWA) and integrates with task completion condition to update parameters, so as to further increase overall deadline hit rate; that is to say, make more mobile device tasks be completed on time

(2) We analyzed and proved the existing of unique Nash equilibrium in noncollaborative game problem of edge server, enabled different edge servers to eventually reach to Nash equilibrium steady state through strategy design, and testified its convergence

(3) Through a simulation test, we verified the convergence of TUSGT strategy and proved its validity by comparing the deadline hit rate of five datum strategies and the average benefit of edge servers

In this paper, Section 2 introduces the related work of resource allocation in MEC; Section 3 describes the system model in MEC; Section 4 provides the idea and relevant certificate of TUSGT; Section 5 shows the result of simulation test and evaluates TUSGT performance in details.

2. Related Work

In existing studies of resource allocation strategies, the main optimization objectives are task delay and systematic energy cost.

Regarding the research of task processing delay optimization, the literature [11] designed a vertically and horizontally synergistic network architecture based on vehicle-mounted edge computing network architecture. Then, by analyzing the relationship between communication, cache, and computing resources, a joint optimization model of the three was proposed, and the asynchronous distributed reinforcement learning was used to realize the solution to task offloading and resource management strategy. This model can obviously reduce overall time delay. The literature [12] expresses the heterogeneous task offloading problem in distributed edge computing environment as a multiperson game problem, and every participator makes offloading decision according to incomplete information. Based on this, the authors designed a heterogeneous edge computing task offloading strategy based on minority game. In this strategy, subtasks are grouped to compete with other tasks for resources. This strategy can reduce more task processing delay.

Regarding the research of systematic energy cost optimization, the literature [13] put forward a convex optimization problem and used offloading priority function to solve the problem. This function sets priority for users, which relies on their own channel gains and local computing energy cost. The author also proposed a low-complexity suboptimal strategy, utilizing average subchannel gains to solve the corresponding mixed integer optimization problem. This strategy can significantly optimize energy cost. The literature [14] considered two different resource allocation strategies for joint optimization: the first strategy paid attention to fairness between users, and the second one paid attention to systematic energy efficiency. Besides, the author expressed energy cost optimization problem as a nonconcave fraction programming problem in collaborative strategies and used a low-complexity optimization strategy to realize optimal allocation. This scheme improves the energy efficiency of wireless mobile edge computing system. According to the problems of high time delay, high energy cost, and low reliability, the literature [15] proposed an offloading model considering time delay and energy cost as well as a dynamic game allocation model based on mobile terminal's reputation value. The improved particle swarm optimization algorithm and Lagrange multiplier method were used to solve this model, respectively. This scheme can effectively reduce total systematic energy cost and enhance offloading and allocation reliability.

All the above research work assume that edge servers would complete any computing tasks from mobile devices unconditionally, without considering their own benefits. This is an idealistic situation and cannot be realized in real application scenarios.

In recent years, some researchers have started to take the edge server benefit into consideration. The literature [16] put forward an auction-based edge computing resources market. This auction mechanism can guarantee authenticity and computing efficiency while maximizing overall benefit, and edge servers will be allocated to the device willing to pay the highest cost. This auction strategy can effectively solve the problem of maximizing the benefit of edge computing service provider. The literature [17] developed a deep learning-based optimal auction strategy for edge resource allocation. It utilized neural network to convert bids and used their own estimation as training data to adjust neural network parameters, so as to optimize loss function. Finally, edge servers were allocated to corresponding devices according to obtained allocation strategy and acquired corresponding benefits. This strategy can effectively promote total benefit of system. The literature [18] designed multi-item auction and data offloading strategy based on congestion game, aiming to maximize the income of mobile operator and reduce the payment of mobile users. This strategy can effectively increase total income of mobile operator and bring down the payment of mobile users to a certain extent. Though above research work have considered the benefit of edge server, they have not taken task delay requirement into full account.

The current research focuses on optimization of delay and energy consumption, but not enough consideration for the benefits of edge servers. Therefore, the starting point of this paper is to optimize the benefits of edge servers. The delay is an important performance evaluation index of the MEC system. Because the mobile device unloads the task, it must return the calculation result within a certain period of time. And while optimizing the revenue of the edge server, it will have a certain impact on the delay, so this paper selects these two indicators as the optimization goals.

The benefit includes a delay factor, which is to allow the edge server to choose the tasks that it can complete as soon as possible. However, if the strategy is only determined according to the initial conditions, limited by the incomplete information of the edge server, there must be a lot of selection conflicts. At this time, the adjustment of the penalty coefficient is to balance the conflict and the benefits of the edge server. And solving the conflict of choice of tasks can significantly improve the overall task completion rate and the overall average benefit of the MEC system. The proposed TUSGT considers task delay requirement and edge server benefit need simultaneously, converts the natural competition relationship between edge servers into a noncollaborative game problem, and designs a game theory-based strategy to solve the problem, maximizing edge server benefit and making more computing tasks be completed as possible.

3. System Model

3.1. Network Architecture. The proposed MEC network architecture is shown in Figure 1. Mobile devices transmit task information require to offload into management server; edge server obtains task information from management server and then sends its choice to the latter. Management server takes charge in collecting edge server choice, updating resource allocation strategies, and updating subsequent parameters. After resource allocation strategy is determined, an edge server starts to perform tasks and transmits the results to corresponding mobile devices.

3.2. Task Model. In the proposed system model, a group of M mobile devices is represented by $MD = \{MD_1, MD_2, \cdots, MD_M\}$; a group of N edge servers is represented by $ES = \{ES_1, ES_2, \cdots, ES_N\}$. Therein, every mobile device only holds one atomic task, and all mobile devices' task set is expressed as $\text{Task} = \{T_1, T_2, \cdots, T_M\}$. Set the information cell group of the m^{th} task as $T_m = \{VI_m, VO_m, L_m, \Omega_m, R_m\}$, in which VI_m denotes to the input data size of task m, VO_m denotes to the output data size of task m, L_m denotes to the required operand of task m, Ω_m denotes to the time delay requirement of task m, and R_m denotes to the reward paid by task m.

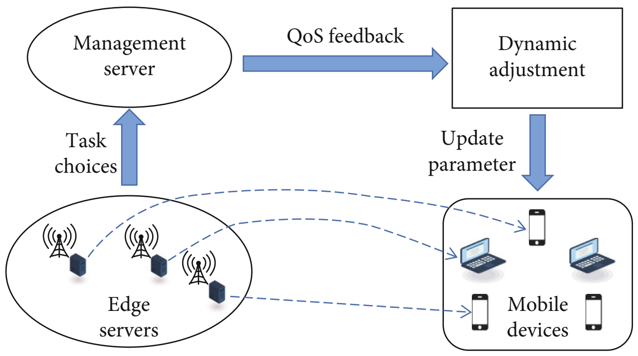

Figure 1: MEC network architecture.

3.3. Communication Model. The paper supposes a mobile device only communicates with one edge server, so when uploading or downloading data, there is no interference from other device transmission. Define the transmission rate between mobile device m and edge server n as shown in Equation (1) [19]:

$$d_{m,n} = B \log_2\left(1 + \frac{p_m g_{m,n}}{\sigma^2}\right), \tag{1}$$

in which B is the bandwidth, p_m is the transmission rate, $g_{m,n}$ represents the channel gains, and σ^2 is the noise power. The calculation formula of channel gain $g_{m,n}$ is shown in Equation (2) [19]:

$$g_{m,n} = G\left(\frac{3 \times 10^8}{4\pi F_c dist_{m,n}}\right)^{PL}, \tag{2}$$

in which G is the antenna gain, F_c is the carrier frequency, $dist_{m,n}$ represents the distance between mobile device m and edge server n, and PL is the path loss factor. According to Equations (1) and (2), the data transmission rate can be calculated by the distance between mobile device and edge server.

3.4. Time Delay and Energy Cost Model. The local computing delay of task is shown in Equation (3), in which f_m is the computing rate of mobile device m.

$$D_m^{\text{local}} = \frac{L_m}{f_m}. \tag{3}$$

In case that task m is offloaded to edge server n, then the corresponding task delay is shown in

$$D_{m,n}^{\text{off}} = D_{m,n}^u + D_{m,n}^d + D_{m,n}^c, \tag{4}$$

$$D_{m,n}^u = \frac{VI_m}{d_{m,n}}, \tag{5}$$

$$D_{m,n}^d = \frac{VO_m}{d_{m,n}}, \tag{6}$$

$$D^{c}_{m,n} = \frac{L_m}{f_n}, \tag{7}$$

in which $D^{u}_{m,n}$ is the task uploading time, $D^{d}_{m,n}$ is the result downloading time, $D^{c}_{m,n}$ is the task computing time on edge server, and f_n is the computing rate of edge server.

On account of the benefit of edge server, it is necessary to figure out the energy cost overhead of edge server, and the greater the energy cost overhead is, the smaller the benefit will be. The energy cost overhead of edge server is shown in

$$c_{m,n} = P_{\text{com},n} \times D^{c}_{m,n} + P_{\text{tra},n} \times \left(D^{u}_{m,n} + D^{d}_{m,n}\right), \tag{8}$$

in which $P_{\text{com},n}$ is the computing power of edge server n and $P_{\text{tra},n}$ is the transmission power of edge server n.

3.5. Benefit and Overhead Model. The cost overhead that an edge server n completes the offloading task of mobile device m is shown in

$$\pi_{m,n} = \begin{cases} c_{m,n}, d_{m,n} > 0, \\ \infty, d_{m,n} < 0. \end{cases} \tag{9}$$

That is to say, when the communication rate between edge server n and mobile device m is normal, the overhead is the cost of edge server; when edge server n cannot communicate with mobile device m, the overhead of edge server is infinite.

Edge servers compete for task according to overhead function (9), and ever edge server attempts to find the optimal strategy, minimizing its overhead. This causes all edge servers do not receive any task to save overhead (not receiving any task means no cost overhead to be paid). However, this result clearly contradicts the purpose of MEC system, and it makes no sense for mobile device to offload task, so mobile devices need to provide certain rewards. The specific benefit obtained by an edge server can be calculated by the initial reward R_m of mobile device task and a penalty function $\text{punish}_{m,n}$ [20]. The computing method of penalty function is shown in

$$\text{punish}_{m,n} = \begin{cases} \dfrac{\lambda_m \Omega_m}{\Omega_m - D^{\text{off}}_{m,n} + \omega}, & D^{\text{off}}_{m,n} \le \Omega_m, \\ \infty, \quad D^{\text{off}}_{m,n} > \Omega_m, \end{cases} \tag{10}$$

in which Ω_m is the required deadline of task m and $D^{\text{off}}_{m,n}$ is the completion time for the server n to complete the offloaded task. ω is a very small nonzero positive number, used to prevent denominator as zero when offloading task completion delay just satisfies task delay requirement. λ is the penalty coefficient, used for dynamic adjustment by reverse update method (see details in Section 4.2). This penalty function is set to give different penalties based on the speed at which edge server completes the task; i.e., the more time intervals left for edge server to complete task, the smaller the penalty will be. Next, it is needed to set a benefit indicator function $u_{m,n}$, by computing which an edge server can make its decision. The benefit indicator function $u_{m,n}$ is defined as shown in

$$u_{m,n} = \begin{cases} \dfrac{R_m \times \left(\Omega_m - D^{off}_{m,n} + \omega\right)}{\lambda_m \times \Omega_m \times N(m) \times c_{m,n}}, & \pi_{m,n} \ne \infty, \\ 0, \quad \pi_{m,n} = \infty, \end{cases} \tag{11}$$

in which $N(m)$ represents the number of edge servers choosing the task m; i.e., the more tasks are selected, the lower the value of the reward indicator function. Benefit refers to what the edge server can ultimately get. The benefit contains multiple factors; the purpose is to allow multiple factors to affect the edge server comprehensively. These factors are of similar importance and can affect the selection of edge servers.

3.6. Task Offloading Model. The system model in this paper is referred to [21, 22]. In order to verify the effect of the scheme, it is considered that the task of the mobile device is inseparable; that is, the task cannot be divided into multiple edge servers for completion. And if the edge server selects the mobile device task, it uses all its computing resources for computing, so one edge server can receive at most one task at the same time. On the basis of above agreement, the paper expresses the allocation strategy of mobile device m as $a_m = \{a_{m,0}, a_{m,1}, \cdots, a_{m,N}\}$, in which $a_{m,n} = \{0, 1\}$; the value of $a_{m,0}$ represents whether to calculate locally. Therefore, the total strategy configuration of MEC system can be obtained as $A = \{a_1, a_2, \cdots, a_M\}$, satisfying $\sum_{i=1}^{M} a_{i,n} \le 1, \forall n \in ES$ and $\sum_{i=1}^{N} a_{m,i} \le 1, \forall m \in \text{Task}$.

3.7. Objective Function. In the competition game on edge server side, the objective of every edge server is to maximize its own benefit. So, the general objective on edge server side is shown in

$$\text{benefit}_{m,n} = \begin{cases} \dfrac{R_m \times \left(\Omega_m - D^{\text{off}}_{m,n} + \omega\right)}{\Omega_m \times c_{m,n}}, & \pi_{m,n} \ne \infty, \\ 0, \quad \pi_{m,n} = \infty, \end{cases} \tag{12}$$

$$\text{maximize} : \sum_{i=1}^{N} \sum_{m \in S_i} \text{benefit}_{m,i}, \tag{13}$$

in which S_i represents the task set selected by edge server i, which is the subset of Task, and $\text{benefit}_{m,i}$ represents the benefit obtained by edge server i to complete mobile device m task. This objective function represents maximizing the total edge server benefit on the edge server side.

On mobile device side, every mobile device needing to offload tasks expects its task to be completed. Thus, the total objective function on mobile device side is shown in

$$\text{maximize} : \sum_{i=1}^{M} \theta_i. \tag{14}$$

θ_i is a binary variable. If the task of mobile device i can be completed within required time delay, then its value is

1; otherwise, it is 0. This objective function represents making more offloading tasks to be completed on time as possible.

4. Two-Way Update Resource Allocation Strategy

The TUSGT designed in this paper optimizes resource allocation strategies from objective functions (13) and (14), respectively. One is to optimize on edge server side, and the other is to update parameters on mobile device side, as shown in Figure 1.

On edge server side, the edge servers do not know each other's configuration and state information, which is a non-collaborative game problem. Since a large number of edge servers may not belong to the same manufacturer in practice, each of them needs to consider their own interests, so it will not be a complete cooperative game problem. In order to obtain higher benefits for itself, the edge server will not hand over all its configuration information to the management server but will only select the most suitable task according to its own configuration. The management server only receives the task selection of the edge server but does not have the configuration information of the edge server. So, the paper designed a game-based joint optimization strategy (GBJOS) to solve this problem. When the GBJOS converges, all edge servers will not actively apply for strategy modification. On mobile device side, the classic EWA algorithm [20] in online learning can be used for updating, to satisfy the objectives on mobile device side.

4.1. Game-Based Joint Optimization Strategy

4.1.1. Game Model. This paper established a game model of multiuser and multiedge server resource allocation in MEC and then made definition for it, as so to analyze this game model.

Definition 1. The offloading task set is $\text{Task} = \{T_1, T_2, \cdots, T_M\}$. When mobile device m does not have enough local resources to complete the task, i.e., $D_m^{\text{local}} \geq \Omega_m$, the mobile device m will decide to offload the task. At this moment, this mobile device will be added to MD set; its task and task-related information will be added in the Task set.

Definition 2. After completing the task m, if edge server n' obtains higher benefit than edge server n, then task m will be allocated to edge server n'.

Definition 3. If in current resource allocation strategy of MEC system, the task m has been allocated to edge server n, then edge server n calculates the value of benefit indicator function (11) for task m; its $N(m)$ value is 1. Otherwise, when this server performs calculation, the $N(m)$ value should be a normal accumulated value.

Definition 4. Task selection threshold N_TO. When the number of mobile device m task selected reaches to the threshold N_TO, that's to say, $N(m) \geq N\_TO$, other edge servers are not allowed to choose the task.

From Definition 1, it is known that now all tasks in Task set need to perform offloading computing, and edge servers will compete for tasks in Task set to gain benefit. Definition 2 ensures to maximize the total benefit of edge servers, that's to say, if the selection of different edge servers conflicts, the task will be allocated to the one with greater benefit. Definition 3 is set to increase the cost of selection conflict between different edge servers, so that the selection of edge servers can be dispersed as possible. Definition 4 is to prevent this game from falling into an infinite loop and guarantee every edge server must complete the final selection within a certain number of times. The specific value of threshold N_TO 的 shall be adjusted according to real data setup.

According to the above definitions, the following lemma is given:

Lemma 5. *After all edge servers hold tasks, they will tend to keep the currently selected task in subsequent selections.*

Proof. Suppose that the server n currently chooses task k and have held it. In the next round of selection, if the server n selects the task k', then there must be:

$$u_{k,n} < u_{k',n} \tag{15}$$

Combined with Eq.(11) and Definition 3, it is obtained that:

$$\frac{R_k \times \left(\Omega_k - D_{k,n}^{off} + \omega\right)}{\lambda_k \times \Omega_k \times 1 \times c_{k,n}} < \frac{R_{k'} \times \left(\Omega_{k'} - D_{k',n}^{off} + \omega\right)}{\lambda_{k'} \times \Omega_{k'} \times N\left(k'\right) \times c_{k',n}} \tag{16}$$

Since the current selection is task k, it is proved the following result appears in the last round of calculation:

$$\max\left\{\forall m \in \{1, 2, \cdots, M\} \middle| \frac{R_m \times \left(\Omega_m - D_{m,n}^{off} + \omega\right)}{\lambda_m \times \Omega_m \times N(m) \times c_{m,n}}\right\} = \frac{R_k \times \left(\Omega_k - D_{k,n}^{off} + \omega\right)}{\lambda_k \times \Omega_k \times N(k) \times c_{k,n}} \tag{17}$$

It is certainly that:

$$\frac{R_k \times \left(\Omega_k - D_{k,n}^{off} + \omega\right)}{\lambda_k \times \Omega_k \times N(k) \times c_{k,n}} \geq \frac{R_{k'} \times \left(\Omega_{k'} - D_{k',n}^{off} + \omega\right)}{\lambda_{k'} \times \Omega_{k'} \times N\left(k'\right) \times c_{k',n}} \tag{18}$$

in which, $N(k) \geq 1$, then,

$$\frac{R_k \times \left(\Omega_k - D_{k,n}^{off} + \omega\right)}{\lambda_k \times \Omega_k \times 1 \times c_{k,n}} \geq \frac{R_k \times \left(\Omega_k - D_{k,n}^{off} + \omega\right)}{\lambda_k \times \Omega_k \times N(k) \times c_{k,n}} \tag{19}$$

It is obtained that

$$\frac{R_k \times \left(\Omega_k - D_{k,n}^{off} + \omega\right)}{\lambda_k \times \Omega_k \times 1 \times c_{k,n}} \geq \frac{R_{k'} \times \left(\Omega_{k'} - D_{k',n}^{off} + \omega\right)}{\lambda_{k'} \times \Omega_{k'} \times N\left(k'\right) \times c_{k',n}} \quad (20)$$

Thus, the contradiction between Equations (20) and (16) is proved. □

Based on above lemmas and definitions, the model is analyzed by game theory. We use $a_{-n} = \{a_1, \cdots, a_{n-1}, a_{n+1}, \cdots, a_N\}$ to represent all resource allocation strategies for edge serves other than edge server n. Providing the selection a_{-n} of all other edge servers, the edge server n will make current round of selection according to a_{-n} and by figuring out the value of Eq.(11). It can be seen that in Eq.(11), for each edge server, only the parameter $N(m)$ is unknown, so edge servers can make strategy selection only if they get the value of $N(m)$, with no need to know the configuration and state information of remaining edge servers.

4.1.2. Model Analysis. This section firstly defines Nash equilibrium and potential game, and then uses potential game to prove the existence of Nash equilibrium in the game [21].

Definition 6. If a resource allocation strategy $A^* = (a_1^*, \cdots, a_N^*)$ is a Nash equilibrium in resource allocation game for multiuser and multiedge server; then, in the case of resource allocation strategy A^*, no edge server can further improve its benefit by unilaterally changing its selection strategy. That is,

$$u_n(a_n^*, a_{-n}^*) \geq u_n(a_n, a_{-n}^*), \quad \forall a_n \in \text{Task}, n \in \text{ES}. \quad (21)$$

Definition 7. If a game has the following properties, it is called an ordinal potential game: $\exists \phi(A)$, for each $n \in ES$, $a_n, a_{n'} \in$ Task, if

$$u_y\left(a_n', a_{-n}\right) > u_n(a_n, a_{-n}), \quad (22)$$

then

$$\phi\left(a_n', a_{-n}\right) > \phi(a_n, a_{-n}). \quad (23)$$

Ordinal potential game has a property, i.e., the existence of a Nash equilibrium as well as finite improvement properties. In this way, any asynchronous optimal response updating process (no more than one participator maximizes its benefit by updating strategy) must be finite and will eventually result in Nash equilibrium [21]. In order to prove that the game in this paper is an ordinal potential game, the potential function of the paper is constructed as

$$\phi(A) = \sum_{i=1}^{N} \sum_{j=1}^{M} \frac{R_j \times \left(\Omega_j - D_{j,i}^{\text{off}} + \omega\right)}{\Omega_j \times c_{j,i}} I_{\{a_{j,i}=1\}}, \quad (24)$$

in which $I_{\{U\}}$ is an indicator function. If U expression is correct, then $I_{\{U\}} = 1$; otherwise, $I_{\{U\}} = 0$.

Theorem 8. *The resource allocation game model between edge servers in the paper is a finite ordinal potential game with potential function (24).*

Proof. Assume that a server $n \in ES$ changes its selection k to k', and the allocation strategy will be updated from A to A'. This will cause an increase of its benefit indicator function value, i.e., $u_{k',n} > u_{k,n}$. According to the definition of potential function, it can be discovered this will result in higher potential function value, i.e., $\phi(A') > \phi(A)$. From the lemma, it can be obtained when selection update occurs, there must be $k = 0$, i.e., not selecting the task $u_{k,n} = 0$. Thus, there are only two situations to consider: (1) $a_{k',n} = 0$ and (2) $a_{k',n} \neq 0$.

Because $k = 0$, then $u_{k,n} = 0$, that is to say,

$$\frac{R_k \times \left(\Omega_k - D_{k,n}^{off} + \omega\right)}{\Omega_k \times c_{k,n}} = 0. \quad (25)$$

(1) In case $a_{k',n} = 0$, then $u_{k',n} > u_{k,n} = 0$, it can be obtained by Equation (11) that

$$\frac{R_{k'} \times \left(\Omega_{k'} - D_{k',n}^{\text{off}} + \omega\right)}{\lambda_{k'} \times \Omega_{k'} \times N\left(k'\right) \times c_{k',n}} > 0 \quad (26)$$

As $\lambda_{k'} > 0, N(k') > 0$, the above equation can be reduced to

$$\frac{R_{k'} \times \left(\Omega_{k'} - D_{k',n}^{\text{off}} + \omega\right)}{\Omega_{k'} \times c_{k',n}} > 0. \quad (27)$$

According to Definition 7, it is needed to compare the potential function under different strategies, and their size can be expressed by the difference of the two. The specific calculation process is as follows:

$$\begin{aligned}\phi\left(A'\right) - \phi(A) = & \sum_{i=1}^{N} \sum_{j=1}^{M} \frac{R_j \times \left(\Omega_j - D_{j,i}^{\text{off}} + \omega\right)}{\Omega_j \times c_{j,i}} I_{\{a'_{j,i}=1\}} I_{\{i \neq n\}} \\ & + \sum_{j=1}^{M} \frac{R_j \times \left(\Omega_j - D_{j,n}^{\text{off}} + \omega\right)}{\Omega_j \times c_{j,n}} I_{\{a'_{j,i}=1\}} \\ & - \sum_{i=1}^{N} \sum_{j=1}^{M} \frac{R_j \times \left(\Omega_j - D_{j,i}^{\text{off}} + \omega\right)}{\Omega_j \times c_{j,i}} I_{\{a_{j,i}=1\}} I_{\{i \neq n\}} \\ & - \sum_{j=1}^{M} \frac{R_j \times \left(\Omega_j - D_{j,n}^{\text{off}} + \omega\right)}{\Omega_j \times c_{j,n}} I_{\{a_{j,i}=1\}}\end{aligned}$$

$$
\begin{aligned}
&= \sum_{j=1}^{M} \frac{R_j \times \left(\Omega_j - D_{j,n}^{\text{off}} + \omega\right)}{\Omega_j \times c_{j,n}} I_{\left\{a'_{j,i}=1\right\}} \\
&\quad - \frac{R_k \times \left(\Omega_k - D_{k,n}^{\text{off}} + \omega\right)}{\Omega_k \times c_{k,n}} \\
&= \frac{R_{k'} \times \left(\Omega_{k'} - D_{k',n}^{\text{off}} + \omega\right)}{\Omega_{k'} \times c_{k',n}} > 0.
\end{aligned}
\tag{28}
$$

(2) In case $a_{k',n} \neq 0$, it can be obtained by Definition 3 that, if strategy update occurs at this time, there must be the following inequation:

$$
\frac{R_{k'} \times \left(\Omega_{k'} - D_{k',n}^{\text{off}} + \omega\right)}{\Omega_{k'} \times c_{k',n}} > \frac{R_{k'} \times \left(\Omega_{k'} - D_{k',n'}^{\text{off}} + \omega\right)}{\Omega_{k'} \times c_{k',n'}}
\tag{29}
$$

Integrating with Definition 7, the potential function under different strategies is compared. The specific calculation process is as follows:

$$
\begin{aligned}
\phi\left(A'\right) - \phi(A) &= \sum_{i=1}^{N}\sum_{j=1}^{M} \frac{R_j \times \left(\Omega_j - D_{j,i}^{\text{off}} + \omega\right)}{\Omega_j \times c_{j,i}} I_{\left\{a'_{j,i}=1\right\}} I_{i\neq n} I_{\left\{i\neq n'\right\}} \\
&\quad + \sum_{j=1}^{M} \frac{R_j \times \left(\Omega_j - D_{j,n}^{\text{off}} + \omega\right)}{\Omega_j \times c_{j,n}} I_{\left\{a'_{j,n}=1\right\}} \\
&\quad + \sum_{j=1}^{M} \frac{R_j \times \left(\Omega_j - D_{j,n'}^{\text{off}} + \omega\right)}{\Omega_j \times c_{j,n'}} I_{\left\{a'_{j,n'}=1\right\}} \\
&\quad - \sum_{i=1}^{N}\sum_{j=1}^{M} \frac{R_j \times \left(\Omega_j - D_{j,i}^{\text{off}} + \omega\right)}{\Omega_j \times c_{j,i}} I_{\left\{a_{j,i}=1\right\}} I_{\{i\neq n\}} I_{\left\{i\neq n'\right\}} \\
&\quad - \sum_{j=1}^{M} \frac{R_j \times \left(\Omega_j - D_{j,n}^{\text{off}} + \omega\right)}{\Omega_j \times c_{j,n}} I_{\left\{a_{j,n}=1\right\}} \\
&\quad - \sum_{j=1}^{M} \frac{R_j \times \left(\Omega_j - D_{j,n'}^{\text{off}} + \omega\right)}{\Omega_j \times c_{j,n'}} I_{\left\{a_{j,n'}=1\right\}} \\
&= \frac{R_{k'} \times \left(\Omega_{k'} - D_{k',n}^{\text{off}} + \omega\right)}{\Omega_{k'} \times c_{k',n}} \\
&\quad - \frac{R_{k'} \times \left(\Omega_{k'} - D_{k',n'}^{\text{off}} + \omega\right)}{\Omega_{k'} \times c_{k',n'}} > 0.
\end{aligned}
\tag{30}
$$

Combing with the results of Equations (28) and (30) as well as Definition 4, it is concluded that the MEC multiuser and multiedge server resource allocation game proposed in the paper is a finite ordinal potential game. Therefore, the game has the characteristics of a finite ordinal potential game, including limited improvement properties. □

4.1.3. Game Strategy Design. The above section verifies the MEC multiuser and multiedge server resource allocation game has Nash equilibrium, and this section introduces the strategy to solve Nash equilibrium in details. There are some classical algorithms for finding the existence of Nash equilibrium, such as optimum response [18] and reinforcement learning [22]. However, the problem of finding Nash equilibrium in the model of this paper is incomplete information; that is to say, all configuration information for each edge server is held only by itself. Thus, an edge server cannot make its own optimal response according to the optimal selection of other edge servers. So this paper chose to use game theory for problem solution. Through the above proof, this paper designed a game-based joint optimization strategy (GBJOS) to solve the competition problem of edge server, as shown in Strategy 9.

Strategy 9. Iteration-based optimal energy efficiency update algorithm.

```
Input Mobile device's task information
Output Resource allocation strategy A
Initialization Initialize the resource allocation strategy of system, set task selection threshold as $N\_TO = M/2$, set maximum iteration times max $Iter = 100$
1) while $converge \neq True$ and $iter <$ max $Iter$ do
2)   $iter \longleftarrow iter + 1$
3)   administrative server transmits Task, A, $N(m)$, $N\_TO$, $\lambda_m$ to each edge server
       receive parameter information
6)     based on Definitions 3 and 4, maximize the value of Eq. (11), select the optimal strategy $br_n$ for them
7)     transmit the strategy $br_n$ to administrative server
8)     if $br_n$ is not null, and currently do not hold $br_n$ then
9)       request strategy change from administrative server
10)    end if
11)  end for
12)  if administrative server does not receive any request then
13)       $converge \longleftarrow True$
14)  else
15)     randomly choose an edger serve and allow it to update strategy
16)     based on this round of received strategy $br_n$, update A
17)     based on this round of strategy update of all edge servers

18)  end if
19) end while
20) Return A
```

GBJOS mainly includes two parts: strategy selection of edge server and collection of strategy update information from administrative server.

(i) Strategy selection of edge server: edge servers obtain the related parameters of Equation (11) from administrative server, for calculating the benefit of all tasks $\{u_n(a_n, a_{-n}), n \in ES, a_n, a'_n \in \text{Task}\}$, choose the task with the largest benefit as this round of strategy selection. In case an edge server does not hold the task selected for this round, it can send a request to administrative server for strategy modification.

(ii) Collection of strategy update information from administrative server: in each round of iteration, the administrative server can receive the current strategy selection of all edge servers as well as the request of some edge servers for modification. If no request for modification is received, it is judged as convergence; otherwise, administrative server shall gather current strategies, then randomly choose an edge server having send request for modification and permit it to change current resource allocation strategy [23]. Meanwhile, the administrative server updates resource allocation strategy and transmits collected parameters to each edge server.

4.1.4. Convergence Analysis. In Theorem 8, we have proved this game is a finite ordinal potential game, so this strategy will converge to Nash equilibrium within in limited number of iteration times. In practical strategy execution, when the administrative server does not receive any request for modification, it is judged as strategy convergence; at this time, the update of resource allocation strategy will be ended.

Next, the time complexity of GBJOS will be analyzed. Within each iteration, every edge server performs task selection in parallel. Therein, some sorting operation and basic operations are involved, so the time complexity of this part is $O(N \log N)$. It is assumed that this strategy will go through C times of iteration before reaching to termination, so the overall time complexity of GBJOS is $O(CN \log N)$. It is clear whether this strategy can converge within limited times relies on whether iteration time C has a upper bound or not.

Theorem 10. *When for* $\forall m \in \{1, 2, \cdots, M\}$, $\forall n \in \{1, 2, \cdots, N\}$, $D^{off}_{m,n} \le \Omega_m$, $R_m > c_{m,n}$ *occurs, GBJOS will terminate after iterating for* $\max\{(N * Q * \Omega_{\max})/\omega, M * N\_TO\}$ *times at most, i.e.,* $C < \max\{(N * Q * \Omega_{\max})/\omega, M * N\_TO\}$.

Proof. During iteration, it is assumed that edge server n updates current task selection k to k', and updates allocation strategy A to A'. Based on Definition 4 and lemma, it is obtained that $k = 0$; then there are two main scenarios to consider: (1) $a_{k',n} = 0$ and (2) $a_{k',n} \ne 0$

(1) $a_{k',n} = 0$: the theoretical maximal benefit is expressed as

$$Q = \max\left\{\forall m \in \{1, 2, \cdots, M\}, \forall n \in \{1, 2, \cdots, N\} \middle| \frac{R_m \times \left(\Omega_m - D^{off}_{m,n} + \omega\right)}{\Omega_m \times c_{m,n}}\right\} \tag{31}$$

According to potential function (24), it can be obtained that

$$0 \le \phi(A) \le \sum_{i=1}^{N} \sum_{j=1}^{M} QI_{\{a_{j,i}=1\}} < NQ. \tag{32}$$

According to the nature of potential game, strategy change will lead to the increase of potential function value. Suppose the minimal granularity of potential game increase to be $Q_{\min}$, that is to say,

$$\phi\left(A'\right) \ge \phi(A) + Q_{\min}. \tag{33}$$

According to Equation (28), it can be obtained that

$$\phi\left(A'\right) - \phi(A) \ge \frac{\omega}{\Omega_{k'}}. \tag{34}$$

Define the maximal time delay as

$$\Omega_{\max} = \max\{\forall m \in \{1, 2, \cdots, M\} | \Omega_m\}. \tag{35}$$

There must be

$$\Omega_{k'} \le \Omega_{\max}. \tag{36}$$

That is to say,

$$\begin{gathered} \phi\left(A'\right) - \phi(A) \ge \frac{\omega}{\Omega_{\max}} > 0, \\ a_{k',n} \ne 0. \end{gathered} \tag{37}$$

□

If the strategy selection of edge servers contradicts, it can be known from Definitions 3 and 4 that for a task, whether a server can successfully snatch it or not, the upper bound of conflict times is N_TO. So, in this case, the upper bound of strategy conflict times is $M * N\_TO$.

Through the analysis of above two cases, it can be concluded that: iteration times $C < \max\{(N * Q * \Omega_{\max})/\omega, M * N\_TO\}$.

4.2. Reverse Update of Parameters. The above-mentioned GBJOS strategy makes edge server to maximize its benefit; finally, all edge servers can reach an agreement. In this section, the penalty coefficient λ in Equation (10) was dynamically adjusted by setting loss function. The penalty coefficient is designed to give different significance scoring to the tasks of different size, and edge servers determine their own strategy selection by working out Equation (11). So, penalty coefficient λ plays an important role in guiding edge server to make strategy selection. If different tasks have great difference in λ, it would be unfair to some tasks, making λ to exert large impact on the value of Equation (11), while ignoring the influence of other parameters; but if different tasks have too small difference in λ, the gap between task priority is too small, then edge server would still make strategy selection according to the initial condition of a task, and the updates do not work as it is expected to be.

In the TUSGT of this paper, penalty coefficient is updated dynamically through online learning framework. The paper sets a loss function φ; each task will be updated to varied extent according to its own data characteristics. By virtue of such automatic dynamic adjustment, tasks are given different priorities, reaching the purpose of improving deadline hit rate. The loss function φ_m for task m is calculated as shown in

$$\varphi_m = VI_m - \frac{\sum_{i=1}^{M} VI_i \times \theta_i}{\sum_{i=1}^{M} \theta_i}, \ \sum_{i=1}^{M} \theta_i \neq 0, \tag{38}$$

in which θ_i refers to whether task i is completed on time or not (see details in Section 3.7). The penalty coefficient λ is updated by EWA algorithm [24], as shown in

$$\lambda_m^{\text{new}} = \begin{cases} \lambda_m^{\text{old}} \times e^{-\eta\varphi_m}, & \sum_{i=1}^{M} \theta_i \neq 0, \\ \lambda_m^{\text{old}} \times \frac{1}{2}, & \sum_{i=1}^{M} \theta_i = 0. \end{cases} \tag{39}$$

When no task is completed on time, the penalty coefficient of all tasks is halved, to increase task attraction. It can be seen from Equation (39) that, in general condition, the priority of the task with large volume of data will be elevated, while the update extent relies on the task completion of current strategy. Therein, the update extent of penalty coefficient also depends on the value of learning rate η, which should be set artificially according to actual initial task information. The value of η also decides the update times of TUSGT; the smaller the update extent is, the more times it will be updated. While updating too much will result in unsatisfactory result, so it needs to be weighed against actual data. The penalty coefficient needs to be adjusted according to the actual task size and the computing power of the edge server. It is only necessary to use the parameters expected in the scene as input and make multiple adjustments. The variation of this coefficient is also affected by the size of the task, but in general, the edge server can be guided to make a choice that is more conducive to the overall task completion rate of the MEC system.

5. Experiment and Result Analysis

In the MEC multiuser and multiedge server resource allocation scenarios set in the simulation test of this paper, all mobile devices and edge servers are evenly distributed in a region of $1000\,\text{m} \times 1000\,\text{m}$. The performance of different strategies is evaluated by deadline hit rate and the average benefit of edge servers. Finally, the convergence of TUSGT strategy and the growth of operation time are analyzed.

5.1. Experimental Parameter Setting. In the simulation test of the paper, there are 30 edge servers, and the performance of TUSGT is evaluated by setting different number of mobile devices. Both task input volume VI_m and computing amount L_m conform to uniform distribution within the range of values, and the output data volume VO_m of mobile device tasks is set to be 0.2 times of VI_m [25]. This initial reward value is proportional to the amount of task data. The reward value will not change during the experiment. If the mobile device is not allocated computing resources, it will miss the opportunity to be calculated in this round. When the edge server completes the calculation of the current task, it can update its reward value to participate in a new round of allocation. This paper refers to the experimental parameter data in literature [19] to set the CPU processing capacity f_m and f_n of mobile device and edge server, which also conform to even distribution within the range of values, aimed at simulating a real environment with various devices of different computing power. The transmission rate of mobile device conforms to even distribution within the range of values, and communication-related parameters such as antenna gain are fixed value [19]. Regarding task delay requirement Ω_m, the time delay requirement of task set in this paper is consistent with even distribution within time interval $[Z \pm 0.3Z]$, in which, Z is the time delay granularity [25]. The smaller the time delay granularity is, the more strict the time delay requirement of task will be. The learning rate is affected by the amount of task data. In the parameter environment of this paper, setting the learning rate to 0.1 has a better effect. In practical applications, multiple adjustments need to be made according to the estimated amount of task data, and a value with better effect is selected. This value is not adjusted during operation. The specific parameter settings are shown in Table 1.

Table 1: Experimental parameters.

Parameters	Value
Number of edge servers N	30
Number of mobile devices M	[20, 80]
Input volume of task VI_m	[20, 100] MB
Calculation amount of task L_m	[500, 1500] Mcycles/MB
Delay requirement granularity of task Z	[100, 300] s
CPU processing capability of mobile devicef_m	[1, 5] Mcycles/s
CPU processing capability of edge serverf_n	[200, 400] Mcycles/s
Bandwidth B	2 MHz
Noise power σ^2	10^{-10} W
Antenna gain G	4.11
Carrier frequency F_c	915 MHz
Path loss factor PL	3
Transmission power of mobile device p_m	[0.1, 0.8] W
Learning rate η	0.1

5.2. Evaluation Metrics. TUSGT considers edge server benefit and deadline hit rate at the same time. Edge server benefit embodies its initiative to perform task; the higher the benefit is, the more the edge server is willing to perform task. Deadline hit rate embodies the performance of allocation strategy, because the MEC system is aimed at completing the task offloaded by mobile deice; the higher the deadline hit rate is, the better the performance of allocation strategy is. So this paper evaluates the performance of different strategies from two aspects: deadline hit rate of MEC system and average benefit of edge server. First, these two indexes are defined as follows:

(1) Deadline hit rate (DHR):

$$\text{DHR} = \frac{M - \sum_{i=1}^{M} \theta_i}{M}, \tag{40}$$

in which θ_i represents whether the task i is completed on time or not and M represents total number of tasks.

(2) Average benefit (AB):

The benefit of edge server n is

$$\sum_{m \in S_n} \text{benefit}_{m,n}, \tag{41}$$

in which, S_n represents the final strategy set of edge server n. The average benefit of edge server is obtained as

$$\text{AB} = \frac{\sum_{i=1}^{N} \sum_{m \in S_i} \text{benefit}_{m,i}}{N}, \tag{42}$$

in which N represents the number of edge servers.

5.3. Comparison of Strategies. The paper selected five allocation strategies to compare with the TUSGT strategy proposed. The five comparison strategies are as follows:

(1) BGTA strategy: a game theory-based task allocation scheme, allowing the player to selfishly compete for task so as to maximize its own benefit [26].

(2) MILP strategy: adopts mixed integer linear programming (MILP) to formulate joint task offloading and resource allocation strategy [27].

(3) Greedy Strategy (GS): a greedy resource allocation strategy, making edge server to always select the task with highest benefit

(4) Random Strategy (RS): a random allocation strategy, making edge server to randomly select task for computing

(5) Ideal strategy: an ideal situation, in which the administrative server performs a global and integrated allocation based on the information of all edge servers and mobile device tasks and takes maximizing the deadline hit rate of system as objective. In this case, all edge servers and mobile devices completely listen to the arrangement of administrative server, without the ability to choose

In the BGTA strategy, the author considers the revenue problem of edge servers, constructs a revenue function, allows edge servers to select the task with the highest benefit at present, and then uses game theory to build an allocation algorithm to solve the conflict problem of task selection and finally solves the allocation strategy. In the MILP strategy, the author takes the monetary cost and task completion delay of the edge server as the optimization goal, constructs it as a MILP problem, and then uses the branch and bound algorithm (The Branch and Bound Algorithm) to solve the approximate optimal solution of the target problem; namely, the optimization problem is regarded as a search tree to search, so as to obtain the final resource allocation strategy.GS adopts the classical greedy algorithm to construct resource allocation strategy, which is a classical solution method. RS reflects the performance of resource allocation problem under unstable strategies, normally the lower bound of allocation strategy performance. Ideal strategy assumes all devices follows the management and do not pursue for any self-interest, and the strategy maker owns the information of all devices and tasks in current environment. But this is not practical, so the performance of ideal strategy is only taken as an upper bound in the comparison of deadline hit rate, to measure the difference between various strategies and ideal upper bound.

5.4. Experimental Results

5.4.1. Deadline Hit Rate. Figure 2 provides the influence of task number on DHR for different resource allocation strategies. In this experiment, due to the number of edge server being fixed as 30, in the process of increasing task number, the overall DHR must show a declining trend, and the fluctuation of effect is caused by the heuristic operation in strategy and the randomness of task generation. It can be seen from the chart that, for the TUSGT proposed in the paper, its deadline hit rate is very close to that of ideal strategy under different number of tasks. But in the case of 60 tasks, only TUSGT and ideal achieve the highest DHR of 0.5. Among them, the effect of RS is not stable, but it can be observed from an overall trend that its effect is far less than TUSGT. The poorest effect of GS contributes to the short sight of edge server. Under these five different number of tasks, as DHR performance is concerned, TUSGT is improved by 22% on average compared to BGTA and by 27% on average compared to MILP.

Figure 3 shows the impact of changing task delay requirement on DHR for different strategies. In the experiment as shown in Figure 3, the number of both edge server and task is set as 30, so DHR is within the range of [0,1]. The increase of delay granularity means the task delay requirement gradually becomes loose, and overall DHR is also rising. Similar to the experiment in Figure 2, RS performance is not stable and its DHR is lower than TUSGT. It can be observed from Figure 3 that, in difference scenarios, TUSGT is slightly lower than Ideal, and the remaining

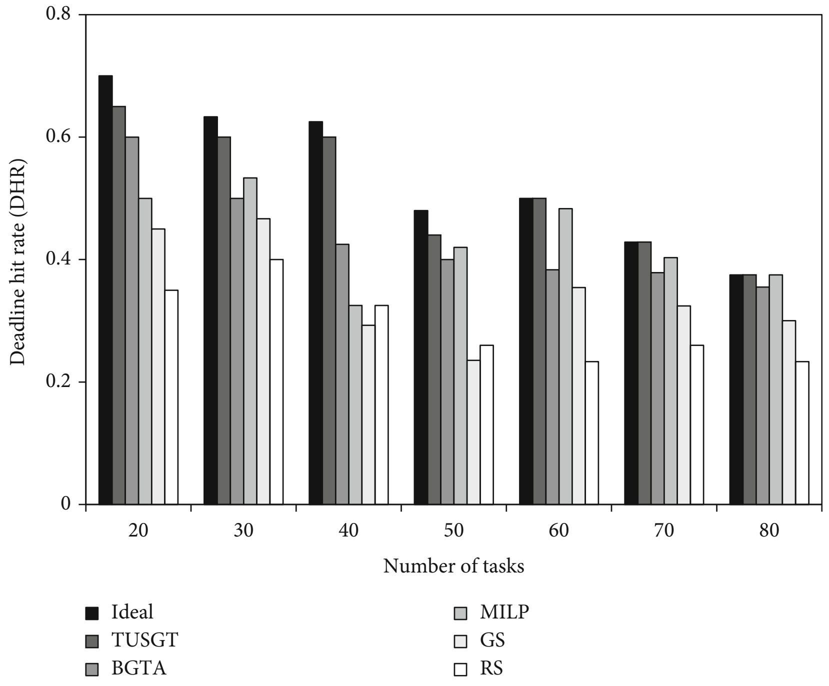

Figure 2: Impact of the number of tasks on DHR.

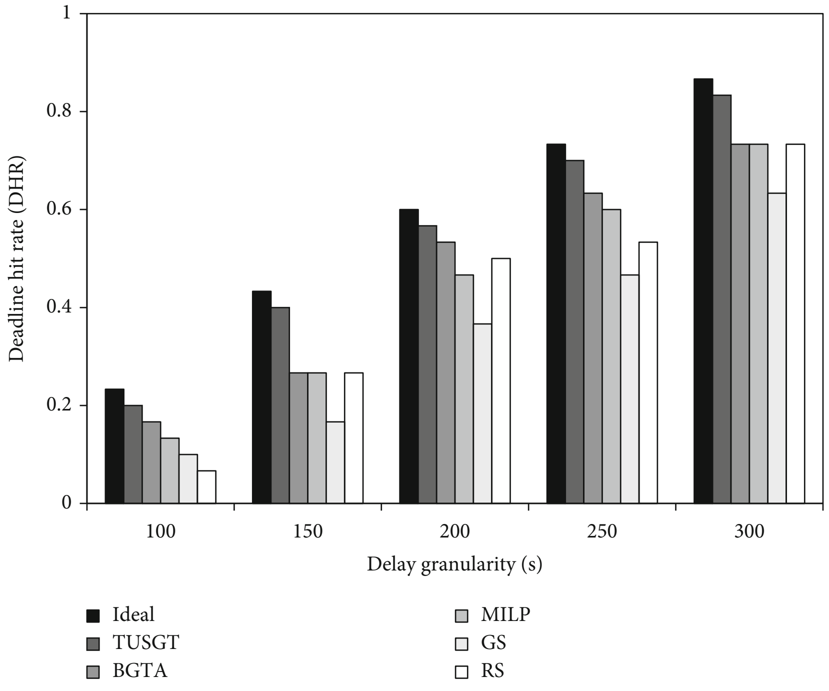

Figure 3: Impact of delay granularity on DHR.

strategies are all lower than TUSGT, which is due to the reverse update penalty coefficient in TUSGT. Under these five different task delay requirements, as DHR performance is concerned, TUSGT is improved by 20% on average compared to BGTA and by 30% on average compared to MILP.

5.4.2. Average Benefit of Edge Server. Figure 4 shows the impact of changing task number on edge server AB for different strategies. In this experiment, there are 30 edge servers, and the number of task is changed to simulate different scenarios. It can be seen from Figure 4 that, in scenarios with different task number, all average benefits of TUSGT are the highest, due to that the game theory-based strategy allows edge servers to compete for task. It is also observed that ideal strategy lags far behind on AB index, because ideal only cares about overall DHR of the system, without

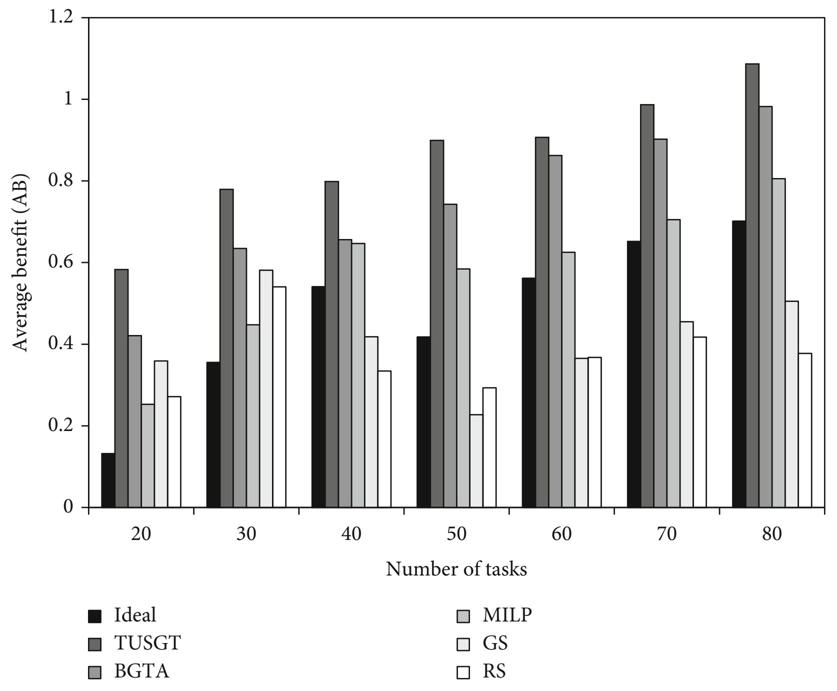

Figure 4: Impact of the number of tasks on AB.

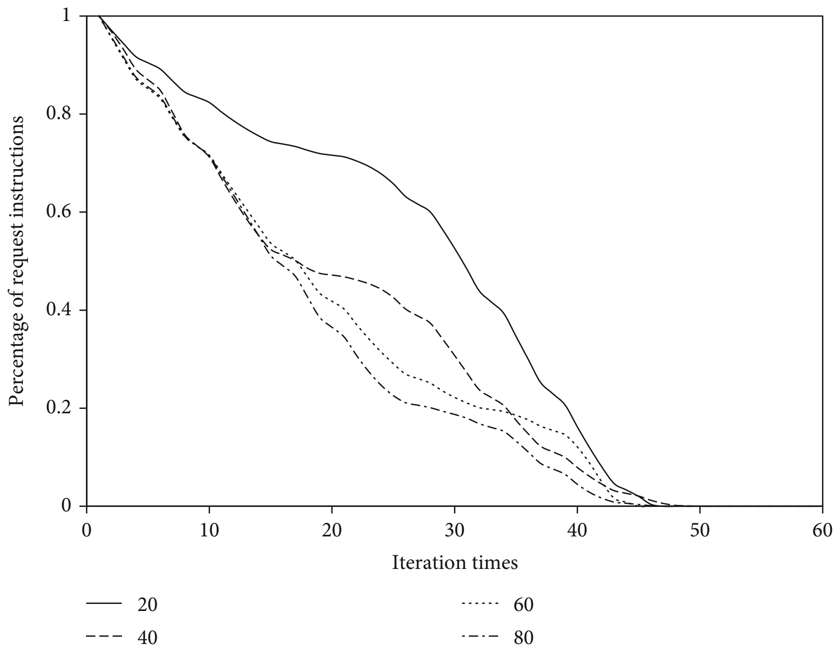

Figure 5: TUSGT convergence curves.

considering the competition relationship and benefit of edge servers. Both RS and GS have a poor effect. RS effect fluctuates greatly due to its instability, and GS sacrifices much DHR, resulting in low AB. Under these five different number of task, as edge server AB is concerned, TUSGT is improved by 22% on average compared to BGTA and by 65% on average compared to MILP.

5.4.3. Convergence and Overhead Analysis. Figure 5 displays the convergence when task number is 20, 40, 60, and 80, and its vertical axis is the percentage of request instruction number which represents the convergence of TUSGT. From the introduction in section 4, it is known that TUSGT convergence mainly lies on the convergence of GBJOS strategy. When there is edge server requesting for strategy modification,

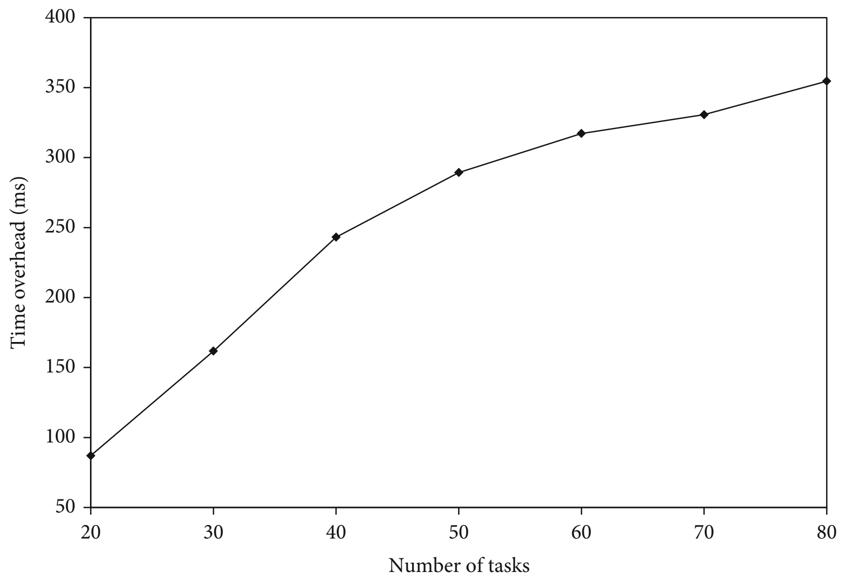

Figure 6: Impact of task number on time overhead.

it represents convergence. It is observed in the figure that, as the iteration times increase, all the numbers of edge servers applying for modifying resource allocation strategy are decreasing under four scenarios, so TUSGT reaches to convergence state after less iteration times. With the increase of task number, TUSGT convergence tends to be smooth, because the more the task number is, the smaller the competition between edge servers will be, and the easier it is to converge.

Figure 6 shows the change of TUSGT execution time as the task number in MEC system increases. The main time overhead of TUSGT is from GBJOS, because the time complexity of reversely updating parameters is very small [25], while GBJOS requires edge server to iterate many times and compete for selection task. Therefore, GBJOS with fast convergence makes TUSGT to have very small time overhead, as shown in Figure 6, the TUSGT time overhead is merely 350 ms in the case of 80 tasks, which would not produce too great influence on task execution. It is also observed from Figure 6 that, as task number increases, the overall TUSGT time overhead tends to increase smaller and smaller. This means even if more tasks are added, TUSGT time overhead would not be elevated too much suddenly.

The above experimental results indicate that the TUSGT proposed in the paper can effectively promote edge server benefit and MEC system DHR. TUSGT has a good convergence, and execution time overhead would not affect the real-time requirement of offloading task in MEC system.

6. Conclusions

This paper built a model for multiuser and multiedge server resource allocation problem in MEC system at first, considered the benefit of edge server, and proposed a two-way update strategy based on game theory (TUSGT) to solve this problem. It is proved that the task competition relationship between edge servers is a finite ordinal potential game and the Nash equilibrium exists; then parameters are updated reversely to adjust task weight, so as to improve overall deadline hit rate. Finally, through simulation test, we verified the convergence nature and time performance of TUSGT. Compared to other datum strategies, TUSGT significantly promotes edge server benefit and overall deadline hit rate. However, TUSGT only solves problem in static scenario, but does not consider decomposable task or that one edge server can receive multiple tasks. The further study will consider dynamic offloading strategy and user movability factors and extend TUSGT solution scenario, to adapt to more general situations.

Data Availability

The data used to support the findings of this study are included within the article.

Conflicts of Interest

The authors declare that there are no conflicts of interest.

Acknowledgments

The authors acknowledge the National Natural Science Foundation of China (Grant: 62072321), the National Science Foundation of the Jiangsu Higher Education Institutions of China (No. 20KJB520002), the Future Network Scientific Research Fund Project (No. FNSRFP-2021-YB-38), and the Suzhou Planning Project of Science and Technology (Nos. SNG2020073 and SS202023). And this work was also supported by Project Funded by the Priority Academic Program Development of Jiangsu Higher Education Institutions.

References

[1] N. Heuveldop, *Ericsson Mobility Report (5g)*, Ericsson, Stockholm, 2018.

[2] F. Wang, J. Xu, X. Wang, and S. Cui, "Joint Offloading and Computing Optimization in Wireless Powered Mobile-Edge Computing Systems," *IEEE Transactions on Wireless Communications*, vol. 17, no. 3, pp. 1784–1797, 2018.

[3] Y. Y. Mao, J. Zhang, S. H. Song, and K. B. Letaief, "Stochastic joint radio and computational resource management for multi-user mobile-edge computing systems," *IEEE Transactions on Wireless Communications*, vol. 16, no. 9, pp. 5994–6009, 2017.

[4] F. A. Samimi, P. K. McKinley, and S. M. Sadjadi, "Mobile Service Clouds: A Self-Managing Infrastructure for Autonomic Mobile Computing Services," in *Self-Managed Networks, Systems, and Services. SelfMan 2006*, A. Keller and J. P. Martin-Flatin, Eds., vol. 3996 of Lecture Notes in Computer Science, Springer, Berlin, 2006.

[5] B. Zhou, A. V. Dastjerdi, R. N. Calheiros, S. N. Srirama, and R. Buyya, "A context sensitive offloading scheme for mobile cloud computing service," in *2015 IEEE 8th International Conference on Cloud Computing*, pp. 869–876, New York, NY, USA, June 2015.

[6] M. Patel, B. Naughton, C. Chan, N. Sprecher, S. Abeta, and A. Neal, "Mobile-edge computing introductory technical white paper," *White Paper, Mobile-Edge Computing (MEC) Industry Initiative*, vol. 29, 2014.

[7] Y. C. Hu, M. Patel, D. Sabella, N. Sprecher, and V. Young, "Mobile edge computing—a key technology towards 5G," *ETSI White Paper*, vol. 11, pp. 1–16, 2015.

[8] Y. Mao, C. You, J. Zhang, K. Huang, and K. B. Letaief, "A survey on mobile edge computing: the communication perspective," *IEEE Communication Surveys and Tutorials*, vol. 19, no. 4, pp. 2322–2358, 2017.

[9] N. Abbas, Y. Zhang, A. Taherkordi, and T. Skeie, "Mobile edge computing: a survey," *IEEE Internet of Things Journal*, vol. 5, no. 1, pp. 450–465, 2018.

[10] P. Mach and Z. Becvar, "Mobile edge computing: a survey on architecture and computation offloading," *IEEE Communication Surveys and Tutorials*, vol. 19, no. 3, pp. 1628–1656, 2017.

[11] L. Liu, C. Chen, J. Feng, Q. Q. Pei, C. He, and Z. B. Dou, "Joint intelligent optimization of task offloading and service caching for vehicular edge computing," *Journal on Communications*, vol. 42, no. 1, pp. 18–26, 2021.

[12] M. Hu, Z. Xie, D. Wu, Y. Zhou, X. Chen, and L. Xiao, "Heterogeneous edge offloading with incomplete information: a minority game approach," *IEEE Transactions on Parallel and Distributed Systems*, vol. 31, no. 9, pp. 2139–2154, 2020.

[13] C. You, K. Huang, H. Chae, and B.-H. Kim, "Energy-efficient resource allocation for mobile-edge computation offloading," *IEEE Transactions on Wireless Communications*, vol. 16, no. 3, pp. 1397–1411, 2017.

[14] L. Y. Ji and S. T. Guo, "Energy-efficient cooperative resource allocation in wireless powered mobile edge computing," *IEEE Internet of Things Journal*, vol. 6, no. 3, pp. 4744–4754, 2019.

[15] J. Qi, H. R. Sun, and K. Gong, "Research on intelligent computing offloading model based on reputation value in mobile edge computing," *Journal on Communications*, vol. 41, no. 7, pp. 141–151, 2020.

[16] Y. Jiao, P. Wang, D. Niyato, and Z. Xiong, "Social welfare maximization auction in edge computing resource allocation for mobile blockchain," in *2018 IEEE International Conference on Communications (ICC)*, pp. 1–6, Kansas City, MO, USA, May 2018.

[17] N. C. Luong, Z. Xiong, P. Wang, and D. Niyato, "Optimal auction for edge computing resource management in mobile blockchain networks: a deep learning approach," in *2018 IEEE International Conference on Communications (ICC)*, pp. 1–6, Kansas City, MO, USA, May 2018.

[18] D. Liu, L. Khoukhi, and A. Hafid, "Decentralized data offloading for mobile cloud computing based on game theory," in *2017 Second International Conference on Fog and Mobile Edge Computing (FMEC)*, pp. 20–24, Valencia, Spain, May 2017.

[19] J. Yan, S. Bi, Y. J. Zhang, and M. Tao, "Optimal task offloading and resource allocation in mobile-edge computing with inter-user task dependency," *IEEE Transactions on Wireless Communications*, vol. 19, no. 1, pp. 235–250, 2020.

[20] D. Y. Zhang and D. Wang, "An integrated top-down and bottom-up task allocation approach in social sensing based edge computing systems," in *IEEE INFOCOM 2019 - IEEE Conference on Computer Communications*, pp. 766–774, Paris, France, April 2019.

[21] D. Monderer and L. S. Shapley, "Potential games," *Games and Economic Behavior*, vol. 14, no. 1, pp. 124–143, 1996.

[22] M. Bowling and M. Veloso, "Multiagent learning using a variable learning rate," *Artificial Intelligence*, vol. 136, no. 2, pp. 215–250, 2002.

[23] X. Chen, L. Jiao, W. Li, and X. Fu, "Efficient multi-user computation offloading for mobile-edge cloud computing," *IEEE/ACM Transactions on Networking*, vol. 24, no. 5, pp. 2795–2808, 2016.

[24] N. Cesa-Bianchi and G. Lugosi, *Prediction, Learning, and Games*, Cambridge University Press, Cambridge, 2006.

[25] F. Mashhadi, S. A. Salinas Monroy, A. Bozorgchenani, and D. Tarchi, "Optimal auction for delay and energy constrained task offloading in mobile edge computing," *Computer Networks*, vol. 183, article 107527, 2020.

[26] D. Zhang, Y. Ma, Y. Zhang, S. Lin, X. S. Hu, and D. Wang, "A real-time and non-cooperative task allocation framework for social sensing applications in edge computing systems," in *2018 IEEE Real-Time and Embedded Technology and Applications Symposium (RTAS)*, pp. 316–326, Porto, Portugal, April 2018.

[27] M. Alkhalaileh, R. N. Calheiros, Q. V. Nguyen, and B. Javadi, "Data-intensive application scheduling on mobile edge cloud computing," *Journal of Network and Computer Applications*, vol. 167, article 102735, 2020.