2020 16th International Conference on Mobility, Sensing and Networking (MSN) | 978-1-7281-9916-0/20/$31.00 ©2020 IEEE | DOI: 10.1109/MSN50589.2020.00129

A Game-Theoretical Approach for Task Offloading in Edge Computing

1st Juan Luo
College of Computer Science and Electronic Engineering Hunan University
Changsha, China
juanluo@hnu.edu.cn

2nd Qian Qian
College of Computer Science and Electronic Engineering Hunan University
Changsha, China
847048556@qq.com

3rd Luxiu Yin
College of Computer Science and Electronic Engineering Hunan University
Changsha, China
yinluxiu@hnu.edu.com

4th Ying Qiao
College of Computer Science and Electronic Engineering Hunan University
Changsha, China
18302961249@163.com

***Abstract*—Edge computing is envisioned as a prominent technology that provides high computing demand services by offloading computation-intensive and delay-sensitive task from mobile or Internet of Things(IoT) devices to nearby edge servers. However, more and more edge servers are deployed near the terminal devices. The uneven distribution of devices will lead to insufficient resource utilization of edge servers, so the social benefits of the edge computing system decrease. In this paper, we propose an effective task offloading strategy in the scenario of multi-users and multi-edge servers. Terminal devices broadcast task offloading request to edge servers, and the edge servers compete for tasks to improve their resource utilization. We use a non-cooperative game to describe the competition of edge servers, and model an optimization problem as the multi-edge servers resource allocation problem. We then design an iterative algorithm to solve the optimization problem and prove that the problem has an unique Nash equilibrium. Simulation results show that the proposed offloading strategy not only improves the resource utilization of edge servers, but also guarantees the demand of terminal devices.**

***Index Terms*—Edge Computing, Task Offloading, Resource Allocation, Non-cooperative Game, Nash Equilibrium**

I. Introduction

In the past decade, mobile and IoT devices, including smart phones, wearable devices, tablets and sensors, etc., are becoming more and more popular. According to Ericsson's report [1], there will be 20 billion mobile and IoT devices are connected to the network by 2023. The increasing popularity of smart devices is driving the development of applications with advanced features, e.g., interactive online gaming, gesture and face recognition, as well as 3D modeling [2] [3]. These applications are resource-hungry, computation-intensive and high energy consumption. Cloud computing can cause high latency and high energy consumption, so it cannot guarantee the quality of service(QoS) of terminal devices [4]. A new solution is needed to deal with explosive computation demands and QoS requirements.

Edge computing is proposed to solve this problem by providing access to computing resources for nearby terminal devices, especially for those devices with limited resources and capabilities. Edge computing deploy edge servers at the edge of network and terminal devices can offload computation task to edge servers [5]. An access point(AP) is deployed next to each edge server. The terminal device can offload its computation task to one edge server. Due to the shortened communication distance and cloud-like computing ability, edge computing reduces task execution delay and energy consumption.

There has been a lot of research work on task offloading in academia and industry. Task offloading refers to how the device chooses a edge server to execute the computation task. Therefore, the optimization goal of the research is mainly to reduce delay [6], reduce energy consumption, reduce energy consumption with constraint of delay [2] and reduce both of two [7], etc. These works mainly optimize the delay and energy consumption for the terminal devices, while ignoring the overhead of edge servers. Because of the uneven distribution of terminal devices, the resource of edge server is underutilized. In order to improve their resource utilization, edge servers will compete for as many tasks as possible. As far as we know, few studies have considered the resource utilization of edge servers.

We aim to propose an effective task offloading strategy in a multi-edge servers environment. We mainly consider the effectiveness of edge servers. The edge servers compete tasks and allocate resources to execute the task in order to improve its own resource utilization. Game theory are widely used in the field of distributed computing [8], suitable for designing distributed mechanisms. The devices broadcasts a task request to the edge servers within the communication range, and the edge server competes for the task. Then the edge server that has won the competition allocates resources to tasks to improve the utilization of resource. We use non-cooperative game to describe the competition among edge server, model an optimization problem as multi-edge servers resource allocation problem, propose an iterative algorithm to solve the problem, and prove that the problem has an unique Nash equilibrium. The main contributions of this papers are as follows:

1) In an edge computing system with multi-edge servers and multi-terminal devices, we propose a novel task offloading strategy. Considering the resource utilization of edge servers, we describe the competition among edge servers as a non-cooperative game problem, which is called a multi-edge server resource allocation game problem. The edge server is a game player. All edge servers can improve their resource utilizaiton by the task offloading strategy, while guaranteed the demands of terminal devices;
2) We analyze and prove the existence and uniqueness of the Nash equilibrium of the multi-edge servers resource allocation game problem. Then, we design an iterative algorithm to find the Nash equilibrium of the problem;
3) In the simulation. We verify the convergence of the proposed algorithm. Then, compared the proposed algorithm with some benchmark algorithms in terms of task offloading computing overhead, energy consumption and edge server resource utilization, which verifies the effectiveness of the proposed algorithm.

The rest of the paper is organized as follows. We first discuss related work in Section II, and introduce the system model in Section III. The description of task offloading, the construction of multi-edge server resource allocation game and the proof of the existence and uniqueness of the game's Nash equilibrium are presented in Section IV. An iterative algorithm to find the Nash equilibrium is proposed in Section V. The simulation results are presented in Section VI. Finally, we conclude in Section VII.

II. RELATED WORK

Task offloading in edge computing has always been a hot research topic in the last decade, and extensive investigation have been conducted.

According to the number of end users, task offloading problems are mainly divided into two categories: single-user scenarios and multi-users scenarios. In the past, many researches focused on the problem of single-user task offloading. Huang et al. [9] proposed a dynamic offloading algorithm based on Lyapunov optimization to improve mobile cloud computing performance while meeting application execution delays. Wang et al. [10] proposed a low-complexity adaptive offloading decision transmission scheduling algorithm based on Lyapunov optimization theory, which optimized the average execution delay and average energy consumption of the task. Ji and Guo [11] studied the offloading and resource allocation of energy-saving calculations in order to maximize energy efficiency under minimum service quality constraints.

In the multi-user scenario, many researchers use game theory to solve the problem of multi-user offloading from the perspective of users. Chen et al. [8] studied the multi-user computing offloading problem of mobile edge cloud computing in a multi-channel wireless interference environment, and explained that the use of a centralized method is an NP problem, so the game theory method is used to achieve efficient distributed computing. Cao et al. [12] proved that the problem of multi-user computing offloading is a potential game and at least one Nash equilibrium exists. They proposed a fully distributed computing offloading (FDCO) algorithm based on machine learning methods, which can converge To a strategic Nash equilibrium without any information exchange. Zheng et al. Li [13] established the M/G/1 queuing model and non-cooperative game framework in the multi-user non-cooperative computing offloading scene. He proved the existence of game Nash equilibrium through theoretical calculations, and designed a distributed algorithm to find Nash equilibrium.

Through the summary of the above related work, we found that there have been a lot of results in the research of task offloading in single-user scenarios. There are also many scholars who use game theory to make offloading decisions in multi-user scenarios, but most of them are from the perspective of the user, considering it as a player. In the scenario of multi-user and multi-edge server, this paper considers the competitive relationship between them from the perspective of the edge server, uses them as players, establishes a game model, and designs an iterative algorithm to find Nash equilibrium.

III. SYSTEM MODEL

In this section, we introduce the system model. The system is composed of multi-users and multi-edge servers. We assume that the terminal device select to offload all the generated computing tasks to the edge servers for execution. Let $U = \{u_1, u_2, \dots, u_n\}$ denotes the n terminal devices. We hold that each terminal device generates a series of task requests according to the frequency that conforms to the Poisson distribution. Let $S = \{s_1, s_2, \dots, s_m\}$ be the corresponding server set. To simplify the model, we set the computing capabilities f of edge servers to be the same. Each of these edge servers is equipped with an AP, which communicates with the terminal device.

The task offloading process of system is shown in Fig. 1. When the task from the device needs to offloaded and processed within its deadline, the device first broadcasts it. Due to the limited coverage of AP, not all edge servers can receive the task request, as shown in Fig. 1(a). Let $S' = \{s_1, s_2, \dots, s_{m'}\}$ denotes the set of edge servers that receive the task request and $x_{i,j}$ denotes whether the edge server j has received the task request $Task_i$. If the edge server has received the ask request then the value is 1; otherwise, 0. The edge server that receives the request allocates resources to the task to compete for it according to its own situation, as shown in Fig. 1(b). Finally, the edge server returns the result of resource allocation to the device, and the device makes the offloading decision, as shown in Fig. 1(c). Next, the communication model and computation model of the system will be introduced.

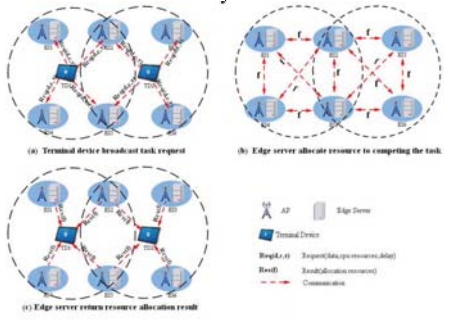
(a) Terminal device broadcast task request
(b) Edge server allocate resource to competing the task
(c) Edge server return resource allocation result
AP
Edge Server
Terminal Device
Req(d,c,a)
Request(data,cpu resources,delay)
Res(f)
Result(allocation resources)
Communication

Fig. 1. The process of task offloading on the multi-users and multi-edge servers system.

A. Communication Model

We assume that the channel bandwidth between the terminal device and the AP is B. Hence, Shannon equation [14] can be used to calculate the data transmission rate of each task on the channel as Eq. (1):

$$r_{i,j} = Blog_2(1 + \frac{p_i h_{i,j}}{\sigma^2}) \tag{1}$$

where p_i is the transmission power of the terminal device, $h_{i,j}$ is the channel gain of the channel and σ^2 represents the noise power on the channel.

B. Computation model

In the computation model, terminal device u_n has a task offloading request,which contains a task $Task_i \triangleq (d_i, c_i, a_i)$. Here d_i denotes the size of data involved in the computation task, c_i denotes the size of computation resources required to accomplish the task, expressed by the CPU operands and a_i denotes the deadline specified by the terminal device for the task. In this paper, the goal of optimization is defined as the computation overhead of the task offloading to the edge server.

According to the communication model in Section III-A, we can calculate the transmission time and energy of the terminal device offloading the task, as shown in Eq. (2) and (3) respectively,

$$\tau^j_{i,trans} = \frac{d_i}{r_{i,j}} \tag{2}$$

$$e_{i,trans}^{j} = p_i \times \frac{d_i}{r_{i,j}} = p_i \times \tau_{i,trans}^{j} \tag{3}$$

where p_i is the transmission power of the device.

The edge server is under loaded as more and more tasks are offloaded. So the offloaded tasks should be put into the cache queue of the edge server to wait for free resources. In this paper, the queuing model of edge server is regarded as single server queuing model (M/M/1). Therefore, the queuing delay of task on the edge server can be calculated by the Eq. (4):

$$\tau_{i,queue}^{j} = \frac{1}{v_j - w_j} \tag{4}$$

where v_j is the service rate and w_j is the request arrival rate of edge server j in unit time. The task starts to be executed when the edge server has enough computing resources. Its computing delay can be expressed as Eq. (5):

$$\tau_{i,com}^{j} = \frac{c_i}{f_j^i} \tag{5}$$

wheref_j^i represents the computation resources allocated to the task by edge server j. In addition, its computing energy consumption can be calculated by the Eq. (6):

$$e_{i,com}^{j} = k \times (\frac{c_i}{f_j^i})^2 \times c_i \tag{6}$$

where k is the energy coefficient, which depends on the chip architecture. The value of k is set to 10^{-9}. According to Eq. (2), (4) and (5), the execution delay of the task on the edge server can be expressed as Eq. (7):

$$t_i^j = \tau_{i,trans}^{j} + \tau_{i,queue}^{j} + \tau_{i,com}^{j} \tag{7}$$

According to Eq. (3) and (6), the energy execution consumption of task on the edge server is as Eq. (8):

$$e_i^j = e_{i,trans}^{j} + e_{i,com}^{j} \tag{8}$$

With the delay and energy consumption model of the task, the computation overhead model for task offloading to edge server j can be defined as Eq. (9):

$$K_i^j = \lambda_i^t \times t_i^j + \lambda_i^e \times e_i^j \tag{9}$$

where λ_i^t and λ_i^e represent the weight of the delay and energy consumption for the task offloading respectively. The two coefficients satisfy the following relationship:

$$\begin{cases} \lambda_i^t + \lambda_i^e = 1 \\ 0 \le \lambda_i^t \le 1 \\ 0 \le \lambda_i^e \le 1 \end{cases} \tag{10}$$

IV. PROBLEM FORMULATION

A. *Multi-edge servers resource allocation Problem*

The edge that can bring the least computation overhead to the task must be selected when the task is offloaded, which not only ensure that the task is completed within the deadline, but also reduce the execution energy consumption to improve the user experience. Therefore, we take minimizing the computation overhead of single task offloading as the optimization goal. Minimizing the computation overhead single task offloading is equivalent to minimizing the computation overhead of all tasks offloading, our objective function is defined as follows:

$$Z_j(f_j^i) = K_i^j = \lambda_i^t \times t_i^j + \lambda_i^e \times e_i^j \tag{11}$$

$$\text{min:} \quad Z_j(f_j^i) \tag{12}$$

$$\text{s. t.} \quad e_{i,trans}^{j} \le \varepsilon_i e_{max}^{j}, \forall i = 1, 2, \cdots, n \tag{13}$$

$$t_i^j \le a_i, \forall i = 1, 2, \cdots, n \tag{14}$$

$$\sum_{j \in E'} x_{ij} \le m, \forall i = 1, 2, \cdots, n \tag{15}$$

$$v_j > w_j, \forall j = 1, 2, \cdots, m' \tag{16}$$

Constraint Eq. (13) is a constraint on the energy of the terminal device to ensure that the energy consumed by the transmission task does not exceed the remaining energy of the terminal device. Constraint Eq. (14) assures that the task can be completed within the delay constraint. Constraint Eq. (15) indicates that not all edge servers will receive the task offloading request. The constraint Eq. (16) guarantees that the number of tasks processed is more than that of arriving tasks on the edge server j per unit time.

B. *Game Formulation*

In this paper, we use game theory to solve the problem and construct a multi-edge servers game model. In the game, the edge servers receiving the task request are considered as the player, and the resource allocated to task $Task_i$ by the edge server in S' is the strategy set. Let $f_{-j} = \{f_1, f_2, \ldots, f_{j-1}, f_{j+1}, \ldots, f_{m'}\}$ denotes the resource allocation results of all edge servers except edge server j. Given the resource allocation results f_{-j} of other edge servers, edge server j would like to allocate the optimal resource amount f_j to the task $Task_i$. The allocation basis is as follows:

$$\min \quad u_j(f_j, f_{-j}), \forall j \in S' \tag{17}$$

where $u_j(f_j, f_{-j})$ is the overhead function of the edge server j, which is defined as Eq. (18) according to Eq. (9):

$$u_j(f_j, f_{-j}) = K_i^j = \lambda_i^t \times t_i^j + \lambda_i^e \times e_i^j \tag{18}$$

We then formulate the problem above as a strategic game $G = (S', \{F_j\}_{j \in S'}, \{u_j\}_{j \in S'})$, where the set of edge servers S' is the set of players, F_j is the set of strategies for player, and the overhead function u_j of each edge server is the cost function to be minimized by player. In the sequel, we call the game G as the multi-servers resource allocation game. To study how the resource allocation conflicts among these edge servers are solved, we investigate whether the game admits at least one Nash equilibrium.

Definition 1(Nash Equilibrium): A resource allocation profile $f^* = \{f_1^*, f_2^*, \ldots, f_j^*, \ldots, f_{m'}^*\}$ is a Nash equilibrium of the multi-edge servers resource allocation game if no edge server can further reduce the computation overhead by unilaterally changing its own resource allocation, i.e.,:

$$u_j(f_j^*, f_{-j}^*) \le u_j(f_j, f_{-j}^*), \forall f_j \in F_j, j \in S' \tag{19}$$

According to the concept of Nash equilibrium, we have the following corollaries:

Corollary 1: In the Nash equilibrium of multi-edge servers resource allocation game, the resource f_j^* allocated by edge server s_j must be the optimal allocation strategy in response to f_{-j}.

Proof: For the edge server s_j, if the allocated resource f_j^* in F_j is not the optimal allocation strategy, there must be another allocation strategy f_j that can reduce the computational overhead of the task, i.e., $u_j(f_j, f_{-j}^*) \le u_j(f_j^*, f_{-j}^*)$. The allocation strategy is changed from f_j^* to f_j, s_j can continue to reduce the computation overhead, but this contradicts with (19). Therefore, in a Nash equilibrium, no edge server can unilaterally reduce the overhead.

C. Game Property

In this part, we analyze the existence and uniqueness of Nash equilibrium in the multi-edge servers resource allocation game. First, two famous lemmas are presented below [15]:

Lemma 1: At least one Nash equilibrium for a non-cooperative game $G = (S', \{F_j\}_{j\in S'}, \{u_j\}_{j\in S'})$ is existed:

(1) The strategy space F_j is a non-empty, convex and compact subset of some Euclidean space.

(2) The overhead function $u_j(f_j, f_{-j})$ is continuous and quasi-convex in F_j.

Lemma 2: A continuous and differentiable function is convex if and only if the second derivative is always greater than or equal to 0 in its domain.

The following corollary shows the existence of Nash equilibrium in the above game.

Corollary 2(existence): There is a Nash equilibrium for the multi-edge servers resource allocation game $G = (S', \{F_j\}_{j\in S'}, \{u_j\}_{j\in S'})$, where u_j is defined by (18).

Proof: Obviously, F_j is a non-empty, convex and compact subset. Next, we calculate the second derivative of $u_j(f_j, f_{-j})$.

The second derivative is shown in Eq. (20):

$$\frac{\partial^2 u_j}{\partial (f_j^i)^2} = \frac{c_i}{(f_j^i)^3} + 2 \times k \times f_j^i \times c_i \tag{20}$$

According to the second derivative, it is easy to verify Eq. (21):

$$\frac{\partial^2 u_j}{\partial (f_j^i)^2} > 0 \tag{21}$$

Thus, $u_j(f_j, f_{-j})$ is convex function of f_j and is quasi-convex. By Lemma 1, there is at least one Nash equilibrium for the multi-edge server resource allocation game G. Corollary 2 holds.

Lemma 3: For a non-cooperative game G whose overhead function is quasi-convex if satisfy Eq. 22:

$$\sum_{\substack{k=1,2,\cdots,m' \\ k\notin j}} |\frac{\partial^2 u_j}{\partial f_j^i \partial f_k^i}| < |\frac{\partial^2 u_j}{\partial (f_j^i)^2}| \tag{22}$$

Then G is dominance-solvable.

The following corollary shows the uniqueness of Nash equilibrium of the above game.

Corollary 3(uniqueness): There is only one unique Nash equilibrium for the multi-edge servers resource allocation game $G = (S', \{F_j\}_{j\in S'}, \{u_j\}_{j\in S'})$, where u_j is defined by (18).

Proof: $\frac{\partial^2 u_j}{\partial f_j^i \partial f_k^i} = 0$, it is easy to verify according to (21):

$$\sum_{\substack{k=1,2,\cdots,m' \\ k\notin j}} |\frac{\partial^2 u_j}{\partial f_j^i \partial f_k^i}| = 0 < |\frac{\partial^2 u_j}{\partial (f_j^i)^2}| \tag{23}$$

By Lemma 3, the game G has a unique Nash equilibrium. Corollary 3 holds.

V. ALGORITHMS

In this section, an iterative algorithm is proposed to find the Nash equilibrium of the game. The details are shown in Algorithm 1.

We set the maximum number of iterations I to make the algorithm converge. The setting of I can ensure that all edge servers no longer update their resource allocations, and the overall time complexity of algorithm is $O(I \times m')$. The update f_j' bases on resources allocated by other edge servers.

Algorithm 1 Game-Based Task Offloading Algorithm(GBTOA)

```
Input: $Task_i$, $E$ and data required for the above formula;
Output: the result set of resource allocation for the game to reach
    Nash equilibrium $f^* = \{f_1^*, f_2^*, \cdots, f_m^*\}$;
 1: Initialize
 2: the initial resources allocated to the task by the edge servers
    that received the task request, namely $f = \{f_1, f_2, \cdots, f_m\}$;
 3: for each edge server $s_j \in S'$ do
 4:     use Eq. (18) to calculate the overhead function
    $u_j(f_j, f_{-j})$;
 5: end for
 6: End Initialize
 7: set the initial iteration: iteration = 1 and set the max iteration:
    Max Iteration;
 8: while iteration $\leq$ Max Iteration do
 9:     for $j = 1$ to $m'$ do
10:         find $f_j'$ that minimizes the overhead function
    $u_j(f_j, f_{-j})$ with $f_{-j}$;
11:         use Eq. (18) to calculate the overhead function
    $u_j(f_j', f_{-j})$;
12:         if $u_j(f_j', f_{-j}) < u_j(f_j, f_{-j})$ then
13:             $f_j = f_j'$, update the value of $f_j$ to $f_j'$;
14:         else
15:             not update $f_j$;
16:         end if
17:     end for
18:     iteration = iteration+1, enter the next iteration;
19: end while
20: for $j = 1$ to $m'$ do
21:     $f_j^* = f_j$, the updated value obtained in the last iteration
    is $f_j^*$;
22: end for
23: return $f^*$
```

VI. SIMULATION

In this section,we provide illustrative results to demonstrate the performance of the proposed algorithm and computation offloading strategy.

A. Parameter Configuration

In the simulation, the terminal device generates 100 task offloading requests fixedly on the basis of Poisson Expectation $q = 1.5$. We select image processing tasks as the computation task data set. According to the actual size of the image,the data size of a task is uniformly selected from [300, 1000] KB, the computation resource required to accomplish the task is determined by the data size, and the CPU cycles required to process each bit of data are between 500 and 1500 cycles. Similarly, the expected deadline of a task is randomly generated within 1s-4s [5]. In addition,the number of devices is varied from 15 to 150 with increment 15. The number of edge servers varies from 10-80. It is assumed that the CPU computation resources of all edge servers is the same, which are all set to 40 GHz. Considering the computation overhead for each task offloading determined by delay and power consumption, the time weight coefficient in this paper is any one of the set $\{0.1, 0.3, 0.5, 0.7, 0.9\}$, and the corresponding energy consumption weight coefficient is $\lambda^e = 1 - \lambda^t$. We use Shannon Equation to calculate the transmission rate between the terminal device and the edge server, where we set the channel bandwidth

$B = 3$ MHz, the transmission power of terminal devices $p_n = 1$ W, the channel gain $h_{n,j} = 10^{-6}$, and the average noise power $\sigma^2 = 10^{-9}$ W , respectively.

In order to verify the convergence of the proposed algorithm, we first show the dynamics of computation overhead of single task during the iteration process on the edge server that receives its offloading request in Fig. 2. We see that with the number of iterations increasing, the computation overhead of single task offloading keeps changing, but eventually converges to a stable point, which illustrates that all edge servers participating in the game have reached a balanced state based on their own resource conditions,i.e., in the Nash equilibrium of the multi-edge server game.

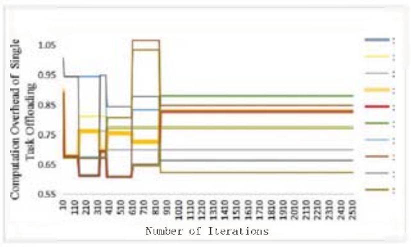

Fig. 2. Dynamics of computation overhead for single task offloading.

Fig. 3 shows the dynamics of overall computation overhead of task offloading in the multi-terminal and multi-edge server system during the iteration. We can observe that the computation overhead decreases when the number of iterations continues to increase. The downward trend of overhead is obvious before 150 times, but it gradually eases after 150 times. Eventually, the system reached a stable state. This shows that the algorithm can achieves Nash equilibrium within a limited number of times (1270 in this simulation experiment).

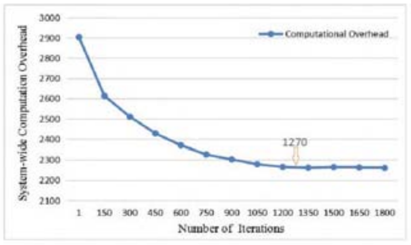

Fig. 3. Dynamics of system-wide computation overhead.

Next, the proposed GBTOA algorithm is compared with benchmark algorithms to evaluate the performance of the algorithm. The benchmark algorithms are RA, RGA, QGA and GA [16].

Fig. 4 shows that the effect of delay weight coefficient on the total computation overhead of the system. Notice that the total computation overhead of GBTOA and other comparison algorithms is increasing in the process of increasing from 0.1 to 0.9, due to the fact that the reduction of the energy consumption weighting coefficient leads to the increase of the computation overhead. Moreover, the proposed GBTOA always obtains the smallest computation overhead than other algorithms as the time weight coefficient increases. When the delay weight coefficient is 0.1, the computation overhead of GBTOA reduces by 25.5% compared to RG and 12.5% lower than that of GA.

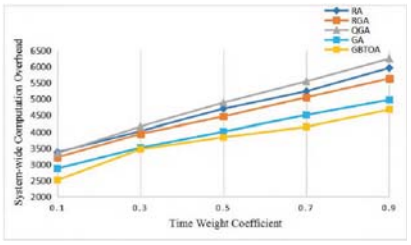

Fig. 4. System-wide computation overhead with different delay weight coefficient.

Fig. 5 and Fig. 6 present the influence of the number of terminal devices on the system-wide computation overhead and energy consumption, respectively. In Fig. 5, we observe the total overhead increases as the number of tasks increases. When the number of terminal devices is 15, GBTOA has the smallest computational overhead, it has a smaller gap with other comparison algorithms. Compared with GA, its performance is improved by 10%. This is because that the edge servers has sufficient resources to handle tasks at this time. As the number of terminal devices increases, the gap between GBTOA and other algorithms is larger, and the advantages are obvious. In Fig. 6, we see that the energy consumption of GBTOA is significantly different from other algorithms when the number of terminal devices is 15. As the number of terminal devices increases, the gap with the three algorithms of RA, RGA, and QGA is gradually expanding. GBTOA reduces energy consumption by 38.5% compared with RA When the number is 120. Even though the performance is reduced compared with GA, it is still lower than GA's energy consumption.

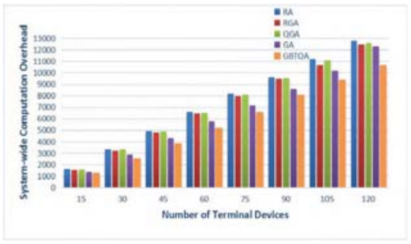

Fig. 5. System-wide computation overhead with number of terminal devices.

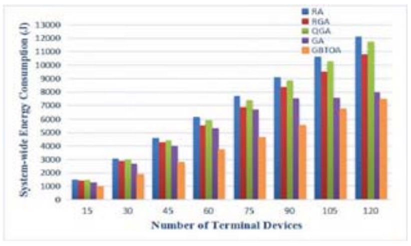

Fig. 6. System-wide energy consumption with number of terminal devices.

We show the system-wide computation overhead and energy consumption with different number of edge servers in Fig. 7 and Fig. 8. We see that, for the metric of computation overhead, the computation overhead of GBTOA is slowly decreasing before 20 edge servers, which indicates that edge server players can allocate more resources to tasks than before by increasing the number within a certain range. There is almost no obvious change in computation overhead after 20, which means that it is impossible to allocate more optimal resources to the task by increasing the number of players. In addition, the computation overhead of GBTOA is always smaller than the overhead of other algorithms. For the metric of energy consumption, the total energy consumption of GBTOA tends to decrease slightly when the number of edge servers increases from 10 to 20. While the number of edge servers exceeds 20, the total energy consumption tends to be stable. It can be seen that the optimal amount of resources that edge servers can allocate to the task through the game is also the optimal solution of GBTOA when the number of edge servers is 20. Even if the number of players increases, the algorithm can still find the same optimal solution within a limited number of times.

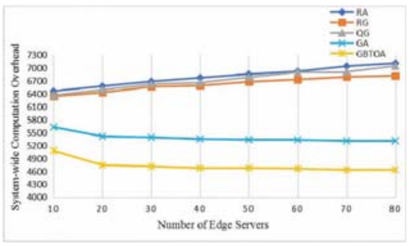

Fig. 7. System-wide computation overhead with different edge servers.

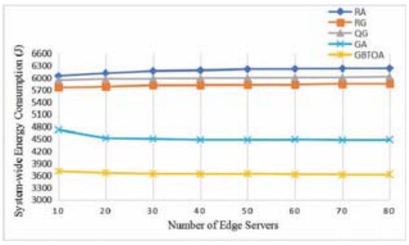

Fig. 8. System-wide energy consumption with different edge servers.

In Fig. 9, We see that more and more tasks are offloaded with the number of terminal devices increasing, the average resource utilization of edge servers of GBTOA and other algorithms has gradually increased, but the average resource utilization of GBTOA is always the highest. In addition, the average resource utilization rate of GBTOA is getting larger and larger than the other algorithms as the number of terminal devices increase before 75.

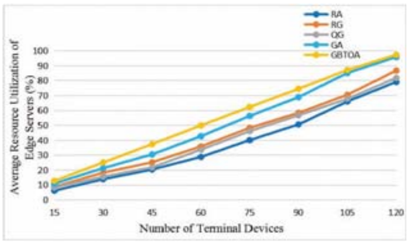

Fig. 9. Average resource utilization rate of edge servers with different number of terminal devices.

VII. CONCLUSION

In scenarios of multi-users and multi-edge servers, this paper proposes a new task offloading strategy. Terminal devices broadcast task offloading requests to edge servers within the communication range and the edge servers allocate resources competition the task according to their own resources situation. We describe the problem as a non-cooperative game, which is called multi-edge servers resource allocation game and theoretically prove that there is a unique Nash equilibrium of the game. Then, we design a iterative algorithm to find Nash equilibrium. We define the task execution delay and energy consumption as the computation overhead of task offloading. When the game reaches the Nash equilibrium, using the computing resources allocated by each edge server, the user selects the edge server offloading task that meets the delay constraint and has the least offloading overhead. Finally, simulation experiments show that the proposed task offloading strategy not only improves the resource utilization of edge servers, but also meets the demand of end users to minimize overhead.

ACKNOWLEDGMENT

This work was supported in part by the National Natural Science Foundation of China under Grant 61972140, and in part by the National Defense Basic Research Plan under Grant JCKY2018110C145.

REFERENCES

[1] N. Heuveldop, "Ericsson mobility report (5g)," *Ericsson, Stockholm*, 06 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. 6, pp. 4177–4190, 2018.
[3] Y. Mao, J. Zhang, S. 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, pp. 5994–6009, 2017.
[4] J. Gubbi, R. Buyya, S. Marusic, and M. Palaniswami, "Internet of things (IoT): A vision, architectural elements, and future directions," *Future Generation Computer Systems*, vol. 29, p. 16451660, 2013.
[5] Liu C , Li K , Liang J and Li, Keqin, "COOPER-MATCH: Job Offloading with A Cooperative Game for Guaranteeing Strict Deadlines in MEC," *IEEE Transactions on Mobile Computing*, vol. PP, pp. 1–1, 06 2019.
[6] Yin, Luxiu and Luo, Juan and Luo, Haibo, "Tasks Scheduling and Resource Allocation in Fog Computing Based on Containers for Smart Manufacturing," *IEEE Transactions on Industrial Informatics*, vol. 14, no. 10, pp. 4712–4721, 2018.
[7] Y. Xiao and M. Krunz, "QoE and power efficiency tradeoff for fog computing networks with fog node cooperation," 05 2017, pp. 1–9.
[8] 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, 10 2015.
[9] Huang, Dong and Wang, Ping and Niyato, Dusit, "A Dynamic Offloading Algorithm for Mobile Computing," *IEEE Transactions on Wireless Communications - TWC*, vol. 11, pp. 1991–1995, 06 2012.
[10] Wang, Junyi and Peng, Jie and Wei, Yanheng and Liu, Didi and Fu, Jielin, "Adaptive application offloading decision and transmission scheduling for mobile cloud computing," *China Communications*, vol. 14, pp. 169–181, 03 2017.
[11] Ji, Luyue and Guo, Songtao, "Energy-Efficient Cooperative Resource Allocation in Wireless Powered Mobile Edge Computing," *IEEE Internet of Things Journal*, vol. PP, pp. 1–1, 11 2018.
[12] H. Cao and J. Cai, "Distributed multi-user Computation Offloading for Cloudlet-Based Mobile Cloud Computing: A Game-Theoretic Machine Learning Approach," *IEEE Transactions on Vehicular Technology*, vol. 67, no. 1, pp. 752–764, 2018.
[13] J. Zheng, Y. Cai, Y. Wu, and X. Shen, "Dynamic Computation Offloading for Mobile Cloud Computing: A Stochastic Game-Theoretic Approach," *IEEE Transactions on Mobile Computing*, vol. PP, pp. 1–1, 06 2018.
[14] T. Q. Dinh, J. Tang, Q. D. La, and T. Q. S. Quek, "Offloading in Mobile Edge Computing: Task Allocation and Computational Frequency Scaling," *IEEE Transactions on Communications*, vol. 65, pp. 3571–3584, 08 2017.
[15] A. W. Roberts, "Convex functions.in Handbook Convex Geometry," *The Netherlands: Elsevier*, vol. 187, pp. 1081–1104, 1993.
[16] J. jie, "Cooperative task scheduling in mobile edge computing system," *University of Electronic Science and Technology of China*, 06 2018.