QoE-Aware Decentralized Task Offloading and Resource Allocation for End-Edge-Cloud Systems: A Game-Theoretical Approach

Ying Chen, *Member, IEEE*, Jie Zhao, Yuan Wu, *Senior Member, IEEE*, Jiwei Huang, *Member, IEEE*, and Xuemin Shen, *Fellow, IEEE*

Abstract—Due to the limited computing resource and battery capability at the mobile devices, the computation-intensive tasks generated by mobile devices can be offloaded to edge servers or cloud for processing. In this paper, we study the multi-user task offloading problem in an end-edge-cloud system, in which all user devices compete for the limited communication and computing resources. Particularly, we first formulate the offloading problem with the goal of maximizing the Quality of Experience (QoE) of the users subject to resource constraints. Since each user focuses on maximizing its own QoE, we reformulate the problem as a Multi-User Task Offloading Game (MUTO-Game). We then identify an important property that for any device, both the communication interference and the degree of computing resource competition can be upper bounded. Based on the property, we further theoretically prove that there exists at least one Nash Equilibrium offloading strategy in the MUTO-Game. We propose the Game-based Decentralized Task Offloading (GDTO) approach to obtain the Nash Equilibrium offloading strategy. Finally, we analyze the upper bound for the convergence time and characterize the performance guarantee of the obtained offloading strategy for the worst case. A series of experimental results are presented, in comparison with both the centralized optimal approach and the approximate approaches.

Index Terms—Task offloading, end-edge-cloud, quality of experience (QoE), game model

1 Introduction

With the rapid development of mobile computing techniques, more and more computation-intensive tasks are generated by applications or services running on mobile devices, such as AR, VR [1], etc. However, the computing resources and the battery capacity of mobile devices are generally limited [2]. A feasible solution is to offload the tasks to remote cloud with sufficient computing resources. However, offloading to cloud suffers from several limitations. In particular, since the cloud is usually located far away from mobile devices, offloading to the cloud would result in high delay [3] and degrade users' Quality of Experience (QoE). Moreover, transmitting all the tasks to the cloud would put a heavy burden on the core networks. To solve the above issues, one promising solution is to take advantage of the recently proposed Mobile Edge Computing (MEC) [4], [5], [6], [7] framework. Based on the MEC framework, the edge servers with computing resources are placed at the network access points (typically in base stations) close to mobile devices. In this way, the service delay for mobile devices can be reduced and users' QoE can be improved. The burden of transmitting tasks from mobile devices on the core networks can also be relieved [8].

Although the edge servers are equipped with some computing resources, the computing capacities are still limited compared with the central cloud [7], [9]. With the rapid increase in the number of mobile applications and mobile devices, the edge servers cannot efficiently process all the offloaded tasks from mobile devices. Therefore, edge servers are typically backed-up by a remote cloud. When the edge servers become heavily loaded by the computation workloads from the mobile devices, part of the tasks can be further offloaded to the cloud [10], [11], [12]. Thus, the task offloading in an end-edge-cloud system has attracted more and more attention from both industry and academia.

However, solving the task offloading problem in the end-edge-cloud system faces several challenges. First, the transmission resources and the computing resources on edge servers are limited [9]. All the user devices in the system have to

- *Ying Chen and Jie Zhao are with Computer School, Beijing Information Science and Technology University, Beijing 100101, China. E-mail: {chenying, zhaojie99723}@bistu.edu.cn.*
- *Yuan Wu is with the State Key Laboratory of Internet of Things for Smart City, University of Macau, Macau, China. E-mail: yuanwu@um.edu.mo.*
- *Jiwei Huang is with the Beijing Key Laboratory of Petroleum Data Mining, China University of Petroleum, Beijing 102249, China. E-mail: huangjw@cup.edu.cn.*
- *Xuemin Shen is with the Department of Electrical and Computer Engineering, University of Waterloo, Waterloo, ON N2L 3G1, Canada. E-mail: sshen@uwaterloo.ca.*

Manuscript received 24 May 2022; revised 15 November 2022; accepted 15 November 2022. Date of publication 18 November 2022; date of current version 5 December 2023.
This work was supported in part by the National Natural Science Foundation of China under Grants 61902029, 61972414, 61973161, and 61991404, in part by the R&D Program of Beijing Municipal Education Commission under Grant KM202011232015, in part by Beijing Nova Program under Grant Z201100006820082, in part by Beijing Natural Science Foundation under Grant 4202066, in part by the Science and Technology Development Fund of Macau SAR under Grant 0162/2019/A3, in part by Guangdong Basic and Applied Basic Research Foundation under Grant 2022A1515011287, and in part by the Guangdong-Macau Joint Laboratory for Advanced and Intelligent Computing under Grant GDST 2020B1212030003.
(Corresponding author: Jiwei Huang.)
Digital Object Identifier no. 10.1109/TMC.2022.3223119

compete for the limited transmission and computing resources [13], and each user device aims at maximizing its own benefit. Therefore, how to achieve a balanced offloading strategy among all the users while maximizing the total benefit of the whole system is a challenging problem. Moreover, the growing solution space size (due to the growing number of the mobile devices and the available channels) also dramatically increases the computational complexity in finding the optimal offloading strategy. There are some heuristic approaches with relatively low complexity that can provide offloading strategies approximated to the globally optimal one. Nevertheless, it is technically difficult to provide a guaranteed performance gap between the approximated one and the globally optimal one.

In this paper, we study the multi-user task offloading problem in the end-edge-cloud system. Our goal is to maximize the QoE of the users subject to the resource constraints. We formulate the problem as a Multi-User Task Offloading Game (MUTO-Game) model, and theoretically prove that there exists at least one Nash Equilibrium offloading solution. Then, we propose a Game-based Decentralized Task Offloading (GDTO) algorithm to solve the game model and reach one of the feasible equilibrium offloading strategies. Theoretical analysis for the convergence time and performance guarantee of the worst case are given. The main contributions of this paper are summarized as follows.

- We study the multi-user task offloading problem in an end-edge-cloud system, where the edge servers are located in base stations and are also connected with a central cloud. Tasks of users can be processed locally on the user devices, or offloaded to be processed on the edge servers through wireless channels, or offloaded to the cloud. Our goal is to maximize all the users' QoE. The offloading constraints include both communication resource constraints and computing resource constraints. The offloading decisions include both the task offloading decisions (i.e., the local processing, the edge/cloud processing) and the channel resource allocation for offloading transmissions.
- Considering that all users are self-interested and aim at optimizing their respectively own QoE, we reformulate the multi-user task offloading problem as a MUTO-Game model. Each game player is the user in the system with the objective of maximizing its own QoE. The solution to the MUTO-Game is defined as its Nash Equilibrium solution. Then, we theoretically demonstrate that for any user in the system, both its communication interference and degree of computing resource competition can be upper bounded (i.e., Lemma 1). Next, we prove, by demonstrating six possible cases, that the MUTO-Game is a potential game and give the corresponding potential function (i.e., the results in Theorem 1), based on which we can establish that our MUTO-Game has at least one feasible Nash Equilibrium strategy.
- We propose the Game-based Decentralized Task Offloading (GDTO) algorithm to obtain the Nash Equilibrium offloading strategy for the end-edge-cloud system. Each user individually makes its own offloading decision, and the GDTO algorithm can be implemented in a distributed way. To theoretically analyze the performance of the GDTO algorithm, we give the upper bound of the number of iterations to obtain an Nash Equilibrium offloading strategy (i.e., Theorem 2). Furthermore, we define the Price of Anarchy (PoA), i.e., the ratio of the QoE obtained by the worst Nash Equilibrium offloading strategy and the QoE of the centralized optimal offloading strategy. We then give the lower bound of the PoA (i.e., Theorem 3).
- We carry out extensive experiments to evaluate our GDTO algorithm. The experiment results show that when the offloading solution space size increases exponentially, the increasing speed of the number of iterations for GDTO to reach an Nash Equilibrium solution is less than linear speed. We also carry out two groups of experiments with different solution space scales to evaluate GDTO. The small scale experiment with the optimal strategy shows that the QoE of our GDTO algorithm is close to that of the centralized optimal solution. The large scale experiment with four approximate algorithms validate the superiority of our GDTO algorithm.

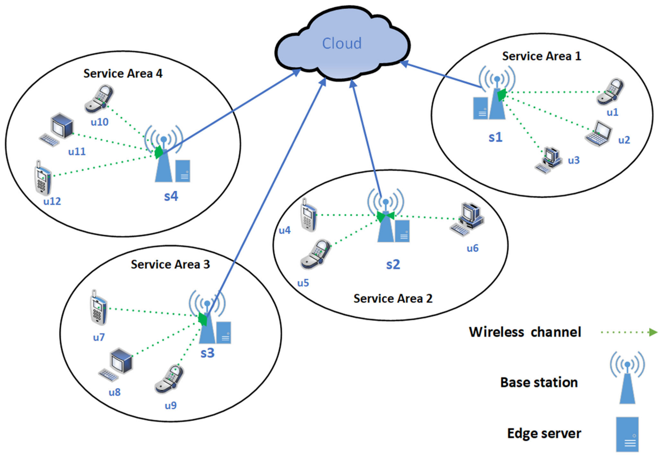

Fig. 1. An example of the system scenario.

The remainder of this paper is organized as follows. Section 2 describes the system model and formulates the multi-user task offloading problem. Section 3 reformulates the offloading problem as the MUTO-Game model, and gives the theoretical analysis for the property of the MUTO-Game. Section 4 proposes the decentralized GDTO algorithm, and provides the theoretical analysis for GDTO's performance. Section 4 provides the experimental evaluation. Section 6 presents the related works and Section 7 gives the conclusion of this paper.

2 System Model and Problem Formulation

2.1 System Model

Fig. 1 depicts the system scenario. There are n User Devices (UDs) represented by $U = \{u_1, u_2, \ldots, u_n\}$ and m base stations represented by $S = \{s_1, s_2, \ldots, s_m\}$. Each UD u_i has a computing task (B_i, X_i, δ_i) to process. Here, B_i represents the task's size (in bits), X_i represents the number of CPU cycles required to complete the task. $\delta_i \in S$ stands for the base station that the UD u_i connects to. Each base station (BS) is equipped with an edge server to provide computing services for the UDs. In this following, we treat the BS δ_i and the edge

TABLE 1
Key Notations

Notations	Definitions
u_i	the ith UD, $i \in \{1, 2, \ldots, n\}$
B_i	the task size of u_i
$\mathcal{U}$	the set of UDs
f_i	the u_i's computing capability (cycles/s)
γ_i	the u_i's weight parameter of computation resource
s_j	the jth edge server, $j \in \{1, 2, \ldots, m\}$
f_{s_j}	the s_j's computing capability (cycles/s)
c_j	the number of channels on edge server s_j
f_{δ_i}	the δ_i's computing capability (cycles/s)
f_{i,δ_i}	the computing capability obtained by u_i on δ_i
λ_i	the decision of offloading method for u_i
k_i	the channel selection of u_i
a_i	offloading decision for u_i, $a_i = (\lambda_i, k_i)$
$W_i^{k_i}$	k_i's bandwidth on BS δ_i
ϖ_0	background noise
$g_i^{k_i}$	the channel gain between u_i and δ_i on wireless channel k_i
ϱ_i	the coefficient of energy consumption per CPU cycle for UD u_i
$r_i^{k_i}$	u_i's data rate on k_i
$\hat{r}$	data rate between BS and cloud
$E_{a_{-i}}(a_i)$	the quality of experience value of offloading decision a_i
α	the rate of change for QoE
β	the reference value of QoE
E_{max}	the maximum QoE
E_{min}	the minimum QoE
T_{max}	the maximum delay
T_{min}	the minimum delay
EC_{max}	the maximum energy consumption
EC_{min}	the minimum energy consumption
$\phi_{-a}(a_i)$	potential function
M_i^p	threshold of wireless communication interference
M_i^γ	threshold of computation capability competition

server δ_i interchangeable. For each BS s_j, there are c_j wireless channels. Table 1 lists the main notations of this paper. UDs can offload tasks to the edge servers for processing. Referring to existing works [10], [12], [14], [15], we consider the scenario that the UDs are pre-assigned to the BS. We also integrate the collaborative edge-cloud computing framework, and when the UDs assigned BS is overloaded, users' tasks can be further offloaded to the central cloud for processing. In this way, the UDs transmit their tasks to the cloud through the BSs. If the channel resources are not enough or exhausted, the UDs will have to execute their tasks locally on the devices. To be more specific, we give the detailed definition of offloading decision for each UD in Definition 1.

Definition 1. *(Offloading Decision a_i) Let $a_i \in \{(0,0) \bigcup (\lambda_i, k_i)\}$ represent the offloading decision of UD u_i, i.e., to execute the task locally, or offload the task to the edge serve or the cloud through which wireless channel. Specifically, $a_i = (0,0)$ represents that the task is executed locally. $\lambda_i = 1$ represents that the task is offloaded to be executed in the edge server δ_i, and $\lambda_i = 2$ represents that the task is offloaded to the cloud. $k_i \in \{1, \ldots, c_{\delta_i}\}$ represents the selected wireless channel of the BS δ_i for sending the task.*

Then, the collective offloading strategy of all the UDs can be defined by Definition 2.

Definition 2. *(Offloading Strategy* **a** *of all the UDs) An offloading strategy for all the UDs is the collection denoted by* $\mathbf{a} = (a_1, \ldots, a_n)$.

2.2 Communication Model

In this paper, we consider that the BS provides multiple available channels, and each UD can choose only one of the channels provided by the BS to access. Multiple users may access the same wireless channel, and there exists interference among the UDs that access the same channel.

2.2.1 Offloading to Edge Server

When multiple UDs communicate with the BS on the same channel, there is interference between these UDs. The Signal-to-Interference-plus-Noise Ratio (SINR) for UD u_i is

$$\Upsilon_i^{k_i} = \frac{p_i g_i^{k_i}}{\varpi_0 + \sum_{u_l \in U/\{u_i\}: k_l = k_i \cap \delta_l = \delta_i} p_l g_l^{k_i}}. \tag{1}$$

ϖ_0 is the background noise variance. p_i is the transmission power of u_i, $g_i^{k_i}$ is the uplink channel gain between u_i and BS δ_i on channel k_i. The transmission rate for UD u_i's communication with BS δ_i via channel k_i is

$$r_i^{k_i} = W_i^{k_i} \log_2(1 + \Upsilon_i^{k_i}), \tag{2}$$

where $W_i^{k_i}$ is the bandwidth on wireless channel k_i. When the number of UDs assigned to the same channel increases, the transmission interference will increase, and the transmission rate will decrease. In this paper, there is a constraint r_{min} for the minimum transmission rate, as shown in Eq. (3). r_{min} is specified by the BS provider [13]. If the transmission rate $r_i^{k_i}$ is smaller than r_{min}, the transmission will be terminated.

$$r_i^{k_i} \geq r_{min}, u_i \in U. \tag{3}$$

According to Eq. (2), the communication delay between u_i and δ_i is

$$\hat{t}_{i,\delta_i} = \frac{B_i}{r_i^{k_i}}. \tag{4}$$

According to [16], the energy consumption of communication for UD u_i is

$$\hat{e}_{i,\delta_i} = p_i \hat{t}_{i,\delta_i} = p_i \frac{B_i}{r_i^{k_i}}. \tag{5}$$

2.2.2 Offloading to Cloud

When the edge servers' resources are not enough, the tasks of UDs can be offloaded to the cloud for processing. In this way, the BS transmits the data to the remote cloud by a high-speed fiber communication link. UD u_i's delay of transmitting from the BS to the cloud is

$$\hat{t}_c = \frac{B_i}{\hat{r}}, \tag{6}$$

where $\hat{r}$ is the transmission rate from the BS to the cloud. Therefore, the total transmission delay of UD u_i is

$$\hat{t}_{i,c} = \hat{t}_{i,\delta_i} + \hat{t}_c = \frac{B_i}{r_i^{k_i}} + \frac{B_i}{\hat{r}}. \tag{7}$$

2.3 Computation Model

Similar to [17], it is considered that the cloud has sufficient computing resources, and the computation delay on the cloud is very small, and can be neglected. Let f_i represent the computing capability of UD u_i. When UD u_i chooses to process the task locally ($a_i = (\lambda_i, k_i) = (0, 0)$), the computation delay of u_i's task is

$$t_i = \frac{X_i}{f_i}. \tag{8}$$

If UD u_i offloads task to the edge server, then, $a_i = (\lambda_i, k_i) = (1, k_i)$. Recall that δ_i represents the BS that u_i connects to. Let f_{δ_i} represent the computing capability of the edge server δ_i. The computing resources of the edge server will be shared by all the tasks of the UDs offloading to δ_i. In this paper, the computing resource that δ_i assigns to u_i is,

$$f_{i,\delta_i} = \frac{\gamma_i}{\sum_{u_l \in U: \lambda_l = \lambda_i \cap \delta_l = \delta_i} \gamma_l} f_{\delta_i}, \tag{9}$$

where γ_i is the weight parameter of UD u_i [18]. $\sum_{u_l \in U: \lambda_l = \lambda_i \cap \delta_l = \delta_i} \gamma_l$ is the sum of the weight parameter of all the UDs offloading task to δ_i. To guarantee the quality of service for each UD, there is a minimum threshold f_{min} as shown in Eq. (10).

$$f_{i,\delta_i} \geq f_{min}, u_i \in U. \tag{10}$$

and the computing energy consumption [19] of UD u_i can be calculated as

$$e_i = \varrho_i X_i, \tag{11}$$

where ϱ_i is the energy consumption factor per CPU cycle for UD u_i.

The computation delay of UD u_i is

$$t_{i,\delta_i} = \frac{X_i}{f_{i,\delta_i}}, \tag{12}$$

2.4 QoE Model

Similar to the related works [16], [18], [19], [20] on task offloading, in this paper, we consider that each device generates a task, and focus on the offloading strategy of the tasks. Thus, similar to the related works, the queueing delay is ignored in this paper. We can obtain the total delay of each UD $u_i \in U$ as,

$$T_i = \begin{cases} t_i, & a_i = (0,0) \\ \hat{t}_{i,\delta_i} + t_{i,\delta_i}, & a_i = (1, k_i) \\ \hat{t}_{i,\delta_i} + \hat{t}_c, & a_i = (2, k_i) \end{cases}. \tag{13}$$

The total energy consumption of each UD $u_i \in U$ is

$$EC_i = \begin{cases} e_i, & a_i = (0,0) \\ \hat{e}_{i,\delta_i}, & a_i \neq (0,0) \end{cases}. \tag{14}$$

Similar to [16], we consider both the latency and energy consumption of each UD, and formulate the cost of each UD as

$$Cost_i = \tau_i^t T_i + \tau_i^e EC_i, \tag{15}$$

where τ_i^t and τ_i^e (with $\tau_i^t + \tau_i^e = 1$) represent the weighted parameters of delay and energy consumption of UD u_i, respectively. Each UD u_i can set its corresponding τ_i^t and τ_i^e based on its own priority and preference. For example, when UD u_i is in a low-battery state and is more concerned about energy consumption, it can set a larger τ_i^e. When UD u_i runs latency-sensitive applications, it can set a larger τ_i^t.

For each UD u_i, there exists an upper bound T_{max} of the maximum delay and a lower bound T_{min} of the minimum delay. There is also an upper bound EC_{max} of the maximum energy consumption and a lower bound EC_{min} of the minimum energy consumption. The maximum delay can be upper bounded by $T_{max} = X_{max}/f$, where $f = \min\{f_1, f_2, \ldots, f_n\}$ is the minimum computing capability and $X_{max} = \max\{X_1, X_2, \ldots, X_n\}$ is the maximum number of CPU cycles for all UDs' tasks. The lower bound of minimum delay corresponds to the user completing task through the edge server in an ideal case. In other words, $T_{min} = B_{min}/r_{max} + X_{min}/f_{max}$, where $f_{max} = \max\{f_{s_1}, f_{s_2}, \ldots, f_{s_m}\}$ represents the maximum computing capability for edge servers, $B_{min} = \min\{B_i, u_i \in U\}$ represents the minimum size of the task for all UDs, $X_{min} = \min\{X_1, X_2, \ldots, X_n\}$ is the minimum number of CPU cycles for all UDs' tasks, $r_{max} = W\log_2(1 + p_{max} g_{max}/\varpi_0)$, $p_{max} = \max\{p_1, p_2, \ldots, p_n\}$ represents the maximum transmission power for all UDs and $g_{max} = \max\{g_i^{k_i}, u_i \in U\}$ represents the maximum channel gain for all UDs and channels. Similarly, $EC_{max} = \varrho_{max} X_{max}$ and $EC_{min} = p_{min} B_{min}/r_{max}$, where $\varrho_{max} = \max\{\varrho_1, \varrho_2, \ldots, \varrho_n\}$ is the maximum energy consumption factor per CPU cycle for all UDs and $p_{min} = \min\{p_1, p_2, \ldots, p_n\}$ represents the minimum transmission power for all UDs.

For each UD u_i, we can obtain $Cost_i \leq \tau_i^t T_{max} + \tau_i^e EC_{max} \leq (\tau_i^t + \tau_i^e)\max\{T_{max}, EC_{max}\} = \max\{T_{max}, EC_{max}\}$, and $Cost_i \geq \tau_i^t T_{min} + \tau_i^e EC_{min} \geq (\tau_i^t + \tau_i^e)\min\{T_{min}, EC_{min}\} = \min\{T_{min}, EC_{min}\}$. Therefore, there exists an upper bound $Cost_{max} = \max\{T_{max}, EC_{max}\}$ of the maximum cost and a lower bound $Cost_{min} = \min\{T_{min}, EC_{min}\}$ of the minimum cost.

Generally speaking, the QoE is often negatively correlated with the cost. When the cost increases, the QoE will decrease. Referring to [21], [22], in this paper, the QoE of each UD u_i is defined as follows:

$$E_{a_{-i}}(a_i) = \begin{cases} E_{min}, & a_i = (0,0) \\ -\alpha \log_2 Cost_i + \beta, & a_i \neq (0,0) \end{cases}, \tag{16}$$

where parameter $\alpha = \frac{E_{max} - E_{min}}{\log_2(Cost_{max}/Cost_{min})} > 0$ and parameter $\beta = \frac{E_{max}\log_2 Cost_{max} - E_{min}\log_2 Cost_{min}}{\log_2(Cost^{max}/Cost^{min})} > 0$. $E_{\max}$ denotes the maximum value of the QoE which corresponds to $Cost_{\min}$, and $E_{\min}$ denotes the minimum value of the QoE which corresponds to $Cost_{\max}$. To be more specific, when the cost approaches $Cost_{min}$, the QoE approaches the maximum value E_{max}. When the cost approaches $Cost_{max}$, the QoE approaches the minimum value E_{min}. In $E_{a_{-i}}(a_i)$, $a_{-i} = (a_1, \ldots, a_{i-1}, a_{i+1}, \ldots a_n)$ represents the offloading decisions of all the UDs excluding UD u_i.

2.5 Problem Formulation

The optimization goal for the task offloading problem is to find the offloading strategy to maximize the sum of QoE of all UDs subject to the resource constraints. The detailed problem formulation is shown in Problem (17).

$$\begin{aligned} &max \textstyle\sum_{u_i \in U} E_{a_{-i}}(a_i) \\ &s.t. \quad (3), (10). \end{aligned} \tag{17}$$

Problem (17) is an NP-hard problem[18]. Solving this problem faces severe difficulties and challenges. First, there is a competitive relationship between the UDs. All the UDs compete for the limited resources. Each rational UD focuses on its own benefit, and wants to obtain more resources to maximize its own QoE. Thus, it is hard for all the competing UDs to reach a stable and equilibrium state. Second, as the number of UDs increases, the scale of the solution space size increases exponentially. Thus, centralized optimization approaches often suffer from high complexity, and it is unrealistic to obtain the desired offloading solutions within acceptable time.

3 Task Offloading Game

We take advantage of game theory to solve the task offloading problem (17) for end-edge-cloud. In the section, we formulate the task offloading game model, and present the theoretical analysis for the property of our offloading game.

3.1 Task Offloading Game Formulation

Here, we reformulate the task offloading problem as a Multi-user Task Offloading Game (MUTO-Game) $P = (U, \{A_i\}_{u_i \in U}, \{E_i\}_{u_i \in U})$. U is the player collection. Each player is the UD and makes the offloading decision $a_i \in \{(0,0) \bigcup (\lambda_i, k_i)\}$. A_i and E_i are the available offloading decision set and the benefit of UD u_i, respectively. The benefit of the UD u_i is represented by its QoE according to Eq. (16).

In this paper, we consider that there is a network service provider (or an agent) for each BS. The service provider (agent) maintains the information of all the users connected to the BS. Each user acquires the information (such as channel information, interference information) from the service provider and sends back user's desirable decision to the service provider. All these players compete for the limited channels and computing resources. Next, we define the offloading solution to this game P by its Nash Equilibrium solution. The detailed definition is given in Definition 3 below.

Definition 3. *(Nash Equilibrium) Given the MUTO-Game $P = (U, \{A_i\}_{u_i \in U}, \{E_i\}_{u_i \in U})$, if no UD can change its decision to increase its QoE, the offloading strategy $\mathbf{a}^* = (a_1^*, a_2^*, \dots, a_n^*)$ can reach a Nash Equilibrium, i.e.*

$$E_{a_{-i}^*}(a_i) \le E_{a_{-i}^*}(a_i^*), \forall u_i \in U, \forall a_i \in A_i. \tag{18}$$

Then, we give Property 1 which demonstrates that the optimal offloading solution to each UD is its best response to other UDs.

Property 1. *For the Nash Equilibrium offloading solution $a^* = (a_1^*, a_2^*, \dots, a_n^*)$ of the MUTO-Game P, UD u_i's optimal offloading decision $a_i^* \in A_i$ is the best response to the offloading decisions a_{-i}^* of other $(n-1)$ UDs.*

Proof. Together with Eqs. (1), (2), (9), (13), and (16), for each UD u_i, given the offloading decisions of other UDs excluding u_i, u_i's QoE depends on a_i. Suppose $a_i^* \in A_i$ is not the best response of UD u_i, then, there must exist another better decision $a_i \in A_i$ such that a_i can increases its QoE and $E_{a_{-i}^*}(a_i) > E_{a_{-i}^*}(a_i^*)$. This contradicts with the condition that $E_{a_{-i}^*}(a_i) \le E_{a_{-i}^*}(a_i^*)$. Therefore, $a_i^* \in A_i$ must be the best response of UD u_i given other UDs' offloading decisions. □

Property 1 shows that if a Nash equilibrium offloading solution exists, then the MUTO-Game allows each individual UD to make its own offloading decision, and all users' decisions together form the overall offloading strategy. In this way, the offloading decision for each individual UD can be obtained in a distributed pattern, reducing the time complexity and improving the efficiency.

3.2 Analysis of Nash Equilibrium Offloading Solution

In this part, we propose the detailed theorem to demonstrate that an Nash Equilibrium solution exists in the MUTO-Game. We propose Lemma 1 which proves that for all the UDs, the transmission interference and the degree of computing resources competition can be upper bounded.

Lemma 1. *For each UD u_i, if it is allocated to channel k_i, then in the transmission process, u_i's transmission interference $\sum_{u_l \in U/\{u_i\}: k_l = k_i \cap \delta_l = \delta_i} p_l g_l^{k_l}$ can be upper bounded by*

$$M_i^p = (p_i g_i^{k_i}) / (2^{\frac{r_{min}}{W}} - 1) - \varpi_0. \tag{19}$$

In addition, if UD u_i decides to complete its task by the edge server in BS δ_i, then, u_i's degree of computing resource competition $\sum_{u_l \in U: \lambda_l = \lambda_i \cap \delta_l = \delta_i} \gamma_l$ can be upper bounded by

$$M_i^\gamma = \frac{\gamma_i}{f_{min}} f_{\delta_i}. \tag{20}$$

Proof. UD u_i can transmit data through channel k_i only if condition (3) is satisfied. That is to say, $r_i^{k_i} \ge r_{min}$. Together with Eq. (1) and Eq. (2), we obtain that $\sum_{u_l \in U/\{u_i\}: k_l = k_i \cap \delta_l = \delta_i} p_l g_l^{k_l} \le (p_i g_i^{k_i}) / (2^{\frac{r_{min}}{W}} - 1) - \varpi_0 = M_i^p$. Similarly, the task of u_i can only be executed on the edge server in BS δ_i only if condition (10) is satisfied, i.e., $f_{i,\delta_i} \ge f_{min}$. Together with Eq. (9), we obtain that $\frac{\gamma_i}{\sum_{u_l \in U: \lambda_l = \lambda_i \cap \delta_l = \delta_i} \gamma_l} f_{\delta_i} \ge f_{min}$. Thus, $\sum_{u_l \in U: \lambda_l = \lambda_i \cap \delta_l = \delta_i} \gamma_l \le \frac{\gamma_i}{f_{min}} f_{\delta_i} = M_i^\gamma$ holds. □

Then, we can prove that our MUTO-Game is a potential game based on the lemma 1. Definition 4 gives a detailed definition of potential games.

Definition 4. *(Potential Game) If a potential function $\phi(a)$ exists and satisfies Eq. (21), the game is a potential game.*

$$E_{a_{-i}}(a_i) \leq E_{a_{-i}}(a'_i) \Rightarrow \phi_{a_{-i}}(a_i) \leq \phi_{a_{-i}}(a'_i), \tag{21}$$

where $u_i \in U$ *and* $a_i, a'_i \in A_i$.

For a potential game, the potential function increases (or decreases) with the increase (or decrease) of the utility function. Next, we give Theorem 1 to prove our MUTO-Game is a potential game.

Theorem 1. *The MUTO-Game is a potential game, and the potential function is*

$$\begin{aligned}\phi_{-a}(a_i) = \\ &-\frac{1}{2}\sum_{u_i\in U} Q_i \sum_{u_l \neq u_i}(p_l g_l^{k_i} I_{\{k_l=k_i\}} + \gamma_l I_{\{\lambda_l=\lambda_i\}}) I_{\{\delta_l=\delta_i \cap a_i=(1,k_i)\}} \\ &-\sum_{u_i\in U} Q_i(\sum_{u_l\neq u_i} p_l g_l^{k_i} I_{\{k_l=k_i\}} + M_i) I_{\{\delta_l=\delta_i\cap a_i=(2,k_i)\}} \\ &-\sum_{u_i\in U} Q_i \eta M_i I_{\{a_i=(0,0)\}},\end{aligned} \tag{22}$$

where $\eta \geq 2$, $Q_i = p_i g_i^{k_i} + \gamma_i$, $M_i = M_i^p + M_i^{\gamma} = \frac{\gamma_i}{f_{min}} f_{\delta_i} + (p_i g_i^{k_i})/(2^{\frac{r_{min}}{W}} - 1) - \varpi_0$.

Proof. For each UD u_i, consider two different offloading decisions a_i and a'_i. We suppose that $E_{a_{-i}}(a_i) \leq E_{a_{-i}}(a'_i)$ and prove Theorem 1 from the following 6 cases. 1) $a_i = (1, k_i)$ and $a'_i = (1, k'_i)$; 2) $a_i = (2, k_i)$ and $a'_i = (2, k'_i)$; 3) $a_i = (2, k_i)$ and $a'_i = (1, k'_i)$; 4) $a_i = (2, k_i)$ and $a'_i = (1, k_i)$; 5) $a_i = (0, 0)$ and $a'_i = (1, k'_i)$; 6) $a_i = (0, 0)$ and $a'_i = (2, k'_i)$. □

Case 1. $a_i = (1, k_i)$ *and* $a'_i = (1, k'_i)$.

According to Eq. (16), we have $Cost(a_i) \geq Cost(a'_i)$. Together with Eqs. (1), (2), (4), and (12), we obtain that

$$\sum_{u_l\in U/\{u_i\}:k_l=k_i\cap\delta_l=\delta_i} p_l g_l^{k_i} \geq \sum_{u_l\in U/\{u_i\}:k_l=k'_i\cap\delta_l=\delta_i} p_l g_l^{k'_i}.$$

Thus,

$$\phi_{a_{-i}}(a_i) - \phi_{a_{-i}}(a'_i) =$$

$$Q_i \sum_{u_l\in U/\{u_i\}} (p_l g_l^{k'_i} I_{\{k_l=k'_i\}} - p_l g_l^{k_i} I_{\{k_l=k_i\}}) I_{\{\delta_l=\delta_i\}} \leq 0.$$

Case 2. $a_i = (2, k_i)$ *and* $a'_i = (2, k'_i)$.

According to Eqs. (13) and (16), $E_{a_{-i}}(a_i) \leq E_{a_{-i}}(a'_i)$ implies $r_i^{k_i} \leq r_i^{k'_i}$. Thus, the proof of Case 2 is similar as that of Case 1. We can obtain $\phi(a_i)_{a_{-i}} \leq \phi_{a_{-i}}(a'_i)$.

Case 3. $a_i = (2, k_i)$ *and* $a'_i = (1, k'_i)$.

In Lemma 1, we prove $\sum_{u_l\in U/\{u_i\}:k_l=k_i\cap\delta_l=\delta_i} p_l g_l^{k_i} + \sum_{u_l\in U/\{u_i\}:\lambda_l=\lambda_i\cap\delta_l=\delta_i} \gamma_l \leq M_i$. Therefore,

$$\phi_{a_{-i}}(a_i) - \phi_{a_{-i}}(a'_i) =$$

$$Q_i[\sum_{u_l\in U/\{u_i\}:\delta_l=\delta_i} (p_l g_l^{k'_i} I_{\{k_l=k'_i\}} + \gamma_l I_{\{\lambda_l=\lambda'_i\}}) - M_i - \sum_{u_l\in U/\{u_i\}:\delta_l=\delta_i} p_l g_l^{k_i} I_{\{k_l=k_i\}}] < 0.$$

Case 4. $a_i = (2, k_i)$ *and* $a'_i = (1, k_i)$.

Similar to the Case 3, Because

$$\sum_{u_l\in U/\{u_i\}:k_l=k_i\cap\delta_l=\delta_i} p_l g_l^{k_i} + \sum_{u_l\in U/\{u_i\}:\lambda_l=\lambda_i\cap\delta_l=\delta_i} \gamma_l \leq M_i.$$

Therefore, $\sum_{u_l\in U/\{u_i\}:\lambda_l=\lambda_i\cap\delta_l=\delta_i} \gamma_l \leq M_i$. We obtain

$$\phi_{a_{-i}}(a_i) - \phi_{a_{-i}}(a'_i) =$$

$$Q_i(\sum_{u_l\in U/\{u_i\}:\delta_l=\delta_i} \gamma_l I_{\{\lambda_l=\lambda'_i\}} - M_i) < 0.$$

Case 5. $a_i = (0, 0)$ *and* $a'_i = (1, k'_i)$.

$\eta \geq 2$ and $\sum_{u_l\in U/\{u_i\}:k_l=k_i\cap\delta_l=\delta_i} p_l g_l^{k_i} + \sum_{u_l\in U/\{u_i\}:\lambda_l=\lambda_i\cap\delta_l=\delta_i} \gamma_l \leq M_i$ imply that

$$\phi_{a_{-i}}(a_i) - \phi_{a_{-i}}(a'_i) =$$

$$Q_i[\sum_{u_l\in U/\{u_i\}:\delta_l=\delta_i} (p_l g_l^{k'_i} I_{\{k_l=k'_i\}} + \gamma_l I_{\{\lambda_l=\lambda'_i\}}) - \eta M_i] < 0.$$

Case 6. $a_i = (0, 0)$ *and* $a'_i = (2, k'_i)$.

Similar to the Case 5, we obtain

$$\phi_{a_{-i}}(a_i) - \phi_{a_{-i}}(a'_i) =$$

$$Q_i[\sum_{u_l\in U/\{u_i\}:k_l=k'_i} p_l g_l^{k'_i} - (\eta - 1)M_i] < 0.$$

Until now, we have proved that Eq. (22) holds for all the 6 cases, which thus completes the whole proof.

4 Game-Based Decentralized Task Offloading Algorithm

In the section, we propose the Game-based Decentralized Task Offloading (GDTO) Approach to solve the original problem for the end-edge-cloud system. The theoretical analysis for the convergence of our GDTO approach and its optimality analysis are also given.

4.1 Decentralized Task Offloading Approach Design

Theorem 1 proves that the MUTO-Game is a potential game. Thus, the MUTO-Game has the Finite Improvement Property (FIP) [23], and the Nash Equilibrium offloading strategy can be obtained through the finite number of iterations. Therefore, we design the GDTO algorithm, i.e., Algorithm 4.1, to find a Nash Equilibrium of the game. Each UD makes the offloading decision in an iterative pattern. Each UD searches for the optimal decisions in parallel and then competes for the update opportunities. Only the UD who wins the update opportunity can update the decision. The GDTO algorithm terminates when no UD wants to change the decision furthermore.

Algorithm 1. Game-Based Decentralized Task Offloading (GDTO) Algorithm

```
Input: $S=\{s_1,s_2,\dots,s_m\}$, $U=\{u_1,u_2,\dots,u_n\}$, and other parameters
Output: all UDs' decision
1:  Initialization:
2:  a=$\{a_1,a_2,\dots,a_n\}$, mobile device $u_i$'s decision $a_i=(\lambda_i,k_i)=(0,0)$, where $i=1\sim n$
3:  End Initialization
4:  repeat
5:    for each UD $u_i \in U$ do
6:        Calculate the current $r_i^{k_i}$ and $f_{i,\delta_i}$
7:        if $f_{i,\delta_i} < f_{min}$ then
8:          update its decision with $a_i=(2,k_i)$
9:        if $r_i^{k_i} < r_{min}$ then
10:         update its decision with $a_i=(0,0)$
11:       Calculate the current QoE
12:   Calculate the current total QoE
13:   for each UD $u_i \in U$ do
14:     for each channel $k_i \in \{1,\dots,c_{\delta_i}\}$ do
15:        Calculate $\sum_{u_i\in U} E_{a_{-i}}(a_i')$ when $u_i$'s task is offloaded
           to $k_i'$ channel and completed in edge server or cloud.
16:     Find a decision that can achieve the highest
        $\sum_{u_i\in U} E_{a_{-i}}(a_i')$ from all possible decisions
17:     if $\sum_{u_i\in U} E_{a_{-i}}(a_i') > \sum_{u_i\in U} E_{a_{-i}}(a_i)$ then
18:        send $a_i'$ to contend for decision update opportunity
19:        if $u_i$ gets a chance to get an update then
20:          chance $u_i$'s decision to $a_i'$
21: until there is no need to update decisions for any user
22: return a
```

At the beginning, for the initial settings, no UD offloads the task and each UD $u_i, \forall u_i \in U$ starts with the offloading decision $a_i=(0,0)$ (Lines 1-3). Next, the GDTO approach allows each UD to update the offloading decision by iteration. Based on the FIP property, the GDTO approach is guaranteed to converge and reach the Nash equilibrium, where no single UD will change its decision any further.

In each iteration, the transmission rate $r_i^{k_i}$ and the allocated computing resources f_{i,δ_i} for each UD u_i are calculated. Then, the constraints of Eq. (3) and Eq. (10) are checked and validated (Lines 5-10). After that, the each UD u_i's current QoE and total QoE of all UDs is calculated. Then, each UD u_i searches for the best offloading decision a_i' (Lines 13-16). The UD u_i's decision to update is a_i' and the previous decision is a_i. Then, the UD compares the new decision with the previous decision. If $\sum_{u_i\in U} E_{a_{-i}}(a_i') > \sum_{u_i\in U} E_{a_{-i}}(a_i)$, a_i' will be sent to the new decision set of all UDs to compete for update opportunities (Lines 17-18). In some studies [13], [24], the competitive process is used to determine the winner in an indeterminate manner (such as by random method). In this paper, we consider the UD with the largest change in QoE utility as the winner. After that, if UD u_i wins the update opportunity, a_i will be updated to a_i'. Decisions of the UDs that do not win will not be updated in the iteration (Lines 19-20). After the winner has updated the decision, the QoE of all UDs that are affected by the winner will be updated accordingly. Finally, when no UD changes the offloading decision, the GDTO algorithm ends and the Nash equilibrium offloading strategy is reached.

The decisions of all the UDs constitute the final offloading strategy a, which is the solution of the GDTO problem (Lines 21-22). Since each individual UD makes its own offloading decision, our GDTO algorithm is a decentralized algorithm.

4.2 Convergence Analysis

After a finite number of iterations, the MUTO-Game will finally reach a Nash Equilibrium offloading strategy because of the FIP property. Next, we prove the upper bound of the converge time measured by the number of iterations, as in Theorem 2.

Theorem 2. *When Q_i, Q_i^p and M_i are non-negative integers for $u_i \in U$, there is an upper limit on the number of iterations, which satisfies*

$$R \le \frac{1}{2} n^2 Q_{max}^2/Q_{min} + nQ_{max}(nQ_{max}^p + M_{max})/Q_{min} + n\eta Q_{max}M_{max}/Q_{min},$$

where $Q_i^p \triangleq p_i g_i^{k_i}$, $Q_{min}^p \triangleq min(Q_i^p)$, $Q_{max}^p \triangleq max(Q_i^p)$, $Q_{max} \triangleq max(Q_i)$, $Q_{min} \triangleq min(Q_i)$ and $M_{max} \triangleq max(M_i)$.

Proof. According to Eq. (22), it holds

$$\begin{aligned} 0 &\ge \phi_{a_{-i}}(a_i) \\ &\ge -\frac{1}{2}\sum_{u_i\in U}\sum_{u_l\in U} Q_{max}Q_{max}I_{\{a_i=(1,k)\}} \\ &\quad - \sum_{u_i\in U} Q_{max}(\sum_{u_l\in U} Q_{max}^p + M_{max})I_{\{a_i=(2,k)\}} \\ &\quad - \sum_{u_i\in U} \eta Q_{max}M_{max}I_{\{a_i=(0,0)\}} \end{aligned}$$

Thus, we have

$$0 \ge \phi_{a_{-i}}(a_i) \ge -\frac{1}{2}n^2Q_{max}^2 - nQ_{max}(nQ_{max}^p + M_{max}) - n\eta Q_{max}M_{max}. \tag{23}$$

If u_i updates its decision from a_i to a_i', u_i's QoE increases, $E(a_i) < E(a_i')$. According to Definition 4, the potential function $\phi_{-a}(a_i)$ meets

$$\phi_{a_{-i}}(a_i') - \phi_{a_{-i}}(a_i) \ge 0 \tag{24}$$

Then, we prove Theorem. 2 by analyzing the following 6 cases. □

Case 1. $a_i=(1,k_i)$ *and* $a_i'=(1,k_i')$.

According to Eq. (22), There is

$$\phi_{a_{-i}}(a_i') - \phi_{a_{-i}}(a_i) = Q_i \sum_{u_l \in U/\{u_i\}} (Q_l^p I_{\{k_l=k_i\}} - Q_l^p I_{\{k_l=k_i'\}}) I_{\{\delta_l=\delta_i\}} > 0 \tag{25}$$

Because Q_i^p is a non-negative integer for $u_i \in U$, there holds

$$\sum_{u_l \in U/\{u_i\}} (Q_l^p I_{\{k_l=k_i\}} - Q_l^p I_{\{k_l=k_i'\}}) I_{\{\delta_l=\delta_i\}} \geq 1.$$

Thus, according to Eq. (25), $\phi(a_i') \geq \phi(a_i) + Q_i \geq \phi(a_i) + Q_{min}$.

Case 2. $a_i = (2, k_i)$ *and* $a_i' = (2, k_i')$.

Similar to Case 1, Inequality (25) can be obtained. Therefore, there exists $\phi(a_i') \geq \phi(a_i) + Q_i \geq \phi(a_i) + Q_{min}$.

Case 3. $a_i = (2, k_i)$ *and* $a_i' = (1, k_i')$.

There exists

$$\phi_{a_{-i}}(a_i') - \phi_{a_{-i}}(a_i) = Q_i[M_i + \sum_{u_l \in U/\{u_i\}} (Q_l^p I_{\{k_l=k_i\}} - Q_l^p I_{\{k_l=k_i'\}} - \gamma_l I_{\{\lambda_l=\lambda_i'\}}) I_{\{\delta_l=\delta_i\}}] > 0.$$

Since $M_i > 0$, $Q_i^p > 0$ and $\gamma_i > 0$ are integers for any $u_i \in U$, there holds $M_i + \sum_{u_l \in U/\{u_i\}} (Q_l^p I_{\{k_l=k_i\}} - Q_l^p I_{\{k_l=k_i'\}} - \gamma_l I_{\{\lambda_l=\lambda_i'\}}) I_{\{\delta_l=\delta_i\}} \geq 1$. Thus, $\phi(a_i') \geq \phi(a_i) + Q_i \geq \phi(a_i) + Q_{min}$.

Case 4. $a_i = (2, k_i)$ *and* $a_i' = (1, k_i)$.

There exists

$$\phi_{a_{-i}}(a_i') - \phi_{a_{-i}}(a_i) = Q_i(M_i - \sum_{u_l \in U/\{u_i\}:\delta_l=\delta_i} \gamma_l I_{\{\lambda_l=\lambda_i'\}}) > 0.$$

Because Q_i and Q_i^p are non-negative integers, we can obtain λ_i is non-negative integers. Therefore, $M_i - \sum_{u_l \in U/\{u_i\}:\delta_l=\delta_i} \gamma_l I_{\{\lambda_l=\lambda_i'\}} \geq 1$. So $\phi(a_i') \geq \phi(a_i) + Q_i \geq \phi(a_i) + Q_{min}$.

Case 5. $a_i = (0, 0)$ *and* $a_i' = (1, k_i')$.

There exists

$$\phi_{a_{-i}}(a_i') - \phi_{a_{-i}}(a_i) = Q_i[\eta M_i - \sum_{u_l \in U/\{u_i\}} (Q_l^p I_{\{k_l=k_i'\}} + \gamma_l I_{\{\lambda_l=\lambda_i'\}}) I_{\delta_l=\delta_i}] > 0$$

Because M_i and Q_i^p are non-negative integers, there is

$$\eta M_i - \sum_{u_l \in U/\{u_i\}:\delta_l=\delta_i} (p_l g_l^{k_i'} I_{\{k_l=k_i'\}} + \gamma_l I_{\{\lambda_l=\lambda_i'\}}) \geq 1.$$

So $\phi(a_i') \geq \phi(a_i) + Q_i \geq \phi(a_i) + Q_{min}$.

Case 6. $a_i = (0, 0)$ *and* $a_i' = (2, k_i')$.

$$\phi_{a_{-i}}(a_i') - \phi_{a_{-i}}(a_i) = Q_i[(\eta - 1)M_i - \sum_{u_l \in U/\{u_i\}} Q_l^p I_{\{k_l=k_i' \cap \delta_l=\delta_i\}}] > 0. \tag{26}$$

We can obtain $(\eta - 1)M_i - \sum_{u_l \in U/\{u_i\}} Q_l^p I_{\{k_l=k_i' \cap \delta_l=\delta_i\}} \geq 1$. Thus, based on inequalities (26), $\phi(a_i') \geq \phi(a_i) + Q_i \geq \phi(a_i) + Q_{min}$.

Together with Case 1 to Case 6, we can get $\phi(a_i') \geq \phi(a_i) + Q_{min}$. Thus, $\phi(a_i') - \phi(a_i) \geq Q_{min}$. In other words, the minimum increased value of the potential function before and after each iteration is Q_{min}. Therefore, according to inequalities (23), the following inequality always holds:

$$0 - \phi_{a_{-i}}(a_i) \leq \frac{1}{2} n^2 Q_{max}^2 + n Q_{max}(n Q_{max}^p + M_{max}) + n\eta Q_{max} M_{max}.$$

We can obtain

$$R \leq \frac{1}{2} n^2 Q_{max}^2 / Q_{min} + n Q_{max}(n Q_{max}^p + M_{max}) / Q_{min} + n\eta Q_{max} M_{max} / Q_{min}.$$

Therefore, we complete the proof of Theorem 2.

4.3 Complexity Analysis

As shown in Algorithm 4.1, in each iteration, each user conducts the operations of calculating its own QoE (Lines 5-11) and searching for its best strategy (Lines 13-16) in a distributed way. For the part of calculating its own QoE, there are only some basic mathematical operations, and the time complexity for this part can be regarded as $\mathcal{O}(1)$. For the part of searching for its best strategy, the time complexity is $\mathcal{O}(c_{max})$, where $c_{max} = \max\{c_1, \ldots, c_m\}$, i.e., the maximum number of channels provided by the BSs. Therefore, the time complexity for each iteration is $\mathcal{O}(1) + \mathcal{O}(c_{max}) = \mathcal{O}(c_{max})$. Furthermore, Theorem 2 in the paper proves the upper bound of number of interactions, i.e., $R \leq \frac{1}{2} n^2 Q_{max}^2 / Q_{min} + n Q_{max}(n Q_{max}^p + M_{max}) / Q_{min}$. Therefore, the time complexity of GDTO algorithm is $\mathcal{O}(c_{max} \times (\frac{1}{2} n^2 Q_{max}^2 / Q_{min} + n Q_{max}(n Q_{max}^p + M_{max}) / Q_{min}))$.

4.4 Analysis of Price of Anarchy

In general, our proposed multi-user offloading game may admit more than one Nash Equilibrium. Moreover, these Nash Equilibrium solutions might be different from the globally optimal solution to Problem (17) before. To quantify the gap between the Nash Equilibrium solution and the globally optimal solution to Problem (17), we adopt the metric of Price of Anarchy (PoA) [25]. In particular, PoA measures the ratio between the worst utility of offloading strategies that achieve Nash Equilibrium and the utility of the exactly optimal offloading strategy. We denote the set of all Nash equilibrium strategies by $\mathbb{N}$ in our MUTO-Game. $\overline{a} = (\overline{a}_1, \ldots, \overline{a}_n)$ denote the centralized optimal offloading strategy. The PoA of the game in the overall QoE is

$$POA_{QoE} = \frac{min_{a^* \in \mathbb{N}} \sum_{u_i \in U} E_{a_{-i}^*}(a_i^*)}{\sum_{u_i \in U} E_{\overline{a}_{-i}}(\overline{a}_i)}, \tag{27}$$

and it can be analyzed by the following Theorem 3.

Theorem 3. *For the MUTO-Game, PoA calculated with Eq. (27) satisfies:*

$$\frac{E_{min}}{-\alpha\log_2(\min\{\frac{B_{min}}{r_{max}}+\frac{X_{min}}{f_{max}},\frac{p_{min}B_{min}}{r_{max}}\})+\beta} \leq POA_{QOE} \leq 1. \tag{28}$$

Proof.

There is $QOE(a^*) \leq QOE(\overline{a})$ for any offloading strategy $a^* \in \mathbb{N}$, i.e., $\frac{\sum_{u_i\in U} E_{a^*_{-i}}(a^*_i)}{\sum_{u_i\in U} E_{\overline{a}_{-i}}(\overline{a}_i)} \leq 1$ Therefore, $POA_{QOE} \leq 1$.

For any user $u_i \in U$, there are the minimum QoE E_{min} and the maximum QoE E_{max}. Thus, for any offloading strategy $a^* \in \mathbb{N}$, the total QoE satisfies

$$\sum_{u_i\in U} E_{a^*_{-i}}(a^*_i) \geq nE_{min}$$

and the total QoE produced by the optimal offloading strategy $\overline{a}$ satisfies

$$\sum_{u_i\in U} E_{\overline{a}_{-i}}(\overline{a}_i) \leq nE_{max}$$

Therefore, combining the above inference with Eq. (27), we can get

$$POA_{QOE} \geq \frac{nE_{min}}{nE_{max}} = \frac{E_{min}}{E_{max}}$$

According to (13), (14), (15) and (16), there is

$$\frac{E_{min}}{E_{max}} = \frac{E_{min}}{-\alpha\log_2(\min\{\frac{B_{min}}{r_{max}}+\frac{X_{min}}{f_{max}},\frac{p_{min}B_{min}}{r_{max}}\})+\beta}.$$

Finally, we can prove (28) of Theorem 3. □

5 Performance Evaluation

In the section, we conduct parameter analysis and comparative experiments to validate the performance and the effectiveness of our GDTO alogrithm.

5.1 Parameter Configuration

We consider that there are 10 service areas in the experiments. Each service area has a BS with an edge server ($|S| = 10$). The UDs in each service area are randomly distributed. The parameter settings are shown in Table 2. For each BS, the background noise ϖ_0 = -100 dBm and wireless channel bandwidth $W = 5$ MHz [13], [24]. For each UD $u_i \in U$, the task's data size is randomly set from 3 MB to 5 MB, the transmission power $p_i =$ 1000 mWatts and the computing capability is 0.5 GHz. The transmission rate from BS to cloud is $\hat{r} = 1$ Mbps. The CPU cycles number to complete the task is $X_i = B_i v$, where $v =$ 1000 cycles/bit according to [26], [27]. Similar to [28], the large-scale path loss is 128.1+37.6$\log_{10} l_i$ dB, where l_i is the distance between BS δ_i and UD u_i, randomly set from 500 m to 1000 m.

TABLE 2
Experimental Settings (1)

Parameter	Value
Distance from user to BS (l_i)	500 m~1000 m
bandwidth of Channel (W)	5 MHz
Data size of task	3 MB~5 MB
Transmission rate of base station ($\hat{r}$)	1 Mbps
Background noise	-100 dBm
Processing density of task (v)	1000 cycles/bit
Large-scale path loss	$128.1 + 37.6\log_{10} l_i$ dB
Computing capability of user	0.5 GHz
Transmission power (p_i)	1000 mWatts

5.2 Experimental Results

We evaluate the convergence time of the GDTO approach with different numbers of users and channels. We also conduct two groups of comparison experiments to validate the GDTO's performance.

5.2.1 Evaluation of Convergence Time

As the execution time of algorithms are greatly affected by the hardware equipment, similar to existing works, we use the number of iterations to represent the convergence time.

Fig. 2 shows the number of iterations of the GDTO for different number of UDs. The number of users is increased from 100 to 1000. We can see from Fig. 2 that the number of iterations increases continuously when the number of UDs is from 100 to 600. When the number of UDs is from 700 to 1000, the number of iterations does not change too much. The phenomena is caused by the competition mechanism of the GDTO algorithm. When the number of users is from 100 to 600, there are sufficient resources to serve the users. Therefore, there are more possible decisions to select for the users, and the iteration number increases. However, when the number of users is from 700 to 1000, a large number of users cannot be offloaded due to insufficient transmission resources.

Fig. 3 shows the number of iterations with different number of channels. The number of UDs is 1000 and the number of channels varies from 5 to 40. We can see from Fig. 3 that the iteration number grows with the increase of channels numbers. This is because more available channels for selection brings more candidate strategies for the users. Nevertheless, when the size of solution space increases exponentially with the number of channels, the increase speed of the iteration number of our GDTO algorithm is lower than linear speed.

5.2.2 Evaluation of Energy Consumption

Table 3 lists the numerical results showing the average energy consumption that users experience from the task offloading and local task processing operations. The number of UDs is set as 1000, and the number of channels provided by each BS varies from 5 to 40 with an increment of 5. We can see from Table 3 that with the

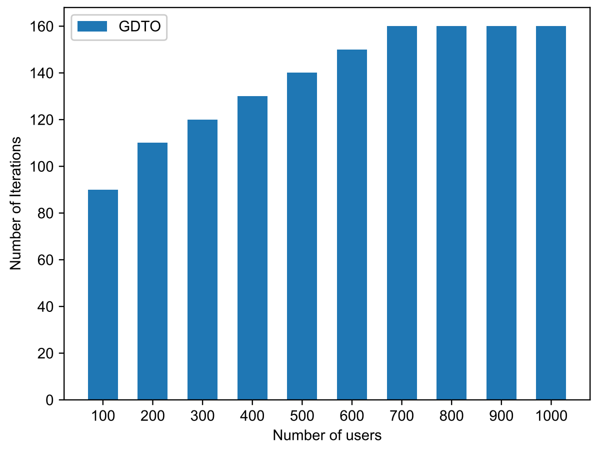

Fig. 2. Number of iterations versus number of users.

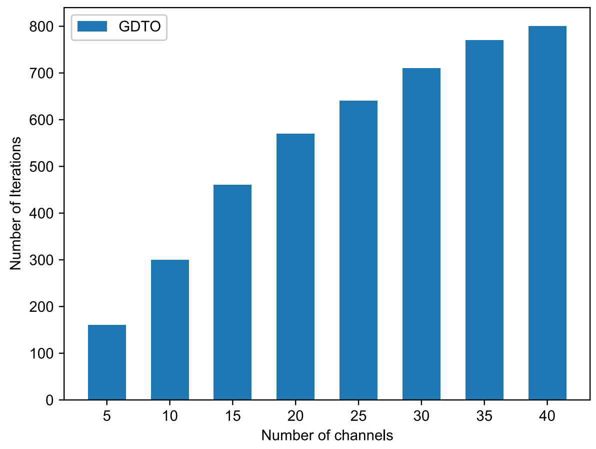

Fig. 3. Number of iterations versus number of channels.

TABLE 3
Energy Consumption

Number of channels for each BS	Average energy consumption of local processing (J)	Average energy consumption of task offloading (J)	Average sum energy consumption (J)
5	30.1	1.7	31.8
10	27.0	2.3	29.3
15	22.9	3.5	26.4
20	19.5	4.5	24.0
25	16.2	5.4	21.7
30	13.6	5.9	19.5
35	10.4	7.0	17.4
40	8.1	7.3	15.4

increase of number of channels, the average energy consumption of local task processing decreases and the average energy consumption for task offloading increases. The reason is that when there are more available channels, there are more transmission resources provided for UDs to offload tasks. Then, the number of tasks offloaded will increase, and the number of tasks processed locally will decrease. Thus, the average energy consumption for local computing decreases and the average energy consumption of task offloading increases. Furthermore, we can also observe from Table 3 that the average energy consumption sum decreases when the number of channels increases. We can draw the conclusion that providing more channel resources will benefit more users in reducing their energy consumption.

5.2.3 Comparison Experiments

To further evaluate our GDTO algorithm, we compare it with 5 other methods shown as below.

- *Optimal*: The optimal method models the problem as an integer programming problem and uses the centralized approach to get a global optimal offloading solution.
- *Random*: The method randomly assigns decisions to each user to complete the task. When there are enough bandwidth resources, the user randomly selects a method (local computing or transmitting the tasks to the BS). When the edge servers' computation resources are sufficient, the user randomly selects local computing, edge computing or cloud computing. Otherwise, the user randomly selects local computing or cloud computing.
- *ICSOC 19*: This heuristic method is extended from [29] and adjusted accordingly to solve our offloading problem. Specifically, each user selfishly and greedily acquires the maximum resources to maximize its QoE while meeting constraints.
- *DGEA 20*: This distributed game-based algorithm DGEA 20 is extended from [30] to be applied to our model. When the users compete for the update opportunities, the winner is selected randomly.
- *CCPM 19*: This scheduling algorithm CCPM 19 is extended from [31] and adjusted accordingly for our model. The users are sorted according to their channel conditions and qualities. And then, based on the sorted user order, the users updates the decision to achieve its own maximum QoE one by one.

When the solution space size is large, the optimal method suffers from high complexity and cannot obtain the offloading strategy within a reasonable time. Therefore, we set two groups of experiments, including the small-scale experiments and the large-scale experiments (Set # 1 and Set # 2), as shown in Table 4. For Set # 1, we evaluate all the four methods, while for Set # 2, the optimal method is omitted.

a. Small-scale Experiment

Fig. 4 shows the average QoE of the 6 algorithms with different number of UDs. We can find that for all the 6 algorithms, the average QoE decreases with the increase of number of UDs. There are two reasons. First, the computing capability of the edge servers are fixed. Adding UDs will gradually use up the resources of the edge servers, thus the number of UDs who cannot offload task will increase. Second, the channel resources are fixed. Adding UDs will result in competition for the channel resources. Therefore, the average QoE will decrease. In addition, when the number of user is smaller than 60, the average QoE of our GDTO algorithm is the same as that of the Optimal algorithm. When the

TABLE 4
Experimental Settings (2)

		n	c_{δ_i}	f_{δ_i}
Set # 1	Set # 1.1	$10 \sim 100$	1	5
	Set # 1.2	50	$1 \sim 6$	5
	Set # 1.3	50	1	$5 \sim 14$
	Set # 2.1	$100 \sim 1000$	5	10
Set # 2	Set # 2.2	1000	$5 \sim 40$	10
	Set # 2.3	1000	5	$10 \sim 100$

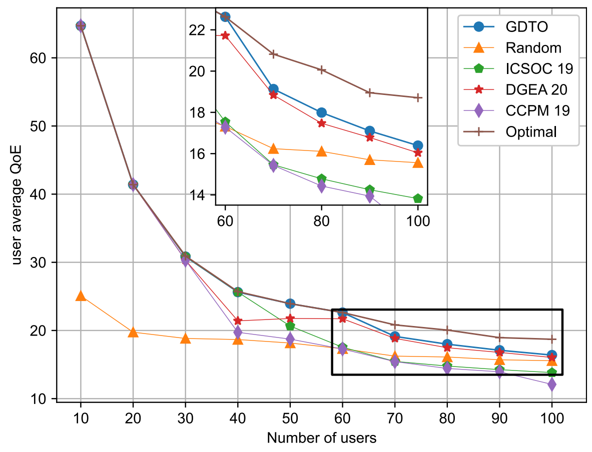

Fig. 4. Average QoE versus number of users (Set #1.1).

number of user exceeds 60, the average QoE of our GDTO algorithm is slightly smaller than the Optimal and still larger than the other 4 algorithms.

Fig. 5 shows the total QoE with the 6 different algorithms when the number of channels is from 1 to 6. We can see that when the number of channels varies from 1 to 5, the total QoEs of the 6 algorithms all increase when the number of channels increases. The reason is that when the number of channels increases, there are more transmission resources and less competition among users, increasing each user's QoE. However, as the number of wireless channels increases from 5 to 6, the QoEs of the algorithms do not improve much. This is because when the number of wireless channels is further increased, the computing resources of the edge servers become the bottleneck and are not sufficient to serve the users. We also find that when number of channels is smaller than 4, the GDTO's QoE is second to that of the Optimal algorithm and better than the other 4 algorithms. When the number of channels is larger than 4, the QoE of our GDTO algorithm and the Optimal algorithm is the same.

Fig. 6 shows the total QoEs of the 6 algorithms when the edge server's computing capability varies from 5 GHz to 14 GHz. The overall QoE of the GDTO, ICSOC 19, DGEA 20, CCPM 19 and the optimal algorithms all increase when the computing capability increases from 5 GHz to 14 GHz. As more computing resources are available, the edge servers can accommodate more tasks. Thus, the overall QoE will increase. The overall QoE of our GDTO algorithm is slightly smaller than that of the optimal algorithm, and larger than the other 4 algorithms.

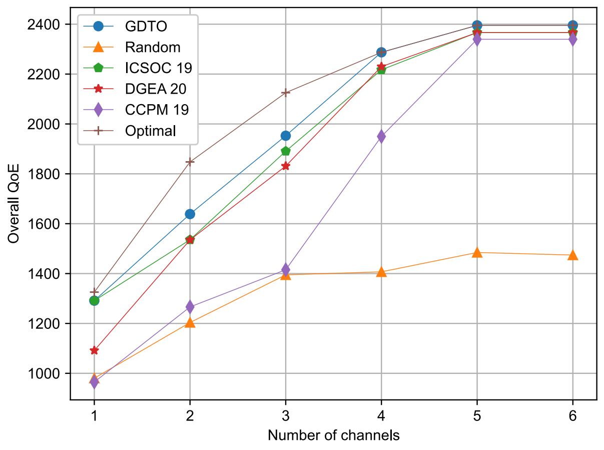

Fig. 5. Total QoE versus number of channels (Set #1.2).

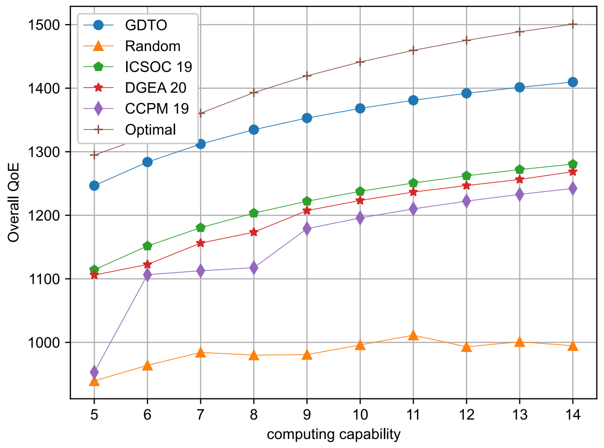

Fig. 6. Total QoE versus computing capability (Set #1.3).

b. Large-scale Experiment

Figs. 7 and 8 show the average QoE and percentage of task offloading of the 5 algorithms with different number of UDs. The number of UDs is increased from 100 to 1000. We can see from Fig. 7 that the QoEs of the 5 algorithms all decrease when number of UDs increases. This is because the transmission resources are limited, and the resources allocated to each user will decrease when there are more users competing for the resources, thus decreasing the average QoE. Nevertheless, the QoE of our GDTO algorithm is still the largest among the 5 algorithms. In Fig. 8, with the increase of number of UDs, the percentage of offloaded tasks with GDTO, ICSOC 19, DGEA 20 and CCPM 19 algorithms all decrease. That is because the number of UDs is increasing, the resources gradually become insufficient. Thus, the percentage of UDs who can offload tasks will decrease. However, the percentage of task offloading with our GDTO algorithm is still the largest among all the 5 algorithms.

Figs. 9 and 10 show the total QoE and percentage of offloading task under the 5 algorithms. The number of wireless channels is from 5 to 40. We can see that for the 5 algorithms, the QoE all increases with the increase of number of channels. This is because when the number of UDs is fixed, the increase in number of channels can bring more transmission resources for users. Therefore,

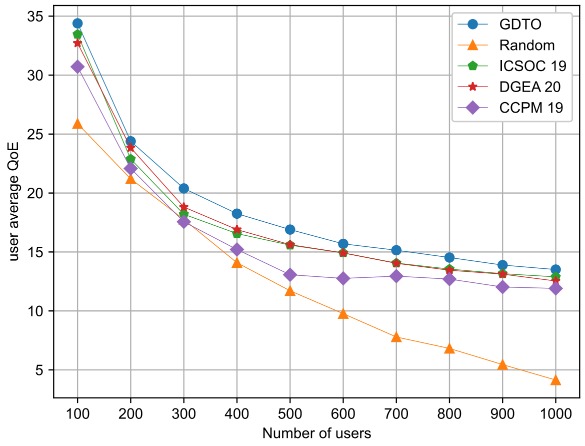

Fig. 7. Average QoE versus number of users (Set #2.1).

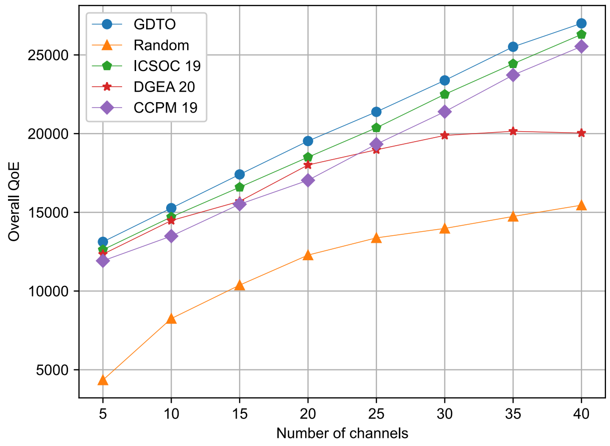

Fig. 9. Total QoE versus number of channels (Set #2.2).

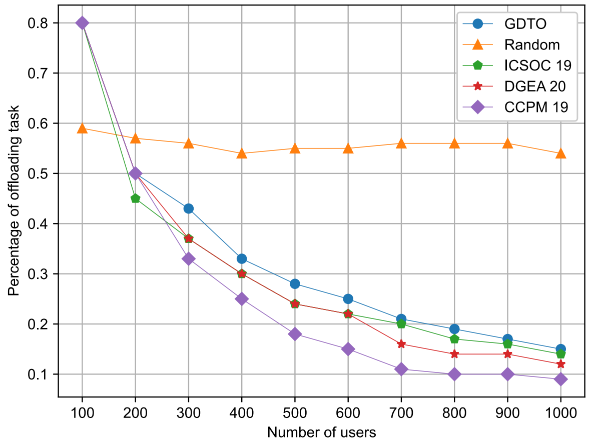

Fig. 8. Percentage of offloading task versus number of users (Set #2.1).

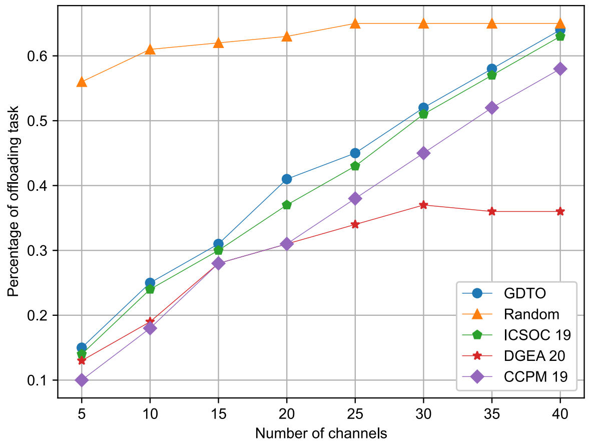

Fig. 10. Percentage of offloading task versus number of channels (Set #2.2).

users can increase the transmission rate and thus improve the overall QoE. In addition, we can find that the overall QoE of GDTO algorithm is always larger than that of the other 4 algorithms. We also find from Fig. 10 that the percentages of task offloading for the GDTO, ICSOC 19, DGEA 20 and CCPM 19 algorithms all rise when the number of channels increases. Actually, more wireless channel resources enable more users to transmit tasks, thus improving the percentage of task offloading. Similarly, we can see that the percentage of task offloading of our GDTO algorithm is larger than the ICSOC 19, DGEA 20 and CCPM 19 algorithms. The percentage of offloading task with the Random algorithm does not change much due to its random offloading mechanism.

We adopt the metric of number of Offloading Beneficial User (OBU) to further evaluate our proposed GDTO algorithm in benefiting the users from offloading tasks. One user is called an OBU user, if the user's obtained QoE with integrated offloading choice is better than its QoE with solely local computing. Then, we say that this user benefits from offloading, and call this user an OBU user.

Fig. 11 shows the number of OBU users with the 5 algorithms under different number of channels. The number of channels is from 5 to 40. We can observe that the numbers of OBU users for the 5 algorithms all become larger with the increase of number of channels. This is because when there are more channels, there are more transmission resources provided for the users to offload their tasks. Then, more users can benefit from offloading tasks and the number of OBU users will increase. We also observe that the number of OBU users with our GDTO algorithm is always the largest among the 5 algorithms, which demonstrates the superiority of our GDTO algorithm in benefiting the users. Besides, the number of the OBU users with the Random algorithm is always the smallest among the 5 algorithm. This is because the Random algorithm does not make effective use of the channel resources to make the offloading decisions.

Fig. 12 shows the overall QoE when edge server's computing capability varies from 10 GHz to 100 GHz. We observe that the overall QoE of the 5 algorithms increases as the computing capability of the edge server increases, and the QoE of our GDTO algorithm is the largest. We also observe that when the computing capacity is further increased (about 70 GHz in Fig. 12), the increase rate of QoE gradually becomes slower. The increase trend of the curves of all the algorithms' QoE is gradually flattened. The reason is that when the computing capability increases to a certain level, the computing resources allocated to the UDs on the edge servers are already sufficient. At this time, the main

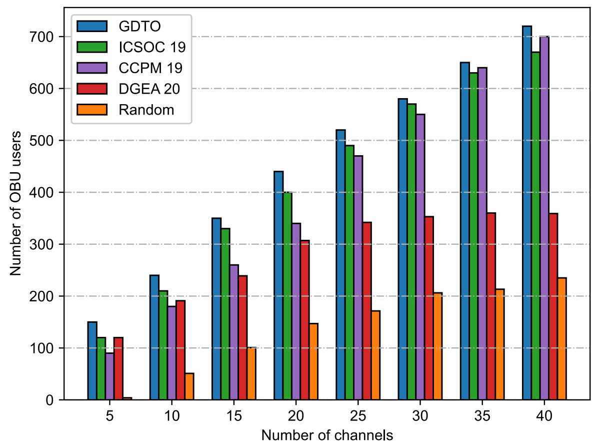

Fig. 11. Number of OBU users versus number of channels (Set #2.2).

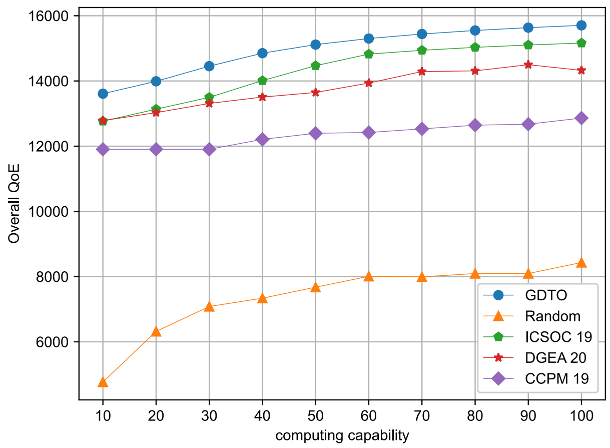

Fig. 12. Total QoE versus computing capability (Set #2.3).

factor affecting the increase of QoE is insufficient channel resource.

6 Related Work

The problem of caching optimization and task offloading in edge computing is an important research topic, and widely studied in academia. Xu et al. [32] studied the service caching problem in MEC networks. An integer linear programming solution and a distributed game-theoretic approach were proposed to minimize the social cost for all network service providers. In [33], Lyu et al. proposed a distributed online learning approach to optimize content placement and delivery at each edge server to achieve optimal cache hit rate and cost effectiveness. Wang et al. [34] regarded the content caching strategy, tasks offloading decision and resource allocation as an optimization problem. it was transformed into a convex problem, and was solved by a multiplier-based alternating directions approach. In [35], Fan et al. designed an iterative algorithm based on Lyapunov optimization to minimize the total task processing latency. Duan et al. [36] transformed and decomposed the task offloading optimization problem into two subproblems and designed an online control scheme based on Long Short Term Memory and Dueling Double DQN to reduce the computation cost of mobile devices and achieve load balancing.

In general, co-channel interference exists in the wireless channel transmission in edge computing scenarios. Assigning too many users to edge servers can lead to serious interference that can affect the user's data rate. In [13], [24], Cui et al. studied the mobile edge device allocation problem of interference perception in edge computing and edge cloud computing respectively. A distributed solution approach was proposed. In [37], Chen et al. considered the computing task offloading of multi-user in a two-layered end-edge system and the tasks were either executed on the edge servers or locally. It was considered that the resources of edge servers were sufficient and could satisfy all the users. Different from their work, our work considers a three-layered end-edge-cloud system, and the tasks can also be executed on the cloud. Furthermore, we consider more realistic scenarios that the edge servers' computing resources are limited, and users compete for the computing resources.

In addition, some studies have taken user's quality of experience (QoE) as an optimization goal. In Lai et al. [21] solved the user assignment problem in edge computing scenarios. A distributed algorithm was proposed to optimize the user's QoE based on game theory. Zheng et al. [22] comprehensively considered the use of spectrum efficiency, user fairness and service satisfaction to evaluate user's quality of experience, described the resource allocation problem as a local cooperative game.

The computing capability of edge servers is limited compared to that of cloud servers.Therefore, it can be considered to combine edge computing and cloud computing to improve the entire system's performance. Some works have studied the computing offloading problem in the edge-cloud systems. In [11], Fantacci et al. used queuing theory to evaluate and optimize three-layer edge-cloud computing, and proposed a allocation algorithm of computing resource to maximize the optimization goal under the constraint of satisfying a specific quality of service(QoS). In [12], Wang et al. considered the heterogeneous system combining mobile edge computing and cloud center, and presented an approach to optimize time latency of the whole system by coordinating task allocation, computing and transmission resources together.

In the edge cloud scenario, users sometimes need to compute their own tasks locally (end-edge-cloud computing). For example, when the connection is unstable or the system resources are insufficient. There were also some works to consider task offloading for end-edge-cloud computing. Sun et al. [38] used a low-complexity hierarchical heuristic method and an inequality method to determine resource allocation under the joint constraints of cache resources and edge server communication and computing resources. Du et al. [39] proposed a low-complexity suboptimal algorithm based on semidefinite relaxation, fractional programming theory and Lagrange duality to solve offloading decisions and resource allocation problems. Ding et al. [18] studied the offloading optimization under two different types of computing architectures, and proposed game algorithms under two architectures based on game theory.

Related works on caching and task offloading in MEC can also be classified by the criteria of distributed approaches and centralized approaches. Some existing works proposed the task offloading approaches which could be implemented in a distributed way. [40] studied the risk-aware data offloading problem in MEC. Each user aimed at maximizing the perceived satisfaction. A non-cooperative game was formulated to study this problem and a distributed algorithm converging to the Nash Equilibrium was proposed. [41] investigated the energy efficient task offloading in a collaborative MEC and autonomous aerial system. The authors adopted the Satisfaction Game and proposed a reinforcement learning-enabled approach to increase Internet of Thing user's satisfaction with the energy cost considered. [42] designed incentives for wireless body area network (WBAN) users to curtain the task offloading amount to achieve the green task offloading for 5G-enabled healthcare systems. The Stackelberge game was adpoted and the optimal solution was obtained by the alternating direction method of multipliers (ADMM)-based approach implemented in a distributed manner. [43] proposed the resource management scheme to minimize the MEC server's energy consumption without compromising WBAN users' QoE. A cooperative framework was proposed, and the Nash bargaining theory was adopted to model the interaction. The closed-form Nash bargaining solutions were also derived. In [44], Feng et al. studied the caching optimization problem in edge networks. The cache control was described as a stochastic difference game, and a distributed cache iterative control algorithm was proposed to finally obtain an optimal edge cache control policy.

Some other works adopted centralized approaches to solve the task offloading problems. [10] considered the scenario of edge computing and cloud computing collaboration with the objective of minimizing the weighted-sum latency of all mobile devices. A resource allocation strategy based on convex optimization was proposed. [12] focused on the task scheduling, resources allocation among devices, multi-layer MEC servers and remote clouds. [45] modeled the computation offloading problem by a Markov decision process, and proposed a deep Q-network-based computation offloading algorithm. [15] jointly considered the computing offloading and interference management in a MEC-enabled network, and formulated the computing offloading, physical resource and MEC computing resource allocation as an optimization problem.

7 Conclusion

In the paper, we investigate the user task offloading problem in a collaborative end-edge-cloud system. We formulate the offloading problem with the goal of maximizing users' QoE while satisfying the communication and computing resource constraints. Then, we reformulate the problem as the MUTO-Game model, and define the optimal offloading strategy by the Nash Equilibrium strategy. We propose the lemma that both the communication interference and the degree of computing resource competition can be upper bounded. Then, we prove that the MUTO-Game is a potential game and there exists at least one Nash Equilibrium offloading strategy. We propose the decentralized GDTO approach which obtains the Nash Equilibrium offloading strategy. Theoretical analysis for the upper bound of convergence time and performance guarantee in the worst case is given. Finally, we conduct extensive experiments, which validate the performance of our GDTO approach.

One future direction is considering the service provider as another type of players in the game, and take advantage of Stackelberg Game to formulate the offloading problem with the goal of social welfare maximization. Another important and interesting future research direction is to adopt the non-orthognal multiple access (NOMA) technique that implements the successive interference cancellation technique to model the interference.

References

[1] Z. Chen et al., "An empirical study of latency in an emerging class of edge computing applications for wearable cognitive assistance," in *Proc. 2nd ACM/IEEE Symp. Edge Comput.*, 2017, pp. 1–14.

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

[3] C. Wang, S. Zhang, Y. Chen, Z. Qian, J. Wu, and M. Xiao, "Joint configuration adaptation and bandwidth allocation for edge-based real-time video analytics," in *Proc. IEEE Conf. Comput. Commun.*, 2020, pp. 257–266.

[4] S. Wang et al., "A cloud-guided feature extraction approach for image retrieval in mobile edge computing," *IEEE Trans. Mobile Comput.*, vol. 20, no. 2, pp. 292–305, Feb. 2021.

[5] J. Zheng, Y. Cai, Y. Wu, and X. Shen, "Dynamic computation offloading for mobile cloud computing: A stochastic game-theoretic approach," *IEEE Trans. Mobile Comput.*, vol. 18, no. 4, pp. 771–786, Apr. 2019.

[6] F. Vhora and J. Gandhi, "A comprehensive survey on mobile edge computing: Challenges, tools, applications," in *Proc. IEEE 4th Int. Conf. Comput. Methodol. Commun.*, 2020, pp. 49–55.

[7] K. Peng, V. Leung, X. Xu, L. Zheng, J. Wang, and Q. Huang, "A survey on mobile edge computing: Focusing on service adoption and provision," *Wirel. Commun. Mobile Comput.*, vol. 2018, 2018.

[8] Y. Wu, K. Ni, C. Zhang, L. P. Qian, and D. H. K. Tsang, "NOMA-assisted multi-access mobile edge computing: A joint optimization of computation offloading and time allocation," *IEEE Trans. Veh. Technol*, vol. 67, no. 12, pp. 12244–12258, Dec. 2018.

[9] J. Hu, K. Li, C. Liu, J. Chen, and K. Li, "Coalition formation for deadline-constrained resource procurement in cloud computing," *J. Parallel Distrib. Comput.*, vol. 149, pp. 1–12, 2021.

[10] J. Ren, G. Yu, Y. He, and G. Y. Li, "Collaborative cloud and edge computing for latency minimization," *IEEE Trans. Veh. Technol*, vol. 68, no. 5, pp. 5031–5044, May 2019.

[11] R. Fantacci and B. Picano, "Performance analysis of a delay constrained data offloading scheme in an integrated cloud-fog-edge computing system," *IEEE Trans. Veh. Technol*, vol. 69, no. 10, pp. 12004–12014, Oct. 2020.

[12] P. Wang, Z. Zheng, B. Di, and L. Song, "HetMEC: Latency-optimal task assignment and resource allocation for heterogeneous multi-layer mobile edge computing," *IEEE Trans. Wireless Commun.*, vol. 18, no. 10, pp. 4942–4956, Oct. 2019.

[13] G. Cui, Q. He, F. Chen, Y. Zhang, H. Jin, and Y. Yang, "Interference-aware game-theoretic device allocation for mobile edge computing," *IEEE Trans. Mobile Comput.*, vol. 21, no. 11, pp. 4001–4012, Nov. 2021.

[14] L. Chen, S. Zhou, and J. Xu, "Computation peer offloading for energy-constrained mobile edge computing in small-cell networks," *IEEE/ACM Trans. Netw.*, vol. 26, no. 4, pp. 1619–1632, Apr. 2018.

[15] C. Wang, F. R. Yu, C. Liang, Q. Chen, and L. Tang, "Joint computation offloading and interference management in wireless cellular networks with mobile edge computing," *IEEE Trans. Veh. Technol*, vol. 66, no. 8, pp. 7432–7445, Aug. 2017.

[16] P. A. Apostolopoulos, E. E. Tsiropoulou, and S. Papavassiliou, "Cognitive data offloading in mobile edge computing for internet of things," *IEEE Access*, vol. 8, pp. 55736–55749, 2020.

[17] Z. Zhou, S. Yu, W. Chen, and X. Chen, "CE-IoT: Cost-effective cloud-edge resource provisioning for heterogeneous IoT applications," *IEEE Internet Things J.*, vol. 7, no. 9, pp. 8600–8614, Sep. 2020.

[18] Y. Ding, K. Li, C. Liu, and K. Li, "A potential game theoretic approach to computation offloading strategy optimization in end-edge-cloud computing," *IEEE Trans. Parallel Distrib. Syst.*, vol. 33, no. 6, pp. 1503–1519, Jun. 2022.

[19] X. Chen, L. Jiao, W. Li, and X. Fu, "Efficient multi-user computation offloading for mobile-edge cloud computing," *IEEE/ACM Trans. Netw.*, vol. 24, no. 5, pp. 2795–2808, May 2016.
[20] T. Fang, F. Yuan, L. Ao, and J. Chen, "Joint task offloading, D2D pairing, and resource allocation in device-enhanced mec: A potential game approach," *IEEE Internet Things J.*, vol. 9, no. 5, pp. 3226–3237, May 2021.
[21] P. Lai et al., "Quality of experience-aware user allocation in edge computing systems: A potential game," in *Proc. IEEE 40th Int. Conf. Distrib. Comput. Syst.*, pp. 223–233, 2020.
[22] J. Zheng, Y. Cai, Y. Liu, Y. Xu, B. Duan, and X. Shen, "Optimal power allocation and user scheduling in multicell networks: Base station cooperation using a game-theoretic approach," *IEEE Trans. Wireless Commun.*, vol. 13, no. 12, pp. 6928–6942, Dec. 2014.
[23] D. Monderer and L. S. Shapley, "Potential games," *Games Econ. Behav.*, vol. 14, no 1, pp. 124–143, 1996.
[24] G. Cui et al., "Interference-aware SaaS user allocation game for edge computing," *IEEE Trans. Cloud Comput.*, vol. 10, no. 3, pp. 1888–1899, Third Quarter, 2020.
[25] T. Roughgarden, *Selfish Routing and the Price of Anarchy*. Cambridge, MA, USA: MIT Press, 2005.
[26] J. Kwak, O. Choi, S. Chong, and P. Mohapatra, "Processor-network speed scaling for energy–delay tradeoff in smartphone applications," *IEEE/ACM Trans. Netw.*, vol. 24, no. 3, pp. 1647–1660, Mar. 2016.
[27] J. Kwak, Y. Kim, J. Lee, and S. Chong, "Dream: Dynamic resource and task allocation for energy minimization in mobile cloud systems," *IEEE J. Sel. Areas Commun.*, vol. 33, no. 12, pp. 2510–2523, Dec. 2015.
[28] P. Lai et al., "Cost-effective user allocation in 5G noma-based mobile edge computing systems," *IEEE Trans. Mobile Comput.*, vol. 21, no. 12, pp. 4263–4278, Dec. 2022.
[29] P. Lai et al., "Edge user allocation with dynamic qualityof service," in *Proc. Int. Conf. Serv.-Oriented Comput.*, 2019, pp. 86–101.
[30] Q. He et al., "A game-theoretical approach for user allocation in edge computing environment," *IEEE Trans. Parallel Distrib. Syst.*, vol. 31, no. 3, pp. 515–529, Mar. 2020.
[31] H. Zeng, X. Zhu, Y. Jiang, Z. Wei, and T. Wang, "A green coordinated multi-cell noma system with fuzzy logic based multi-criterion user mode selection and resource allocation," *IEEE J. Sel. Topics Signal Process.*, vol. 13, no. 3, pp. 480–495, Mar. 2019.
[32] Z. Xu et al., "Near-optimal and collaborative service caching in mobile edge clouds," *IEEE Trans. Mobile Comput.*, to be published, doi: 10.1109/TMC.2022.3144175.
[33] X. Lyu, C. Ren, W. Ni, H. Tian, R. P. Liu, and X. Tao, "Distributed online learning of cooperative caching in edge cloud," *IEEE Trans. Mobile Comput.*, vol. 20, no. 8, pp. 2550–2562, Aug. 2021.
[34] C. Wang, C. Liang, F. R. Yu, Q. Chen, and L. Tang, "Computation offloading and resource allocation in wireless cellular networks with mobile edge computing," *IEEE Trans. Wireless Commun.*, vol. 16, no. 8, pp. 4924–4938, Aug. 2017.
[35] W. Fan et al., "Collaborative service placement, task scheduling, and resource allocation for task offloading with edge-cloud cooperation," *IEEE Trans. Mobile Comput.*, to be published, doi: 10.1109/TMC.2022.3219261.
[36] S. Duan et al., "MOTO: Mobility-aware online task offloading with adaptive load balancing in small-cell MEC," *IEEE Trans. Mobile Comput.*, to be published, doi: 10.1109/TMC.2022.3220720.
[37] X. Chen, L. Jiao, W. Li, and X. Fu, "Efficient multi-user computation offloading for mobile-edge cloud computing," *IEEE/ACM Trans. Netw.*, vol. 24, no. 5, pp. 2795–2808, May 2015.
[38] C. Sun et al., "Task offloading for end-edge-cloud orchestrated computing in mobile networks," in *Proc. IEEE Wirel. Commun. Netw. Conf.*, 2020, pp. 1–6.
[39] J. Du, L. Zhao, J. Feng, and X. Chu, "Computation offloading and resource allocation in mixed fog/cloud computing systems with min-max fairness guarantee," *IEEE Trans. Commun.*, vol. 66, no. 4, pp. 1594–1608, 2018.
[40] P. A. Apostolopoulos, E. E. Tsiropoulou, and S. Papavassiliou, "Risk-aware data offloading in multi-server multi-access edge computing environment," *IEEE/ACM Trans. Netwo.*, vol. 28, no. 3, pp. 1405–1418, Jun. 2020.
[41] N. Irtija, I. Anagnostopoulos, G. Zervakis, E. E. Tsiropoulou, H. Amrouch, and J. Henkel, "Energy efficient edge computing enabled by satisfaction games and approximate computing," *IEEE Trans. Green Commun. Netw.*, vol. 6, no. 1, pp. 281–294, Mar. 2021.
[42] P. K. Bishoyi and S. Misra, "Enabling green mobile-edge computing for 5G-based healthcare applications," *IEEE Trans. Green Commun. Netw.*, vol. 5, no. 3, pp. 1623–1631, Sep. 2021.
[43] P. K. K. Bishoyi and S. Misra, "Towards energy-and cost-efficient sustainable MEC-assisted healthcare systems," *IEEE Trans. Sustain. Comput.*, early access, Apr. 26, 2022, doi: 10.1109/TSUSC.2022.3170508.
[44] H. Feng, S. Guo, D. Liu, and Y. Yang, "Mean-field game theory based optimal caching control in mobile edge computing," *IEEE Trans. Mobile Comput.*, early access, Jul. 25, 2022, doi: 10.1109/TMC.2022.3193764.
[45] X. Chen, H. Zhang, C. Wu, S. Mao, Y. Ji, and M. Bennis, "Optimized computation offloading performance in virtual edge computing systems via deep reinforcement learning," *IEEE Internet Things J.*, vol. 6, no. 3, pp. 4005–4018, Jun. 2019.

Ying Chen (Member, IEEE) received the PhD degree in computer science and technology from Tsinghua University, Beijing, China, in 2017, where she was a joint PhD student with the University of Waterloo, Waterloo, ON, Canada from 2016 to 2017. She is an associate professor with the Computer School, Beijing Information Science and Technology University, Beijing. Her current research interests include Internet of Things, mobile edge computing, wireless net works and communications, machine learning, etc. She is the recipient of the Best Paper Award with IEEE SmartIoT 2019, the 2016 Google PhD Fellowship Award, and the 2014 Google Anita Borg Award, 2022 OUTSTANDING CONTRIBUTION AWARD in 18th EAI CollaborateCom, respectively. She serves/served the leading guest editor of Springer JCC, TPC member of IEEE HPCC, and PC member of IEEE Cloud, CollaborateCom, IEEE CPSCom, CSS, etc. She is also the Reviewer of several journals such as the *IEEE Wireless Communications Magazine*, *IEEE Transactions on Dependable and Secure Computing*, *IEEE Internet of Things Journal*, *IEEE Transactions on Computers*, *IEEE Transactions on Cloud Computing*, and *IEEE Transactions on Services Computing*.

Jie Zhao is currently working toward the MEng degree in computer science and technology, the Beijing Information Science and Technology University, China. His current research interests include edge computing, Internet of Things, and Game theory.

Yuan Wu (Senior Member, IEEE) received the PhD degree in electronic and computer engineering from the Hong Kong University of Science and Technology, in 2010. He is currently an associate professor with the State Key Laboratory of Internet of Things for Smart City, University of Macau, Macao, China, and also with the Department of Computer and Information Science, University of Macau. During 2016-2017, He was a visiting scholar with Department of Electrical and Computer Engineering, University of Waterloo. His research interests include resource management for wireless networks, green communications and computing, edge computing and edge intelligence, and energy informatics. He received the Best Paper Award from the IEEE ICC'2016, WCSP'2016, IEEE TCGCC'2017, and IWCMC'2021. He is currently on the editorial board of *IEEE Transactions on Vehicular Technology*, *IEEE Transactions on Network Science and Engineering*, and *IEEE Internet of Things Journal*.

Jiwei Huang (Member, IEEE) received the BEng and PhD degrees in computer science and technology from Tsinghua University, Beijing, China, in 2009 and 2014, respectively. He was a Visiting Scholar with the Georgia Institute of Technology, Atlanta, GA, USA. He is currently a professor and the Dean with the Department of Computer Science and Technology, China University of Petroleum, Beijing, and the Director of Beijing Key Laboratory of Petroleum Data Mining, Beijing. He has authored or coauthored one book and more than 60 articles in international journals and conference proceedings, including *IEEE Transactions on Services Computing*, *IEEE Transactions on Cloud Computing*, *IEEE Transactions on Vehicular Technology*, *IEEE Internet of Things Journal*, *ACM Sigmetrics*, and IEEE ICWS, etc. His research interests include services computing, Internet of Things, and edge computing. He is currently on the Editorial Board of the Chinese Journal of Electronics and Scientific Programming.

Xuemin Shen (Fellow, IEEE) received the PhD degree in electrical engineering from Rutgers University, New Brunswick, NJ, USA, in 1990. He is a University Professor with the Department of Electrical and Computer Engineering, University of Waterloo, Canada. His research focuses on network resource management, wireless network security, Internet of Things, 5G and beyond, and vehicular networks. He is a registered Professional Engineer of Ontario, Canada, an Engineering Institute of Canada Fellow, a Canadian Academy of Engineering Fellow, a Royal Society of Canada Fellow, a Chinese Academy of Engineering Foreign Member, and a Distinguished Lecturer of the *IEEE Vehicular Technology Society and Communications Society*. He received the Canadian Award for Telecommunications Research from the Canadian Society of Information Theory (CSIT) in 2021, the R.A. Fessenden Award, in 2019 from IEEE, Canada, Award of Merit from the Federation of Chinese Canadian Professionals (Ontario), in 2019, James Evans Avant Garde Award, in 2018 from the *IEEE Vehicular Technology Society*, Joseph LoCicero Award in 2015 and Education Award, in 2017 from the IEEE Communications Society (ComSoc), and Technical Recognition Award from Wireless Communications Technical Committee (2019) and AHSN Technical Committee (2013). He has also received the Excellent Graduate Supervision Award in 2006 from the University of Waterloo and the Premier's Research Excellence Award (PREA), in 2003 from the Province of Ontario, Canada. He served as the Technical Program Committee Chair/Co-Chair for IEEE Globecom'16, IEEE Infocom'14, IEEE VTC'10 Fall, IEEE Globecom'07, and the Chair for the IEEE ComSoc Technical Committee on Wireless Communications. He is the President of the IEEE ComSoc. He was the Vice President for Technical & Educational Activities, Vice President for Publications, Member-at-Large on the Board of Governors, Chair of the Distinguished Lecturer Selection Committee, and Member of IEEE Fellow Selection Committee of the ComSoc. He served as the Editor-in-Chief of the *IEEE IoT Journal*, *IEEE Network*, and *IET Communications*.