







# Efficient Multi-Task Computation Offloading Game for Mobile Edge Computing

Shuhui Chu , Chengxi Gao , *Member, IEEE*, Minxian Xu , *Member, IEEE*, Kejiang Ye , *Senior Member, IEEE*, Zhu Xiao , *Senior Member, IEEE*, and Chengzhong Xu , *Fellow, IEEE*

**Abstract**—Mobile edge computing emerges to serve mobile users with low-latency computation offloading in edge networks, which are resource-constrained with massive users and workloads. However, existing communication and computing resource allocation schemes for offloaded tasks aren't efficient enough, where finished tasks still occupy resources, wasting constrained resources. Besides, the multi-user offloading is usually for scenarios of one task per user, ignoring real-world *multi-task* offloading scenarios where each user has multiple tasks, lack generality and flexibility. Meanwhile, local computing resource allocation schemes in multi-task scenarios ignore resource readjustment, causing low resource utilization. To solve these problems, we propose ECO-GAME, an efficient multi-task offloading scheme, which dynamically allocates bandwidth and computing resources to unfinished tasks, resulting in high resource utilization. We initially formulate the multi-task offloading problem as the game minimizing each user's cost, which is NP-hard. Thus we re-formulate the game utilizing potential games to optimize user's objective either locally or globally, and prove the existence of its Nash equilibrium. We then design an efficient multi-task offloading algorithm to obtain an approximate solution in polynomial time, together with computational complexity analysis. We further conduct performance evaluation on ECO-GAME utilizing price of anarchy. Numerical results demonstrate the efficiency of ECO-GAME, and show ECO-GAME reduces 49.2% cost over the state-of-the-art work, and scales well with the increasing number of tasks and users.

**Index Terms**—Multi-task, mobile edge computing, computation offloading, potential games, Nash equilibrium.

## I. INTRODUCTION

THE development of the mobile edge computing and the Internet of Things (IoT) technology comes with a massive amount of mobile devices. Many mobile applications on mobile devices, including natural language processing, face recognition, interactive gaming and augmented reality, are emerging in the market in recent years, which generally require intensive computation [1], [2]. However, mobile devices are constrained in computing capability. The conflict between resource-constrained mobile devices and resource-consuming applications drives the development of mobile cloud computing (MCC) [3]. In MCC, the computation tasks are offloaded from mobile devices to resource-rich cloud infrastructures for execution, which are however located physically far away from mobile devices, causing high latency for data exchange with the cloud. To reduce data transmission delays during offloading, mobile edge computing (MEC) has been proposed [4]. MEC deploys the cloud infrastructures at the edge of the wireless networks, whose physical locations are in the vicinity of the mobile users [5], [6], [7]. Therefore, for computation offloading, MEC is able to provide much lower delays and jitters than MCC, thus satisfying the demands of the delay-sensitive applications.

The mobile edge networks come with a massive amount of mobile user devices and a large amount of data and task load from these devices, where the quick increase in the number of mobile devices and task workload has brought a big burden to the resource-constrained communication networks. Therefore, it is necessary and urgent to fully utilize the limited resources of networks. However, most of existing works implicitly assume that the bandwidth and computing resources allocated to each offloaded task remain *unchanged* during an offloading and execution period regardless of the released resources by finished tasks, thus leading to resource waste and sub-optimal solutions. Besides, they fail to consider the real-world *multi-task* computation offloading scenarios where each user has multiple computation tasks and the local computing resources allocated to each task can be re-adjusted throughout its local execution with the resource release by finished tasks, therefore lack generality and flexibility.

Specifically, the limitations manifest in four perspectives:

Manuscript received 12 August 2022; revised 22 October 2023; accepted 5 November 2023. Date of publication 13 November 2023; date of current version 6 February 2024. This work was supported in part by the National Key R&D Program of China under Grant 2021YFB3300200, in part by Science and Technology Development Fund, Macau SAR under Grant 0081/2022/A2, in part by NSFC and The Science and Technology Development Fund, Macau SAR under Grant 0123/2022/AFJ, in part by National Natural Science Foundation of China under Grants 62072451, 92267105, 62272152, and 62102408, in part by Guangdong Special Support Plan under Grant 2021TQ06X990, in part by Shenzhen Basic Research Program under Grants JCYJ20200109115418592, JCYJ20220818101610023, and JCYJ20220531100804009, in part by Open Research Fund from Guangdong Laboratory of Artificial Intelligence and Digital Economy [Shenzhen (SZ)] under Grant GML-KF-22-22, in part by Shenzhen Science and Technology Program under Grants JCYJ20220530160408019 and RCBS20210609104609044, and in part by CAAI-Huawei MindSpore Open Fund, Guangdong Basic and Applied Basic Research Foundation under Grant 2023A1515011915. (*Corresponding authors: Chengxi Gao; Chengzhong Xu.*)

Shuhui Chu and Chengzhong Xu are with the State Key Lab of IoTSC, Department of Computer and Information Science, University of Macau, Taipa, Macau 999078, China (e-mail: yc07445@um.edu.mo; czxu@um.edu.mo).

Chengxi Gao, Minxian Xu, and Kejiang Ye are with the Shenzhen Institute of Advanced Technology, Chinese Academy of Sciences, Shenzhen 518055, China (e-mail: chengxi.gao@siat.ac.cn; mx.xu@siat.ac.cn; kj.ye@siat.ac.cn).

Zhu Xiao is with the Shenzhen Research Institute, Hunan University, Shenzhen 518055, China (e-mail: zhuxiao@hnu.edu.cn).

Digital Object Identifier 10.1109/TSC.2023.3332140

*Waste of Wireless Communication Resources:* For computation offloading in MEC, like many related works [8], [9], [10], the wireless communication resources are constrained. The majority of existing works propose solutions about how each mobile user selects an appropriate wireless channel to minimize its computational offloading cost of time. However, they assume that the bandwidth resources allocated to each offloaded task remain unchanged, even though other offloaded tasks have completed transmission. In other words, in their bandwidth resource allocation schemes, once the allocated bandwidth is determined, it is fixed throughout the transmission of the task and can not be re-adjusted although other tasks have completed transmission. This leads to the waste of wireless bandwidth resources, and is inferior for the resource-constrained scenarios. Therefore, it is necessary to propose an efficient bandwidth allocation scheme that re-allocates the wireless bandwidth from the transmission-completed tasks to the transmission-uncompleted tasks to fully utilize resources and improve computational offloading performance in resource-constrained scenarios.

*Waste of MEC Computing Resources:* According to [8], [11], the computing resources in MEC are limited. The majority of existing works only propose allocating MEC computing resources to mobile users' tasks that choose offloading equally for fairness, and imply that the allocated MEC computing resources remain unchanged during a computation offloading period, even though other tasks have finished computing. In other words, in their computing resource allocation schemes, once the allocated computing resources are determined, they are fixed throughout the execution of the task and can not be re-adjusted although other tasks have finished computing. This leads to the waste of MEC computing resources, and is inferior for the resource-constrained scenarios. Thus, it is necessary to propose an efficient computing resource allocation scheme that re-allocates the computing resources from the computation-completed tasks to the computation-uncompleted tasks to fully utilize the resources and improve computation offloading performance in the resource-constrained scenarios.

*Lack of Consideration of the Multi-Task Offloading Scenarios:* The majority of existing solutions assume the multi-user offloading scenarios where each user has only one computation task, and try to reach optimal solutions. However, in a variety of real-world applications, the smart device usually needs to perform multiple tasks simultaneously [11], [12], [13], [14], [15]. For example, the intelligent camera may have to simultaneously perform several tasks such as the compression of video data, real-time object recognition and so on. Therefore, the previous solutions are not suitable for the *multi-task* offloading scenarios where each user has a set of computation tasks to offload, since the optimization objectives change into the granularity of user (a group of tasks) instead of a single task. In this case, it's essential to take the *multi-task* offloading scenarios into consideration for generality and re-formulate the optimization problem in the granularity of user.

*Lack of Consideration of the Local Computing Resource Re-Adjustment for Each Task:* As is known, mobile devices are constrained in computing capability, while existing works such as [13], [16], [17], [18], [19] propose the simple local computing

models which only consider that the local computing resources allocated to each task are fixed throughout its local execution once determined. They do not consider that when other tasks finish computing, the corresponding allocated local computing resources are released and the local computing resources of uncompleted tasks can be re-adjusted. Their local computing resource allocation schemes ignore the resource readjustment, resulting in the waste of local computing resources and low resource utilization. Therefore, it is necessary to propose an efficient local computing resource allocation scheme that takes the release and re-adjustment of the local computing resources into consideration to improve the utilization of constrained local computing resources.

To solve all aforementioned challenging problems, we propose ECO-GAME, a multi-task computation offloading algorithm together with efficient wireless bandwidth and computing resource allocation schemes. In ECO-GAME, we consider the cases that the bandwidth and computing resources released by the completed tasks are re-allocated to the uncompleted tasks, and each user has multiple computation tasks, which is abstracted to the *multi-task* offloading problem. Then, we take advantage of the tools of *game theory* to formulate the multi-task offloading problem as the strategic game, so that each mobile user makes its decisions locally according to its own interests. Our *contributions* are outlined below.

- We take into consideration the real-world scenarios where each user has multiple tasks to compute in the MEC computation offloading problem to ensure the generality. Besides, for constrained resources, we take the resource release and readjustment into consideration, and propose efficient wireless bandwidth and edge computing resource allocation schemes for offloaded tasks and an efficient local computing resource allocation scheme for tasks computed locally, which re-allocate bandwidth and computing resources of completed tasks to uncompleted tasks, resulting in high resource utilization. Based on the aforementioned efficient resource allocation schemes, we propose the computation models.
- Based on game theory, we initially formulate the multi-task offloading problem as the computation offloading game, which minimizes each user's average cost for computing all its generated tasks. However it's NP-hard to solve the game. Thus we utilize the tool of potential games to re-formulate the game to optimize each user's objective either locally or globally, thereby solving the solution in polynomial time. For the re-formulated game, its Nash equilibrium is further proved to exist by constructing corresponding potential function to show that it belongs to potential games.
- We propose the multi-task offloading algorithm based on the formulated game (ECO-GAME), to obtain an approximate solution in polynomial time via its Nash equilibrium. Our ECO-GAME is distributed and each user can make its decisions locally and independently, ensuring the privacy and confidentiality. Further, we perform an analysis about the low computational complexity of ECO-GAME, and derive an upper bound on the number of slots required for

ECO-GAME to terminate. Besides, performance evaluation on ECO-GAME is given by utilizing the metric of price of anarchy (PoA) and the upper and lower bounds for PoA are derived from the perspective of the system cost.

- We conduct numerous experiments, and the experimental results demonstrate the efficiency of ECO-GAME. The results show ECO-GAME minimizes each user's cost while also reducing the computation cost of its each task, ensuring the user's QoS (quality of service). Besides, the results illustrate ECO-GAME can be implemented in a real time manner, and reduce the system cost by 49.2% over JPBR [20], and scale well as the number of tasks on each user increases and the number of users increases.

The remainder of this paper is organized as follows. The related work in MEC computation offloading is discussed in Section II. Section III introduces both system model and problem formulation. We formulate the computation offloading game and propose the computation offloading algorithm in Sections IV and V respectively. Section VI gives numerical results, and the paper is concluded in Section VII.

## II. RELATED WORK

In last few years, computation offloading has increasingly played a critical role in the area of mobile edge computing [21]. Many existing works solve the computation offloading problems in different perspectives. We summarize related works as follows.

Typically, the optimization goals of the computation offloading works focus either on energy consumption or on latency, or on both [22]. Some existing works [23], [24] investigated the computation offloading problems for minimizing energy consumption. [23] optimized the energy consumption of the hosts in cloud computing environment by considering their computation, communication, and cooling energy consumption. [24] investigated the problem of energy minimization together with resource allocation, within the constraints on the computing and wireless resources and the time latency. Some others [25], [26], [27], [28] investigated the latency-minimization problems in MEC offloading systems. [29] modelled the edge-edge and edge-cloud collaborative offloading problem as the Markov decision process to support low-latency applications. Both energy and latency are critical factors in computation offloading, and there are some works like [30], [31], [32], [33] optimizing energy and time simultaneously in computation offloading. [30] applied the vehicular nodes to provide the computing resources for solving the Cloudlet node overload problem via task offloading. [31] aimed to minimize the total cost of energy and latency for all tasks using the reinforcement learning scheme. However, [31] only focused on the system level, ignoring the individual level.

A number of recent works applied the game theoretical approach to investigate the distributed multi-user computation offloading schemes [8], [9], [10], [20], [34], [35], [36], [37], [38], [39]. [9] formulated the partial offloading problem in multi-MEC system as a convex optimization problem maximizing each user's satisfaction, which was further confronted as a non-cooperative game. [20], [36] considered the computation

offloading problem with the wireless and cloud resource allocation, which was analysed from a game theoretic perspective. [34] formulated the distributed computation offloading problem in single-channel communication scenarios as the strategic game, and its Nash equilibrium was proved to exist. [35] extended [34] by considering multi-channel communication scenarios, and Nash equilibrium of the reformulated game was also proved to exist by using the tool of potential games. [39] formulated the distributed offloading problem for Internet of Vehicles as a game, and designed a self-learning based algorithm to obtain its Nash equilibrium. However, all works above do not consider the wireless and cloud resources from the completed tasks are re-allocated to the uncompleted tasks. Besides, they only focus on the multi-user offloading scenarios, where however each mobile user has only one single computation task to perform.

For the multi-task offloading scenario, where each mobile user has a group of computation tasks [11], [11], [12], [13], [14], [16], [17], [18], [40], [41] studied the multi-server multi-task offloading problem for minimizing the overall delay of all tasks, where the communication and computing resource allocations were jointly optimized. [17] studied the problem of joint task offloading scheduling and resource allocation in vehicular edge computing and decomposed it into a two-layer optimization problem. [41] proposed a joint task offloading and resource allocation with edge-edge cooperation and formulated it as a potential game. However, they do not consider the reallocations of these resources to uncompleted tasks. [12] formulated the task offloading issue as a stochastic game based on multi-agent imitation learning, in which each device minimizes its average completion time of all the tasks it generates. However, [12] does not consider the contention for communication and computation resources among multiple offloaded tasks. [18] considered resource allocation, compression and security issues in computation offloading, which however doesn't consider the contention for local computing resources among multiple tasks. [13] investigated the multi-vehicle multi-task offloading problem with task delay requirement constraints. [16] investigated the multi-task offloading problem through nonorthogonal multiple access in mobile edge computing. However, these works do not consider the local computing resources from the completed tasks are re-allocated to the uncompleted tasks.

The most related to our work are [20], [32], [42]. Our previous work in [32] focused on the distributed multi-channel offloading problem in homogeneous multi-channel communication scenarios, and that the cloud computing resources allocated to each offloaded task kept unchanged throughout its execution. Different from [32], we investigate the multi-task offloading problem in the heterogeneous multi-channel communication scenarios in this paper, and consider cloud computing resources from the completed tasks are re-allocated to the uncompleted tasks. [42] considered the mobile users offloaded the tasks via a shared channel, and only the users currently offloading their tasks could be allocated the time slots dynamically. Different from [42], our work takes the cloud computing resource contention among multiple offloaded tasks into consideration. Besides, different from [20], our work considers the communication resources and cloud computing resources from the completed tasks are



re-allocated to the uncompleted tasks. Moreover, our work considers the multi-task computation offloading scenarios, where each user has multiple computation tasks. And different from the existing multi-task offloading works, our work considers the local computing resources from the completed tasks are re-allocated to the uncompleted tasks.

### III. SYSTEM MODEL AND PROBLEM FORMULATION

Consider a general MEC network architecture consisting of a wireless base station together with edge server resources deployed in the vicinity and a group of  $\mathcal{N} = \{1, 2, \dots, N\}$  mobile device users, each mobile device user having multiple independent computation tasks [13], [16] (e.g., a smart camera user may run multiple independent tasks of video data compression and real-time target recognition in parallel [16]). Any of the tasks can be either computed on the device locally or offloaded to edge servers for execution via one of a group of  $\mathcal{M} = \{1, 2, \dots, M\}$  wireless channels. Here the number of computation tasks generated by user  $n$  is denoted as  $k_n$ . Each user  $n$ 's task  $i$  is denoted as  $(D_{n,i}, L_{n,i})$ . Here  $D_{n,i}$  denotes the size of transmission data (program files, input parameters, etc) when user  $n$  offloads its task  $i$ .  $L_{n,i}$  denotes the number of CPU cycles required for user  $n$ 's task  $i$ . For user  $n$ , some methods such as call graph analysis can be applied to obtain the information of  $D_{n,i}$  and  $L_{n,i}$  for each task  $i$  [35]. User  $n$ 's task  $i$ 's computation offloading decision is denoted as  $a_{n,i} \in \{0\} \cup \mathcal{M}$ . Here  $a_{n,i} = 0$  implies user  $n$  determines to execute task  $i$  on the device locally, and  $a_{n,i} > 0$  implies user  $n$  determines to offload task  $i$  to edge servers for computation via the channel  $a_{n,i}$ . The decisions of all the tasks of user  $n$  make user  $n$ 's strategy  $\mathbf{a}_n = (a_{n,1}, a_{n,2}, \dots, a_{n,k_n})$ . Further, all users' strategies constitute the decision profile  $\mathbf{a} = (\mathbf{a}_1, \mathbf{a}_2, \dots, \mathbf{a}_N)$ . Like many other works including [34], [35], we make an assumption that all the users stay unchanged within a computation offloading period.<sup>1</sup> Considering constrained computing and wireless resources, we next discuss our models of computing and wireless resource allocation respectively. Note that the major notations used in this paper are summarized in Table I.

#### A. Computing Resource Allocation

1) *Local Computing Resource Allocation*: We denote user  $n$ 's computing capability on its device as  $F_n^l$  (i.e., number of CPU cycles per second). For user  $n$ 's task  $i$ , if it chooses local computing ( $a_{n,i} = 0$ ) and the other tasks of user  $n$  choose offloading ( $a_{n,j} > 0, \forall j \in \{1, 2, \dots, k_n\} \setminus \{i\}$ ), the allocated computing resource of user  $n$ 's task  $i$  is  $F_n^l$ . Due to the limited local computing resources, when other tasks of user  $n$  also choose local computing, they compete for local computing resources. In order to ensure fairness among these independent tasks, we consider that the computing resources of user  $n$  will be equally shared among these tasks that choose local computing. Therefore, when  $a_{n,i} = 0$ , the computing resources allocated to user  $n$ 's task  $i$  are computed as  $\frac{F_n^l}{\sum_{j=1}^{k_n} I_{\{a_{n,j}=0\}}}$ . The value of the

<sup>1</sup>In other words, mobile users cannot dynamically arrive or depart or change their tasks' decisions within a computation offloading period.

TABLE I  
TABLE OF NOTATIONS

Symbol	Description (Unit)
$N$	Number of users
$M$	Number of channels
$k_n$	Number of tasks generated by user $n$
$\mathbf{a}_n$	User $n$ 's strategy consisting of decisions of all tasks of user $n$
$a_{n,i}$	Decision of user $n$ 's task $i$
$\mathbf{a}_{-(n,i)}$	Decisions of all tasks of all users except user $n$ 's task $i$
$\mathbf{a}_{-n}$	Strategies of all users except user $n$
$\mathbf{a}_{n,-i}$	Decisions of user $n$ 's all tasks except task $i$
$\mathbf{a}$	Decision profile consisting of strategies of all users
$\mathbf{a}^*$	The decision profile of Nash equilibrium
$B_m$	Total bandwidth on the wireless channel $m$ (Mbps)
$D_{n,i}$	Size of transmission data when user $n$ offloads task $i$ (Mb)
$L_{n,i}$	Number of CPU cycles required for user $n$ 's task $i$ (Gcycle)
$F^c$	Computing capability of MEC servers (GHz)
$F_n^l$	User $n$ 's computing capability on its device (GHz)
$C_{n,i}^l(\mathbf{a})$	Local computing cost of user $n$ 's task $i$
$C_{n,i}^e(\mathbf{a})$	Edge computing cost of user $n$ 's task $i$
$C_{n,i}(\mathbf{a})$	Cost for executing user $n$ 's task $i$
$\mathcal{N}$	The set of users
$\mathcal{S}_n$	Strategy space of user $n$
$T_n$	User $n$ 's cost function
$U_{n,i}$	Utility function of user $n$ 's task $i$
$\mathcal{T}$	Set of all tasks of all users
$\mathcal{K}_n$	Set of all tasks of user $n$
$\mathcal{S}_{n,i}$	Strategy space of task $i$ of user $n$
$\Gamma_0$	The original game formulated based on the problem
$\Gamma_n$	The potential game for user $n$
$\Gamma$	The final game reformulated based on $\Gamma_0$ and $\Gamma_n$
$\Delta_n(t)$	User $n$ 's set of task decision update at slot $t$
$\Phi(\mathbf{a})$	The potential function

indicator function  $I_{\{E\}}$  is 1 when  $E$  is true, and 0 when  $E$  is false. The existing works such as [13], [16] assume a simple local computing model, where the allocated local computing resources cannot be re-adjusted throughout the computation once determined. We put forward a complex local computing model with consideration of resource release and re-adjustment in this paper. Specifically, when the task computation is finished, the allocated computing resources are released immediately, and the released resources are re-allocated evenly to those tasks that have not yet finished the computation, thereby improving the resource utilization. Therefore, when  $a_{n,i} = 0$ , throughout the computation of user  $n$ 's task  $i$ , the allocated local computing resources may be dynamically increased in stages with the end of the computation of other tasks that choose local computing.

2) *Edge Computing Resource Allocation*: Due to constrained MEC computing resources, when multiple tasks choose edge computing, they compete for edge computing resources. In order to ensure fairness among these independent offloaded tasks, we consider that edge computing resources are equally allocated to all the offloaded tasks. Edge computing resources allocated to the offloaded task  $i$  of user  $n$  depend on the decisions of all the users' tasks. Therefore, when  $a_{n,i} > 0$ , we compute  $\frac{F^c}{\sum_{l=1}^N \sum_{k=1}^{k_l} I_{\{a_{l,k}>0\}}}$  as the computing resources that are allocated to task  $i$  of user  $n$ . Here  $F^c$  is the computing capability of MEC servers. However, given  $\mathbf{a}$ , most related works including [8] consider the computing resources allocated to each task that chooses offloading remain unchanged throughout its computation. In this paper, we take the resource release and re-adjustment into consideration, namely, the computing resources of any offloaded

task get released as soon as it finishes computation, and the released computing resources from the computation-completed tasks are re-allocated evenly to the computation-uncompleted tasks, in order to be work conserving. Therefore, when  $a_{n,i} > 0$ , throughout the computation of user  $n$ 's task  $i$ , the allocated computing resources may be dynamically increased in stages with the end of the computation of other offloaded tasks.

### B. Wireless Bandwidth Resource Allocation

The bandwidth resources of the wireless channel are shared by the tasks that choose to offload via the channel. Unlike [32] which considers the homogeneous multi-channel communication scenarios, here we focus on the scenarios of heterogeneous multi-channel communication and denote  $B_m$  as the total bandwidth on the wireless channel  $m$ . When multiple tasks choose to offload via channel  $m$ , they compete for the bandwidth resources of channel  $m$ . In order to ensure fairness among these independent offloaded tasks, we consider that wireless bandwidth resources are equally allocated to all the tasks that choose the same channel to offload. Therefore, when  $a_{n,i} > 0$ , the bandwidth resources allocated to user  $n$ 's task  $i$  are computed as  $\frac{B_{a_{n,i}}}{\sum_{l=1}^N \sum_{k=1}^{k_l} I_{\{a_{l,k}=a_{n,i}\}}}$ , where  $B_{a_{n,i}}$  is the total bandwidth on the channel  $m = a_{n,i}$ . However, given  $\mathbf{a}$ , most related works consider the bandwidth allocated to each offloaded task keeps unchanged in the entire data transmission. In this paper, we take the resource release and re-adjustment into consideration, namely, the bandwidth resources of any offloaded task get released once completing transmission, and the released bandwidth from the transmission-completed tasks is re-allocated evenly to the transmission-uncompleted tasks, in order to be work conserving. Therefore, for user  $n$ 's task  $i$ , when  $a_{n,i} > 0$ , the allocated bandwidth throughout its transmission may get increased dynamically with the end of the data transmission of other offloaded tasks on the same channel.

### C. Computation Model

We next discuss the response time when user  $n$ 's task  $i$  chooses local computing on user  $n$ 's device and edge computing on edge servers, respectively.

1) *Local Computing*: For user  $n$ 's task  $i$ , when  $a_{n,i} = 0$ , the local computing time of user  $n$ 's task  $i$  is related to all of user  $n$ 's tasks that choose local computing. First of all, for all of user  $n$ 's tasks that choose local computing, we sort their CPU cycles in ascending order (if they are equal, they are sorted according to the ID of the task). The sequence of the sorted CPU cycles can be denoted as  $L_{n,1'}, L_{n,2'}, \dots, L_{n,k'}$ , where  $k = \sum_{j=1}^{k_n} I_{\{a_{n,j}=0\}}$ . Here when  $1 \leq j \leq k$ ,  $L_{n,j'}$  denotes the  $j$ -th smallest CPU cycles among user  $n$ 's  $k$  tasks that choose local computing.

Then, based on previous analysis of local computing resource allocation, we can calculate the local computing time of the task whose CPU cycles are  $L_{n,1'}$ , as  $L_{n,1'}/\frac{F_n^l}{k} = \frac{kL_{n,1'}}{F_n^l}$ . Due to  $L_{n,1'} \leq L_{n,2'}$ , the task whose CPU cycles are  $L_{n,1'}$  finishes earlier than the task whose CPU cycles are  $L_{n,2'}$ . As soon as the task finishes computation, its computing resources are released and re-allocated to computation-uncompleted tasks.

Therefore, the computing time of the task whose CPU cycles are  $L_{n,2'}$ , can be divided into two parts. One part is the time before the task whose CPU cycles are  $L_{n,1'}$  finishes, which is equal to  $\frac{kL_{n,1'}}{F_n^l}$ . The other part is the time after the task whose CPU cycles are  $L_{n,1'}$  finishes, which is equal to  $(L_{n,2'} - L_{n,1'})/\frac{F_n^l}{k-1} = \frac{(k-1)(L_{n,2'} - L_{n,1'})}{F_n^l}$ . Hence, the total computing time of the task whose CPU cycles are  $L_{n,2'}$ , can be calculated as  $\frac{kL_{n,1'}}{F_n^l} + \frac{(k-1)(L_{n,2'} - L_{n,1'})}{F_n^l} = \frac{L_{n,1'} + (k-1)L_{n,2'}}{F_n^l}$ . Similarly, the computing time of the task whose CPU cycles are  $L_{n,3'}$ , can be calculated as  $\frac{L_{n,1'} + L_{n,2'} + (k-2)L_{n,3'}}{F_n^l}$ . Further, when  $2 \leq j \leq k$ , the computing time of the task whose CPU cycles are  $L_{n,j'}$ , can be calculated as

$$\frac{L_{n,1'} + L_{n,2'} + \dots + L_{n,(j-1)'} + (k-j+1)L_{n,j'}}{F_n^l} \quad (1a)$$

$$= \frac{\sum_{i=1}^{k_n} I_{\{a_{n,i}=0\}} \left( I_{\{L_{n,i} < L_{n,j'}\}} L_{n,i} + I_{\{L_{n,i} \geq L_{n,j'}\}} L_{n,j'} \right)}{F_n^l} \quad (1b)$$

$$= \frac{L_{n,j'}}{F_n^l} \sum_{i=1}^{k_n} I_{\{a_{n,i}=0\}} \left( I_{\{L_{n,i} < L_{n,j'}\}} \frac{L_{n,i}}{L_{n,j'}} + I_{\{L_{n,i} \geq L_{n,j'}\}} \right) \quad (1c)$$

$$= \frac{L_{n,j'}}{F_n^l} \sum_{i=1}^{k_n} I_{\{a_{n,i}=0\}} \min \left\{ \frac{L_{n,i}}{L_{n,j'}}, 1 \right\}. \quad (1d)$$

Here  $\min\{A, B\}$  is a function where  $\min\{A, B\} = B$  if  $A > B$  and  $\min\{A, B\} = A$  otherwise. As we show before, when  $j = 1$ , the computing time of the task whose CPU cycles are  $L_{n,j'}$ , is computed as  $\frac{kL_{n,1'}}{F_n^l}$ , which also satisfies (1d).

Therefore, according to (1d), the local computing time of user  $n$ 's task  $i$  whose CPU cycles are  $L_{n,i}$ , is computed as

$$T_{n,i}^l(\mathbf{a}) = \frac{L_{n,i}}{F_n^l} \sum_{j=1}^{k_n} \min \left\{ \frac{L_{n,j}}{L_{n,i}}, 1 \right\} I_{\{a_{n,j}=0\}}. \quad (2)$$

2) *Edge Computing*: When user  $n$ 's task  $i$  is offloaded to edge servers for execution (i.e.,  $a_{n,i} > 0$ ), its edge execution time is related to all offloaded tasks of all users. Based on previous analysis of edge computing resource allocation, with similar argument to the derivation of (2), for user  $n$ 's task  $i$ , its edge execution time can be calculated as

$$T_{n,i,exe}^c(\mathbf{a}) = \frac{L_{n,i}}{F^c} \sum_{j=1}^N \sum_{k=1}^{k_j} \min \left\{ \frac{L_{j,k}}{L_{n,i}}, 1 \right\} I_{\{a_{j,k}>0\}}. \quad (3)$$

In addition, for edge computing, computation offloading leads to additional time cost for data transmission. For user  $n$ 's task  $i$ , when  $a_{n,i} > 0$ , its data transmission time is related to all users' tasks on channel  $a_{n,i}$ . Based on previous analysis of wireless bandwidth resource allocation, with similar argument to the derivation of (2), for user  $n$ 's task  $i$ , its data transmission time

is calculated as

$$T_{n,i,off}^c(\mathbf{a}) = \frac{D_{n,i}}{B_{a_{n,i}}} \sum_{j=1}^N \sum_{k=1}^{k_j} \min \left\{ \frac{D_{j,k}}{D_{n,i}}, 1 \right\} I_{\{a_{j,k}=a_{n,i}\}}. \quad (4)$$

#### D. Problem Formulation

First of all, we consider the response time as the cost of computing user  $n$ 's task  $i$ . We next introduce the cost of user  $n$ 's task  $i$  in the cases of local computing and edge computing respectively.

For user  $n$ 's task  $i$ , when  $a_{n,i} = 0$ , the cost of local computing is computed as

$$C_{n,i}^l(\mathbf{a}) = T_{n,i}^l(\mathbf{a}), \quad (5)$$

where  $T_{n,i}^l(\mathbf{a})$  is given in (2).

For user  $n$ 's task  $i$ , when  $a_{n,i} > 0$ , we compute the cost of edge computing as

$$C_{n,i}^c(\mathbf{a}) = T_{n,i,off}^c(\mathbf{a}) + T_{n,i,exe}^c(\mathbf{a}), \quad (6)$$

where  $T_{n,i,off}^c(\mathbf{a})$  and  $T_{n,i,exe}^c(\mathbf{a})$  are given in (4) and (3) respectively. Here the time cost of computing results back to user  $n$  is ignored, since the size of computing results is generally small enough compared to  $D_{n,i}$  [34], [35].

Based on (5) and (6), the cost for computing user  $n$ 's task  $i$  can be calculated as

$$C_{n,i}(\mathbf{a}) = C_{n,i}^l(\mathbf{a})I_{\{a_{n,i}=0\}} + C_{n,i}^c(\mathbf{a})I_{\{a_{n,i}>0\}}. \quad (7)$$

Further, we define user  $n$ 's cost as the average cost (i.e., average response time) for computing all the tasks generated by user  $n$ . For each user, the objective is to minimize its own cost. This multi-task offloading problem is formulated as

$$\min_{\mathbf{a}_n} \frac{\sum_{i=1}^{k_n} C_{n,i}(\mathbf{a}_n, \mathbf{a}_{-n})}{k_n}, \forall n \in \mathcal{N}, \quad (8)$$

which implies the trade-off between optimality and fairness.  $\mathbf{a}_{-n} = (\mathbf{a}_1, \dots, \mathbf{a}_{n-1}, \mathbf{a}_{n+1}, \dots, \mathbf{a}_N)$  represents strategies of all users except user  $n$ .

In order to solve the problem formulated above and to obtain efficient users' strategies, we take advantage of game theory to formulate the corresponding game in the following section.

## IV. COMPUTATION OFFLOADING GAME

We utilize the game theoretic approach to solve the formulated multi-task offloading problem. Game theory is commonly used to design distributed schemes that allow users to self-organize to develop strategies based on their interactions, ultimately obtaining mutually satisfactory solutions in which no user wants to change its strategy unilaterally.

#### A. Game Formulation

In the formulated multi-task offloading problem, each user's objective is to minimize its own cost, which is formulated as

$$\min_{\mathbf{a}_n \in \mathcal{S}_n \triangleq \{0, 1, \dots, M\}^{k_n}} T_n(\mathbf{a}_n, \mathbf{a}_{-n}), \forall n \in \mathcal{N}. \quad (9)$$

According to (8), user  $n$ 's cost function is denoted as

$$T_n(\mathbf{a}_n, \mathbf{a}_{-n}) = \frac{\sum_{i=1}^{k_n} C_{n,i}(\mathbf{a}_n, \mathbf{a}_{-n})}{k_n}. \quad (10)$$

In the computation offloading problem above where users compete for bandwidth and computing resources, each user is selfish and aims to minimize its own cost. Then a straightforward approach to solve the problem is to formulate it as the game  $\Gamma_0 = (\mathcal{N}, \{\mathcal{S}_n\}_{n \in \mathcal{N}}, \{T_n\}_{n \in \mathcal{N}})$ . Here  $\mathcal{N}$  represents user set,  $\mathcal{S}_n$  represents user  $n$ 's strategy space, and  $T_n(\mathbf{a}_n, \mathbf{a}_{-n})$  represents user  $n$ 's cost function which is the utility function in games.

We solve the offloading problem using the significant solution concept of Nash equilibrium in games in this paper. Specifically, Nash equilibrium indicates a mutually satisfactory solution for all users in which no user wants to change its strategy unilaterally. Thus it's a relatively stable state. In the game  $\Gamma_0$ , denote its Nash equilibrium as  $\mathbf{a}^* = (\mathbf{a}_1^*, \mathbf{a}_2^*, \dots, \mathbf{a}_N^*)$  when there doesn't exist any user that can unilaterally change its strategy at  $\mathbf{a}^*$  to decrease its own cost, and this is expressed as

$$T_n(\mathbf{a}_n, \mathbf{a}_{-n}^*) \geq T_n(\mathbf{a}_n^*, \mathbf{a}_{-n}^*), \forall n \in \mathcal{N}, \mathbf{a}_n \in \mathcal{S}_n. \quad (11)$$

However, given  $\mathbf{a}_{-n}$ , the optimization problem for user  $n$  in (9) is a combinatorial optimization problem over the  $k_n$ -dimensional discrete space, which is NP-hard just like the similar analysis in [35]. To find an approximate solution in polynomial time, we discuss the distributed solution of the problem above using the potential game  $\Gamma_n = (\mathcal{K}_n, \{\mathcal{S}_{n,i}\}_{i \in \mathcal{K}_n}, \{U_{n,i}\}_{i \in \mathcal{K}_n})$  for user  $n$ , where  $\mathcal{K}_n = \{1, 2, \dots, k_n\}$  denotes the set of all tasks of user  $n$ ,  $\mathcal{S}_{n,i} \triangleq \{0, 1, \dots, M\}$  represents user  $n$ 's task  $i$ 's strategy space, and  $U_{n,i}$  in (14) represents the utility function of user  $n$ 's task  $i$ . By introducing the game  $\Gamma_n$  for user  $n$ , we can reduce the dimensionality of the search space from  $k_n$  to 1, and find an approximate solution in polynomial time. The details are as follows.

First of all, we introduce the concept of potential games [43] as given in [35].

The potential game  $\Gamma_n$  is a game with a potential function  $\Phi_n(\mathbf{a}_n, \mathbf{a}_{-n})$  which satisfies that for each  $i \in \mathcal{K}_n$ ,  $a_{n,-i} \in \prod_{i' \neq i} \mathcal{S}_{n,i'}$ , and  $a'_{n,i}, a_{n,i} \in \mathcal{S}_{n,i}$ , if

$$U_{n,i}(a'_{n,i}, a_{n,-i}, \mathbf{a}_{-n}) < U_{n,i}(a_{n,i}, a_{n,-i}, \mathbf{a}_{-n}), \quad (12)$$

we have

$$\Phi_n(a'_{n,i}, a_{n,-i}, \mathbf{a}_{-n}) < \Phi_n(a_{n,i}, a_{n,-i}, \mathbf{a}_{-n}). \quad (13)$$

Here  $a_{n,-i}$  represents the decisions of user  $n$ 's all other tasks except task  $i$ . Note that  $\mathbf{a} = (\mathbf{a}_n, \mathbf{a}_{-n}) = (a_{n,i}, a_{n,-i}, \mathbf{a}_{-n})$ . For user  $n$ 's task  $i$ , its utility function  $U_{n,i}$  is defined as follows.

$$\begin{aligned} U_{n,i}(a_{n,i}, a_{n,-i}, \mathbf{a}_{-n}) &= I_{\{a_{n,i}>0\}} \left[ \frac{D_{n,i}}{B_{a_{n,i}}} \left( \sum_{k=1}^{k_n} \min \left\{ \frac{D_{n,k}}{D_{n,i}}, 1 \right\} I_{\{a_{n,k}=a_{n,i}\}} - 1 \right) \right. \\ &\quad \left. + \frac{D_{n,i}}{B_{a_{n,i}}} \sum_{j=1}^N \sum_{k=1}^{k_j} \min \left\{ \frac{D_{j,k}}{D_{n,i}}, 1 \right\} I_{\{a_{j,k}=a_{n,i}\}} \right] \end{aligned}$$

$$\begin{aligned}
& + \frac{L_{n,i}}{F^c} \left( \sum_{k=1}^{k_n} \min \left\{ \frac{L_{n,k}}{L_{n,i}}, 1 \right\} I_{\{a_{n,k}>0\}} - 1 \right) \\
& + \frac{L_{n,i}}{F^c} \sum_{j=1}^N \sum_{k=1}^{k_j} \min \left\{ \frac{L_{j,k}}{L_{n,i}}, 1 \right\} I_{\{a_{j,k}>0\}} \Bigg] \\
& + I_{\{a_{n,i}=0\}} \left[ \frac{L_{n,i}}{F_n^l} \left( \sum_{j=1}^{k_n} \min \left\{ \frac{L_{n,j}}{L_{n,i}}, 1 \right\} I_{\{a_{n,j}=0\}} - 1 \right) \right. \\
& \left. + \frac{L_{n,i}}{F_n^l} \sum_{j=1}^{k_n} \min \left\{ \frac{L_{n,j}}{L_{n,i}}, 1 \right\} I_{\{a_{n,j}=0\}} \right]. \quad (14)
\end{aligned}$$

Note that for user  $n$ 's task  $i$ , when it chooses edge computing, its utility function is equal to the first six terms in (14), and when it chooses local computing, its utility function is equal to the last three terms in (14).

In the potential game  $\Gamma_n$ , let the potential function  $\Phi_n$  coincide with user  $n$ 's cost function  $T_n$ , and user  $n$ 's task  $i$ 's utility function  $U_{n,i}$  in (14) is derived from  $T_n$ . Then, the potential game  $\Gamma_n$  is expressed as

$$(\Gamma_n) : \min_{a_{n,i} \in \mathcal{S}_{n,i}} U_{n,i}(a_{n,i}, a_{n,-i}, \mathbf{a}_{-n}), \forall i \in \mathcal{K}_n. \quad (15)$$

We have the theorem as follows.

**Theorem 1:** In potential game  $\Gamma_n$ , when the utility function  $U_{n,i}$  of user  $n$ 's task  $i$  is given in (14), the potential function of the game coincides with the cost function  $T_n$  of user  $n$ .

*Proof:* According to the definition of potential games, to prove Theorem 1, we show that  $U_{n,i}(a'_{n,i}, a_{n,-i}, \mathbf{a}_{-n}) < U_{n,i}(a_{n,i}, a_{n,-i}, \mathbf{a}_{-n})$  implies  $T_n(a'_{n,i}, a_{n,-i}, \mathbf{a}_{-n}) < T_n(a_{n,i}, a_{n,-i}, \mathbf{a}_{-n})$ . To facilitate the analysis, we next consider three situations: a)  $a_{n,i} > 0$  and  $a'_{n,i} > 0$ ; b)  $a_{n,i} > 0$  and  $a'_{n,i} = 0$ ; c)  $a_{n,i} = 0$  and  $a'_{n,i} > 0$ .

First, we note the fact that  $A \min\{\frac{B}{A}, 1\} = B \min\{\frac{A}{B}, 1\}$  where  $A$  and  $B$  are both positive numbers, which is used in the following derivations.

Here let  $d_{j,k,n,i} = \min\{\frac{D_{j,k}}{D_{n,i}}, 1\} I_{\{a_{j,k}=a_{n,i}\}}$ ,  $d'_{j,k,n,i} = \min\{\frac{D_{j,k}}{D_{n,i}}, 1\} I_{\{a_{j,k}=a'_{n,i}\}}$  and  $l_{j,k,n,i} = \min\{\frac{L_{j,k}}{L_{n,i}}, 1\} I_{\{a_{j,k}>0\}}$  for ease of presentation.

a) according to (14), we know that the condition  $U_{n,i}(a'_{n,i}, a_{n,-i}, \mathbf{a}_{-n}) < U_{n,i}(a_{n,i}, a_{n,-i}, \mathbf{a}_{-n})$  implies that

$$\begin{aligned}
& \frac{D_{n,i}}{B_{a_{n,i}}} \left( \sum_{k=1}^{k_n} d_{n,k,n,i} - 1 + \sum_{j=1}^N \sum_{k=1}^{k_j} d_{j,k,n,i} \right) \\
& - \frac{D_{n,i}}{B_{a'_{n,i}}} \left( \sum_{k=1}^{k_n} d'_{n,k,n,i} + \sum_{j=1}^N \sum_{k=1}^{k_j} d'_{j,k,n,i} + 1 \right) > 0. \quad (16)
\end{aligned}$$

According to (10) and (16), we can derive that

$$\begin{aligned}
& T_n(a_{n,i}, a_{n,-i}, \mathbf{a}_{-n}) - T_n(a'_{n,i}, a_{n,-i}, \mathbf{a}_{-n}) \\
& = \frac{1}{k_n} \left[ \frac{D_{n,i}}{B_{a_{n,i}}} \left( \sum_{k=1}^{k_n} d_{n,k,n,i} - 1 + \sum_{j=1}^N \sum_{k=1}^{k_j} d_{j,k,n,i} \right) \right.
\end{aligned}$$

$$\left. - \frac{D_{n,i}}{B_{a'_{n,i}}} \left( \sum_{k=1}^{k_n} d'_{n,k,n,i} + \sum_{j=1}^N \sum_{k=1}^{k_j} d'_{j,k,n,i} + 1 \right) \right] > 0. \quad (17)$$

b) since  $a_{n,i} > 0$ ,  $a'_{n,i} = 0$ , and  $U_{n,i}(a'_{n,i}, a_{n,-i}, \mathbf{a}_{-n}) < U_{n,i}(a_{n,i}, a_{n,-i}, \mathbf{a}_{-n})$ , according to (14), we know that

$$\begin{aligned}
& \frac{D_{n,i}}{B_{a_{n,i}}} \left( \sum_{k=1}^{k_n} d_{n,k,n,i} - 1 + \sum_{j=1}^N \sum_{k=1}^{k_j} d_{j,k,n,i} \right) \\
& + \frac{L_{n,i}}{F^c} \left( \sum_{k=1}^{k_n} l_{n,k,n,i} - 1 + \sum_{j=1}^N \sum_{k=1}^{k_j} l_{j,k,n,i} \right) \\
& > \frac{L_{n,i}}{F_n^l} \left( 2 \sum_{j=1}^{k_n} \min \left\{ \frac{L_{n,j}}{L_{n,i}}, 1 \right\} I_{\{a_{n,j}=0\}} + 1 \right). \quad (18)
\end{aligned}$$

According to (10) and (18), we can derive that

$$\begin{aligned}
& T_n(a_{n,i}, a_{n,-i}, \mathbf{a}_{-n}) - T_n(a'_{n,i}, a_{n,-i}, \mathbf{a}_{-n}) \\
& = \frac{1}{k_n} \left[ \frac{D_{n,i}}{B_{a_{n,i}}} \left( \sum_{k=1}^{k_n} d_{n,k,n,i} - 1 + \sum_{j=1}^N \sum_{k=1}^{k_j} d_{j,k,n,i} \right) \right. \\
& + \frac{L_{n,i}}{F^c} \left( \sum_{k=1}^{k_n} l_{n,k,n,i} - 1 + \sum_{j=1}^N \sum_{k=1}^{k_j} l_{j,k,n,i} \right) \\
& \left. - \frac{L_{n,i}}{F_n^l} \left( 2 \sum_{j=1}^{k_n} \min \left\{ \frac{L_{n,j}}{L_{n,i}}, 1 \right\} I_{\{a_{n,j}=0\}} + 1 \right) \right] > 0. \quad (19)
\end{aligned}$$

c) when  $a_{n,i} = 0$  and  $a'_{n,i} > 0$ , we can also show that  $U_{n,i}(a'_{n,i}, a_{n,-i}, \mathbf{a}_{-n}) < U_{n,i}(a_{n,i}, a_{n,-i}, \mathbf{a}_{-n})$  implies  $T_n(a'_{n,i}, a_{n,-i}, \mathbf{a}_{-n}) < T_n(a_{n,i}, a_{n,-i}, \mathbf{a}_{-n})$  by a similar analysis as in b).

Combining the results in the three situations above, we now have shown that  $U_{n,i}(a'_{n,i}, a_{n,-i}, \mathbf{a}_{-n}) < U_{n,i}(a_{n,i}, a_{n,-i}, \mathbf{a}_{-n})$  implies  $T_n(a'_{n,i}, a_{n,-i}, \mathbf{a}_{-n}) < T_n(a_{n,i}, a_{n,-i}, \mathbf{a}_{-n})$ . Then, according to the definition of potential games, given user  $n$ 's task  $i$ 's utility function  $U_{n,i}$  in (14), the game  $\Gamma_n$  is the potential game together with its potential function coinciding with the cost function  $T_n$  of user  $n$ , and Theorem 1 gets proved.  $\square$

The solution of the potential game  $\Gamma_n$  is Nash equilibrium. Monderer and Shapley have proven that in potential games, the corresponding potential function is minimized either locally or globally at Nash equilibrium [43]. According to Theorem 1, we know the potential function of game  $\Gamma_n$  coincides with user  $n$ 's cost function  $T_n$ . Hence, we conclude that  $T_n$  is minimized either locally or globally at Nash equilibrium of  $\Gamma_n$ . Then, we can achieve the goal of optimizing  $T_n$  by optimizing  $U_{n,i}$  for each task  $i$ , finally finding an approximate solution in polynomial time.

Based on previous analysis, we re-formulate the multi-task offloading problem as the game  $\Gamma = (\mathcal{T}, \{\mathcal{S}_{n,i}\}_{n \in \mathcal{N}, i \in \mathcal{K}_n},$



$\{U_{n,i}(a_{n,i}, a_{-(n,i)})\}_{n \in \mathcal{N}, i \in \mathcal{K}_n}$ , where  $\mathcal{T} = \{(n,i) | n \in \mathcal{N}, i \in \mathcal{K}_n\}$  represents the set of all tasks of all users,  $\mathcal{S}_{n,i}$  represents user  $n$ 's task  $i$ 's strategy space, and  $U_{n,i}(a_{n,i}, a_{-(n,i)})$  in (14) represents the utility function of user  $n$ 's task  $i$ . Then, the game  $\Gamma$  is expressed as

$$(\Gamma) : \min_{a_{n,i} \in \mathcal{S}_{n,i}} U_{n,i}(a_{n,i}, a_{-(n,i)}), \forall n \in \mathcal{N}, i \in \mathcal{K}_n. \quad (20)$$

Here  $a_{-(n,i)}$  denotes the decisions of all tasks of all users except user  $n$ 's task  $i$ . Note that  $\mathbf{a} = (a_{n,i}, a_{-(n,i)})$ . Here the game  $\Gamma$  is an approximate version of the original game  $\Gamma_0$ , and the solution of  $\Gamma$  is also an approximate solution in polynomial time.

We next consider Nash equilibrium as the solution of the computation offloading game  $\Gamma$ . Specifically, in game  $\Gamma$ , a decision profile  $\mathbf{a}^* = (a_{1,1}^*, \dots, a_{1,k_1}^*, a_{2,1}^*, \dots, a_{2,k_2}^*, \dots, a_{N,1}^*, \dots, a_{N,k_N}^*)$  is regarded as the Nash equilibrium when no task of any user can unilaterally change its decision at  $\mathbf{a}^*$  to decrease its own utility function, and this is expressed as

$$\begin{aligned} U_{n,i}(a_{n,i}, a_{-(n,i)}^*) &\geq U_{n,i}(a_{n,i}^*, a_{-(n,i)}^*), \\ \forall n \in \mathcal{N}, i \in \mathcal{K}_n, a_{n,i} &\in \mathcal{S}_{n,i}. \end{aligned} \quad (21)$$

### B. Existence of Nash Equilibrium

Here let's discuss the existence of Nash equilibrium of game  $\Gamma$ . As is known, there always exists the Nash equilibrium in potential games [43]. Therefore, Nash equilibrium of  $\Gamma$  can be proved to exist by showing that  $\Gamma$  belongs to potential games.

In order to prove that game  $\Gamma$  belongs to potential games, we construct its corresponding potential function as

$$\begin{aligned} \Phi(\mathbf{a}) &= \sum_{i=1}^N \sum_{j=1}^{k_i} \left( \frac{1}{2} C_{i,j}^c(\mathbf{a}) I_{\{a_{i,j} > 0\}} + C_{i,j}^l(\mathbf{a}) I_{\{a_{i,j} = 0\}} \right) \\ &+ \frac{1}{2} \sum_{i=1}^N \sum_{j=1}^{k_i} \left( \frac{D_{i,j} I_{\{a_{i,j} > 0\}}}{B_{a_{i,j}}} \sum_{k=1}^{k_i} \min \left\{ \frac{D_{i,k}}{D_{i,j}}, 1 \right\} I_{\{a_{i,k} = a_{i,j}\}} \right) \\ &+ \frac{1}{2} \sum_{i=1}^N \sum_{j=1}^{k_i} \left( \frac{L_{i,j} I_{\{a_{i,j} > 0\}}}{F^c} \sum_{k=1}^{k_i} \min \left\{ \frac{L_{i,k}}{L_{i,j}}, 1 \right\} I_{\{a_{i,k} > 0\}} \right). \end{aligned} \quad (22)$$

**Theorem 2:** Given the potential function in (22), the computation offloading game  $\Gamma$  belongs to potential games.

*Proof:* Based on the definition of potential games, Theorem 2 gets proved if we show that  $U_{n,i}(a_{n,i}, a_{-(n,i)}) > U_{n,i}(a'_{n,i}, a_{-(n,i)})$  implies  $\Phi(a_{n,i}, a_{-(n,i)}) > \Phi(a'_{n,i}, a_{-(n,i)})$ . To facilitate the analysis, we next consider three situations: a)  $a_{n,i} > 0$  and  $a'_{n,i} > 0$ ; b)  $a_{n,i} > 0$  and  $a'_{n,i} = 0$ ; c)  $a_{n,i} = 0$  and  $a'_{n,i} > 0$ .

a) based on (14), we know the condition  $U_{n,i}(a_{n,i}, a_{-(n,i)}) > U_{n,i}(a'_{n,i}, a_{-(n,i)})$  implies that

$$\frac{D_{n,i}}{B_{a_{n,i}}} \left( \sum_{k=1}^{k_n} d_{n,k,n,i} - 1 + \sum_{j=1}^N \sum_{k=1}^{k_j} d_{j,k,n,i} \right) - \frac{D_{n,i}}{B_{a'_{n,i}}} \left( \sum_{k=1}^{k_n} d'_{n,k,n,i} + \sum_{j=1}^N \sum_{k=1}^{k_j} d'_{j,k,n,i} + 1 \right) > 0. \quad (23)$$

According to (22) and (23), we can derive that

$$\begin{aligned} &\Phi(a_{n,i}, a_{-(n,i)}) - \Phi(a'_{n,i}, a_{-(n,i)}) \\ &= \frac{1}{2} \left[ \frac{D_{n,i}}{B_{a_{n,i}}} \left( \sum_{j=1}^N \sum_{k=1}^{k_j} d_{j,k,n,i} - 1 \right) + \frac{D_{n,i}}{B_{a_{n,i}}} \sum_{j=1}^N \sum_{k=1}^{k_j} d_{j,k,n,i} \right. \\ &\quad \left. - \frac{D_{n,i}}{B_{a'_{n,i}}} \sum_{j=1}^N \sum_{k=1}^{k_j} d'_{j,k,n,i} - \frac{D_{n,i}}{B_{a'_{n,i}}} \left( \sum_{j=1}^N \sum_{k=1}^{k_j} d'_{j,k,n,i} + 1 \right) \right] \\ &\quad + \frac{1}{2} \left[ \frac{D_{n,i}}{B_{a_{n,i}}} \left( \sum_{k=1}^{k_n} d_{n,k,n,i} - 1 \right) + \frac{D_{n,i}}{B_{a_{n,i}}} \sum_{k=1}^{k_n} d_{n,k,n,i} \right. \\ &\quad \left. - \frac{D_{n,i}}{B_{a'_{n,i}}} \sum_{k=1}^{k_n} d'_{n,k,n,i} - \frac{D_{n,i}}{B_{a'_{n,i}}} \left( \sum_{k=1}^{k_n} d'_{n,k,n,i} + 1 \right) \right] \\ &= \frac{D_{n,i}}{B_{a_{n,i}}} \left( \sum_{k=1}^{k_n} d_{n,k,n,i} - 1 + \sum_{j=1}^N \sum_{k=1}^{k_j} d_{j,k,n,i} \right) \\ &\quad - \frac{D_{n,i}}{B_{a'_{n,i}}} \left( \sum_{k=1}^{k_n} d'_{n,k,n,i} + \sum_{j=1}^N \sum_{k=1}^{k_j} d'_{j,k,n,i} + 1 \right) > 0. \end{aligned} \quad (24)$$

b) since  $a_{n,i} > 0$ ,  $a'_{n,i} = 0$ , and  $U_{n,i}(a_{n,i}, a_{-(n,i)}) > U_{n,i}(a'_{n,i}, a_{-(n,i)})$ , according to (14), we know

$$\begin{aligned} &\frac{D_{n,i}}{B_{a_{n,i}}} \left( \sum_{k=1}^{k_n} d_{n,k,n,i} - 1 + \sum_{j=1}^N \sum_{k=1}^{k_j} d_{j,k,n,i} \right) \\ &\quad + \frac{L_{n,i}}{F^c} \left( \sum_{k=1}^{k_n} l_{n,k,n,i} - 1 + \sum_{j=1}^N \sum_{k=1}^{k_j} l_{j,k,n,i} \right) \\ &> \frac{L_{n,i}}{F_n^l} \left( 2 \sum_{j=1}^{k_n} \min \left\{ \frac{L_{n,j}}{L_{n,i}}, 1 \right\} I_{\{a_{n,j} = 0\}} + 1 \right). \end{aligned} \quad (25)$$

According to (22) and (25), we can derive that

$$\begin{aligned} &\Phi(a_{n,i}, a_{-(n,i)}) - \Phi(a'_{n,i}, a_{-(n,i)}) \\ &= \frac{1}{2} \left[ \frac{D_{n,i}}{B_{a_{n,i}}} \sum_{j=1}^N \sum_{k=1}^{k_j} d_{j,k,n,i} + \frac{D_{n,i}}{B_{a_{n,i}}} \left( \sum_{j=1}^N \sum_{k=1}^{k_j} d_{j,k,n,i} - 1 \right) \right. \\ &\quad \left. + \frac{L_{n,i}}{F^c} \sum_{j=1}^N \sum_{k=1}^{k_j} l_{j,k,n,i} + \frac{L_{n,i}}{F^c} \left( \sum_{j=1}^N \sum_{k=1}^{k_j} l_{j,k,n,i} - 1 \right) \right] \\ &\quad - \frac{L_{n,i}}{F_n^l} \left( 2 \sum_{j=1}^{k_n} \min \left\{ \frac{L_{n,j}}{L_{n,i}}, 1 \right\} I_{\{a_{n,j} = 0\}} + 1 \right) \\ &\quad + \frac{1}{2} \left[ \frac{D_{n,i}}{B_{a_{n,i}}} \sum_{k=1}^{k_n} d_{n,k,n,i} + \frac{D_{n,i}}{B_{a_{n,i}}} \left( \sum_{k=1}^{k_n} d_{n,k,n,i} - 1 \right) \right] \end{aligned}$$



$$\begin{aligned}
& + \frac{1}{2} \left[ \frac{L_{n,i}}{F^c} \sum_{k=1}^{k_n} l_{n,k,n,i} + \frac{L_{n,i}}{F^c} \left( \sum_{k=1}^{k_n} l_{n,k,n,i} - 1 \right) \right] \\
& = \frac{D_{n,i}}{B_{a_{n,i}}} \left( \sum_{k=1}^{k_n} d_{n,k,n,i} - 1 + \sum_{j=1}^N \sum_{k=1}^{k_j} d_{j,k,n,i} \right) \\
& + \frac{L_{n,i}}{F^c} \left( \sum_{k=1}^{k_n} l_{n,k,n,i} - 1 + \sum_{j=1}^N \sum_{k=1}^{k_j} l_{j,k,n,i} \right) \\
& - \frac{L_{n,i}}{F_n^l} \left( 2 \sum_{j=1}^{k_n} \min \left\{ \frac{L_{n,j}}{L_{n,i}}, 1 \right\} I_{\{a_{n,j}=0\}} + 1 \right) > 0. \quad (26)
\end{aligned}$$

c) when  $a_{n,i} = 0$  and  $a'_{n,i} > 0$ , we can also show that  $U_{n,i}(a_{n,i}, a_{-(n,i)}) > U_{n,i}(a'_{n,i}, a_{-(n,i)})$  implies  $\Phi(a_{n,i}, a_{-(n,i)}) > \Phi(a'_{n,i}, a_{-(n,i)})$  by a similar analysis as in b).

Combining the results above, we now have proved the game  $\Gamma$  belongs to potential games.  $\square$

According to Theorem 2, the game  $\Gamma$  belongs to potential games. Hence, Nash equilibrium of  $\Gamma$  is proven to exist. At  $\Gamma$ 's Nash equilibrium  $\mathbf{a}^*$ , the strategy  $\mathbf{a}_n^*$  of any user  $n$ , which is the subset of  $\mathbf{a}^*$ , is the Nash equilibrium of the corresponding potential game  $\Gamma_n$ , which minimizes the cost function  $T_n$  either locally or globally and is an approximate solution in polynomial time for the original multi-task computation offloading problem. Then, based on the formulated game  $\Gamma$ , we next propose an efficient algorithm to find  $\mathbf{a}^*$ .

## V. COMPUTATION OFFLOADING ALGORITHM

In this section, we put forward the multi-task offloading solution in polynomial time via a Nash equilibrium of the game  $\Gamma$ . In order to achieve the Nash equilibrium, we propose ECO-GAME, an Efficient multi-task Computation Offloading algorithm based on the formulated GAME, as shown in Algorithm 1. We next give the computational complexity analysis about the proposed algorithm. We further conduct the performance evaluation on the proposed algorithm by utilizing the metric of price of anarchy.

### A. Algorithm Design

We next show the algorithm design in detail in this subsection. As is known, there is a significant property of finite improvement for potential games [43]. And we have already proved the game  $\Gamma$  belongs to potential games. Therefore based on the facts above, let no more than one user improve one task's decision in an iteration and the iteration process is guaranteed to achieve the Nash equilibrium after finite times, which is the core of the ECO-GAME algorithm. In ECO-GAME, the iteration process is completed in one time slot, which is carried out synchronously by all users in parallel. Within the time slot  $t$ , the ECO-GAME algorithm includes the following two phases:

a) Collecting task offloading cost: based on the decision profile  $\mathbf{a}(t)$  at slot  $t$ , the edge servers can calculate the task offloading cost  $C_{n,i}^c(m, a_{-(n,i)}(t))$  for user  $n$ 's task  $i$  when the task chooses to offload via the channel  $m \in \mathcal{M}$  and the

---

### Algorithm 1: Efficient Multi-Task Computation Offloading Algorithm (ECO-GAME).

---

1: **Initialization:**

2: Let  $t = 0$

3: Make decision  $a_{n,i}(t) = 0$  of each user  $n$ 's task  $i$

4: For all users in parallel:

5: **repeat**

6: **for all**  $i$  in  $\mathcal{K}_n$  **do**

7: Collect the task offloading cost  $C_{n,i}^c(m, a_{-(n,i)}(t))$  for choosing each channel  $m \in \mathcal{M}$  from edge servers

8: Calculate  $\Delta_n(t)$  (set of task decision update) using (27)

9: **if**  $\Delta_n(t) \neq \emptyset$  **then**

10: Transmit the request for updating task decision to edge servers

11: **if** Receive the approval from edge servers **then**

12: Compute the best task decision update  $(i^*, a^*)$  from  $\Delta_n(t)$  using (28)

13: Update decision  $a_{n,i^*}(t+1) = a^*$

14: Let  $a_{n,i}(t+1) = a_{n,i}(t)$  for each of the other tasks except task  $i^*$

15: Send  $(i^*, a^*)$  to edge servers to update  $\mathbf{a}(t+1)$  for next slot

16: **else**

17: Let  $a_{n,i}(t+1) = a_{n,i}(t)$  for next slot for each of the tasks

18: Update slot  $t = t + 1$

19: **until** Receive the end message from edge servers

---

decisions of other tasks are given in  $\mathbf{a}(t)$ , and report it to user  $n$ . Here in Lines 6-7 of Algorithm 1, each user  $n$  collects the task offloading cost  $C_{n,i}^c(m, a_{-(n,i)}(t))$  for its task  $i \in \mathcal{K}_n$  when the task chooses to offload via each channel  $m \in \mathcal{M}$ , from the edge servers.

b) Updating task decision: here allow only one user to update its one task's current decision, as in Lines 8-17 of Algorithm 1. According to the task offloading cost (i.e.,  $\{C_{n,i}^c(m, a_{-(n,i)}(t)), i \in \mathcal{K}_n, m \in \mathcal{M}\}$ ) collected in phase a), each user  $n$  calculates the set of task decision update as

$$\Delta_n(t) \triangleq \{(i, a') \mid i \in \mathcal{K}_n,$$

$$a' = \arg \min_{a \in \mathcal{S}_{n,i}} U_{n,i}(a, a_{-(n,i)}(t)) \wedge$$

$$U_{n,i}(a', a_{-(n,i)}(t)) < U_{n,i}(a_{n,i}(t), a_{-(n,i)}(t))\}. \quad (27)$$

Based on the collected information, in the calculation of (27), each user  $n$  does not need to know the related information of the tasks of other users, such as data size, which ensures privacy and confidentiality. The condition  $\Delta_n(t) \neq \emptyset$  implies that user  $n$  expects to improve its one task's decision such that its cost gets reduced. In this case, user  $n$  transmits the request for updating task decision to the edge servers. Otherwise, user  $n$  transmits nothing. Then, the edge servers transmit the approval to one user that is randomly selected from a set of users that

have transmitted requests. User  $k$  chooses the best task decision update  $(i^*, a^*)$  from  $\Delta_k(t)$  once it receives the approval. Here  $(i^*, a^*)$  is computed as

$$i^*, a^* = \arg \max_{i, a \in \Delta_k(t)} \left( U_{k,i}(a_{k,i}(t), a_{-(k,i)}(t)) - U_{k,i}(a, a_{-(k,i)}(t)) \right). \quad (28)$$

Then user  $k$  will update the decision of task  $i^*$  as  $a_{k,i^*}(t+1) = a^*$ , and keep the other tasks' decisions unchanged as  $a_{k,i}(t+1) = a_{k,i}(t)$  at next slot, where  $i \neq i^*$ . Then, user  $k$  sends the task decision update  $(i^*, a^*)$  to edge servers to update the decision profile  $\mathbf{a}(t+1)$  at next slot. Users without receiving the approval keep the tasks' decisions unchanged as  $a_{n,i}(t+1) = a_{n,i}(t)$  at next slot.

According to Theorem 2, ECO-GAME converges to the Nash equilibrium of game  $\Gamma$  within finite slots. The edge servers transmit the end message to all users when requests are no longer received in a time slot, and ECO-GAME terminates when the message is received by all users. Finally the decisions of users' tasks in the last time slot are taken as their final decisions.

### B. Computational Complexity Analysis

We next give the computational complexity analysis about the ECO-GAME algorithm in this subsection. Within a time slot, all users in parallel perform the operations in Lines 6-17 of ECO-GAME, the majority of which consist of some basic mathematical computations. Here we care about the calculations of the task decision update in Line 8 and the best task decision update in Line 12. Let  $K_{\max} = \max_{i \in \mathcal{N}} \{k_i\}$ . In calculation of  $\Delta_n(t)$  in (27), for each task  $i$  in  $\mathcal{K}_n$ , user  $n$  first calculates  $U_{n,i}(a, a_{-(n,i)}(t))$  for all  $a$  in  $\mathcal{S}_{n,i}$  with a complexity of  $\mathcal{O}(2(K_{\max} - 1))$ , and then finds the minimum of  $M + 1$  values of  $U_{n,i}$  with a complexity of  $\mathcal{O}(M + 1)$ , and finally a comparison is needed with a complexity of  $\mathcal{O}(1)$ . Hence, the complexity of calculating  $\Delta_n(t)$  is  $\mathcal{O}(K_{\max}(2K_{\max} + M))$ . The calculation of  $(i^*, a^*)$  in (28) involves the maximum operation over at most  $K_{\max}$  elements with a complexity of  $\mathcal{O}(K_{\max})$ . Hence, within a single time slot, the computational complexity is  $\mathcal{O}(K_{\max}(2K_{\max} + M + 1))$ . Here we can see that, the larger the number of each user's tasks as well as number of wireless channels, the higher the computational complexity. Denote  $C$  as the number of time slots required for ECO-GAME to terminate. Thus the computational complexity in total is  $\mathcal{O}(CK_{\max}(2K_{\max} + M + 1))$ . Let  $F_{\max}^l = \max_{i \in \mathcal{N}} \{F_i^l\}$ ,  $B_{\max} = \max_{m \in \mathcal{M}} \{B_m\}$ ,  $B_{\min} = \min_{m \in \mathcal{M}} \{B_m\}$ ,  $K = \sum_{i=1}^N k_i$  and  $y = \min\{\frac{1}{B_{\max}^2}, \frac{1}{B_{\max} F^c F_{\max}^l}\}$ . We have the following result.

**Theorem 3:** When  $D_{n,i}$ ,  $L_{n,i}$ ,  $B_m$ ,  $F^c$  and  $F_n^l$  are integers for any  $n \in \mathcal{N}$ ,  $i \in \mathcal{K}_n$ ,  $m \in \mathcal{M}$ , the number of time slots  $C$  for convergence satisfies that

$$C \leq \left\lceil \sum_{i=1}^N \sum_{j=1}^{k_i} \max \left\{ \frac{k_i L_{i,j}}{F_i^l}, \frac{K}{2} \left( \frac{D_{i,j}}{B_{\min}} + \frac{L_{i,j}}{F^c} \right) \right\} \right\rceil$$

$$+ \frac{1}{2} \sum_{i=1}^N \sum_{j=1}^{k_i} \frac{D_{i,j} k_i}{B_{\min}} + \frac{1}{2} \sum_{i=1}^N \sum_{j=1}^{k_i} \frac{L_{i,j} k_i}{F^c} - \sum_{i=1}^N \sum_{j=1}^{k_i} \min \left\{ \frac{L_{i,j}}{F_i^l}, \frac{1}{2} \left( \frac{D_{i,j}}{B_{\max}} + \frac{L_{i,j}}{F^c} \right) \right\} \right\rceil / y. \quad (29)$$

*Proof:* According to (22), we know that

$$\begin{aligned} & \sum_{i=1}^N \sum_{j=1}^{k_i} \min \left\{ \frac{L_{i,j}}{F_i^l}, \frac{1}{2} \left( \frac{D_{i,j}}{B_{\max}} + \frac{L_{i,j}}{F^c} \right) \right\} \leq \Phi(\mathbf{a}) \\ & \leq \sum_{i=1}^N \sum_{j=1}^{k_i} \max \left\{ \frac{k_i L_{i,j}}{F_i^l}, \frac{K}{2} \left( \frac{D_{i,j}}{B_{\min}} + \frac{L_{i,j}}{F^c} \right) \right\} \\ & + \frac{1}{2} \sum_{i=1}^N \sum_{j=1}^{k_i} \frac{D_{i,j} k_i}{B_{\min}} + \frac{1}{2} \sum_{i=1}^N \sum_{j=1}^{k_i} \frac{L_{i,j} k_i}{F^c}. \end{aligned} \quad (30)$$

Within a time slot, assume user  $n$  improves the task  $i$ 's decision  $a_{n,i}$  to  $a'_{n,i}$  to reduce user  $n$ 's cost. We next show this results in a reduction of the corresponding potential function by at least  $y$ , namely,

$$\Phi(a_{n,i}, a_{-(n,i)}) \geq \Phi(a'_{n,i}, a_{-(n,i)}) + y. \quad (31)$$

Based on the proof of Theorem 2, we consider three situations:

a)  $a_{n,i} > 0$  and  $a'_{n,i} > 0$ ; b)  $a_{n,i} > 0$  and  $a'_{n,i} = 0$ ; c)  $a_{n,i} = 0$  and  $a'_{n,i} > 0$ .

- since  $D_{n,i}$  and  $B_m$  are integers for any  $n \in \mathcal{N}$ ,  $i \in \mathcal{K}_n$ ,  $m \in \mathcal{M}$ , according to (24), we have  $\Phi(a_{n,i}, a_{-(n,i)}) - \Phi(a'_{n,i}, a_{-(n,i)}) \geq \frac{1}{B_{\max}^2} \geq y$ . Here we omit the derivation due to the space limitation.
- since  $D_{n,i}$ ,  $L_{n,i}$ ,  $B_m$ ,  $F^c$  and  $F_n^l$  are integers for any  $n \in \mathcal{N}$ ,  $i \in \mathcal{K}_n$ ,  $m \in \mathcal{M}$ , according to (26), we then have  $\Phi(a_{n,i}, a_{-(n,i)}) - \Phi(a'_{n,i}, a_{-(n,i)}) \geq \frac{1}{B_{\max} F^c F_{\max}^l} \geq y$ . Here we omit the derivation due to the space limitation.
- we also have  $\Phi(a_{n,i}, a_{-(n,i)}) - \Phi(a'_{n,i}, a_{-(n,i)}) \geq y$  by a similar analysis as in b).

Therefore, according to (30) and (31), we can calculate the upper bound of  $C$  as in (29).  $\square$

### C. Price of Anarchy

So far, we have proposed the ECO-GAME algorithm to achieve the Nash equilibrium solution. Now we conduct the performance evaluation on the ECO-GAME algorithm by utilizing the metric of price of anarchy (PoA) [44], which is commonly used in game theory. Specifically, PoA is used to quantify the performance gap between the approximate solution obtained by the ECO-GAME algorithm and the optimal solution, from the perspective of the system cost [35] which denotes the total cost of all the users.

Here the PoA is defined as the ratio of the system cost between the worst Nash equilibrium and optimal solution  $\mathbf{a}^*$ , which is denoted as

$$PoA = \frac{\max_{\mathbf{a} \in \mathcal{A}} \sum_{n \in \mathcal{N}} T_n(\mathbf{a})}{\sum_{n \in \mathcal{N}} T_n(\mathbf{a}^*)}. \quad (32)$$

Here  $\mathcal{A}$  denotes the set of Nash equilibria of game  $\Gamma$ . The smaller PoA means the approximate solution obtained by the algorithm has a better performance. Let  $C_{i,j,\max} = \max\{\frac{L_{i,j}}{F_i^l} \sum_{k=1}^{k_i} \min\{\frac{L_{i,k}}{L_{i,j}}, 1\}, \frac{D_{i,j}(K-M+1)}{B_{\min}} + \frac{L_{i,j}K}{F^c}\}$ ,  $C_{i,j,\min} = \min\{\frac{L_{i,j}}{F_i^l}, \frac{D_{i,j}}{B_{\max}} + \frac{L_{i,j}}{F^c}\}$ , and  $C_{i,j,\text{ini}} = \frac{L_{i,j}}{F_i^l} \sum_{k=1}^{k_i} \min\{\frac{L_{i,k}}{L_{i,j}}, 1\}$ . We next provide the upper and lower bounds for the PoA.

**Theorem 4:** From the perspective of the system cost, PoA for the game  $\Gamma$  satisfies

$$\frac{\sum_{i=1}^N \frac{1}{k_i} \min\left\{\sum_{j=1}^{k_i} C_{i,j,\text{ini}}, \sum_{j=1}^{k_i} C_{i,j,\max}\right\}}{\sum_{i=1}^N \frac{1}{k_i} \sum_{j=1}^{k_i} C_{i,j,\min}} \geq \text{PoA} \geq 1. \quad (33)$$

*Proof:*  $\mathbf{a}^*$  and  $\mathbf{a} \in \mathcal{A}$  denote the optimal solution, and an arbitrary Nash equilibrium of  $\Gamma$  respectively. Obviously, we have  $\sum_{n \in \mathcal{N}} T_n(\mathbf{a}) \geq \sum_{n \in \mathcal{N}} T_n(\mathbf{a}^*)$  and then  $\text{PoA} \geq 1$ .

During the initialization of the algorithm, the decision  $a_{i,j}$  of each user  $i$ 's task  $j$  is 0 and the cost of user  $i$  is  $\frac{1}{k_i} \sum_{j=1}^{k_i} \frac{L_{i,j}}{F_i^l} \sum_{k=1}^{k_i} \min\{\frac{L_{i,k}}{L_{i,j}}, 1\} = \frac{1}{k_i} \sum_{j=1}^{k_i} C_{i,j,\text{ini}}$ . The cost of user  $i$  gets reduced with the algorithm. Thus, for an arbitrary Nash equilibrium  $\mathbf{a}$ , we have

$$T_i(\mathbf{a}) \leq \frac{1}{k_i} \sum_{j=1}^{k_i} C_{i,j,\text{ini}}. \quad (34)$$

For each user  $i$ 's task  $j$ , when  $a_{i,j} = 0$ , we have  $C_{i,j}(\mathbf{a}) \leq \frac{L_{i,j}}{F_i^l} \sum_{k=1}^{k_i} \min\{\frac{L_{i,k}}{L_{i,j}}, 1\}$ . Obviously the number of tasks that choose the channel with the bandwidth of  $B_{\min}$  is at most  $K - M + 1$ . When  $a_{i,j} > 0$ , we have  $C_{i,j}(\mathbf{a}) \leq \frac{D_{i,j}(K-M+1)}{B_{\min}} + \frac{L_{i,j}K}{F^c}$ . Therefore, for user  $i$ 's task  $j$ , we have  $C_{i,j}(\mathbf{a}) \leq \max\{\frac{L_{i,j}}{F_i^l} \sum_{k=1}^{k_i} \min\{\frac{L_{i,k}}{L_{i,j}}, 1\}, \frac{D_{i,j}(K-M+1)}{B_{\min}} + \frac{L_{i,j}K}{F^c}\} = C_{i,j,\max}$ . Thus, we have

$$T_i(\mathbf{a}) \leq \frac{1}{k_i} \sum_{j=1}^{k_i} C_{i,j,\max}. \quad (35)$$

According to (34) and (35), for any user  $i$  at  $\mathbf{a}$ , we have that

$$T_i(\mathbf{a}) \leq \frac{1}{k_i} \min\left\{\sum_{j=1}^{k_i} C_{i,j,\text{ini}}, \sum_{j=1}^{k_i} C_{i,j,\max}\right\}. \quad (36)$$

For each user  $i$ 's task  $j$  at  $\mathbf{a}^*$ , when  $a_{i,j}^* = 0$ , we have  $C_{i,j}(\mathbf{a}^*) \geq \frac{L_{i,j}}{F_i^l}$ . When  $a_{i,j}^* > 0$ , we have  $C_{i,j}(\mathbf{a}^*) \geq \frac{D_{i,j}}{B_{\max}} + \frac{L_{i,j}}{F^c}$ . Therefore, for user  $i$ 's task  $j$ , we have  $C_{i,j}(\mathbf{a}^*) \geq \min\{\frac{L_{i,j}}{F_i^l}, \frac{D_{i,j}}{B_{\max}} + \frac{L_{i,j}}{F^c}\} = C_{i,j,\min}$ . Thus, for any user  $i$  at  $\mathbf{a}^*$ , we have that

$$T_i(\mathbf{a}^*) \geq \frac{1}{k_i} \sum_{j=1}^{k_i} C_{i,j,\min}. \quad (37)$$

According to (36) and (37), we have that

$$\text{PoA} = \frac{\max_{\mathbf{a} \in \mathcal{A}} \sum_{n \in \mathcal{N}} T_n(\mathbf{a})}{\sum_{n \in \mathcal{N}} T_n(\mathbf{a}^*)}$$

TABLE II  
SIMULATION ENVIRONMENT

Component	Description
CPU	Intel Core i7-9700
RAM	16 GB
OS	Microsoft Windows 10
IDE	PyCharm 2020
PL	Python 3.8

TABLE III  
SIMULATION PARAMETERS

Parameter	Value
The user number $N$	30
The number of tasks $k_n$ on each user	[1, 10]
The number of the wireless channels $M$	5
The channel bandwidth $B_m$ for channel $m$	99.7 Mbps
User $n$ 's task $i$ 's data size $D_{n,i}$	[1, 500] Mb
User $n$ 's task $i$ 's number of CPU cycles $L_{n,i}$	[1, 50] Gcycles
The local computing capability $F_n^l$	[0.5, 1.0] GHz
The total computing capability of MEC servers $F^c$	100 GHz

$$\leq \frac{\sum_{i=1}^N \frac{1}{k_i} \min\left\{\sum_{j=1}^{k_i} C_{i,j,\text{ini}}, \sum_{j=1}^{k_i} C_{i,j,\max}\right\}}{\sum_{i=1}^N \frac{1}{k_i} \sum_{j=1}^{k_i} C_{i,j,\min}}. \quad (38)$$

□

## VI. NUMERICAL RESULTS

In this section, we implement numerous experiments to evaluate the ECO-GAME performance, which are conducted in the PyCharm IDE on the laptop with 3.00 GHz Intel Core i7-9700 CPU and 16 GB RAM. The specific components used in the simulation environment are shown in Table II. In our experiments, unless otherwise noted, the simulation parameter settings are as follows, as given in Table III. The parameters used in our experiments are real-world values obtained from the recent related works. Specifically, the coverage range of the wireless base station is 50 m, where there are  $N = 30$  users scattered randomly like [35]. The number of computation tasks generated by each user  $n$  is set to an integer taken from [1, 10] randomly and uniformly [16]. The number of the wireless channels is set to 5, that is,  $M = 5$  [32], [35]. We set the channel bandwidth  $B_m = 99.7$  Mbps for channel  $m$  [8]. User  $n$ 's computation task  $i$ 's data size  $D_{n,i}$  is drawn from [1, 500] Mb randomly and uniformly [30]. User  $n$ 's task  $i$ 's number of CPU cycles  $L_{n,i}$  is drawn from [1, 50] Gcycles randomly and uniformly [8]. According to [8], [20], the local computing capability  $F_n^l$  is drawn from a continuous uniform distribution on [0.5, 1.0] GHz. Like [20], the total computing capability of MEC servers  $F^c$  is set to 100 GHz.

To evaluate the performance of the Nash equilibrium achieved by the ECO-GAME algorithm in terms of computation cost, we compare ECO-GAME with three baselines: 1) *CCAT* (Cloud Computing by All Tasks): based on the communication and computing resource allocation schemes proposed in this paper, all users randomly choose the channels to offload all their tasks to edge servers for execution. 2) *LCAT* (Local Computing by All Tasks): each mobile device user performs all its tasks on

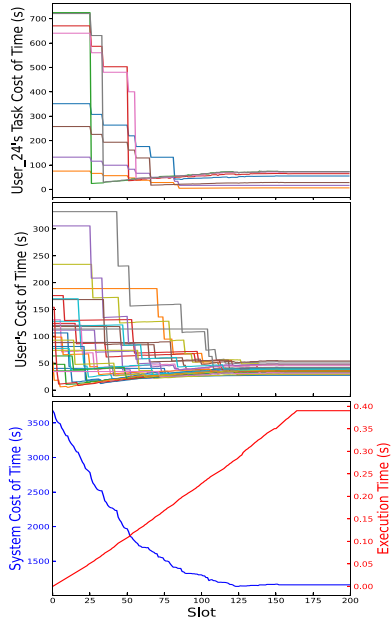


Fig. 1. Computation time cost of user\_24's 8 tasks in different slots (at the top), computation time cost of 30 users in different slots (in the middle), and system cost of time of 30 users in different slots and algorithm execution time (at the bottom).

its device. 3) *JPBR* [20]: it considers the case of constrained communication and cloud computing resources, and that both communication and cloud computing resources allocated to each task that chooses offloading remain unchanged within a computation offloading period. The results shown are the averages of 100 experiments, together with 95% confidence intervals.

A. Convergence of the ECO-GAME Algorithm

To verify the convergence of ECO-GAME, we apply the ECO-GAME algorithm and show the results in Fig. 1. Specifically, Fig. 1 shows the computation cost of user\_24's 8 tasks in different slots (at the top), in which we use user\_24 as the representative user and we can see each task's computation cost (i.e., latency) keeps decreasing and finally stabilizes. This demonstrates our ECO-GAME algorithm minimizes each user's cost while also reducing the computation cost of its each task, thus ensuring the user's QoS. Fig. 1 shows the computation cost of 30 users in different slots in the middle, in which the user's computation cost keeps decreasing on the whole and eventually stabilizes. Fig. 1 shows the system cost of 30 users in different slots and the algorithm execution time at the bottom, in which the system cost keeps decreasing and finally stabilizes. These results indicate that the computation cost converges to a fixed point of Nash equilibrium of the formulated game within finite slots, which verify the convergence of ECO-GAME. Besides, we can also see that the algorithm execution time increases linearly with the number of slots and finally stabilizes, which indicates ECO-GAME converges and terminates in a short time (less than 0.4 s), ensuring the real-time performance of ECO-GAME. This ensures the user obtains the optimal offloading strategy quickly, improving the user's quality of experience.

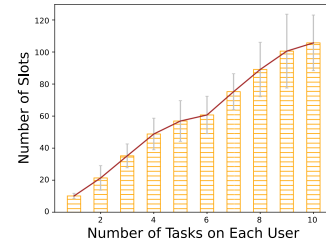


Fig. 2. Number of slots with various numbers of tasks on each user.

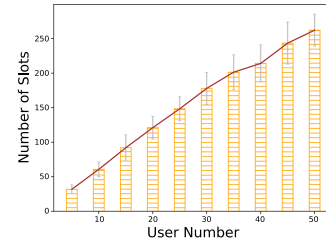


Fig. 3. Number of slots with various user numbers.

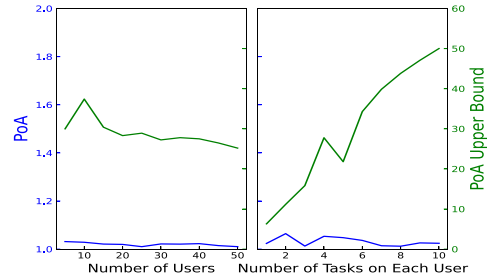


Fig. 4. PoA and PoA upper bound with different number of users (at the left), and with different number of tasks on each user (at the right).

Then we evaluate the convergence time of ECO-GAME in Fig. 2 with different number of tasks on each user (the user number  $N = 10$ ), and in Fig. 3 with different number of users. We can see that, when ECO-GAME converges, the number of slots grows nearly linearly with the number of tasks on each user, and the number of users, respectively. Therefore we conclude that ECO-GAME converges quickly and scales well as the number of tasks on each user increases and the number of users increases.

Further we evaluate the algorithm performance of ECO-GAME with the PoA and PoA upper bound under different number of users and under different number of tasks on each user respectively. And the results are shown in Fig. 4. We can see that the results show the PoA is close to 1 in all cases which means the performance of ECO-GAME is close to optimal. Besides, PoA is fairly insensitive to the number of users and the number of tasks on each user. The PoA upper bound is smaller in the case of a small number of tasks on each user, since in this case the ECO-GAME and the optimal solution prefer the task computing locally. In short, the small performance gap between the ECO-GAME and the optimal solution demonstrates ECO-GAME has a good performance.



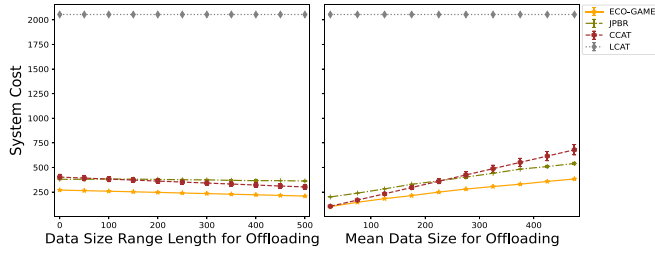


Fig. 5. System cost with various lengths of the data size range (at the left), where the mean data size is 250.5 Mb. System cost with various mean data sizes (at the right), where the data size range length is 50.

### B. Results With Different Data Sizes

To evaluate the performance of ECO-GAME with different lengths of the data size ( $D_{n,i}$ ) range, we first set the user number  $N = 10$  and conduct the experiments with various lengths of the data size range, and show the results in Fig. 5. Note that for different ranges, their mean values are set to be the same. Fig. 5 shows the system cost under various lengths of the data size range at the left, in which ECO-GAME can reduce cost by up to 42.3%, 32.8% and 89.8% and reduce average cost by 35.8%, 31.6% and 88.3% over JPBR, CCAT and LCAT, respectively. This is because ECO-GAME makes more efficient use of the constrained wireless communication resources. ECO-GAME performs much better than LCAT, which illustrates the benefits of computation offloading. Besides, ECO-GAME performs better than JPBR with different length. This is because that JPBR does not consider resources from the completed tasks are re-allocated to the uncompleted tasks, seriously wasting the limited resources, while ECO-GAME considers this case. When the length is small, CCAT performs worse than JPBR, however, CCAT performs better than JPBR as the length increases. This is because that we adopt the bandwidth resource allocation scheme proposed in this paper in CCAT, which re-allocates the bandwidth from the transmission-completed tasks to the transmission-uncompleted tasks. This illustrates the efficiency of the bandwidth resource allocation scheme. ECO-GAME performs better than CCAT. This is because ECO-GAME considers that when too many computation tasks compete for limited communication resources and cloud computing resources, some of the tasks will choose local computing, thereby reducing users' cost. While CCAT considers all tasks choose cloud computing even if it's more beneficial for some tasks to be performed locally.

To evaluate the performance of ECO-GAME with different mean data sizes, we first set the user number  $N = 10$  and run experiments with different mean data sizes, and show the results in Fig. 5. Note that for different mean data sizes, their corresponding range lengths are set to be the same. Fig. 5 shows the system cost under various mean data sizes at the right, in which ECO-GAME can reduce cost by up to 49.2%, 43.6% and 95.0% and reduce average cost by 34.1%, 29.3% and 87.6% over JPBR, CCAT and LCAT, respectively. The system cost by ECO-GAME, JPBR, and CCAT gets increased as the mean data size increases, because larger mean data size for offloading incurs

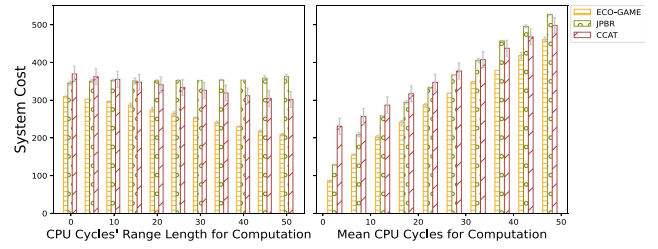


Fig. 6. System cost with various lengths of the CPU cycles' range (at the left), where the mean CPU cycles are 25.5 Gcycles. System cost with various mean CPU cycles (at the right), where the CPU cycles' range length is 5.

larger transmission time delay. Besides, CCAT performs better than JPBR when the mean data size is small, which indicates the efficiency of the resource allocation schemes proposed in this paper. When the mean data size is large, CCAT performs worse than JPBR. This is because a larger mean data size leads to larger transmission cost which seriously affects the performance of task computation offloading, and CCAT considers all tasks are offloaded for computation. ECO-GAME performs better than JPBR and CCAT with similar analysis as in last paragraph. Observe that LCAT performs much worse than ECO-GAME, JPBR, and CCAT. To show the experimental results more clearly, we remove LCAT in the following figures since it performs much worse than the other three schemes, according to our large scale experiment results in the following parts.

### C. Results With Different CPU Cycles

To evaluate the performance of ECO-GAME with different lengths of CPU cycles' range ( $L_{n,i}$ ), we first set the user number  $N = 10$  and conduct the experiments with various lengths of the CPU cycles' range at the same mean value, and show the results in Fig. 6. Fig. 6 shows the system cost under various lengths of the CPU cycles' range at the left, in which ECO-GAME can reduce cost by up to 42.3%, 30.8% and 89.8% and reduce average cost by 25.5%, 21.6% and 88.1% over JPBR, CCAT and LCAT, respectively. This is because ECO-GAME makes more efficient use of the constrained MEC computing resources. ECO-GAME performs better than CCAT and LCAT. Besides, ECO-GAME performs better than JPBR as the length increases. This is because that JPBR does not consider that the computing resources from the computation-completed tasks are re-allocated to the computation-uncompleted tasks, seriously wasting the limited computing resources, while ECO-GAME considers this case. When the length is small, CCAT performs worse than JPBR, however, CCAT performs better than JPBR as the length increases. This is because that we adopt the computing resource allocation scheme proposed in this paper in CCAT, which re-allocates the computing resources from the computation-completed tasks to the computation-uncompleted tasks. This illustrates the efficiency of the computing resource allocation scheme.

In order to evaluate the impact of the mean CPU cycles on ECO-GAME, we first set the user number  $N = 10$  and conduct the experiments with various mean CPU cycles with the same

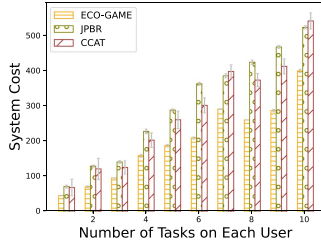


Fig. 7. System cost with various numbers of tasks on each user.

length of the CPU cycles' range, and show the results in Fig. 6. Fig. 6 shows the system cost under various mean CPU cycles at the right, in which ECO-GAME can reduce cost by up to 32.3%, 62.2% and 89.8% and reduce average cost by 18.3%, 23.3% and 84.2% over JPBR, CCAT and LCAT, respectively. The system cost by ECO-GAME, JPBR, CCAT, and LCAT gets increased as the mean CPU cycles increase, because larger mean CPU cycles for computation incur larger execution time. Besides, with different mean CPU cycles, ECO-GAME performs better than JPBR. This is because that ECO-GAME considers resources from the completed tasks are re-allocated to the uncompleted tasks, while JPBR does not consider this case. ECO-GAME performs better than CCAT. When the mean CPU cycles are large, CCAT performs better than JPBR because we adopt the computing resource allocation scheme proposed in this paper in CCAT. This illustrates the efficiency of the computing resource allocation scheme.

#### D. Results With Different Numbers of Tasks on Each User

To evaluate the performance of ECO-GAME with different number of tasks on each user, we first set the user number  $N = 10$  and run experiments with each user  $n$  having the number of  $k_n = 1, 2, 3, \dots, 10$  tasks respectively, and show the results in Fig. 7. Fig. 7 shows the system cost under various numbers of tasks on each user, in which ECO-GAME can reduce cost by up to 45.3%, 41.8%, and 90.1% and reduce average cost by 34.6%, 29.3%, and 89.1% over JPBR, CCAT, and LCAT, respectively. This demonstrates the efficiency of ECO-GAME. Specifically, with different number of tasks on each user, ECO-GAME performs better than JPBR. This is because ECO-GAME considers resources from the completed tasks are re-allocated to the uncompleted tasks. While JPBR does not take this into consideration. Besides, CCAT performs better than JPBR on the whole, which demonstrates the efficiency of the resource allocation schemes proposed in this paper.

#### E. Results With Different Numbers of Users

To evaluate the performance of ECO-GAME with different numbers of users, we conduct the experiments in the settings of the user number  $N = 5, 10, 15, \dots, 50$  respectively, and show the results in Fig. 8. Fig. 8 shows the system cost under various numbers of users, in which ECO-GAME can reduce cost by up to 42.3%, 52.6%, and 93.5% and reduce average cost by 31.7%, 41.4%, and 78.6% over JPBR, CCAT, and LCAT, respectively.

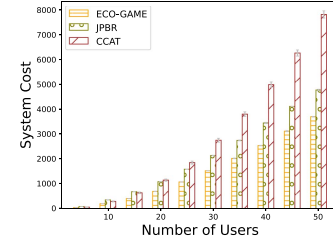


Fig. 8. System cost with various numbers of users.

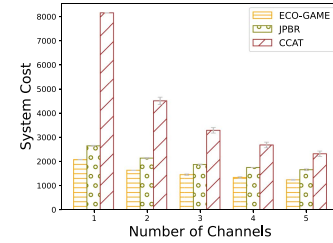


Fig. 9. System cost with various numbers of wireless channels.

This illustrates the efficiency of ECO-GAME. ECO-GAME considers, in the case of constrained bandwidth and computing resources, the reallocation of the bandwidth resources from the transmission-completed tasks to the transmission-uncompleted tasks, and of the computing resources from the computation-completed tasks to the computation-uncompleted tasks. While JPBR does not consider this. JPBR considers that, in the case of constrained communication and cloud computing resources, both communication and cloud computing resources allocated to each offloaded task remain unchanged within a computation offloading period. In short, the experimental results show that ECO-GAME outperforms JPBR in the computation cost, which indicates the effectiveness of the bandwidth and computing resource allocation schemes proposed in this paper.

#### F. Results With Other Parameter Settings

To see the effects of the other parameter settings on the experimental results, in this subsection we conduct additional experiments with other parameter settings for a deeper study. Specifically, we carry out the experiments with four kinds of other parameter settings, and they are number of channels  $M$ , channel bandwidth  $B_m$ , local computing capability  $F_n^l$ , and computing capability of MEC servers  $F^c$  respectively. The corresponding results are shown in Figs. 9, 10, 11 and 12 respectively. In addition, we conduct a deeper study about the special case of multiple users with single tasks, where we use DMCO [32] that has investigated this special case as one of the baselines to evaluate the efficiency of ECO-GAME. And the results are shown in Fig. 13.

1) *Parameter of  $M$* : Fig. 9 shows the system cost under various numbers of wireless channels, where  $M$  is set to 1, 2, 3, 4, 5 respectively. In Fig. 9, we can see that ECO-GAME reduces cost by up to 25.1% and 74.4% and reduces average cost by 23.2% and 58.0% over JPBR and CCAT respectively. 2) *Parameter*

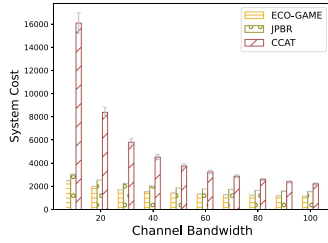


Fig. 10. System cost with various channel bandwidths.

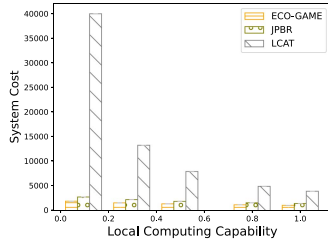


Fig. 11. System cost with various local computing capabilities.

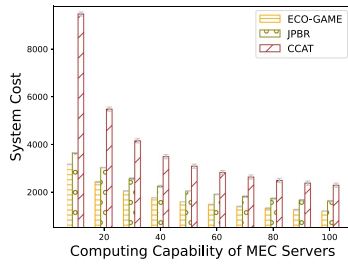


Fig. 12. System cost with various MEC computing capabilities.

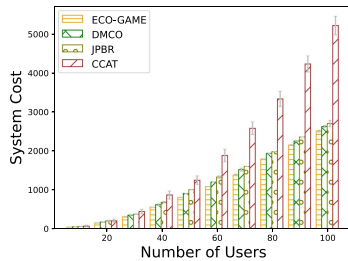


Fig. 13. System cost with various numbers of users with single tasks.

of  $B_m$ : Fig. 10 shows the system cost under various channel bandwidths, where  $B_m$  is set to 10, 20, 30, ..., 100 respectively. In Fig. 10, we can see that ECO-GAME reduces cost by up to 25.1% and 83.8% and reduces average cost by 22.5% and 60.7% over JPBR and CCAT respectively. 3) Parameter of  $F_n^l$ : Fig. 11 shows the system cost under various local computing capabilities, where  $F_n^l$  is set to 0.1, 0.3, 0.5, 0.8, 1.0 respectively. In Fig. 11, we can see that ECO-GAME reduces cost by up to 29.9% and 95.0% and reduces average cost by 26.8% and 82.3% over JPBR and LCAT respectively. 4) Parameter of  $F^c$ : Fig. 12 shows the system cost under various computing capabilities of MEC servers, where  $F^c$  is set to 10, 20, 30, ..., 100 respectively.

In Fig. 12, we can see that ECO-GAME reduces cost by up to 25.1% and 66.2% and reduces average cost by 21.1% and 49.9% over JPBR and CCAT respectively.

Fig. 13 shows the system cost under various numbers of users with single tasks, where the number of users is set to 10, 20, 30, ..., 100 respectively and each user has only one single task. In Fig. 13, we can see that ECO-GAME reduces cost by up to 14.2%, 26.5%, and 51.9% and reduces average cost by 9.5%, 16.2%, and 39.8% over DMCO, JPBR and CCAT respectively. The results demonstrate the efficiency of ECO-GAME. ECO-GAME outperforms DMCO since DMCO does not consider the reallocation of cloud computing resources to uncompleted tasks while ECO-GAME considers this. This also indicates the effectiveness of the cloud computing resource allocation scheme proposed in this paper.

We can draw the following conclusions from the above experimental results. First, ECO-GAME outperforms JPBR, CCAT, and LCAT in terms of the computation cost, which demonstrates the effectiveness of ECO-GAME. Second, the system cost by ECO-GAME, JPBR, CCAT, and LCAT gets reduced as the values of the above four parameters increase, where  $M$  and  $B_m$  correspond to communication resources and  $F_n^l$  and  $F^c$  correspond to computing resources. This is because the contention among tasks gets mitigated with the increase in constrained resources. Third, the system cost by ECO-GAME varies smoothly with these parameters, while the system cost by CCAT and LCAT has a large variation, especially when the parameter values are small. This is because CCAT offloads all tasks to the edge (LCAT computes all tasks locally), which is unwise in resource-constrained situations and may lead to excessive computation cost. While ECO-GAME makes trade-offs between the contentions among various limited resources based on the current resource situation, and makes reasonable offloading decisions, resulting in better performance.

## VII. CONCLUSION

MEC serves mobile users with wireless communication and cloud computing resources in the vicinity in mobile edge networks for low-latency computation offloading. However, existing works mostly consider the resources allocated to each offloaded task keep unchanged during a computation offloading period, leading to the constrained resource waste. Besides, the majority of related works assume each user offloads one task, whose solutions are not suitable for the realistic scenario where each user has multiple computation tasks to offload. To solve these problems, we propose efficient resource allocation schemes for the multi-task offloading problem, which re-allocate the resources from the completed tasks to the uncompleted tasks. Taking response time into account, we utilize the game theoretic approaches to formulate the multi-task offloading problem as the strategic game, together with the analysis about the existence of its Nash equilibrium. We then propose the ECO-GAME algorithm to obtain an approximate solution in polynomial time via the Nash equilibrium, together with the computational complexity analysis and performance evaluation using PoA. The experimental results indicate that ECO-GAME reduces 49.2%



cost than JPBR, and scales well as the number of tasks on each user increases and the number of users increases.

For our work, the limitation is that we do not evaluate our proposed algorithm in a real-world MEC system. And our future direction is to apply our ECO-GAME algorithm into practice. One possible way is to extend our model in the autonomous driving scenarios, where the autonomous vehicles offload the tasks of perception, prediction and planning to the nearby infrastructures for execution. And the nearby vehicles whose computation resources are under-utilized could also be used to execute the offloaded tasks so as to relieve the infrastructure overloads.

## REFERENCES

- [1] M. Goudarzi, H. Wu, M. Palaniswami, and R. Buyya, "An application placement technique for concurrent IoT applications in edge and fog computing environments," *IEEE Trans. Mobile Comput.*, vol. 20, no. 4, pp. 1298–1311, Apr. 2021.
- [2] Y. Siriwardhana, P. Porabage, M. Liyanage, and M. Ylianttila, "A survey on mobile augmented reality with 5G mobile edge computing: Architectures, applications, and technical aspects," *IEEE Commun. Survey Tuts.*, vol. 23, no. 2, pp. 1160–1192, Second Quarter, 2021.
- [3] H. Dinh, C. Lee, D. Niyato, and P. Wang, "A survey of mobile cloud computing: Architecture, applications, and approaches," *Wireless Commun. Mobile Comput.*, vol. 13, no. 18, pp. 1587–1611, 2013.
- [4] Y. Mao, C. You, J. Zhang, K. Huang, and K. Letaief, "A survey on mobile edge computing: The communication perspective," *IEEE Commun. Survey Tuts.*, vol. 19, no. 4, pp. 2322–2358, Fourth Quarter, 2017.
- [5] N. Abbas, Y. Zhang, A. Taherkordi, and T. Skeie, "Mobile edge computing: A survey," *IEEE Internet of Things J.*, vol. 5, no. 1, pp. 450–465, Feb. 2018.
- [6] C. Feng, P. Han, X. Zhang, B. Yang, Y. Liu, and L. Guo, "Computation offloading in mobile edge computing networks: A survey," *J. Netw. Comput. Appl.*, vol. 202, 2022, Art. no. 103366.
- [7] T. Taleb, K. Samdanis, B. Mada, H. Flinck, S. Dutta, and D. Sabella, "On multi-access edge computing: A survey of the emerging 5G network edge cloud architecture and orchestration," *IEEE Commun. Survey Tuts.*, vol. 19, no. 3, pp. 1657–1681, Third Quarter, 2017.
- [8] 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.
- [9] P. Apostolopoulos, G. Fragkos, E. Tsiropoulou, and S. Papavassiliou, "Data offloading in UAV-assisted multi-access edge computing systems under resource uncertainty," *IEEE Trans. Mobile Comput.*, vol. 22, no. 1, pp. 175–190, Jan. 2023.
- [10] Z. Ning et al., "Dynamic computation offloading and server deployment for uav-enabled multi-access edge computing," *IEEE Trans. Mobile Comput.*, vol. 22, no. 5, pp. 2628–2644, May 2023.
- [11] Z. Wan, D. Xu, D. Xu, and I. Ahmad, "Joint computation offloading and resource allocation for noma-based multi-access mobile edge computing systems," *Comput. Netw.*, vol. 196, 2021, Art. no. 108256.
- [12] X. Wang, Z. Ning, and S. Guo, "Multi-agent imitation learning for pervasive edge computing: A decentralized computation offloading algorithm," *IEEE Trans. Parallel Distrib. Syst.*, vol. 32, no. 2, pp. 411–425, Feb. 2021.
- [13] S. Li, S. Lin, L. Cai, W. Li, and G. Zhu, "Joint resource allocation and computation offloading with time-varying fading channel in vehicular edge computing," *IEEE Trans. Veh. Technol.*, vol. 69, no. 3, pp. 3384–3398, Mar. 2020.
- [14] Y. Sahni, J. Cao, L. Yang, and Y. Ji, "Multi-hop multi-task partial computation offloading in collaborative edge computing," *IEEE Trans. Parallel Distrib. Syst.*, vol. 32, no. 5, pp. 1133–1145, May 2021.
- [15] P. Dai, Y. Huang, K. Hu, X. Wu, H. Xing, and Z. Yu, "Meta reinforcement learning for multi-task offloading in vehicular edge computing," *IEEE Trans. Mobile Comput.*, to be published, doi: [10.1109/TMC.2023.3247579](https://doi.org/10.1109/TMC.2023.3247579).
- [16] Y. Wu, B. Shi, L. Qian, F. Hou, J. Cai, and X. S. Shen, "Energy-efficient multi-task multi-access computation offloading via NOMA transmission for IoTs," *IEEE Trans. Ind. Inform.*, vol. 16, no. 7, pp. 4811–4822, Jul. 2020.
- [17] J. Gao, Z. Kuang, J. Gao, and L. Zhao, "Joint offloading scheduling and resource allocation in vehicular edge computing: A two layer solution," *IEEE Trans. Veh. Technol.*, vol. 72, no. 3, pp. 3999–4009, Mar. 2023.
- [18] I. Elgendy, W. Zhang, Y. Zeng, H. He, Y. Tian, and Y. Yang, "Efficient and secure multi-user multi-task computation offloading for mobile-edge computing in mobile IoT networks," *IEEE Trans. Netw. Service Manag.*, vol. 17, no. 4, pp. 2410–2422, Dec. 2020.
- [19] L. Qian, Y. Wu, F. Jiang, N. Yu, W. Lu, and B. Lin, "NOMA assisted multi-task-access mobile edge computing via deep reinforcement learning for industrial internet of things," *IEEE Trans. Ind. Inform.*, vol. 17, no. 8, pp. 5688–5698, Aug. 2021.
- [20] S. Jošilo and G. Dán, "Selfish decentralized computation offloading for mobile cloud computing in dense wireless networks," *IEEE Trans. Mobile Comput.*, vol. 18, no. 1, pp. 207–220, Jan. 2019.
- [21] P. Mach and Z. Becvar, "Mobile edge computing: A survey on architecture and computation offloading," *IEEE Commun. Survey Tuts.*, vol. 19, no. 3, pp. 1628–1656, Third Quarter, 2017.
- [22] Z. Liao et al., "Distributed probabilistic offloading in edge computing for 6G-enabled massive Internet of Things," *IEEE Internet Things J.*, vol. 8, no. 7, pp. 5298–5308, Apr. 2021.
- [23] W. Zhang, R. Yadav, Y. Tian, S. Tyagi, I. Elgendy, and O. Kaiwartya, "Two-phase industrial manufacturing service management for energy efficiency of data centers," *IEEE Trans. Ind. Inform.*, vol. 18, no. 11, pp. 7525–7536, Nov. 2022.
- [24] T. Nguyen, L. Le, and Q. Le-Trung, "Computation offloading in MIMO based mobile edge computing systems under perfect and imperfect CSI estimation," *IEEE Trans. Services Comput.*, vol. 14, no. 6, pp. 2011–2025, Nov/Dec. 2021.
- [25] U. Saleem, Y. Liu, S. Jangsher, X. Tao, and Y. Li, "Latency minimization for D2D-enabled partial computation offloading in mobile edge computing," *IEEE Trans. Veh. Technol.*, vol. 69, no. 4, pp. 4472–4486, Apr. 2020.
- [26] L. Yang, B. Liu, J. Cao, Y. Sahni, and Z. Wang, "Joint computation partitioning and resource allocation for latency sensitive applications in mobile edge clouds," *IEEE Trans. Services Comput.*, vol. 14, no. 5, pp. 1439–1452, Sep/Oct. 2021.
- [27] X. Dai, Z. Xiao, H. Jiang, and J. C. S. Lui, "UAV-assisted task offloading in vehicular edge computing networks," *IEEE Trans. Mobile Comput.*, to be published, doi: [10.1109/TMC.2023.3259394](https://doi.org/10.1109/TMC.2023.3259394).
- [28] X. Dai et al., "Offloading dependent tasks in edge computing with unknown system-side information," *IEEE Trans. Services Comput.*, to be published, doi: [10.1109/TSC.2023.3320674](https://doi.org/10.1109/TSC.2023.3320674).
- [29] C. Sun, X. Wu, X. Li, Q. Fan, J. Wen, and V. C. M. Leung, "Cooperative computation offloading for multi-access edge computing in 6G mobile networks via soft actor critic," *IEEE Trans. Netw. Sci. Eng.*, to be published, doi: [10.1109/TNSE.2021.3076795](https://doi.org/10.1109/TNSE.2021.3076795).
- [30] R. Yadav, W. Zhang, O. Kaiwartya, H. Song, and S. Yu, "Energy-latency tradeoff for dynamic computation offloading in vehicular fog computing," *IEEE Trans. Veh. Technol.*, vol. 69, no. 12, pp. 14 198–14 211, Dec. 2020.
- [31] R. Yadav et al., "Smart healthcare: RL-based task offloading scheme for edge-enabled sensor networks," *IEEE Sensors J.*, vol. 21, no. 22, pp. 24 910–24 918, Nov. 2021.
- [32] S. Chu, Z. Fang, S. Song, Z. Zhang, C. Gao, and C. Xu, "Efficient multi-channel computation offloading for mobile edge computing: A game-theoretic approach," *IEEE Trans. Cloud Comput.*, vol. 10, no. 3, pp. 1738–1750, Third Quarter, 2022.
- [33] Z. Xiao et al., "Multi-objective parallel task offloading and content caching in D2D-aided MEC networks," *IEEE Trans. Mobile Comput.*, vol. 22, no. 11, pp. 6599–6615, Nov. 2023.
- [34] X. Chen, "Decentralized computation offloading game for mobile cloud computing," *IEEE Trans. Parallel Distrib. Syst.*, vol. 26, no. 4, pp. 974–983, Apr. 2015.
- [35] 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, Oct. 2016.
- [36] S. Jošilo and G. Dán, "Wireless and computing resource allocation for selfish computation offloading in edge computing," in *Proc. IEEE Conf. Comput. Commun.*, 2019, pp. 2467–2475.
- [37] A. Shakarami, A. Shahidinejad, and M. Ghobaei-Arani, "A review on the computation offloading approaches in mobile edge computing: A game-theoretic perspective," *Softw. Pract. Experience*, vol. 50, no. 9, pp. 1719–1759, 2020.
- [38] S. Jošilo and G. Dán, "Computation offloading scheduling for periodic tasks in mobile edge computing," *IEEE/ACM Trans. Netw.*, vol. 28, no. 2, pp. 667–680, Apr. 2020.



- [39] Q. Luo, C. Li, T. Luan, W. Shi, and W. Wu, "Self-learning based computation offloading for internet of vehicles: Model and algorithm," *IEEE Trans. Wireless Commun.*, vol. 20, no. 9, pp. 5913–5925, Sep. 2021.
- [40] F. Chai, Q. Zhang, H. Yao, X. Xin, R. Gao, and M. Guizani, "Joint multi-task offloading and resource allocation for mobile edge computing systems in satellite IoT," *IEEE Trans. Veh. Technol.*, vol. 72, no. 6, pp. 7783–7795, Jun. 2023.
- [41] W. Fan et al., "Game-based task offloading and resource allocation for vehicular edge computing with edge-edge cooperation," *IEEE Trans. Veh. Technol.*, vol. 72, no. 6, pp. 7857–7870, Jun. 2023.
- [42] E. Meskar, T. D. Todd, D. Zhao, and G. Karakostas, "Energy aware offloading for competing users on a shared communication channel," *IEEE Trans. Mobile Comput.*, vol. 16, no. 1, pp. 87–96, Jan. 2017.
- [43] D. Monderer and L. S. Shapley, "Potential games," *Games Econ. Behav.*, vol. 14, no. 1, pp. 124–143, 1996.
- [44] T. Roughgarden, *Selfish Routing and the Price of Anarchy*. Cambridge, MA, USA: MIT Press, 2005.



**Shuhui Chu** received the BE and MS degrees in computer science and technology from Jilin University, Changchun, China, in 2017 and 2020, respectively. She is currently working toward the PhD degree in computer science with the University of Macau. Her research interests include mobile edge computing, game theory, and reinforcement learning.



**Chengxi Gao** (Member, IEEE) received the BS and MS degrees from the Department of Computer Science, Northeastern University, China, and the PhD degree from the Department of Computer Science, City University of Hong Kong. He is an assistant professor with the Shenzhen Institute of Advanced Technology (SIAT), Chinese Academy of Sciences (CAS). Before joining CAS, he was a research associate with the City University of Hong Kong. His research interests include data center networking, networking system, and computation offloading.



**Minxian Xu** (Member, IEEE) received the BSc and MSc degrees in software engineering from the University of Electronic Science and Technology of China, in 2012 and 2015, respectively, and the PhD degree from the University of Melbourne, in 2019. He is currently an associate professor with the Shenzhen Institute of Advanced Technology, Chinese Academy of Sciences. His thesis was awarded the 2019 IEEE TCSC Outstanding PhD Dissertation Award. His research interests include resource scheduling and optimization in cloud computing.



**Kejiang Ye** (Senior Member, IEEE) received the BSc and PhD degrees from Zhejiang University, in 2008 and 2013, respectively. He was also a joint PhD student with the University of Sydney from 2012 to 2013. After graduation, he works as post-doc researcher with Carnegie Mellon University from 2014 to 2015 and Wayne State University from 2015 to 2016. He is currently a professor with the Shenzhen Institute of Advanced Technology, Chinese Academy of Science. His research interests focus on the performance, energy, and reliability of cloud computing and network systems.



**Zhu Xiao** (Senior Member, IEEE) received the MS and PhD degrees in communication and information systems from Xidian University, China, in 2007 and 2009, respectively. From 2010 to 2012, he was a research fellow with the Department of Computer Science and Technology, University of Bedfordshire, U.K. He is currently a full professor with the College of Computer Science and Electronic Engineering, Hunan University, China. His research interests include mobile edge computing, Internet of Vehicles, and intelligent transportation systems. He is serving

as an associated editor of *IEEE Transactions on Intelligent Transportation Systems*.



**Chengzhong Xu** (Fellow, IEEE) received the PhD degree from the University of Hong Kong, in 1993. He is currently a chair professor of computer science and the dean with the Faculty of Science and Technology, University of Macau. Prior to this, he was with the Faculty of Wayne State University, USA, and the Shenzhen Institute of Advanced Technology, Chinese Academy of Science, China. He has published more than 400 papers and more than 100 patents. His research interests include cloud computing and data-driven intelligent applications. He was the chair

of IEEE Technical Committee on Distributed Processing from 2015 to 2019.