Computer Networks 178 (2020) 107334

Contents lists available at ScienceDirect

Computer Networks

journal homepage: www.elsevier.com/locate/comnet

The partial computation offloading strategy based on game theory for multi-user in mobile edge computing environment

Shuchen Zhou[a,b,*], Waqas Jadoon[a]

[a] Department of Computer Science, COMSATS University Islamabad, Abbottabad Campus 22060, Islamabad, Pakistan
[b] Institute of International Education, Huanghuai University, Zhumadian, 463000, P.R. China

ARTICLE INFO

Keywords:
Mobile edge computing
Partial computation offloading
Game theory
NOMA

ABSTRACT

Mobile edge computing provides powerful computing for mobile device users to broaden their computation capability. However, due to the limited bandwidth of the wireless channel, both the energy consumption for mobile device's computation offloading and the service latency maybe increase. In this paper, we focus on the partial computation offloading problem for multi-user in mobile edge computing environment with multi-wireless channel. The computation overhead model is built based on game theory. Then, the existence of Nash equilibrium is proven. Furthermore, the partial computation offloading algorithm with low time complexity is given to achieve the Nash equilibrium. In addition, another partial computation task offloading mechanism for mobile device users, who has enough energy and only pay attention to the computation time overhead, is given to reduce the computation overhead. Finally, extensively numerical experiments are conducted. The experimental results show that our proposed algorithm can achieve better performance than the compared algorithms in the mobile edge computing environment.

1. Introduction

In recent years, the mobile devices (MDs) have been applied by more and more people. Meanwhile, mobile device users (MDUs) expect to run more and more desktop-level applications on their own MDs, such as interactive gaming, face recognition, augmented reality, natural language processing, etc[1]. However, these applications are typically high energy consumption, demanding intensive computation, time-sensitive and resource-hungry. Moreover, each MD has the limited computation resources, storage size and energy. Thus, with the limitations, the MDs can not satisfy the increasing resource needs of the mobile applications. For this problem, mobile computation offloading is a effective way to broaden the computation capability of MDs. Mobile computation offloading could enable part or all of the computation tasks from resource-limited MDs to be dynamically migrated to powerful computing platforms.

Mobile edge computing, as a new computing paradigm, has been widely applied to many fields, such as healthcare, video analysis, mobile big data analysis, connected vehicles, smart grid, smart building control, etc[2]. Mobile edge computing place substantial compute resources and storage resources at the edge of the internet, in close proximity to the MDs, such as smart phone, sensors, tablets, etc. Thus, mobile edge computing provides powerful computing for MDUs to broaden their computation capability.

Mobile edge computing could better promote the development of mobile computation offloading. MDs offload their computation tasks via wireless access to edge server (ES) to broaden their computation capability. However, due to the limited bandwidth of the wireless channel (WC), both computation offloading energy of MDs and the service latency maybe increase. Meanwhile, MDUs are acutely sensitive to energy and delay. Therefore, in order to reduce MD's energy consumption and service latency, it is very important value and significance to research the resource allocation problem of WC for multi-MDU computation offloading in mobile edge computing environment.

Generally, mobile applications can be classified into three groups[3], i.e., data partitioned oriented applications (DPOAs), code partitioned oriented applications (CPOAs) and continuous execution applications (CEAs). For DPOAs, such as the file/figure compression application and the virus scan application, the amount of the processed data is known in advance. The data could be divided into several independent subsets. Then, by taking advantage of the parallelism, a portion of these subsets could be processed on MDs and the rest is addressed on ES [4]. A CPOA can be divided into several subtasks. Some of them can be processed in parallel. But, maybe some of them are addressed sequentially, i.e., the output data of one subtask is the input data of another subtask. The relationship among these subtasks can form a directed acyclic graph (DAG). Then, we can build the mathematical model according to the DAG. For CEAs, such as the cloud mobile gaming or other interactive

* Corresponding author.
E-mail address: zshuchen@aliyun.com (S. Zhou).

https://doi.org/10.1016/j.comnet.2020.107334
Received 8 December 2019; Received in revised form 4 May 2020; Accepted 23 May 2020
Available online 3 June 2020

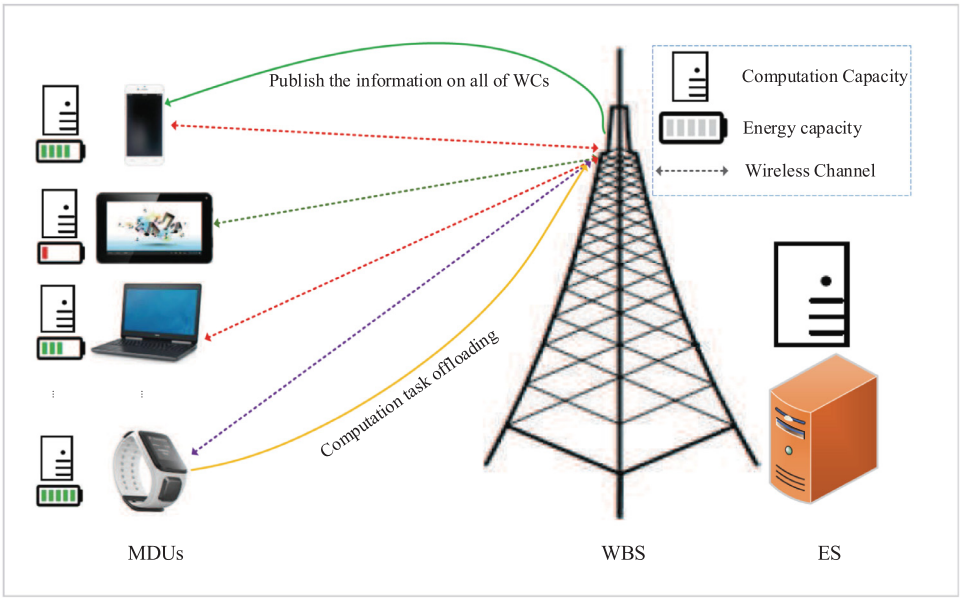

Fig. 1. An illustration of mobile edge computing.

applications, the running time of the applications can not be known in advance. Generally, the average delay is taken as the metric of these applications [5].

In this paper, we focus on the scenario where some MDUs are covered by a wireless base-station (WBS) within an ES. For each MDU, there are several WCs to chosen. Each MD is running DPOAs. For a DPOA, MDU can accomplish it on his/her own MD, on ES or on both of them. The scenario is shown in Fig. 1. In this scenario, the computation offloading mechanism for multi-user is studied. The performance of the computation offloading mechanism is mainly affected by the transmission rate of WC[1]. Thus, how to choose a proper WC from multiple WCs among multiple MDUs for computing offloading becomes a challenging question. If too many MDUs simultaneously offload their tasks to ES via the same WC, there will exist severe interference to each other so that the data transmission rate of the computation task offloading will be reduced. In this case, the computation overhead, such as time overhead, energy overhead, etc., will be higher than that of MD computing. So, in order to achieve the efficient computation task offloading mechanism for MDUs in mobile edge computing environment, there are three key challenges to be tackled as follows.

- How should one MDU choose between the MD computing (on his/her own MD) and the ES computing (via computation offloading)?
- How can one MDU choose a proper WC for computation task offloading to achieve high data transmission rate if the MDU chooses ES computing?
- How many should one MDU offload computation tasks to ES via the chosen WC so that his/her computation overhead is minimized?

To address these problems, we model the computation task offloading problem among multiple MDUs for the mobile edge computing in a multi-WC environment as a partial computation offloading game. The corresponding partial computation offloading algorithm is designed to achieve the Nash equilibrium of the game so that the computation overhead of each MDU is minimized. The main contributions are described as follows.

- *Partial Computation Offloading Game Formulation*: By considering the time overhead and energy overhead, the utility function of the partial computation offloading problem is built. Moreover, the partial computation offloading problem is formulated as the partial computation offloading game based on the utility function and the decision space.
- *Analysis of The Partial Computation Offloading Game Properties*: We analyze the structural property of the partial computation offloading game and show that the game is non-cooperative game within continuous strategic space. According to the property of this type of game, the Nash equilibrium is achieved. Furthermore, another partial computation task offloading mechanism for the MDU, who has enough energy and only pay attention to the computation time overhead, is given to reduce the computation overhead.
- *Partial Computation Offloading Algorithm Design*: By designing the partial computation task offloading algorithm, we obtain the Nash equilibrium of the partial computation offloading game. Numerical results show that our proposed algorithm can achieve the minimum computation overhead in the mobile edge computing environment.

The rest of this paper is organized as follows. Section 2 introduces the related works on computation offloading problem in mobile edge computing environment. Then, the system model is built in Section 3. Moreover, the properties of the partial computation offloading game are discussed in Section 4. Section 5 designs the partial computation offloading algorithm and describes the algorithm by pseudo codes. Furthermore, the performance of the proposed algorithm is analyzed by the numerical experiments in Section 6. Finally, the current and future works are summarized in Section 7.

2. RELATED WORK

Many previous works have paid attention to the computation offloading problem in the mobile edge computing environment with multi-user or multi-wireless channel. Table 1 describes the comparison results of some computation offloading strategies.

Chen *et al.* [1] adopted game theory to formulate the computation offloading decision process and the utility function was the weighted sum of time overhead and energy overhead. The balance between time overhead and energy overhead was obtained by minimize the utility function. Sardellitti *et al.* [4] minimized the overall MDUs' energy consumption, while meeting latency constraints, by jointly optimizing the radio resources and the computational resources. Apostolopoulos *et al.* [6] formulated the joint problem of autonomous mobile edge computing

Table 1
The comparison of some computation offloading strategies[1].

Ref.	Env.	NoC	NoU	Objective	Met.	WS
[1]	MECC	M	M	Minimizing time and energy	GT	3*G*/4*G*
[4]	MECC	M	M	Minimizing energy	OT	5G
[6]	MEC	S	M	Minimizing QoS	GT	3*G*/4*G*
[7]	MEC	M	M	Minimizing time and energy	OT	3*G*/4*G*
[8]	MEC	M	S	Minimizing time and energy	OT	3*G*/4*G*
[9]	MECC	M	S	Minimizing energy	OT	3*G*/4*G*
[10]	MCC	M	M	Minimizing energy	OT	3*G*/4*G*
[11]	MEC	M	M	Minimizing energy	OT	3*G*/4*G*
[12]	MECC	S	M	Minimizing time and energy	OT	3*G*/4*G*
[13]	MEC	S	M	Minimizing time and energy	OT	3*G*/4*G*
[15]	MECHAB	S	M	Minimizing time and energy	FL	*TDMA*
Our paper	MEC	M	M	Minimizing time and energy	GT	5G

[1] Ref.-Reference. Env.-Environment. NoC-Number of channels. NoU-Number of users. Met.-Method. WS-Wireless system. MECC-Mobile edge cloud computing. MEC-Mobile edge computing. MCC-Mobile cloud computing. MECHAB-Mobile edge computing-enabled High altitude balloons. M-Multiple. S-Single. GT-Game theory. OT-Optimization theory. FL-Federated learning. TDMA-Time division multiple access.

servers' operation and MDs' QoS satisfaction as a minority game. Tran *et al.* [7] adopted the mixed integer nonlinear program to model the joint problem of task offloading and resource allocation for maximizing the users' task offloading gains. Li [8] tried to achieve the balance among the minimization of time overhead, minimization of energy overhead and the minimization cost-performance ratio for the single user in multiple mobile edge computing environment. Labidi *et al.* [9] devised two solution for the computation offloading decision process, aiming to minimize the average energy consumption of single MD and guaranteeing the average delay service of the applications on the MD. Zhao *et al.* [10] presented the computation offloading algorithm for multi-MDU scenario by nonlinear programming with high time complexity. Moreover, Zhao assumed the MDUs had the same channel quality and the same computing capabilities. This case is not realistic for the real network. You and Huang [11] designed the partial computation offloading process based on the different channel qualities, the energy overhead of MDs, and the fairness among the MDs. Meanwhile, they given a sub-optimal offloading decision algorithm, But, this solution would lead to negligibly higher energy overhead of MDs comparing with the optimal solution. Muoz *et al.* [12] adopted convex function theory to achieve the a balance solution between energy overhead and time overhead for the multi-MDU in edge computing environment. Also, Mao *et al.* [13] designed the computation offloading algorithm based on Lyapunov optimization for multi-MDU mobile edge computing system. Moreover, the Gauss-Seidel method was applied to obtain the balance solution between energy overhead and time overhead. Chen *et al.* [14] proposed to leverage the artificial intelligence for next generation wireless networks to effectively provide ultra reliable low latency communications and pervasive connectivity for Internet of Things. With this idea, Wang *et al.* [15] adopted the federated learning technology to minimize the weighted sum of the energy and time consumption in mobile edge computing-enabled High altitude balloons.

However, in Table 1, most of works adopt the centralized theory for the computation offloading and have high time complexity to achieve the balance solution between time overhead and energy overhead. Moreover, the wireless systems in most of works are the 3G/4G. In our paper, we adopt the distributed idea from game theory to study the computation offloading problem for multi-MDU scenario in multi-WC mobile edge computing environment with 5G wireless system. The objective is to minimize the weighted sum of energy overhead and time overhead, while designing the partial computation offloading algorithm with low time complexity. Comparing with the previous works, the differences are described as follows.

- The game theory is adopted for the partial computation offloading problem in the mobile edge computing.
- We study the mobile edge computing environment, in which multiple MDUs share multiple wireless channels.
- The wireless system studied by us is 5G and the non-orthogonal multiple access (NOMA) technique is adopted to manage the uplink/downlink communications of MDUs

In addition, some scholars focus on the number of edge servers in mobile edge computing environment. For instance, Ren *et al.* [16] minimized the number of the edge servers based on the integer linear programming, while ensuring QoS constraints. However, it is assumed that each WBS is endowed with an edge server in most of works. Such as, Ouyang *et al.* [17] assumed that each WBS was endowed with an edge server and proposed a novel adaptive user-managed service placement mechanism. Chen *et al.* [18] investigated the collaborative service placement for maximizing the system performance in mobile edge computing environment, in which each WBS had an edge server, due to the limited capacity of each edge server. Bouet *et al.* [19] studied the resource optimization problem by considering the server size, the server number and the server operation area, each of which had an edge server. Moreover, Burger *et al.* [20] indicated that a high number of ESs was preferable compared to few larger ESs to reduce the system latency.

In this paper, like most of works, we also assume that each WBS has an ES and the ES has enough capacity for addressing the workloads of the MDUs that are covered by the ES.

3. SYSTEM MODEL AND GAME FORMULATION

In this section, the system model is detailedly introduced. Firstly, the system model of mobile edge computing under our consideration is described. Then, the communication and computation models are introduced in details. Finally, the game-based partial computation offloading model for the mobile edge computing is proposed.

3.1. System model

In this paper, a mobile edge computing environment, which is composed by U MDUs and one WBS with an ES, is considered. It is assumed that the ES has enough capacity for addressing the workloads of the MDUs who are covered by the corresponding WBS. The set of MDUs is defined as $\mathcal{U} = \{1, 2, \ldots, U\}$. Each MD has some DPOAs, such as the virus scan application, the file/figure compression application, etc[21]. Through WBS, the computation tasks of DPOAs on MDs can be offloaded to the ESs in proximity deployed by the telecom operators (TOs). For convenience, the set of MDUs remains unchanged during a period for computation offloading, such as several hundred milliseconds, and can

change during different periods[1], such as [1]. Like [22], we assume full granularity in data partition, i.e., one DPOA task can be partitioned into several subtasks with any size, despite that only several partitions are possible in practice. So, in our paper, the optimal solution of computation task offloading could be taken as a performance upper-bound of realistic offloading strategies. Because both the communication model and the computation model play very important roles in mobile edge computing environment, they will be introduced in details as follows.

3.2. Communication model

In this section, the communication model for wireless access in mobile edge computing environment is introduced in detail. In our paper, the 5G cellular network is consider and NOMA technique is adopted to manage the uplink/downlink communications of MDUs. Meanwhile, we consider the scenario that the each MDU covered by an certain ES has multi-WC provided by the corresponding WBS to be chosen as the link between the corresponding MD and ES for computation offloading, but each MDU only can choose one WC for data transmission during a period of computation offloading. $\mathscr{L} = \{1, 2, \ldots, L\}$ is taken as the set of wireless channels. y_u is regarded as the channel decision of MDU u. $y_u = l$ means that MDU u chooses WC l as the link between his/her MD and the corresponding ES for computation offloading. $y_u = 0$ implies that MDU u decides to finish the computation tasks on his/her own MD. $\boldsymbol{y} = (y_1, y_2, \ldots, y_U)$ is defined as the WC decision profile of all MDUs. Let x_u denote the ratio of the size of offloading computation tasks to the size of overall computation tasks. In other words, x_u is the computation offloading decision of MDU u. Considering our assumption, $x_u \in [0, 1]$. $x_u = 0$ if MDU u chooses to compute his/her all of tasks locally on his/her own MD, and $x_u > 0$ if MDU u decides to offload all or part of tasks to ES by the WC. Let $\boldsymbol{x} = (x_1, x_2, \ldots, x_U)$ be the computation offloading decision profile of all MDUs. The set of users who choose the same channel as MDU u is denoted by $\mathscr{U}^{y_u} = \{u_1^{y_u}, u_2^{y_u}, \ldots, u_{U'}^{y_u}\}$, $U' \leq U$. Assume that the channel gains of U' MDs follow the order that $g_1^{y_u} \leq g_2^{y_u} \leq \ldots \leq g_{U'}^{y_u}$. For NOMA, the successive interference cancellation (SIC) technique is adopted to decode signals effectively [23]. The decoding order is from the user with better channel gain to the user with worse channel gain [24]. $r_u(\boldsymbol{x}, \boldsymbol{y})$ is taken as the uplink data rate of MDU u that decides to partially or completely offload his/her computation tasks to ES through the wireless channel. According to [24–26], $r_u(\boldsymbol{x}, \boldsymbol{y})$ is described as follows.

$$r_u(\boldsymbol{x}, \boldsymbol{y}) = \omega log_2(1 + \frac{p_u g_u^{y_u}}{\omega\sigma^2 + \sum_{i=1}^{u^{y_u}-1} p_i g_i^{y_u}}), \tag{1}$$

where p_u is regarded as MDU u's transmission power. For simplification, each sub-channel has ω bandwidth. There is no interference existing between channels because different bandwidths are allocated to different channels [23]. σ^2 is taken as the power spectral density of the additive white Gaussian noise. u_u^y is the sort number of MDU u about channel gain in WC y_u.

Without loss of generality, we assume that the channel gains of U MDUs follow the order that $g_1 \leq g_2 \leq \ldots < g_U$. $\{i \in \mathscr{U} | y_i = y_u, g_i \leq g_u, i \neq u\}$ is regarded as the set of MDUs who have the same sub-channel with MDU u and whose channel gains are less than or equal to that of MDU u. Thus, $\sum_{i=1}^{u_u^y-1} p_i g_i^{y_u} = \sum_{\{i \in \mathscr{U} | y_i = y_u, g_i \leq g_u, i \neq u\}} p_i g_i$. So,

$$r_u(\boldsymbol{x}, \boldsymbol{y}) = \omega log_2(1 + \frac{p_u g_u}{\omega\sigma^2 + \sum_{\{i \in \mathscr{U} | y_i = y_u, g_i \leq g_u, i \neq u\}} p_i g_i}), \tag{2}$$

where g_u represents the channel gain between MDU u and the WBS.

3.3. Computation model

In this section, the computation model will be described. Firstly, we assume that MDU u has a computation task J_u, where $J_u \triangleq (s_u, d_u)$. The task J_u can be determined to accomplished on MDU u' MD or on the corresponding ES[2] Here, s_u is taken as the size of computation task J_u and d_u is denoted as the total number of CPU cycles for accomplishing the computation task J_u. The information of s_u and d_u could be achieved by MDU u with the methods in [27], [28], etc. Next, the computation overhead in terms of both energy consumption and computation time consumption for both MD and ES computing approaches will be discussed in detail.

3.3.1. MD Computing

For this case, the MDU u will accomplish his/her all or part of computation task in his/her MD, i.e., MDU u's all or part of computation task will be executed locally. Let c_u^{MD} denote the computation capability (i.e., CPU cycles per second) of MD which belongs to MDU u. Obviously, different MDs cloud have different computation capability. If the computation task J_u is addressed only by MDU u, the computation execution time is

$$t_u^{MD} = \frac{d_u}{c_u^{MD}}. \tag{3}$$

Correspondingly, the computation energy of the computation task J_u that is finished only on by MDU u is

$$e_u^{MD} = e_u' d_u, \tag{4}$$

where e_u' denotes the energy consumption coefficient, i.e., the consumed energy per CPU cycle. e_u' can be achieved by the realistic measurement methods in [29], usually, $e_u' = 10^{-11}(c_u^{MD})^2$.

Then, the MD computation execution time for all or part of task J_u is given as

$$t_u^{MD}(x_u) = \frac{d_u(1 - x_u)}{c_u^{MD}}, \tag{5}$$

where $t_u^{MD}(x_u)$ is the MD computation execution time of MDU u, $(1 - x_u)$ represents the rate of computation task J_u that is executed on MDU u's MD, and $d_u(1 - x_u)$ denotes the total number of CPU cycles for accomplishing the corresponding rate of computation task J_u.

Furthermore, the MD computation energy for all or part of task J_u takes the form

$$e_u^{MD}(x_u) = e_u' d_u(1 - x_u), \tag{6}$$

where $e_u^{MD}(x_u)$ is the MD computation energy of MDU u.

Finally, (3), (4), (5) and (6) combine to form the MD computing overhead in terms of the computation execution time and the computation energy as follows:

$$O_u^{MD}(x_u) = \frac{w_u^t t_u^{MD}(x_u)}{t_u^{MD}} + \frac{(1 - w_u^t) e_u^{MD}(x_u)}{e_u^{MD}}, \tag{7}$$

where Equation (3) and (4) are used to normalize the computation execution time and the computation energy. $w_u^t \in [0, 1]$ implies that the weighting parameter of the computation execution time. Obviously, different MDUs may have different weighting parameters which depend on the what the MDUs focus on. For instance, when MDU u has a MD with low battery, i.e., he/she mainly pay attention to the computation energy, he/she could give more weight to the computation energy, moreover, let $w_u^t = 0$, which means MDU u only focuses on the computation energy. Similarly, when MDU u has a computation task that is sensitive to computation delay, i.e., he/she mainly pay attention to the computation execution time, he/she could give more weight to the computation

[1] The scenario where MEUs can depart and leave dynamically during a period for the computation jobs offloading is ignored and will be studied in a future work.

[2] Here, we assume that the ES has the ability to accomplished the tasks submitted by the users who belong to the ES. For the future work, we will study the scenario where the cloud and edge server coordinately process the tasks.

execution time, moreover, let $w_u^t = 1$, which means MDU u only focuses on the computation execution time. Hence, the weighting parameter of one certain MDU is changing dynamically if the computation tasks of MDU u have different demands. But, for simplicity of our study, we assume that the weighting parameter of each MDU is unchanged during a period and could be set as different value in different periods.

3.3.2. ES Computing

For this case, the MDU u will offload his/her all or part of computation task into ES in proximity deployed by the TO via WC and the ES will accomplish the computation task on behalf of the MDU. For offloading the computation task, the extra overhead in terms of time and energy will be incurred for transmitting the all or part of computation task via WC. Taking into account the communication model in (2), the extra overhead in terms of time and energy of MDU u for transmitting all or part of computation task J_u is described, respectively, as follows:

$$t_{u,tr}^{ES}(\boldsymbol{x}, \boldsymbol{y}) = \frac{s_u x_u}{r_u(\boldsymbol{x}, \boldsymbol{y})}, \tag{8}$$

and

$$e_{u,tr}^{ES}(\boldsymbol{x}, \boldsymbol{y}) = \frac{p_u s_u x_u}{r_u(\boldsymbol{x}, \boldsymbol{y})}, \tag{9}$$

When the computation task is offloaded into ES, ES will execute it. Let c_u^{ES} denote the computation capability (i.e., CPU cycles per second) that is assigned to MDU u by the ES. Based on our assumptions mentioned above, the computation time of ES for MDU u's computation task can be determined by

$$t_{u,exe}^{ES}(x_u) = \frac{d_u x_u}{c_u^{ES}}. \tag{10}$$

Then, according to (3), (4), (8), (9) and (10), the ES computing overhead in terms of time and energy takes the form

$$O_u^{ES}(\boldsymbol{x}, \boldsymbol{y}) = \frac{w_u^t(t_{u,tr}^{ES}(\boldsymbol{x}, \boldsymbol{y}) + t_{u,exe}^{ES}(x_u))}{t_u^{MD}} + \frac{(1 - w_u^t)e_{u,tr}^{ES}(\boldsymbol{x}, \boldsymbol{y})}{e_u^{MD}}. \tag{11}$$

For simplification, we neglect the time overhead for the ES to send the computation result back to MDU, because, for many applications (e.g., natural language processing, face recognition, etc.), the size of computation result is commonly much smaller than that of computation task [1], [30], [31], [32].

From the communication and computation models mentioned above, we can know that the computation offloading is not always beneficial for MDUs. If too many MDUs choose the same WC to offload their computation tasks to ES, the uplink data rate of each MDU in the WC may decrease. In this case, the energy consumption of computation task transmission may be more than that of computation task with local execution. Similarly, the overall time delay of both computation task transmission to ES and execution on ES may be more than that of computation task with local execution. So, it is more beneficial that the MDU choose to accomplish his/her computation task locally on the MD.

4. PARTIAL COMPUTATION OFFLOADING GAME FOR MULTI-USER IN MULTI-WC

In this section, the game theory is adopted to address the partial computation offloading problem due to the inherent characteristics of the wireless systems and the mobile devices.

The inherent characteristics of the wireless systems[33]. First, the wireless systems is more and more complicated due to dynamic interactions, protocol heterogeneity, network size, which make it difficult to optimize, analyze, and model wireless network performance. In such contexts, game theory is able to better understand the complex interactions, and design more efficient, scalable, and robust protocols. Second, due to the limited resources in wireless communications and the increasing number of mobile devices, resource scarcity and hence competition among mobile users becomes more severe. This competition for limited resources closely matches with game theory rationale.

The inherent characteristics of the users [34–36]. Mobile devices may be autonomous entities with different computing capabilities and tasks, and thus with individual interests in terms of response time and energy consumption requirements. Moreover, they may be interested in maximizing their own performance. In this case, the game theory is able to better reflect the user rationality.

4.1. Game formulation

In this section, the partial computation offloading decision problem for multi-user is considered based on game theory. Considering that x_u is the computation offloading decision of MDU u, $x_{-u} = (x_1, \ldots, x_{u-1}, x_{u+1}, \ldots, x_U)$ is used to denote the computation offloading decisions by all other users except MDU u. Similarly, y_u is the WC decision of MDU u, y_{-u} is taken as the WC decision of all other users except MDU u. When other MDUs' computation offloading decisions x_{-u} and WC decision y_{-u} are given, MDU u will choose the proper computation offloading decision x_u and the proper WC y_u to minimize his/her own computation overhead in terms of computation execution time and computation energy. Let $V_u(\boldsymbol{x}, \boldsymbol{y})$ denote the computation overhead of MDU u in terms of computation execution time and energy. Then, according to the communication model and the computation model, $V_u(\boldsymbol{x}, \boldsymbol{y})$ takes the form

$$V_u(\boldsymbol{x}, \boldsymbol{y}) = O_u^{MD}(x_u) + O_u^{ES}(\boldsymbol{x}, \boldsymbol{y}). \tag{12}$$

For MDU u, the objective is to minimize his/her own computation overhead in terms of computation execution time and energy, i.e.,

$$\min_{x_u \in [0,1], y_u \in \mathscr{Y} \bigcup \{0\}} V_u(\boldsymbol{x}, \boldsymbol{y}), \forall u \in \mathscr{U}. \tag{13}$$

Then, the computation overhead problem above is formulated as a non-cooperative game in strategic form

$$\Gamma(\boldsymbol{x}, \boldsymbol{y}) = (\mathscr{U}, \{(x_u, y_u)\}_{u \in \mathscr{U}}, \{V_u(\boldsymbol{x}, \boldsymbol{y})\}_{u \in \mathscr{U}}), \tag{14}$$

where the set of MDUs $\mathscr{U}$ is regarded as the set of players, the set of strategies for player u is defined as a decision tuple set $\{(x_u, y_u)\}_{x_u \in [0,1], y_u \in \mathscr{Y}}$, and the computation overhead function V_u is the utility function for player u. According to the knowledge, which introduces the computation overhead function $V_u(\boldsymbol{x}, \boldsymbol{y})$, mentioned above, we can know that player u prefer to minimize his/her own utility function. In the sequel, the non-cooperative game $\Gamma(\boldsymbol{x}, \boldsymbol{y})$ is called as the partial computation offloading game for multi-user. Now, the concept of Nash equilibrium of the partial computation offloading game is introduced as follows:

Definition 1. The Nash equilibrium of the partial computation offloading game for multi-user is $(\boldsymbol{x}^*, \boldsymbol{y}^*) = ((x_1^*, y_1^*), \ldots, (x_U^*, y_U^*))$ if at the equilibrium $(\boldsymbol{x}^*, \boldsymbol{y}^*)$, no user can further reduce his/her computation overhead by unilaterally changing his/her strategy, i.e.,

$$V_u((x_u^*, y_u^*), (x_{-u}^*, y_{-u}^*)) \leq V_u((x_u, y_u), (x_{-u}^*, y_{-u}^*)), \tag{15}$$

for $\forall (x_u, y_u) \in \{(x_u, y_u)\}_{x_u \in [0,1], y_u \in \mathscr{Y}}$ and $\forall u \in \mathscr{U}$.

4.2. Game properties

In this section, the properties of the partial computation offloading game model (14) will be discussed. Firstly, the partial computation offloading game model with two groups of variables, i.e. $\boldsymbol{x}$ and $\boldsymbol{y}$, is converted to the partial computation offloading game model with single group of variables. Then, the existence of the Nash equilibrium of the later is discussed, which also is the existence discussion of the Nash equilibrium of the former.

For simplification, the decision profiles $\boldsymbol{x}$ and $\boldsymbol{y}$ are redefined as decision profile $\boldsymbol{h}$, where $\boldsymbol{h} = (h_1, h_2, \ldots, h_U)$, and $h_u \in [0, L]$ is the offloading decision of MDU u. When $h_u = 0$, the MDU u choose to accomplish

his/her overall task locally on his/her own MD. $\langle h_u \rangle$ is defined as the ration of the offloading task to the computation task of MDU u. $\langle h_u \rangle$ is of the form

$$\langle h_u \rangle = \begin{cases} 1 \quad \text{if} \quad h_u = l, l \in \mathscr{L}, \\ \{h_u\} \quad \text{if} \quad h_u \in \bigcup_{l \in \mathscr{L}} (l-1, l), \\ 0 \quad \text{if} \quad h_u = 0. \end{cases} \tag{16}$$

where $\{h_u\}$ the fractional part of h_u.

Then, the uplink data rate $r_u(\boldsymbol{x}, \boldsymbol{y})$ of MDU u is converted to $r_u(\boldsymbol{h})$, i.e.

$$r_u(\boldsymbol{h}) = \omega log_2(1 + \frac{p_u g_u}{\omega\sigma^2 + \sum_{\{i \in \mathscr{U} | [h_i] = [h_u] > 0, g_i \leq g_u, i \neq u\}} p_i g_i}). \tag{17}$$

Meanwhile, the MD computation execution time, MD computation energy and MD computing overhead are successively converted to

$$t_u^{MD}(\langle h_u \rangle) = \frac{d_u(1 - \langle h_u \rangle)}{c_u^{MD}}, \tag{18}$$

$$e_u^{MD}(\langle h_u \rangle) = e_u^{'} d_u (1 - \langle h_u \rangle), \tag{19}$$

$$O_u^{MD}(\langle h_u \rangle) = \frac{w_u^t t_u^{MD}(\langle h_u \rangle)}{t_u^{MD}} + \frac{(1 - w_u^t) e_u^{MD}(\langle h_u \rangle)}{e_u^{MD}}. \tag{20}$$

Similarly, the transmitting time, the transmitting energy, ES computation time and ES computation overhead are successively converted to

$$t_{u,tr}^{ES}(\boldsymbol{h}) = \frac{s_u \langle h_u \rangle}{r_u(\boldsymbol{h})}, \tag{21}$$

$$e_{u,tr}^{ES}(\boldsymbol{h}) = \frac{p_u s_u \langle h_u \rangle}{r_u(\boldsymbol{h})}, \tag{22}$$

$$t_{u,exe}^{ES}(\langle h_u \rangle) = \frac{d_u \langle h_u \rangle}{c_u^{ES}}, \tag{23}$$

$$O_u^{ES}(\boldsymbol{h}) = \frac{w_u^t (t_{u,tr}^{ES}(\boldsymbol{h}) + t_{u,exe}^{ES}(\langle h_u \rangle))}{t_u^{MD}} + \frac{(1 - w_u^t) e_{u,tr}^{ES}(\boldsymbol{h})}{e_u^{MD}}. \tag{24}$$

Furthermore, the computation overhead of MDU u in terms of computation execution time and computation energy takes another form

$$V_u(\boldsymbol{h}) = O_u^{MD}(\langle h_u \rangle) + O_u^{ES}(\boldsymbol{h}), \tag{25}$$

and the objective of MDU u is

$$\min_{h_u \in [0, L]} V_u(\boldsymbol{h}), \forall u \in \mathscr{U}. \tag{26}$$

Finally, the partial computation offloading game model $\Gamma(\boldsymbol{x}, \boldsymbol{y})$ with two groups of variables is transformed to $\Gamma(\boldsymbol{h})$ with one group of variables, i.e.,

$$\Gamma(\boldsymbol{h}) = (\mathscr{U}, \{h_u\}_{u \in \mathscr{U}}, \{V_u(\boldsymbol{h})\}_{u \in \mathscr{U}}). \tag{27}$$

Then, the partial computation offloading decisions by all other users except MDU u could be defined as $h_{-u} = (h_1, \ldots, h_{u-1}, h_{u+1}, \ldots, h_U)$.

Definition 2. The Nash equilibrium of the partial computation offloading game for multi-user is $\boldsymbol{h}^* = (h_1^*, \ldots, h_U^*)$ if at the equilibrium $\boldsymbol{h}^*$, no user can further reduce his/her computation overhead by unilaterally changing his/her strategy, i.e.,

$$V_u(h_u^*, h_{-u}^*) \leq V_u(h_u, h_{-u}^*), \tag{28}$$

for $\forall h_u \in \{h_u\}$ and $\forall u \in \mathscr{U}$.

Lemma 1. *For the partial computation offloading game, if MDU u at Nash equilibrium $\boldsymbol{h}^*$ chooses to offload his/her computation task (i.e., $h^* > 0$), then MDU u must be a beneficial edge computing user.*

Obviously, if a MDU, who chooses to offload his/her computation task to ES, is not a beneficial edge computing user at the Nash equilibrium, then this user can improve its benefit by adjusting the offloading task ration of entire task, even just completing his/her computation task on his/her own MD. Of course, this case contradicts with the fact that no user can further reduce his/her computation overhead by unilaterally changing his/her strategy at Nash equilibrium. Furthermore, Nash equilibrium ensures that the MDUs can obtain a mutually satisfactory offloading strategy and no MDU has the incentive to deviate. Meanwhile, the partial computation offloading game can achieve a nice self-stability property. This property is very suitable for the scenario in this paper, because the MDs are owned by different MDUs and they may make their own offloading decisions in their own interests.

Theorem 1. *A Nash equilibrium exists in the partial computation offloading game* $\Gamma(\boldsymbol{h}) = (\mathscr{U}, \{h_u\}_{u \in \mathscr{U}}, \{V_u(\boldsymbol{h})\}_{u \in \mathscr{U}})$.

Proof. According to (16), we can know that, in the interval $(0, L]$, $\langle h_u \rangle$ is a periodic function in terms of h_u, i.e.,

$$\langle h_u \rangle(z) = \langle h_u \rangle(z+1), z \in (0, 1), l \in \mathscr{L}. \tag{29}$$

Obviously, during each period, the strategy space for each MDU is non-empty, convex, compact subset of the Euclidean space. According to (20), (24) and (25), we obtain

$$V_u(\boldsymbol{h}) = 1 + (\frac{w_u^t c_u^{MD}}{c_u^{ES}} - 1)\langle h_u \rangle + \frac{w_u^t s_u c_u^{MD} + (1 - w_u^t) p_u s_u / e_u^{'}}{r_u(\boldsymbol{h}) d_u} \langle h_u \rangle. \tag{30}$$

From (30), we can know that the h_u is apparently continuous in the interval of one period. Then, we will prove that the computation overhead of MDU u $V_u(\boldsymbol{h})$ has concavity during each period.

Let h_u and $h_u^{'}$ be the decision profile of MDU u. According to (30), we have

$$\begin{aligned} V_u(\frac{h_u + h_u^{'}}{2}, h_{-u}) = & 1 + (\frac{w_u^t c_u^{MD}}{c_u^{ES}} - 1) < \frac{h_u + h_u^{'}}{2} > + \\ & \frac{w_u^t s_u c_u^{MD} + (1 - w_u^t) p_u s_u / e_u^{'}}{r_u(\boldsymbol{h}) d_u} \cdot < \frac{h_u + h_u^{'}}{2} >, \end{aligned} \tag{31}$$

and

$$\begin{aligned} \frac{V_u(h_u, h_{-u}) + V_u(h_u^{'}, h_{-u})}{2} = & 1 + (\frac{w_u^t c_u^{MD}}{c_u^{ES}} - 1) \frac{< h_u > + < h_u^{'} >}{2} + \\ & \frac{w_u^t s_u c_u^{MD} + (1 - w_u^t) p_u s_u / e_u^{'}}{2 d_u} (\frac{\langle h_u \rangle}{r_u(\boldsymbol{h})} + \frac{\langle h_u^{'} \rangle}{r_u^{'}(\boldsymbol{h})}). \end{aligned} \tag{32}$$

where $r_u^{'}(\boldsymbol{h})$ is the interference received by MDU u with decision profile $(h_u^{'}, h_{-u})$. When the decision $\lceil h_u \rceil = \lceil h_u^{'} \rceil$, i.e., MDU u uses the same WC with these two decisions, $\langle \frac{h_u + h_u^{'}}{2} \rangle = \frac{\langle h_u \rangle + \langle h_u^{'} \rangle}{2}$ and $r_u(\boldsymbol{h}) = r_u^{'}(\boldsymbol{h})$. In this case, $V_u(\frac{h_u + h_u^{'}}{2}, h_{-u}) = \frac{V_u(h_u, h_{-u}) + V_u(h_u^{'}, h_{-u})}{2}$. When the decision $\lceil h_u \rceil \neq \lceil h_u^{'} \rceil$, i.e., MDU u changes his/her WC choice from WC $\lceil h_u \rceil$ to WC $\lceil h_u^{'} \rceil$, $\langle \frac{h_u + h_u^{'}}{2} \rangle > \frac{\langle h_u \rangle + \langle h_u^{'} \rangle}{2}$. Meanwhile, the relationship between $r_u(\boldsymbol{h})$ and $r_u^{'}(\boldsymbol{h})$ has three scenarios, i.e., $r_u(\boldsymbol{h}) > r_u^{'}(\boldsymbol{h})$, $r_u(\boldsymbol{h}) = r_u^{'}(\boldsymbol{h})$ and $r_u(\boldsymbol{h}) < r_u^{'}(\boldsymbol{h})$. However, if a MDU choose to change his/her choice from WC $\lceil h_u \rceil$ to WC $\lceil h_u^{'} \rceil$, the interference in WC $\lceil h_u^{'} \rceil$ must be less than that in WC $\lceil h_u \rceil$. Otherwise, the MDU prefers no change due to the increase of his/her computation overhead. So, there exists only one relationsip between $r_u(\boldsymbol{h})$ and $r_u^{'}(\boldsymbol{h})$ suiting for the scenario studied by us, i.e., $r_u(\boldsymbol{h}) < r_u^{'}(\boldsymbol{h})$. Therefore,

$$\begin{aligned} & \frac{w_u^t s_u c_u^{MD} + (1 - w_u^t) p_u s_u / e_u^{'}}{r_u(\boldsymbol{h}) d_u} \cdot < \frac{h_u + h_u^{'}}{2} > \leq \\ & \frac{w_u^t s_u c_u^{MD} + (1 - w_u^t) p_u s_u / e_u^{'}}{2 d_u} (\frac{\langle h_u \rangle}{r_u(\boldsymbol{h})} + \frac{\langle h_u^{'} \rangle}{r_u^{'}(\boldsymbol{h})}). \end{aligned} \tag{30}$$

According to Lemma 1, we can know $\frac{w_u^t d_u}{c_u^{ES}} - \frac{w_u^t d_u}{c_u^{MD}} - (1 - w_u^t) e_u^{'} d_u \leq 0$. Thus,

$$(\frac{w_u^t c_u^{MD}}{c_u^{ES}} - 1) < \frac{h_u + h_u^{'}}{2} > \leq (\frac{w_u^t c_u^{MD}}{c_u^{ES}} - 1) \frac{< h_u > + < h_u^{'} >}{2}. \tag{31}$$

Taking into account (30) and (31), we get the form

$$V_u(\frac{h_u + h_u^{'}}{2}, h_{-u}) < \frac{V_u(h_u, h_{-u}) + V_u(h_u^{'}, h_{-u})}{2}. \tag{32}$$

Therefore, the computation overhead of MDU u $V_u(\boldsymbol{h})$ is strictly concave with respect to h_u during each periodic interval. Accordingly, during each periodic interval, the Nash equilibrium of the partial computation offloading game $\Gamma(\boldsymbol{h})$ exists [37].

Furthermore, if each periodic interval is regarded as one choice for each MDU, the strategy space of the partial computation offloading game $\Gamma(\boldsymbol{h})$ becomes $\mathscr{L}\bigcup\{0\}$. In this case, the Nash equilibrium existence of the partial computation offloading game $\Gamma(\boldsymbol{h})$ has been proofed by Chen in [1].

All in all, the Nash equilibrium of the partial computation offloading game $\Gamma(\boldsymbol{h})$ exists in strategy space $\{h_u\}$. The proof is now completed. □

The Nash equilibrium solution of MDU u is

$$h_u^* = \{h_u | V_u(h_u, h_{-u}) \le V_u(h_u^{'}, h_{-u}), \forall h_u^{'} \in [0, L] \ for \ given \ h_{-u}\}. \tag{33}$$

Thus, the Nash equilibrium of the partial computation offloading game $\Gamma(\boldsymbol{h})$ is

$$\begin{aligned}\boldsymbol{h}^* = \quad & \{(h_1^*, \dots, h_U^*) | V_u(h_u^*, h_{-u}^*) \le V_u(h_u, h_{-u}^*),\\ & \forall u \in \mathscr{U}, \forall h_u \in [0, L] \ for \ given \ h_{-u}^*\}.\end{aligned} \tag{34}$$

Theorem 2. *Given a decision profile* $\boldsymbol{h}$*, MDU* u *achieves beneficial edge computing if his/her interference from other users* $\gamma_u(\boldsymbol{h}) \triangleq \sum_{\{i\in\mathscr{U}|[h_i]=[h_u]>0, g_i\le g_u, i\ne u\}} p_i g_i$ *on the chosen WC* $\lceil h_u \rceil$ *satisfies that*

1) $\gamma_u(\boldsymbol{h}) \le T_u$, *if* $\frac{C_u^{ES}}{2C_u^{ES}+c_u^{MD}} > \langle h_u \rangle$, *or*

2) $\gamma_u(\boldsymbol{h}) \le G_u(h_u)$, *if* $\frac{C_u^{ES}}{2C_u^{ES}+c_u^{MD}} \le \langle h_u \rangle$,

where

$$T_u(h_u) \triangleq \frac{p_u g_u}{2^{\frac{w_u^t s_u c_u^{MD} + (1-w_u^t) p_u s_u / e_u^{'}}{(1+C_u^{MD}/C_u^{ES}) d_u \omega}} - 1} - \omega\sigma^2 \tag{35}$$

and

$$G_u(h_u) \triangleq \frac{p_u g_u}{2^{\frac{[w_u^t s_u c_u^{MD} + (1-w_u^t) p_u s_u / e_u^{'}]\langle h_u \rangle}{(1-\langle h_u \rangle - C_u^{MD}\langle h_u \rangle / C_u^{ES}) d_u \omega}} - 1} - \omega\sigma^2. \tag{36}$$

In addition, $\lceil h_u \rceil$ *denotes the minimum integer that is not less than* h_u.

Proof. For a given and indivisible computation task, the MD computation overhead is not less than the ES computation overhead if MDU u is a beneficial edge computing user, i.e.,

$$O_u^{MD}(\langle h_u \rangle) \ge O_u^{ES}(\boldsymbol{h}). \tag{37}$$

Taking into account (20) and (24), (37) is equivalent to

$$\begin{aligned}&\frac{w_u^t t_u^{MD}(\langle h_u \rangle)}{t_u^{MD}} + \frac{(1-w_u^t) e_u^{MD}(\langle h_u \rangle)}{e_u^{MD}} \ge\\ &\frac{w_u^t (t_{u,tr}^{ES}(\boldsymbol{h}) + t_{u,exe}^{ES}(\langle h_u \rangle))}{t_u^{MD}} + \frac{(1-w_u^t) e_{u,tr}^{ES}(\boldsymbol{h})}{e_u^{MD}}.\end{aligned} \tag{38}$$

Furthermore, taking into account (18)-(19) and (21)-(23), we have

$$\begin{aligned}&w_u^t \cdot \frac{d_u(1-\langle h_u \rangle)}{c_u^{MD}} / \frac{d_u}{c_u^{MD}} + (1-w_u^t) e_u^{'} d_u (1-\langle h_u \rangle) / (e_u^{'} d_u) \ge\\ &w_u^t \left(\frac{s_u \langle h_u \rangle}{r_u(\boldsymbol{h})} + \frac{d_u \langle h_u \rangle}{c_u^{ES}}\right) / \frac{d_u}{c_u^{MD}} + (1-w_u^t) \frac{p_u s_u \langle h_u \rangle}{r_u(\boldsymbol{h})} / (e_u^{'} d_u).\end{aligned} \tag{39}$$

That is

$$r_u(\boldsymbol{h}) \ge \frac{[w_u^t s_u c_u^{MD} + (1-w_u^t) p_u s_u / e_u^{'}]\langle h_u \rangle}{(1-\langle h_u \rangle - C_u^{MD}\langle h_u \rangle / C_u^{ES}) d_u}. \tag{40}$$

Then, according to (17), we have

$$\sum_{\{i\in\mathscr{U}|[h_i]=[h_u]>0, g_i\le g_u\}} p_i g_i \le \frac{p_u g_u}{2^{\frac{[w_u^t s_u c_u^{MD} + (1-w_u^t) p_u s_u / e_u^{'}]\langle h_u \rangle}{(1-\langle h_u \rangle - C_u^{MD}\langle h_u \rangle / C_u^{ES}) d_u \omega}} - 1} - \omega\sigma^2, \tag{41}$$

i.e., $\gamma_u(\boldsymbol{h}) \le G_u(h_u)$. Furthermore, we take the first order derivatives of (30) with respect to h_u in one periodic interval, which can be written as follows:

$$\frac{\partial V_u(\boldsymbol{h})}{\partial h_u} = \frac{w_u^t c_u^{MD}}{c_u^{ES}} - 1 + \frac{w_u^t s_u c_u^{MD} + (1-w_u^t) p_u s_u / e_u^{'}}{r_u(\boldsymbol{h}) d_u}. \tag{42}$$

Combining with (38), we know

$$\frac{\partial V_u(\boldsymbol{h})}{\partial h_u} < 0. \tag{43}$$

Then, we have

$$\sum_{\{i\in\mathscr{U}|[h_i]=[h_u]>0, g_i\le g_u\}} p_i g_i < \frac{p_u g_u}{2^{\frac{w_u^t s_u c_u^{MD} + (1-w_u^t) p_u s_u / e_u^{'}}{(1+C_u^{MD}/C_u^{ES}) d_u \omega}} - 1} - \omega\sigma^2, \tag{44}$$

i.e., $\gamma_u(\boldsymbol{h}) \le T_u$.

Obviously, $\gamma_u(\boldsymbol{h}) \le \min\{T_u, G_u(h_u)\}$. Then, we have $\gamma_u(\boldsymbol{h}) \le T_u$, if $\frac{C_u^{ES}}{2C_u^{ES}+c_u^{MD}} > \langle h_u \rangle$, or $\gamma_u(\boldsymbol{h}) \le G_u(h_u)$, if $\frac{C_u^{ES}}{2C_u^{ES}+c_u^{MD}} \le \langle h_u \rangle$. The proof is now completed. □

Corollary 1. *According to the scenario that we study, we have follow conclusions:*

(1) *The MDU u will offload his/her all computation task to ES if he/she has a chance to offload computation task;*
(2) *In order to further reduce the computation overhead, for the time-sensitive MDU u, the ration of the offloading task to the computation task is determined by*

$$\langle h_u \rangle = \frac{1}{\frac{C_u^{MD} s_u}{d_u r_u(\boldsymbol{h})} + \frac{C_u^{MD}}{C_u^{ES}} + 1}, \tag{45}$$

if MDU u has enough energy for his/her computation overhead.

Proof. According to (30), we know $\frac{\partial V_u(\boldsymbol{h})}{\partial h_u} > 0$, i.e., $V_u(\boldsymbol{h})$ is a monotonically increasing and continuous function in each periodic interval. Therefore, MDU u will choose the largest value of h_u in the periodic interval to minimize his/her own computation overhead. But, for the time-sensitive MDU, who has enough energy for his/her computation overhead, although the computation overhead of MDU u obtain the minimum value with the method mentioned above, it still has the decrease possibility. This is because we neglect the computation time of MDU and ES in parallel. In order to illustrate this point, an not strict example is introduced as follows.

We assume that MDU u has a computation task. If MDU u complete his/her computation task on his/her own MD, it takes 800ms. If the task is processed only on ES, it takes 300ms. Now, if we choose a reasonable ration of the offloading task so that the local computation time is equal to the ES computation time, such as each computation time is 200ms within the ration, the overall time to complete the task will be 200ms. Here, it clearly implies that, for the time-sensitive MDU, the task that is processed only on ES has the lowest computation overhead, but, it may still have the decrease possibility in terms of computation time due to the parallel computation. In this case, to obtain the reasonable ration of the offloading task, we let

$$t_u^{MD}(\langle h_u \rangle) = t_{u,tr}^{ES}(\boldsymbol{h}) + t_{u,exe}^{ES}(\langle h_u \rangle). \tag{46}$$

Then, taking into account (18), (21) and (23), we have (45).

It is noteworthy that the two conclusion of this Corollary don't has contradiction. Because the former is to minimize the overall computation overhead. Although the latter obtains lower time overhead than the former, this needs to increase energy overhead. Actually, for the latter, the overall computation overhead in terms of time and energy is increased. Therefore, for latter, we assume that the MDU has enough energy for his/her computation overhead. Now, the proof is completed. □

Theorem 1 implies that the partial computation offloading game process can reach a Nash equilibrium. Theorem 2 gives the WC environment that is suitable for the computation task offloading of each MDU. Meanwhile, Corollary 1 represents how the MDUs offload their computation task. Next section, the algorithm design of the partial computation offloading game will be described.

5. PARTIAL COMPUTATION OFFLOADING ALGORITHM

In this section, the partial computation offloading algorithm is designed in algorithm 1. Firstly, the task offloading decision of each MDU is initialized to 0 (Algorithm 1 Line 1–3). In each slot, each MDU transmit the pilot signal on the WC to the WBS within ES (Algorithm 1 Line 6). Then, the WBS will send the power information of all WCs to each MDU (Algorithm 1 Line 7). Furthermore, the available WC set A_u is achieved base on the power information and Theorem 2 (Algorithm 1 Line 8–9). If the available WC set is not empty, the MDU will choose the WC with the minimum interference to send request-to-update (RTU) message to ES for contending for the decision update opportunity. Without loss of generality, if there exist multiple WCs with the same interference, one WC will be randomly chosen from these WCs (Algorithm 1 Line 11). Accordingly, if MDU obtains the update-permission choice, he/she will choose his/her task offloading decision based on Corollary 1 for next slot. Otherwise, his/her current task offloading decision will be kept for next slot (Algorithm 1 12–15). Moreover, if there is no available WC for MDU u, also, his/her current task offloading decision will be kept for next slot (Algorithm 1 Line 18). Finally, when each MDU doesn't update their task offloading decision and choose the same task offloading decision at next slot, i.e., $h_u(t+1) = h_u(t)$, algorithm 1 will stop.

In each slot, all users will in parallel execute the operations on Line 5–19 in Algorithm 1. Most of the operations are only some basic arithmetical calculations. The key part is to find the WC with the minimum interference on Line 9–11 in Algorithm 1, which will sort L WCs and find the WC with the minimum interference. The time complexity of the sorting operation over L WCs is $O(L\log L)$. Therefore, the time complexity in each slot for the task offloading decision is $O(L\log L)$. Let S be the number of slots that is taken for Algorithm 1 to terminate. Then, the time complexity of Algorithm 1 is $O(SL\log L)$. According to Corollary 1, if MDU u has a chance to offload computation task, he/she will offload his/her all computation task to ES. This scenario is similar to the scenario studied by [1]. So, S has the upper bound, i.e., $S \leq \frac{P_{\max}^2 U^2}{2P_{\min}} + \frac{P_{\max} T_{\max} U}{P_{\min}}$, where $P_{\max} = \max_{u\in\mathscr{U}}\{p_u g_u\}$, $P_{\min} = \min_{u\in\mathscr{U}}\{p_u g_u\}$, and $T_{\max} = \max_{u\in\mathscr{U}}\{T_u, G_u(h_u)\}$.

In Algorithm 1, the slotted time structure is considered to update the partial computation offloading decision. During each slot, the operations on Line 5–19 in Algorithm 1 will be executed on each MDU in parallel, which mainly includes two stages, i.e., the power information collection and the offloading decision update. In the power information collection, each MDU choose his/her offloading decision during current slot and transmit the corresponding pilot signal to WBS via currently selected WC. Then, WBS builds the tuple set of power information $\kappa_l(\boldsymbol{h(t)}) \triangleq \{(p_i, g_i)|i \in \mathscr{U}, \lceil h_i \rceil = l\}$ of WC $l \in \mathscr{L}$. Furthermore, WBS feedbacks power information of all WCs to each MDU. In the offloading decision update, each MDU computes the channel interferences $\pi_u(h_u(t), h_{-u}(t)) \triangleq \{\sum_{(p_i,g_i)\in\kappa_l(\boldsymbol{h(t)}), g_i\leq g_u} p_i g_i, l \in \mathscr{L}\}$, $u \in \mathscr{U}$ on different channels. Then, the correspondingly improved update set of the offloading decisions is achieved, i.e.,

$$A_u = \{h_u^{'}|h_u^{'} = arg \min_{h_u\in[0,L]} V_u(h_u, h_{-u}(t)), V_u(h_u^{'}, h_{-u}(t)) < V_u(h_u(t), h_{-u}(t))\}. \tag{47}$$

If $A_u \neq \phi$, MDU u will choose the WC with the minimum interference to send request-to-update (RTU) message to ES for contending for the decision update opportunity. Otherwise, MDU u will locally finish his/her computation task, i.e., $h_u(t+1) = h_u(t)$. Furthermore, for each WC, WBS will randomly select one MDU from MDUs who simultaneously contend for the same WC. Meanwhile, WBS sends the update-permission (UP) message to the selected MDUs for making offloading decision update, i.e., $h_u(t+1) \in A_u$. MDUs who don't receive UP message will keep their offloading decisions, i.e., $h_u(t+1) = h_u(t)$.

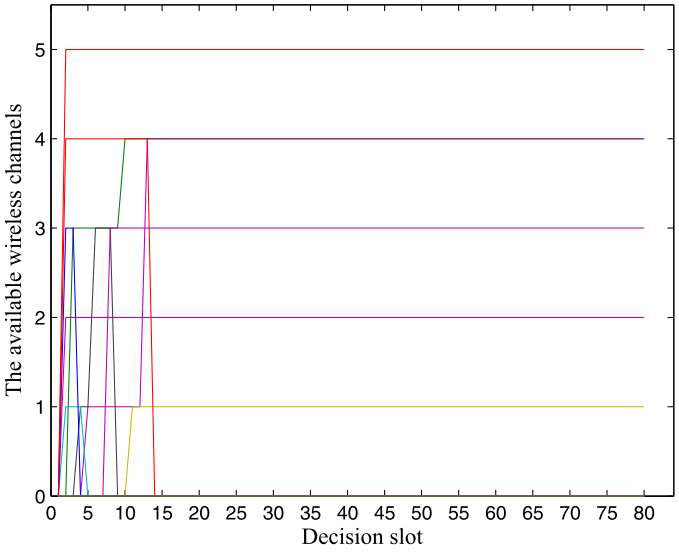

Fig. 2. The decision process by the MDUs.

6. NUMERICAL RESULTS

In this section, the partial computation offloading algorithm will be evaluated by numerical simulation. According to the experiment setting in [1,32], we design our experimental environment. In order to verify the availability of the proposed algorithm, we consider a scenario where the WBS has a circular coverage region with the diameter of 50m, and 35 MDUs are randomly scattered over the coverage region. Meanwhile, the WBS has 5 WCs ($L = 5$), and the WC bandwidth is 5 MHz ($\omega = 5$ MHz). The transmission power of each MDU is 100 mWatts ($p_u = 100$ mWatts), and the background noise power is -100 dBm ($\varpi_0 = -100$ dBm). Furthermore, the channel gain between MDU u and the WBS $g_u = dis(u, WBS)^{-\alpha}$, where $dis(u, WBS)$ is taken as the distance between MDU u and the WBS, and α is the path loss factor. According to [1,32], we set $\alpha = 4$.

Moreover, the face recognition application is regarded as the computation. Its data size is 420 KB, and the total number of the required CPU cycles is 1000 Megacycles. The computational capability of a MDU is randomly assigned from the set {0.5, 0.8, 1.0} GHz so that the MDUs have the heterogenous computational capability. Meanwhile, the computational capability of ES is 100 GHz. For each MDU, the weighting parameter of the computation execution time is randomly assigned from the set {0, 0.5, 1}. For ease of understanding, the parameter setting is shown in Table 2.

Firstly, the process of WC choice for all MDUs by the proposed partial computation offloading algorithm is shown in Fig. 2. Fig. 2 implies that the WC choice process will converge to stable state (i.e., the Nash equilibrium of the partial computation offloading game for multi-user) with the increase of the number of the decision slots. Thereby, all MDUs obtain their own optimal WC choice. Fig. 3 depicts the dynamics of the computation overhead of each MDU with the increase of the number of the decision slots. Obviously, after the WC choice is accomplished, the computation overhead of each MDU converge to a stable value. When the decision slot is 3, the MDU with blue line has the largest computation overhead than his/her other decision slot. This is because this MDU choose a available WC at decision slot 2. But, another users also choose the same WC with this MDU at decision 3. In this case, the computation overhead of this MDU became large according to (17). Thus, this MDU user need make a decision of WC choice to reduce his/her computation overhead at decision slot 4, i.e., the computation overhead of this MDU reduce at decision slot 4.

Table 2
List of simulation data parameters.

Parameter	Numerical value	Unit	Parameter	Numerical value	Unit
U	35	-	s_u	420	KB
L	5	-	d_u	1000	Megacycles
ω	5	MHz	C_u^{MD}	{0.5, 0.8, 1.0}	GHz
p_u	100	mWatts	C_u^{ES}	100	GHz
ϖ_0	-100	dBm	ω_u^t	{0, 0.5, 1.0}	-

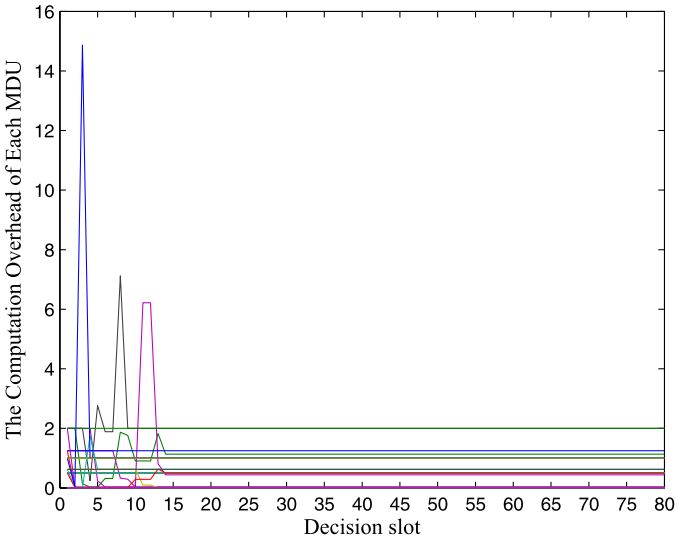

Fig. 3. The computation overhead of each MDU.

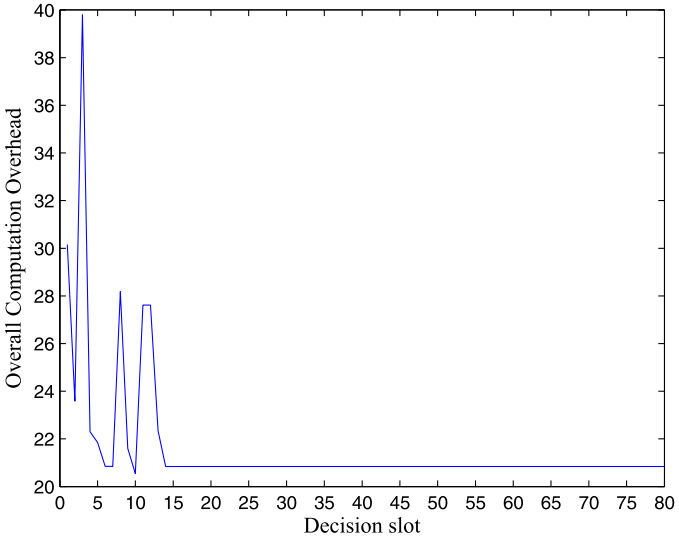

Fig. 4. The overall overhead of the system.

Furthermore, Fig. 4 describes the overall computation overhead of the mobile edge computing system. From Fig. 4, we can know that our proposed partial computation offloading algorithm can achieve optimal equilibrium by MDUs' game. Similar to Fig. 3, the overall computation overhead increases firstly and then decreases in the WC decision process. Finally, it achieves the equilibrium value. Fig. 5 shows the change trend of the number of the iterations of the decision process with different number of MDUs. With the increase of the number of MDUs, the mobile edge computation system needs to take more time to obtain the Nash equilibrium, i.e., the number of the iterations becomes large. In Fig. 5, the red line represents the change trend of the number of iterations with

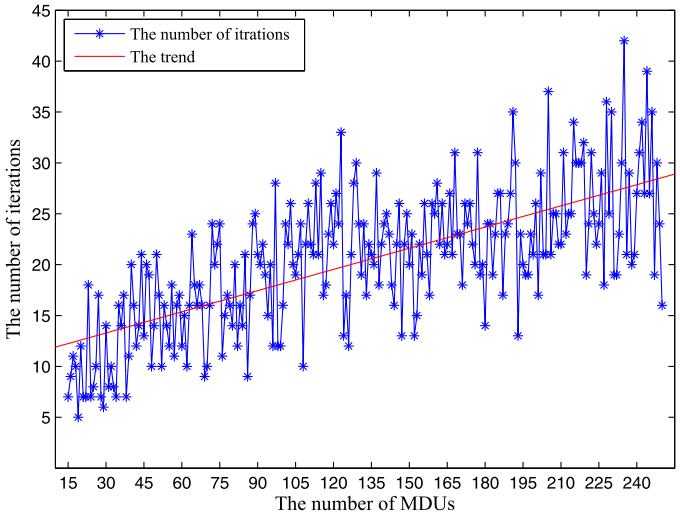

Fig. 5. The number of iterations with different number of the MDUs.

the increase of the number of MDUs. However, this trend just is the dynamic change of the overall system. This doesn't imply that it must suit for single sample. For instance, when the number of MDUs is 18, the number of iterations is 10. But when the number of MDUs is 19, the corresponding number of iterations is 5. The former number of iterations is twice as much as the latter number of iterations. This depends on not only the number of MDUs, but the channel gain, the randomness that the ES chooses MDUs, etc.

Furthermore, in order to verify the performance of our proposed algorithm, we compare the proposed algorithm with follow algorithms:

(1) **MD computing by all users (MDCBAU):** each MDU will accomplishes their tasks on his/her own MD. In this case, there no exist the phenomenon of the network resource contention so that the potential performance degradation will be avoided.
(2) **Edge computing by all users (ECBAU):** each MDU will offload all of their tasks to the ES, and ES will process MDUs' tasks. Meanwhile, the WCs will be randomly assigned to each MDU. In this scenario, the MDUs may be myopic, because each one doesn't consider the interference of other MDUs.
(3) **Centralized optimum based on cross entropy method (COCEM):** the globally optimal solution is achieved by the cross entropy method. The cross entropy method is efficient in finding near-optimal solutions for complex combinatorial optimization problems by adopting the randomized searching technique, which has been shown in [38].

With different number of MDUs, the comparison experiments are executed. Each experiment is repeat 50 times, and the average computation overhead of the mobile edge computing system is achieved. Fig. 6 describes the dynamic change of the computation overhead of the mobile edge computing system. Obviously, with the increase of the number of MDUs, the computation overhead has a increase trend on the whole. However, this trend just is the dynamic change of the overall system. This doesn't imply that it must suit for single sample. The rea-

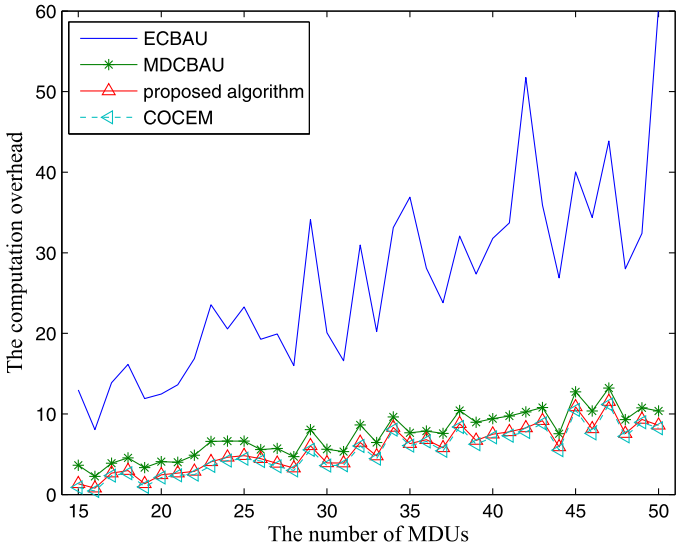

Fig. 6. The computation overhead of three algorithms with different number of the MDUs.

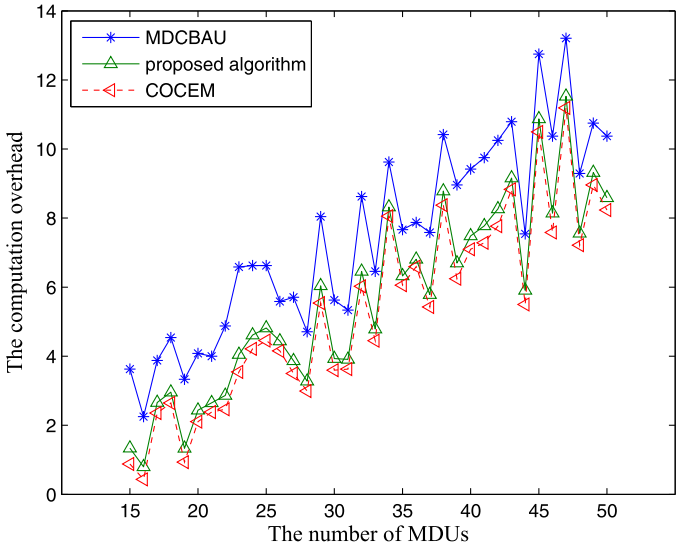

Fig. 7. The computation overhead of two algorithms with different number of the MDUs.

son is same with that of Fig. 5. Also, we can know, the performance of ECBAU algorithm is worst than that of other three algorithms, i.e., the computation overhead of the ECBAU algorithm far outweighs that of other three algorithms. Furthermore, in order to observe the compared result of other three algorithms, Fig. 7 is made.

From Fig. 7, we can know, whatever the number of MDUs is, the performance of MDCBAU algorithm always is worse than that of the proposed partial computation offloading algorithm. For example, when the number of MDUs is 30, the computation overhead of MDCBAU algorithm achieves up to 43.06% increase over that of the proposed algorithm. Averagely, the computation overhead of MDCBAU algorithm is improved up to 30.59% comparing with that of the proposed algorithm. In the MDCBAU algorithm, in order to avoid the potential performance degradation, all MDUs choose to accomplish their tasks on his/her own MDs. But, for a single MDU, he/she can obtain lower computation overhead by edge computing if there no exists the interference of other MDUs. Obviously, our proposed algorithm takes advantage of the edge computing and reasonably assigns the network resources for each MDUs. So, the performance of the proposed algorithm is better than

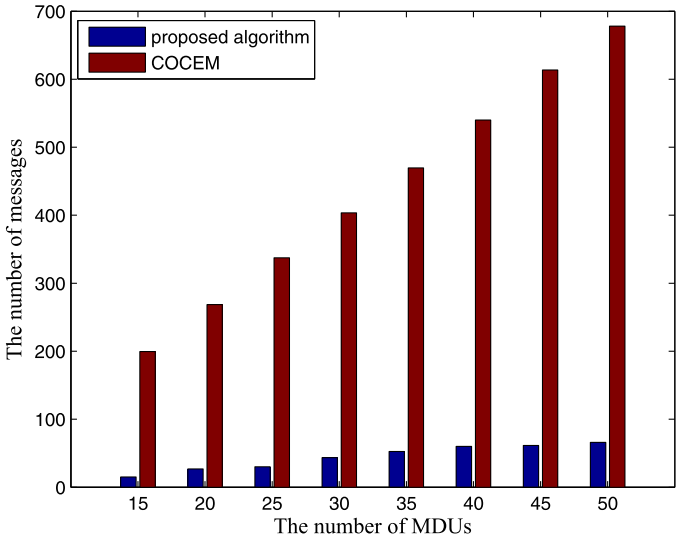

Fig. 8. The number of controlling and signaling messages.

that of MDCBAU algorithm. However, the performance of the proposed algorithm is worse than that of COCEM algorithm. For instance, when the number of MDUs is 30, the proposed algorithm gets up to 9.42% performance reduction over that of COCEM algorithm. Averagely, the computation overhead of the proposed algorithm is improved up to 6.98% comparing with that of COCEM algorithm. COCEM algorithm is the centralized computation offloading algorithm, which is different with the proposed distributed computation offloading algorithm.

Furthermore, in order to illustrate the advantage of the proposed distributed computation offloading algorithm, the number of controlling and signaling messages between MDUs and ES is shown in Fig. 8. Obviously, the number of controlling and signaling messages in the proposed algorithm is far less than that in COCEM algorithm. For example, when the number of MDUs is 30, the number of controlling and signaling messages in the proposed algorithm achieves up to 87.11% reduction over that in COCEM algorithm. Averagely, the number of controlling and signaling messages in the proposed algorithm can obtain up to 89.87% reduction over that in COCEM algorithm. The proposed algorithm belongs to the distributed computation offloading algorithm, in which the WC and decision update information only is exchanged by the MDUs who want update offloading decision. However, COCEM algorithm is the centralized computation offloading algorithm, in which each MDU should exchange his/her device parameters to ES, for instance, MD computation capacity, the channel gain, the transmission power, etc. So, the proposed algorithm can achieve better performance on the number of controlling and signaling messages, comparing with COCEM algorithm.

In order to verify the effectiveness of the Corollary 1 (2), we only consider the time-sensitive MDUs and each MDU has enough energy to accomplish themselves tasks. The task offloading decision $h_u(t+1)$ is made based on Corollary 1 (2), which is used to replace the decision process of $h_u(t+1)$ on Line 13 in Algorithm 1. This improved algorithm 1 is labeled as algorithm 2 that is named as "the partial computation offloading by only considering time overhead (PCOTO)" algorithm. Then, the MDCBAU algorithm and Algorithm 1 are taken as the comparison algorithms to evaluate the performance of PCOTO algorithm in terms of the time overhead of the mobile edge computing system.

Fig. 9 describes the dynamic change of the time overhead with different number of MDUs. Obviously, whatever the number of MDUs is, the time overhead of MDCBAU algorithm (i.e., the time overhead of MD computing) far outweighs that of other two algorithms. Furthermore, in order to observe the compared result of other two algorithms, Fig. 10 is made. From Fig. 10, we can know, whatever the number of MDUs is, the time overhead of Algorithm 1 is higher than that of Algorithm

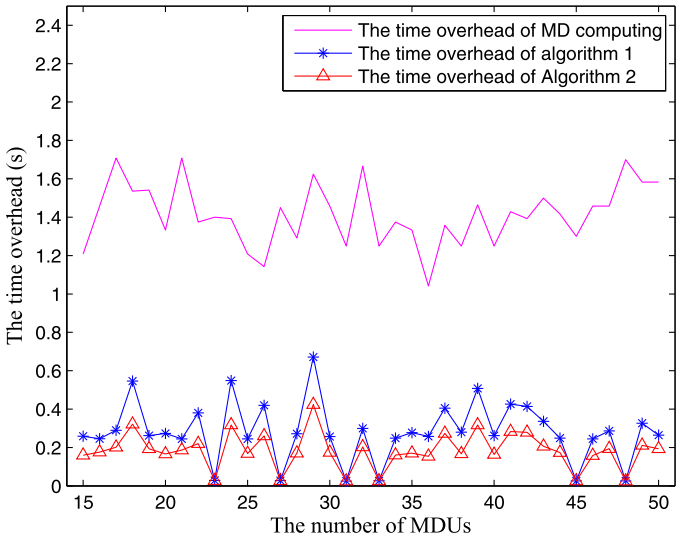

Fig. 9. The time overhead of three algorithms with different number of the MDUs.

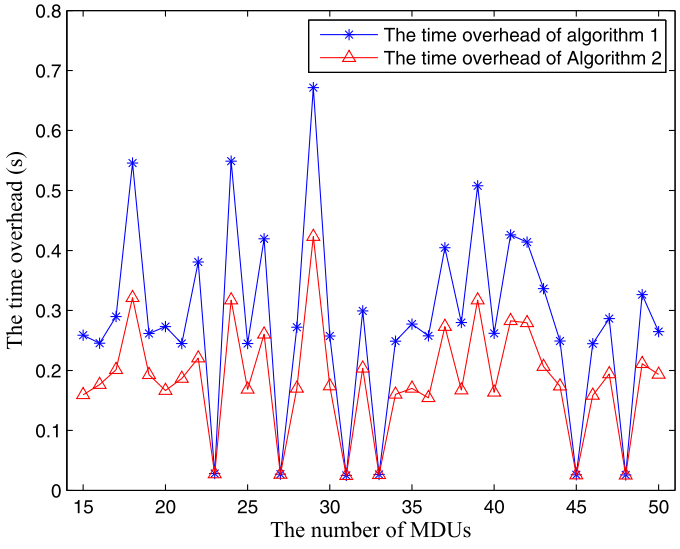

Fig. 10. The time overhead of two algorithms with different number of the MDUs.

2. For example, when the number of MDUs is 31, the time overhead of algorithm 1 is improved up to 2.03% comparing with that of Algorithm 2. Averagely, the time overhead of Algorithm 1 achieves up to 53.86% increase over that of Algorithm 2. When the task offloading decision is made, Algorithm 1 considers not only the time overhead, but also the energy overhead. However, Algorithm 2 only consider the time overhead and the energy of MDs is enough. Actually, Algorithm 2 reduces the time overhead at the expense of energy overhead. Therefore, the performance of Algorithm 2 is worse than that of Algorithm 1 if both time and energy overhead are considered. But, the performance of Algorithm 2 is better than that of Algorithm 1 if MDUs have enough energy and only pay attention to the time overhead.

In this section, we verify that the Nash equilibrium of the partial computation offloading game exists by experiments. Furthermore, in order to verify the effectiveness of the proposed partial computation offloading algorithm, MDCBAU algorithm and ECBAU algorithm are taken as the compared algorithms. The experimental results imply that the system computation overhead by the proposed partial computation offloading algorithm is less than that by MDCBAU algorithm or ECBAU algorithm. Finally, the conclusion (2) in Corollary 1 also is verified by experiments. The results represent that the system time overhead can be further reduced by conclusion (2) comparing with MDCBAU algorithm and Algorithm 1 if MDUs have enough energy and only pay attention to the time overhead.

Algorithm 1 Partial Computation Offloading Algorithm.

initialization:
each MDU **chooses** the task offloading decision $h_u(0) = 0$.
end initialization
repeat
  for each MDU u and each decision slot t in parallel.
  transmit the pilot signal on the chosen WC $h_u(t)$ to the WBS within ES.
  receive the power information of all WCs from the WBS.
  compute the each WC interference.
  acheive the available WC set A_u by **Theorem 2**.
  if $A_u \neq \phi$ **then**
    through the WC with the minimum interference, **send** RTU message to the ES for contending for the decision update opportunity. If there exist multiple WCs with the same interference, one WC will be randomly chosen from these WCs.
    if receive the UP message from ES **then**
      choose the task offloading decision $h_u(t+1)$ based on **Corollary 1 (1)** and the available WCs for next slot.
    else
      $h_u(t+1) = h_u(t)$ \\ The task offloading decision is unchange for next slot.
    end if
  else
    $h_u(t+1) = h_u(t)$ \\ The task offloading decision of current slot is chosen for next slot.
  end if
until END message from ES is received by each MDU

7. Conclusion

In this paper, we focus on the partial computation offloading problem for multi-user in mobile edge computing environment with multi-wireless channel. The computation overhead model is built based on game theory. Then, the existence of Nash equilibrium is proven. Furthermore, the partial computation offloading algorithm with low time complexity is given to achieve the Nash equilibrium. In addition, another partial computation task offloading mechanism for mobile device users, who has enough energy and only pay attention to the computation time overhead, is given to reduce the computation overhead. Finally, extensive experiments are conducted. The experimental results show that the system computation overhead by the proposed partial computation offloading algorithm is less than that by MDCBAU algorithm or ECBAU algorithm. Furthermore, the conclusion (2) in Corollary 1 also is verified by experiments. The results represent that the system time overhead can be further reduced by conclusion (2) comparing with MDCBAU algorithm and Algorithm 1 if MDUs have enough energy and only pay attention to the time overhead.

For the future work, the machine learning algorithms and the power control technology will be considered to solve the computation offloading problem in the mobile edge computing environment with real-time changes.

Declaration of Competing Interest

The authors declare that there is no conflict of interests regarding the publication of this paper.

Supplementary material

Supplementary material associated with this article can be found, in the online version, at doi:10.1016/j.comnet.2020.107334

CRediT authorship contribution statement

Shuchen Zhou: Conceptualization, Methodology, Software, Validation, Formal analysis, Writing - original draft, Writing - review & editing. **Waqas Jadoon:** Supervision.

References

[1] X. Chen, L. Jiao, W. Li, X. Fu, Efficient multi-user computation offloading for mobile-edge cloud computing, IEEE/ACM Trans. Networking 24 (5) (2016) 2795–2808, doi:10.1109/TNET.2015.2487344.

[2] N. Abbas, Y. Zhang, A. Taherkordi, T. Skeie, Mobile edge computing: a survey, IEEE Internet Things J. 5 (1) (2018) 450–465, doi:10.1109/JIOT.2017.2750180.

[3] O. Muñoz, A. Pascual-Iserte, J. Vidal, Optimization of radio and computational resources for energy efficiency in latency-constrained application offloading, IEEE Trans. Veh. Technol. 64 (10) (2015) 4738–4755, doi:10.1109/TVT.2014.2372852.

[4] S. Sardellitti, G. Scutari, S. Barbarossa, Joint optimization of radio and computational resources for multicell mobile-edge computing, IEEE Trans. Signal Inf. Process. Networks 1 (2) (2015) 89–103, doi:10.1109/TSIPN.2015.2448520.

[5] M. Kamoun, W. Labidi, M. Sarkiss, Joint resource allocation and offloading strategies in cloud enabled cellular networks, in: 2015 IEEE International Conference on Communications (ICC), 2015, pp. 5529–5534, doi:10.1109/ICC.2015.7249203.

[6] P.A. Apostolopoulos, E.E. Tsiropoulou, S. Papavassiliou, Game-theoretic learning-based qos satisfaction in autonomous mobile edge computing, in: 2018 Global Information Infrastructure and Networking Symposium (GIIS), 2018, pp. 1–5, doi:10.1109/GIIS.2018.8635770.

[7] T.X. Tran, D. Pompili, Joint task offloading and resource allocation for multi-server mobile-edge computing networks, IEEE Trans. Veh. Technol. 68 (1) (2019) 856–868, doi:10.1109/TVT.2018.2881191.

[8] K. Li, Computation offloading strategy optimization with multiple heterogeneous servers in mobile edge computing, IEEE Trans. Sustainable Comput. (2019), doi:10.1109/TSUSC.2019.2904680. 1–1

[9] W. Labidi, M. Sarkiss, M. Kamoun, Energy-optimal resource scheduling and computation offloading in small cell networks, in: 2015 22nd International Conference on Telecommunications (ICT), 2015, pp. 313–318, doi:10.1109/ICT.2015.7124703.

[10] Y. Zhao, S. Zhou, T. Zhao, Z. Niu, Energy-efficient task offloading for multiuser mobile cloud computing, in: 2015 IEEE/CIC International Conference on Communications in China (ICCC), 2015, pp. 1–5, doi:10.1109/ICCChina.2015.7448613.

[11] C. You, K. Huang, Multiuser resource allocation for mobile-edge computation offloading, in: 2016 IEEE Global Communications Conference (GLOBECOM), 2016, pp. 1–6, doi:10.1109/GLOCOM.2016.7842016.

[12] O. Muñoz, A. Pascual Iserte, J. Vidal, M. Molina, Energy-latency trade-off for multiuser wireless computation offloading, in: 2014 IEEE Wireless Communications and Networking Conference Workshops (WCNCW), 2014, pp. 29–33, doi:10.1109/WCNCW.2014.6934856.

[13] Y. Mao, J. Zhang, S.H. Song, K.B. Letaief, Power-delay tradeoff in multi-user mobile-edge computing systems, in: 2016 IEEE Global Communications Conference (GLOBECOM), 2016, pp. 1–6, doi:10.1109/GLOCOM.2016.7842160.

[14] M. Chen, U. Challita, W. Saad, C. Yin, M. Debbah, Artificial neural networks-based machine learning for wireless networks: a tutorial, IEEE Communications Surveys & Tutorials 21 (4) (2019) 3039–3071, doi:10.1109/comst.2019.2926625.

[15] S. Wang, M. Chen, C. Yin, W. Saad, C.S. Hong, S. Cui, H.V. Poor, Federated learning for task and resource allocation in wireless high altitude balloon networks arXiv:2003.09375v12020.

[16] Y. Ren, F. Zeng, W. Li, L. Meng, A low-cost edge server placement strategy in wireless metropolitan area networks, in: 2018 27th International Conference on Computer Communication and Networks (ICCCN), 2018, pp. 1–6, doi:10.1109/ICCCN.2018.8487438.

[17] T. Ouyang, R. Li, X. Chen, Z. Zhou, X. Tang, Adaptive user-managed service placement for mobile edge computing: An online learning approach, in: IEEE INFOCOM 2019 - IEEE Conference on Computer Communications, 2019, pp. 1468–1476, doi:10.1109/INFOCOM.2019.8737560.

[18] L. Chen, C. Shen, P. Zhou, J. Xu, Collaborative service placement for edge computing in dense small cell networks, IEEE Trans. Mob. Comput. (2019), doi:10.1109/TMC.2019.2945956. 1–1

[19] M. Bouet, V. Conan, Mobile edge computing resources optimization: ageo-clustering approach, IEEE Trans. Netw. Serv. Manage. 15 (2) (2018) 787–796, doi:10.1109/TNSM.2018.2816263.

[20] V. Burger, J.F. Pajo, O.R. Sanchez, M. Seufert, C. Schwartz, F. Wamser, F. Davoli, P. Tran-Gia, Load dynamics of a multiplayer online battle arena and simulative assessment of edge server placements, in: Proceedings of the 7th International Conference on Multimedia Systems, 2016, pp. 1–9.

[21] Y. Wang, S. Min, X. Wang, W. Liang, J. Li, Mobile-edge computing: partial computation offloading using dynamic voltage scaling, IEEE Trans. Commun. 64 (10) (2016) 4268–4282.

[22] O. Muñoz, A. Pascual-Iserte, J. Vidal, Optimization of radio and computational resources for energy efficiency in latency-constrained application offloading, IEEE Trans. Veh. Technol. 64 (10) (2015) 4738–4755.

[23] F. Jia, H. Zhang, H. Ji, X. Li, Distributed resource allocation and computation offloading scheme for cognitive mobile edge computing networks with noma, in: 2018 IEEE/CIC International Conference on Communications in China (ICCC), 2018, pp. 553–557, doi:10.1109/ICCChina.2018.8641192.

[24] Y. Pan, M. Chen, Z. Yang, N. Huang, M. Shikh-Bahaei, Energy-efficient noma-based mobile edge computing offloading, IEEE Commun. Lett. 23 (2) (2019) 310–313, doi:10.1109/LCOMM.2018.2882846.

[25] Y. Wu, L.P. Qian, K. Ni, C. Zhang, X. Shen, Delay-minimization nonorthogonal multiple access enabled multi-user mobile edge computation offloading, IEEE J Sel Top Signal Process 13 (3) (2019) 392–407, doi:10.1109/JSTSP.2019.2893057.

[26] , Resource allocation in next-Generation broadband wireless access networks, C. Singhal, S. De (Eds.), IGI Global, 2017, doi:10.4018/978-1-5225-2023-8.

[27] B.-G. Chun, S. Ihm, P. Maniatis, M. Naik, A. Patti, Clonecloud: Elastic execution between mobile device and cloud, in: Proceedings of the Sixth Conference on Computer Systems, EuroSys '11, ACM, New York, NY, USA, 2011, pp. 301–314, doi:10.1145/1966445.1966473.

[28] L. Yang, J. Cao, Y. Yuan, T. Li, A. Han, A. Chan, A framework for partitioning and execution of data stream applications in mobile cloud computing, ACM SIGMETRICS Performance Evaluation Review 40 (4) (2013) 23–32, doi:10.1145/2479942.2479946.

[29] Y. Wen, W. Zhang, H. Luo, Energy-optimal mobile application execution: Taming resource-poor mobile devices with cloud clones, in: 2012 Proceedings IEEE INFOCOM, 2012, pp. 2716–2720, doi:10.1109/INFCOM.2012.6195685.

[30] F. Wang, J. Xu, Z. Ding, Optimized multiuser computation offloading with multi-antenna noma, in: 2017 IEEE Globecom Workshops (GC Wkshps), 2017, pp. 1–7, doi:10.1109/GLOCOMW.2017.8269088.

[31] K. Kumar, Y. Lu, Cloud computing for mobile users: can offloading computation save energy? Computer (Long Beach Calif) 43 (4) (2010) 51–56, doi:10.1109/MC.2010.98.

[32] X. Chen, Decentralized computation offloading game for mobile cloud computing, IEEE Trans. Parallel Distrib. Syst. 26 (4) (2015) 974–983, doi:10.1109/TPDS.2014.2316834.

[33] , Game theory for wireless communications and networking, Y. Zhang, M. GUIZANI (Eds.), CRC Press, 2011, doi:10.1201/b10975.

[34] S. Josilo, G. Dan, Joint management of wireless and computing resources for computation offloading in mobile edge clouds, IEEE Trans. Cloud Comput. (2019), doi:10.1109/tcc.2019.2923768. 1–1

[35] S. Josilo, G. Dan, Computation offloading scheduling for periodic tasks in mobile edge computing, IEEE/ACM Trans. Networking 28 (2) (2020) 667–680, doi:10.1109/tnet.2020.2968209.

[36] A. MacKenzie, S. Wicker, Game theory in communications: motivation, explanation, and application to power control, in: GLOBECOM01. IEEE Global Telecommunications Conference (Cat. No.01CH37270), IEEE, 2001, doi:10.1109/glocom.2001.965533.

[37] Z. Han, D. Niyato, W. Saad, T. Baar, A. Hjrungnes, Game theory in wireless and communication networks: Theory, models, and applications, 1st, Cambridge University Press, New York, NY, USA, 2012.

[38] R.Y. Rubinstein, D.P. Kroese, The cross-Entropy method, Springer New York, 2004, doi:10.1007/978-1-4757-4321-0.

Shuchen Zhou received the BS degree in Computer Science and Technology from Henan university, Henan, China in 2007, and Master degree in Computer Application Technology from Henan university, Henan, China in 2010. He is currently a lecturer in Institute of International Education, Huanghuai University, China. His research interests include edge computing, information hiding, digital watermarking, etc. He has published more than 8 papers.