Collaborative Computation Offloading for Multiaccess Edge Computing Over Fiber–Wireless Networks

Hongzhi Guo, *Member, IEEE* and Jiajia Liu, *Senior Member, IEEE*

***Abstract*—By offloading the computation tasks of the mobile devices (MDs) to the edge server, mobile-edge computing (MEC) provides a new paradigm to meet the increasing computation demands from mobile applications. However, existing mobile-edge computation offloading (MECO) research only took the resource allocation between the MDs and the MEC servers into consideration, and ignored the huge computation resources in the centralized cloud computing center. Moreover, current MEC hosted networks mostly adopt the networking technology integrating cellular and backbone networks, which have the shortcomings of single access mode, high congestion, high latency, and high energy consumption. Toward this end, we introduce hybrid fiber–wireless (FiWi) networks to provide supports for the coexistence of centralized cloud and multiaccess edge computing, and present an architecture by adopting the FiWi access networks. The problem of cloud-MEC collaborative computation offloading is studied, and two schemes are proposed as our solutions, i.e., an approximation collaborative computation offloading scheme, and a game-theoretic collaborative computation offloading scheme. Numerical results corroborate that our solutions not only achieve better offloading performance than the available MECO schemes but also scale well with the increasing number of computation tasks.**

***Index Terms*—Mobile-edge computing, multi-access edge computing, MEC, computation offloading, collaborative computation offloading, fiber-wireless, FiWi.**

I. Introduction

With the popularity of smart mobile devices (MDs) such as smart phone, smart watch/band, virtual reality (VR) glass, etc. especially with the advent of more and more new Internet-of-Things (IoT) devices like shared bike, and shared mobile power supply, lots of new applications have emerged and attracted great attentions [1], [2]. For these new applications, e.g., face/fingerprint/iris recognition, augmented/virtual reality, natural language processing, and interactive gaming, they have one trait in common, i.e., computation intensive and high energy consumption [3]–[5]. However, due to MDs' physical size constraint, these applications are always restricted by MDs' limited computing capacity and battery power. The conflict between computation intensive applications and resource-constrained MDs brings an unprecedented challenge for the future mobile business development.

To address this challenge, the previously used method is to offload the MDs' computation tasks to the resource-rich cloud by adopting the mobile cloud computing technology, i.e., mobile computation offloading (MCO) [6]–[8]. However, it is noticed that traditional MCO approaches have the drawbacks of long latency and poor reliability caused by the data transmission through the wide area network. Considering the massive MDs and the low-latency requirement of new mobile applications, traditional MCO schemes are facing increasing difficulty to meet the MDs' new computation tasks. Recently, mobile-edge computing (MEC), which can provide cloud computing capabilities in close proximity to the mobile users, has been proposed as a key technique toward 5G [9], [10]. Offloading the MDs' computation tasks to the MEC server in close proximity, i.e., mobile-edge computation offloading (MECO), is envisioned to be a promising solution to address the above challenges [11]–[15]. Compared to traditional MCO schemes, MECO can achieve lower latency and higher reliability, and a number of research work on MECO has been proposed over the past several years [16], [17].

However, we note that existing research on MECO only took the computation resource allocation between the MDs and the MEC servers into consideration, and ignored the huge computation resources in the centralized cloud computing (CCC) center, e.g., Amazon AWS, Microsoft Azure, and Google GCE. Facing the increasing number of MDs and mobile applications, the resource bottleneck of MEC servers has been becoming more and more prominent, especially taking the network operators' capital expenditure (CAPEX) and operating expense (OPEX) into consideration. Moreover, although mobile users may experience long latency during the data exchange with the centralized cloud via the wide area network (WAN), the centralized cloud can provide the users with stringent cloud processing capability. These two technologies are complementary. Furthermore, it is noticed that the latency and the reliability in the WAN have been well improved with the widespread popularity of the passive optical networks (PONs) recently. Therefore, making the most of the

Manuscript received August 16, 2017; revised November 19, 2017; accepted December 22, 2017. Date of publication January 8, 2018; date of current version May 14, 2018. This work was supported in part by the National Natural Science Foundation of China (61771374, 61771373, and 61601357), in part by the China 111 Project under Grant B16037, and in part by the Fundamental Research Fund for the Central Universities JB171501. The review of this paper was coordinated by Igor Bisio. *(Corresponding author: Jiajia Liu.)*

H. Guo and J. Liu are with the State Key Laboratory of Integrated Services Networks, School of Cyber Engineering, Xidian University, Xi'an 710071, China (e-mail: hzguo@xidian.edu.cn; liujiajia@xidian.edu.cn).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TVT.2018.2790421

centralized cloud and distributed MEC resources and designing a collaborative computation offloading mechanism appear to be particularly necessary and important.

Moreover, it is noticed that current MEC Hosted Networks mostly adopt the networking technology integrating cellular and backbone networks, which have the shortcomings of single access mode, high congestion, high latency and high energy consumption. Obviously, limited by the outdated MEC Hosted Networks, this type of single access mode (i.e., one or more than one MEC servers connected to a cellular base station (BS)) has a poor flexibility. As stated before, with the popularity of PONs, a number of research work has been carried out to adopt PONs for cloud computing [18], [19]. Furthermore, we note that ESTI MEC group has changed its name from "Mobile-Edge Computing" to "Multi-access Edge Computing" very recently so as to support more access modes, e.g., WiFi and wired access [20]. By leveraging the complementary advantages of low-latency, high reliability, large capacity in PONs, and high mobility, good scalability, and supporting diverse wireless access technologies in wireless networks, hybrid fiber-wireless (FiWi) networks are envisioned to be a promising technology to support the coexistence of centralized cloud and distributed MEC services and to provide multiple radio access modes [21]–[25].

Toward this end, we introduce FiWi networks to provide supports for the coexistence of centralized cloud and multi-access edge computing, and study the problem of collaborative computation offloading in this paper. In particular, our major contributions are summarized as follows:

- In order to provide supports for the coexistence of centralized cloud and multi-access edge computing, we present a generic architecture by adopting the hybrid FiWi access networks.
- We study the problem of collaborative computation offloading with centralized cloud and MEC in a multi-channel interference environment, and formulate it as a constrained optimization problem, with the objective to minimize all the MDs' energy consumption while satisfying the MDs' computation execution time constraint given the number of wireless channels.
- In order to overcome the computational complexity of solving the collaborative computation offloading problem, we propose an approximation collaborative computation offloading scheme, where an approximation greedy strategy is adopted.
- To overcome the drawback of centralized management in the approximation approach (e.g., privacy concerns) and to provide supports for multi-user multi-MEC scenarios, we further propose a distributed collaborative computation offloading scheme by adopting the game theory, i.e., a game-theoretic collaborative computation offloading scheme. Besides, we analyze the Nash equilibrium and convergence of the proposed distributed scheme and give a brief theoretical proof.

The remainder of this paper is organized as follows. Section II discusses some related work. Section III presents the architecture for centralized cloud and MEC over FiWi networks. In Section IV, we first discuss the system model and formulate the problem of collaborative computation offloading with centralized cloud and MEC, and then present our step-by-step solutions. Moreover, Section V gives some analysis and discussions on our proposed schemes, and Section VI presents our numerical findings. Finally, Section VII concludes the whole paper.

II. Related Work

As a key technique in MEC, MECO has attracted significant attention and a number of MECO schemes have been proposed in recent years [26]–[28]. Chen *et al.* [11] studied the multi-user computation offloading problem for MEC in a multi-channel wireless interference environment, and proposed a game-theoretic computation offloading algorithm as their solution, where superior computation offloading performance in terms of energy consumption and computation execution time was reported. Zheng *et al.* [26] studied the problem of multi-user computation offloading for mobile cloud computing with active/inactive mobile users, and formulated the mobile users' dynamic offloading decision process as a stochastic game. After that, an efficient multi-agent stochastic learning algorithm was proposed as their solution.

Moreover, in order to reduce MDs' energy consumption and to further prolong their battery life, some researchers focused on introducing energy harvesting or dynamic voltage scaling technologies into the design of MECO schemes [5], [27]. Mao *et al.* [27] studied the dynamic MECO problem with energy harvesting devices, and proposed a low-complexity online algorithm as their solution, i.e., the Lyapunov optimiazation-based dynamic computation offloading algorithm, which jointly decides the offloading decision, the CPU-cycle frequencies for mobile execution, and the MDs' transmit power for computation offloading. In order to minimize both the tasks' execution latency and the MD's energy consumption in the MECO scenario where a single MD can offload tasks to multiple wireless access points (APs), Dinh *et al.* [13] presented a computational offloading framework by jointly optimizing the task allocation decision and the MD's CPU frequency, and two semidefinite relaxation (SDR) based algorithms were proposed as their solutions for the fixed and the elastic CPU frequency cases, respectively.

Besides, there are also some researchers focusing on introducing PONs to provide supports for cloud computing/edge computing services [1], [18], [19]. For instance, Reaz *et al.* [19] proposed an access network design integrating a cloud with a hybrid wireless-optical broadband access network (WOBAN), i.e., cloud-integrated WOBAN, which has the benefits of resource utilization improvement, higher scalability, and better flexibility to diverse cloud services. Moreover, Rimal *et al.* [1], [29] explored the possibilities of empowering integrated FiWi access networks to offer MEC capabilities in the 5G era, and studied the envisioned design scenarios for MEC over FiWi networks. After that, they proposed a unified resource management scheme for MEC-over-Ethernet-based FiWi networks as their exploratory work for MEC over FiWi networks in the 5G era.

However, existing research work on MECO only focused on the computation offloading between the MDs and the MEC

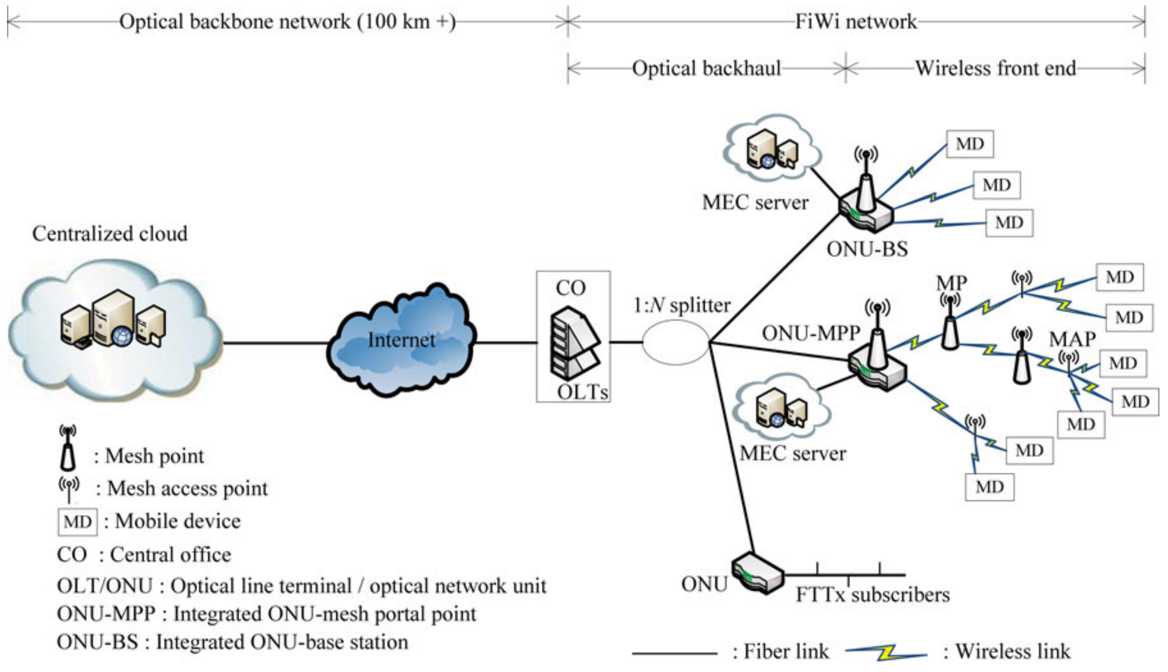


Fig. 1. Generic architecture for centralized cloud and multi-access edge computing over FiWi networks.

servers, and ignored the huge computation resources in the CCC center. In this paper, we will design a collaborative computation offloading scheme for centralized cloud and multi-access edge computing by adopting hybrid FiWi access networks as MEC Hosted Networks, where the computation resources of the MDs, the MEC and CCC servers can be utilized adequately.

III. Architecture for Collaborative Computation Offloading Over FiWi Networks

Fig. 1 depicts a generic architecture with coexistence of centralized cloud and MEC over FiWi networks, which consists of the optical backbone network connecting to the CCC center thousands of miles away, and the FiWi Hosted Networks providing MEC services. It is noticed that the long latency with increasing distance has been one of the critical issues in traditional backbone networks. However, with the development of optical fiber technology, this issue can be well addressed by adopting ultra-low loss and low latency fibers, and this type of optical fiber technology has being implemented by some telecom operators [30], [31].

The FiWi networks can further be divided into two parts, i.e., the optical backhaul, which provides the users with broadband access and connectivity between the MDs and the CCC servers, and the wireless front end providing the MDs with access services [32], [33]. In this paper, we adopt a widespread and cost-effective Ethernet Passive Optical Network (EPON)/10G-EPON with its tree-and-branch topology in the optical backhaul. Generally, EPON can cover a range of about 20 km. In EPON, one or more than one optical line terminals (OLTs) at the service provider's central office (CO) form the root, and connect to the optical backbone via a fiber link so as to provide cloud computing services. Three subsets of optical network units (ONUs) at the premises, which connect to the OLT via a $1 : N$ ($1 : 32$ or $1 : 64$) splitter, form the leaf nodes. By adopting the OLTs located at it, the CO is responsible for managing the information transmission between the MDs and their associated ONUs as well as acting as a gateway to other networks.

In particular, the fist subset of ONUs, which is located as the premises of residual/business subscribers, provides fiber-to-the-X (FTTx) services (e.g., FTTH, and FTTO) to the wired subscribers. The second subset of ONUs equipped with a cellular BS, i.e., the collocated ONU-BS, serves as the interface between the optical backhaul and the wireless front end and provides the MDs with wireless access services. Furthermore, the third subset of ONUs is equipped with mesh portal points (MPPs), i.e., the collocated ONU-MPPs, and connects the wireless mesh network (WMN) in the wireless front end to the optical backhaul of the FiWi network. Thereinto, the mesh access points (MAPs) provide direct wireless access services to the MDs within their coverage area, and the mesh points (MPs) placed between the MPPs and the MAPs act as the relay nodes in the WMN.

In the wireless front end, a MD, e.g., smart phone, smart watch, VR glass, IoT device, may access the Internet directly via the ONU-BS or through the multi-hop WMN network. Note that these have no influence on our following computation offloading design, since the hops between a MD and an ONU-BS/MPP do not affect the computation model of the processing time. In order to provide the MDs with MEC services, one or more MEC servers are connected to the ONU-BSs/MPPs via dedicated fiber links. In this paper, only one MEC server connected to a single ONU-BS/MPP is considered, as illustrated in Fig. 1, and more than one MEC servers connected to a ONU-BS/MPP with resource allocation among them will be investigated in our future work.

By adopting our proposed architecture, the computation tasks from the MDs can either be executed locally by adopting the MDs' own computation resources, or be offloaded to the MEC server in close proximity and be accomplished on the MEC server, or be offloaded to the CCC server thousands of mile away and be executed on the CCC server. Depending on the MDs' computation tasks, the computing ability of the MEC servers, and the consumed energy of the MDs. and the processing time of the computation tasks, we can design corresponding energy-efficient computation offloading strategies by adopting the centralized cloud and distributed MEC resources collaboratively, as to be studied in greater detail in the following sections.

IV. Collaborative Computation Offloading With Centralized Cloud and MEC

In this section, we first introduce the system model, and then formulate the problem of collaborative computation offloading with centralized cloud and MEC. After that, we present our solutions to the problem, i.e., an approximation collaborative computation offloading scheme, and a distributed game-theoretic collaborative computation offloading scheme.

A. System Model

As depicted in Fig. 1, we consider a set of collocated MDs, which is denoted by $\mathcal{N} = \{1, 2, ..., N\}$. Without loss of generality, we assume that for each MD i, there is only one computationally intensive task T_i to be completed within a period of time, and each computation task is atomic and cannot be divided. Moreover, the computation task T_i can be denoted as $T_i = \{m_i, c_i, t_i^{\max}\}$, where m_i is the size of computation input data, e.g., program codes, input data, and input parameters, c_i denotes the required CPU cycles to finish the task T_i, and $T_i^{\max}$ is the maximum permissible latency for accomplishing T_i. A MD can apply the methods in [34], [35], and [36] to obtain the information of c_i for each type of computation tasks. For each MD i, there exists a wireless BS through which it can offload its computation task T_i to the MEC server in close proximity or the CCC server thousands of miles away. The wireless BS can be a WiFi AP, a macro eNB, or a small cell according to ETSI's latest action on renaming mobile-edge computing to multi-access edge computing. The case that a MD connects to more than one wireless BSs, e.g., LTE-A heterogenous network (HetNet), will be investigated in future work.

For the wireless BS, we suppose that there are K orthogonal frequency channels denoted as $\mathcal{K} = \{1, 2, ..., K\}$ and denote $\lambda_i \in \{0, 1, -1\}$ as the offloading decision of MD i, where $\lambda_i = 0$ means that MD i chooses to execute its computation task T_i locally by adopting its own computing ability, the offloading decision $\lambda_i = 1$ means that MD i decides to offload task T_i to the MEC server via a wireless channel, and the offloading decision $\lambda_i = -1$ is that MD i chooses to offload its task T_i to the CCC server along the FiWi and the optical backbone networks. Similar to other MECO research [11], [27], we consider a quasi-static scenario where the MD set $\mathcal{N}$ remains unchanged during a computation offloading period.

In the following, we present the computation model in terms of the energy consumption and the processing time by computing locally, offloading to the MEC server, and offloading the CCC server, respectively.

1) Local Execution Model: For the offloading decision $\lambda_i = 0$, MD i decides to execute its computation task T_i locally, we denote f_i^{loc} and γ_i^{loc} as the computing ability and the consumed energy per CPU cycle of MD i, respectively. Thus, the processing time and the energy consumption of task T_i by computing locally can be calculated as

$$t_i^{loc} = \frac{c_i}{f_i^{loc}}, \tag{1}$$

and

$$e_i^{loc} = c_i \cdot \gamma_i^{loc}. \tag{2}$$

2) Mobile-Edge Execution Model: For the offloading decision $\lambda_i = 1$, MD i chooses to offload its computation task T_i to the MEC server in close proximity, and the task will be executed on the MEC server. In such case, the processing time of task T_i includes not only the computation execution time on the MEC server but also the time cost for transmitting T_i via wireless access and the dedicated fiber link. Note that for most mobile applications, such as fingerprint/face/iris recognition, etc., the size of computation outcome is much smaller than that of the computation input data, we neglect the time cost for sending the computation results from the MEC server to the MDs. Therefore, the total processing time for task T_i by offloading to the MEC server can be computed as

$$t_i^{mec} = \frac{m_i}{r_i} + \frac{m_i}{c} + \frac{c_i}{f_i^{mec}}. \tag{3}$$

In (3), c is the uplink data rate of the optical fiber network, f_i^{mec} is the computing ability of the MEC server, which can be easily obtained by calling the kernel code. r_i denotes the uplink data rate of transmitting from MD i to the wireless BS via a wireless access channel, which can be given as

$$r_i = \omega \log_2 \left(1 + \frac{p_i \cdot g_i^s}{\sigma + \sum_{k \in \mathcal{N} \setminus \{i\} : \lambda_k = \lambda_i} p_k \cdot g_k^s} \right). \tag{4}$$

Here, ω and σ are the channel bandwidth and the background noise power, respectively, and g_i^s denotes the channel gain between MD i and BS s. Note that we adopt the wireless interference model while solving the computation offloading problem, where multiple MDs share the same spectrum resource simultaneously. Moreover, the corresponding energy consumption of MD i by offloading to the MEC server can be given as

$$e_i^{mec} = p_i \cdot t_i^{mec}. \tag{5}$$

In (5), p_i is the transmit power of MD i.

3) Centralized Cloud Execution Model: For the offloading decision $\lambda_i = -1$, MD i will offload its computation task T_i to the CCC center thousands of miles away via the FiWi and the optical backbone networks, and the CCC server will accomplish the computation task on behalf of MD i. Note that centralized cloud such as Amazon AWS and Microsoft Azure, always has sufficient cloud computation resources and very strong computing ability, so that the MDs' computation requirements can be well satisfied and the execution time by adopting the CCC server can be neglected.

To simplify our computation, only the transmission delay specific to the distance and the uplink propagation delay for the upstream transmission of the given computation data are calculated, since the the transmission delay specific to the distance dominates the whole packet delay [37], [38]. Thus, the total processing time of task T_i by offloading to the CCC server can be calculated by adding up the time cost for transmitting T_i via the wireless access, the transmission delay from the wireless BS to the CCC server via the optical backbone network, and the uplink propagation delay of the long-reach optical backbone network, i.e.,

$$t_i^{ccc} = \frac{m_i}{r_i} + n\frac{m_i}{c} + \tau, \tag{6}$$

Here, n is the number of optical amplifying from the ONU-BS to the CCC center thousands of miles away, and τ denotes the uplink propagation delay. The total consumed energy in this case is then given as

$$e_i^{ccc} = p_i \cdot t_i^{ccc}. \tag{7}$$

B. Problem Formulation

For each task T_i, it can be either computed locally by adopting the computing ability of MD i, or be offloaded to the MEC server in close proximity to MD i, or be offloaded the CCC server thousands of miles away. According to different execution and offloading strategies, the latency t_i and the consumed energy e_i of task T_i may be different. Considering that most MDs are battery-driven, our target is to make the most of the centralized cloud and distributed MEC resources, and to design an energy-efficient collaborative computation offloading scheme so as to minimize the consumed energy of MDs while satisfying the maximum permissible latency of all computation tasks and the total number of wireless channels. Specifically, the problem of optimal collaborative computation offloading among the MDs, the CCC and the MEC servers can be defined as follows.

Definition 1: Cloud-MEC Optimal Collaborative computation Offloading (CMOCO): Given the initial information, such as all the MDs' computation tasks, the computing ability and the consumed energy per CPU cycle of both the MDs and the MEC server, the transmit power of each MD, the number of wireless channels, and the uplink data rate of the wireless and the optical networks, the problem of CMOCO is find an optimal computation offloading decision profile of all the MDs that minimizes the total energy consumption while satisfying the maximum permissible processing time of each computation task and the total number of wireless channels, where the computation resources of the MDs, the MEC and the CCC servers are collaboratively utilized.

Based on the system model in Section IV-A, the problem of CMOCO can be formulated as

$$\begin{aligned} &\min_{\{\lambda_i\}} \sum_{i=1}^{N} \lambda_i \cdot e_i \\ &\text{s.t. } C1: \lambda_i \cdot t_i \le t_i^{\max},\ \forall\, i \in \mathcal{N}, \\ &\qquad C2: \sum_{i=1}^{N} |\lambda_i| \le K, \quad \forall\, i \in \mathcal{N}, \\ &\qquad C3: \lambda_i \in \{0, 1, -1\}, \quad \forall\, i \in \mathcal{N}. \end{aligned} \tag{8}$$

According to (3)–(7), e_i (i.e., the consumed energy of task T_i) can be given as

$$e_i = \begin{cases} e_i^{loc}, & \text{if } \lambda_i = 0, \\ e_i^{mec}, & \text{if } \lambda_i = 1, \\ e_i^{ccc}, & \text{if } \lambda_i = -1. \end{cases} \tag{9}$$

In (8), The constraints $C1$ ensure the maximum permissible latency for accomplishing task T_i, and the constraints $C2$ guarantee that the maximum occupied wireless channels by the MDs cannot be more than K, i.e., the total number of channels possessed by the wireless BS. For the constraints $C3$, they state that only one offloading decision can be chosen for each computation task T_i.

Moreover, we can see that with more computation tasks being offloading to the MEC/CCC server, the uplink data rate via wireless access decreases, resulting in a larger computation overhead in terms of processing time and energy consumption. Therefore, an energy-efficient collaborative computation offloading scheme not only depends on the execution time constraints of all tasks but also depends on the wireless transmission channel interference among the MDs. To solve the problem of CMOCO, it is to find an optimal computation offloading decision profile of all MDs, which has the minimum total energy consumption while satisfying the maximum execution latency constraints of all the computation tasks and the total wireless channel constraint. Unfortunately, this problem is NP-hard and we cannot find an optimal solution to it in polynomial time.

Theorem 1: The problem of CMOCO over FiWi networks is NP-hard.

Proof: From the discussions above, an optimal solution to the CMOCO problem is a computation decision list that minimizes the total energy consumption of the MDs while satisfying the maximum processing time of all the computation tasks and the total wireless channel constraint. It is noticed that this problem can be easily reduced to the classical maximum cardinality bin packing problem, which is NP-hard [38]. In particular, we can regard the N MDs and the K wireless channels in CMOCO as the items and the bins in the classical maximum cardinality bin packing problem, respectively. Therefore, the problem of CMOCO over FiWi networks is NP-hard. ■

C. Solutions

1) An Optimal Enumeration Collaborative Computation Offloading Solution: In order to solve the problem of CMOCO,

Algorithm 1: An optimal enumeration collaborative computation offloading algorithm (OECCO).

```
Input: $\mathcal{N}, \mathcal{K}, T_i; f_i^{loc}, f_i^{mec}; \gamma_i^{loc}, p_i; c, \tau, w, \sigma, \forall\, i \in \mathcal{N}$.
Output: an optimal computation offloading decision list
    $S$, and the total minimum energy consumption $E_{\min}$;
    otherwise, NIL.
 1: enumerate all optional offloading decision
    combinations of $\lambda_i$;
 2: discard these offloading decision combinations that
    fail to meet the given latency constraints of the
    computation tasks and the total number of wireless
    channels;
 3: compute the total energy consumption of remaining
    offloading decision combinations, and find an overall
    offloading decision combination $S$, which has the total
    minimum energy consumption $E_{\min}$;
 4: output $S$ and $E_{\min}$; otherwise, output NIL.
```

an intuitive method is to enumerate all possible offloading decision combinations of the MDs, and to find an optimal offloading decision combination of all the MDs that has the total minimum energy consumption while satisfying the MDs' maximum permissible execution time given the number of wireless channels, i.e., an optimal enumeration collaborative computation offloading scheme (OECCO). The details of OECCO are briefly described in Algorithm 1.

For OECCO in Algorithm 1, the following conclusion can be easily proved.

Proposition 1: OECCO in Algorithm 1 is surely able to obtain an optimal solution to the CMOCO problem.

Proof: Since OECCO traverses the whole solution space and enumerates all optional offloading decision combinations, it is obvious that OECCO in Algorithm 1 is able to obtain an optimal solution to the CMOCO problem.

OECCO is simple and effective, but it has a very high computational complexity, since OECCO has to enumerate all optional offloading decision combinations and compute their corresponding energy consumption so as to find an optimal offloading decision combination with the lowest total energy consumption. Thus, OECCO is unworkable for a large number of computation tasks, and it is just presented as a benchmark for our following schemes. More details on the computational complexity of OECCO will be discussed in Section V.

2) An Approximation Collaborative Computation Offloading Solution: As OECCO is unworkable for a large number of computation tasks, we further propose an approximation solution to the CMOCO problem, i.e., an approximation collaborative computation offloading scheme (ACCO). The details of ACCO are given in Algorithm 2.

In particular, ACCO in Algorithm 2 mainly consists of three stages. In the first stage from step 2 to step 7, we compute the processing time of each task T_i via computing locally, offloading to the MEC/CCC server without wireless interference, and discard those tasks failing to meet their maximum permissible execution latency constraints, i.e., $\min\{t_i^{loc}, t_i^{mec}, t_i^{ccc}\} > t_i^{\max}, \forall\, i \in \mathcal{N}$.

Algorithm 2: An approximation collaborative computation offloading algorithm (ACCO).

```
Input: $\mathcal{N}, \mathcal{K}, T_i; f_i^{loc}, f_i^{mec}, \gamma_i^{loc}, p_i; c, \tau, w, \sigma, \forall\, i \in \mathcal{N}$.
Output: an optimal offloading decision set $S$, and the
    total minimum energy consumption $E_{\min}$; otherwise,
    NIL.
 1: Initialize $\mathcal{M} = \mathcal{P} = S^{loc} = S^{mec} = S^{ccc} = \emptyset$,
    $E_{\min} = 0$, and the number of occupied wireless
    channels $k = 0$;
 2: for all $i \in \mathcal{N}$ do
 3:     compute $t_i^{loc}$, $t_i^{mec}$, and $t_i^{ccc}$ without wireless
        interference;
 4:     if $\min\{t_i^{loc}, t_i^{mec}, t_i^{ccc}\} > t_i^{\max}$ then
 5:       $\mathcal{N} = \mathcal{N} \backslash \{i\}$;
 6:     end if
 7: end for
 8: if $\mathcal{N} == \emptyset$ then
 9:     return NIL;
10: else
11:     for all $j \in \mathcal{N}$ do
12:       if $t_j^{loc} > t_j^{\max}$ &amp;&amp; $|\mathcal{M}| < K$ then
13:         $\mathcal{N} = \mathcal{N} \backslash \{j\}, \mathcal{M} = \mathcal{M} \cup \{j\}, \mathcal{P} = \mathcal{P} \cup \{j\}$;
14:         execute offloading procedure in Procedure 1;
15:       end if
16:     end for
17: end if
18: $\mathcal{P} = \emptyset$;
19: while $\mathcal{N} \neq \emptyset$ do
20:     for all $l \in \mathcal{N}$ do
21:       $\mathcal{N} = \mathcal{N} \backslash \{l\}, k = \min\{K, |\mathcal{N}| + |\mathcal{M}|\}$;
22:       compute $e_l^{loc}$, $e_l^{mec}$, and $e_l^{ccc}$;
23:       if $\min\{e_l^{mec}, e_l^{ccc}\} \leq e_l^{loc}$ && $|\mathcal{M}| < K$ then
24:         $\mathcal{M} = \mathcal{M} \cup \{l\}, \mathcal{P} = \mathcal{P} \cup \{l\}$;
25:         execute offloading procedure in Procedure 1;
26:       else
27:         $S^{loc} = S^{loc} \cup \{l\}$;
28:       end if
29:     end for
30: end while
31: compute the total energy consumption, $E_{\min}$;
32: return $S = \{S^{loc}, S^{mec}, S^{ccc}\}$ and $E_{\min}$;
```

Procedure 1: an offloading decision procedure.

```
1: for all $p \in \mathcal{P}$ do
2:     compute $e_p^{mec}, e_p^{ccc}$;
3:     if $e_p^{mec} \leq e_p^{ccc}$ then
4:       $S^{mec} = S^{mec} \cup \{p\}$;
5:     else
6:       $S^{ccc} = S^{ccc} \cup \{p\}$;
7:     end if
8: end for
```

Then, in the second stage from step 11 to step 16, we pick out the computation tasks that cannot be executed locally on the MDs from the remaining computation tasks, i.e., $t_i^{loc} > t_i^{\max}$ and $|\mathcal{M}| < K$. For these computation tasks, we compute their energy consumption separately by adopting offloading to the MEC server and by offloading to the CCC server, i.e., e_i^{mec} and e_i^{ccc}, and choose the offloading decisions with the lower energy consumption. Finally, for the remaining computation tasks, which can be executed locally on the MDs, or be offloaded to the MEC server via wireless access, or be offloading to the CCC server via the FiWi and optical backbone networks, we iteratively update the number of occupied wireless channels, and compute the consumed energy separately by adopting the offloading decisions loc, mec, and ccc. After that we compare the energy consumption among them and choose the offloading decisions with the lowest energy consumption.

For ACCO in Algorithm 2, we have the following conclusion.

Proposition 2: ACCO in Algorithm 2 can give a near-optimal solution to the problem of CMOCO.

Proof: According to the forall loop of steps 20–29 in Algorithm 2, ACCO adopts a greedy strategy to choose MD l's offloading decision, by adopting which the consumed energy of MD l is minimum. However, considering that the total number of wireless channels is limited and we randomly choose a MD l from the set of remaining MDs $\mathcal{N}$, ACCO in Algorithm 2 can only give a near-optimal solution to the problem of CMOCO. Locally optimal decisions cannot guarantee an optimal solution to CMOCO due to the random selection of MD l. ■

3) A Game-Theoretic Collaborative Computation Offloading Solution: Compared to OECCO, ACCO provides us with a practical collaborative computation offloading scheme as ACCO has a much higher computational efficiency and can give a near-optimal solution. However, we note that ACCO is a centralized optimization scheme, so that a centralized control center is required to collect the local parameters of the MDs and to manage the computation offloading over the whole network. If we leave privacy concerns out of consideration, it may not matter much with the scenarios of multi-users connecting to a wireless BS. But for the scenarios with more than one wireless BSs, more problems on the location of the centralized control center and the computation offloading among multiple MEC severs should be addressed. Thus, a distributed scheme without centralized management is required.

Moreover, it is noticed that game theory is widely adopted in solving the decision-making problems among multiple rational game players with contrasting goals [39]–[41]. Therefore, we develop a game-theoretic collaborative offloading solution to CMOCO in the following, i.e., a game-theoretic collaborative computation offloading scheme (GT-CCO). In particular, we can regard each MD in CMOCO as a rational game player, which reacts to other players' offloading decisions by carrying out its current optimal offloading decision. After a number of steps, all the users self-organize into a mutually equilibrium state, i.e., the Nash equilibrium, at which no user can further reduce its consumed energy by changing its offloading decision.

To simplify our following algorithm description and analysis, some notations are predefined as follows. We denote $\lambda_{-i} = \{\lambda_1, ..., \lambda_{i-1}, \lambda_{i+1}, ..., \lambda_N\}$ as the offloading decision profile of all the MDs excluding MD i during a computation offloading period. Given other MDs' offloading decisions, MD i would like to carry out its current optimal offloading decision so as to minimize its energy consumption, i.e.,

$$\begin{aligned} &\min_{\lambda_i \in \{0,1,-1\}} L_i(\lambda_i, \lambda_{-i}) \\ &\text{s.t. } \lambda_i \cdot t_i \le t_i^{\max}, \ \forall\, i \in \mathcal{N}, \\ &\qquad \sum_{i=1}^{N} |\lambda_i| \le K, \quad \forall\, i \in \mathcal{N}, \\ &\qquad \lambda_i \in \{0, 1, -1\}, \quad \forall\, i \in \mathcal{N}, \end{aligned} \tag{10}$$

Here, $L_i(\lambda_i, \lambda_{-i})$ is the energy consumption function of MD i, and the current optimal offloading decision λ_i^* of MD i can be defined as

Definition 2: Current optimal offloading decision: Given other MDs' offloading decision profile λ_{-i}, the current optimal offloading decision λ_i^* of MD i is an optimal offloading decision by adopting which MD i can achieve the minimum energy consumption at the next time slot, i.e., $L_i(\lambda_i^*, \lambda_{-i}) \le L_i(\lambda_i, \lambda_{-i}), \forall\, i \in \mathcal{N}$.

According to (2), (5), and (7), $L_i(\lambda_i, \lambda_{-i})$ can be computed as

$$L_i(\lambda_i, \lambda_{-i}) = \begin{cases} e_i^{loc}, & \text{if } \lambda_i = 0, \\ e_i^{mec}, & \text{if } \lambda_i = 1, \\ e_i^{ccc}, & \text{if } \lambda_i = -1. \end{cases} \tag{11}$$

After that, we can formulate the problem of CMOCO as a strategic game $\Gamma = (\mathcal{N}, \{\Lambda_i\}_{i \in \mathcal{N}}, \{L_i\}_{i \in \mathcal{N}})$, where the set of MDs $\mathcal{N}$ is the set of rational game players, Λ_i is the strategy set for player i, and the energy consumption function of MD i, i.e., $L_i(\lambda_i, \lambda_{-i})$, is the cost function to be minimized by player i.

The details of GT-CCO are briefly described in Algorithm 3. During each decision slot in GT-CCO, all the MDs first calculate their current optimal offloading decisions. Then the MDs whose offloading strategies should be updated in next time slot contend for the decision update opportunity. If MD j wins the update opportunity, it updates its offloading decision, otherwise, its offloading decision keeps unchanged. After a number of iterations, when no MD needs to update its offloading decision, that is all the MDs come to the Nash equilibrium state, the algorithm stops and an optimal offloading decision list of all the MDs can be obtained. More details on the Nash equilibrium and convergence of GT-CCO will be discussed in Section V.

For GT-CCO in Algorithm 3, the following conclusion can be easily proved.

Proposition 3: GT-CCO in Algorithm 3 can give a near-optimal solution to the problem of CMOCO.

Proof: It is noticed that at each iteration of GT-CCO, each MD calculates its current optimal offloading decision given other MDs' offloading decision profile, and MD j who wins the decision update opportunity updates its offloading decision. However, similar to ACCO in Algorithm 2, GT-CCO in Algorithm 3 choose MD j in random way, and locally

Algorithm 3: A game-theoretic collaborative computation offloading scheme (GT-CCO).

```
Input: $\mathcal{N}, \mathcal{K}, T_i; f_i^{loc}, f_i^{mec}, \gamma_i^{loc}, p_i; c, \tau, w, \sigma, \forall\, i \in \mathcal{N}$.
Output:   an optimal offloading decision set $S$, and the
     total minimum energy consumption $E_{\min}$.
 1: Initialize the number of occupied wireless channel
    $k = 0$, the offloading decision during time slot $t$
    $\lambda_i(t) = 0$, and $\lambda_i = 0, \forall\, i \in \mathcal{N}$;
 2: for each decision slot $t$ do
 3:    for all $i \in \mathcal{N}$ do
 4:      compute $r_i$ and $\lambda_i^*(t+1)$ with given $t_i^{\max}$ and the
         total wireless channel constraint;
 5:      store the MDs whose offloading strategies should
         be updated into $U(t)$;
 6:    end for
 7:    if $U(t) \neq \emptyset$ then
 8:      each MD in $U(t)$ contends for the decision update
         opportunity;
 9:      if MD $j$ wins the decision update opportunity then
10:        $\lambda_j = \lambda_j(t+1)$, and update $\lambda_j$ in $S$;
11:        broadcast the update message to other MDs;
12:      else
13:        $\lambda_j = \lambda_j(t)$;
14:      end if
15:    end if
16: end for
17: compute the total minimal energy consumption $E_{\min}$
    based on the calculated offloading decision list $S$;
18: return $S$ and $E_{\min}$.
```

optimal decisions cannot guarantee a global optimal solution to CMOCO. Therefore, GT-CCO in Algorithm 3 can only give a near-optimal solution to the problem of CMOCO. ∎

V. Analysis and Discussions

In this section, we analyze the Nash equilibrium and convergence of GT-CCO in Algorithm 3, and present some discussions on the computational complexity of our proposed schemes in this paper.

A. Analysis of Nash Equilibrium and Convergence

According to the discussions in Section IV-C3, we first give the definition of Nash equilibrium as follows.

Definition 3: Nash equilibrium (NE): For a computation offloading decision profile $\lambda^* = \{\lambda_1^*, \lambda_2^*, ..., \lambda_N^*\},\ \lambda^* \in \{\Lambda_i\}_{i\in\mathcal{N}}$, it is a NE for the collaborative computation offloading game if neither MD can further reduce its energy consumption by deviating unilaterally from the profile λ^*, i.e.,

$$L_i(\lambda_i^*, \lambda_{-i}^*) \le L_i(\lambda_i, \lambda_{-i}^*), \forall\, \lambda_i \in \Lambda_i, i \in \mathcal{N}. \tag{12}$$

For the collaborative computation offloading game, we have the following conclusion.

Theorem 2: The collaborative computation offloading game $\Gamma = (\mathcal{N}, \{\Lambda_i\}_{i\in\mathcal{N}}, \{L_i\}_{i\in\mathcal{N}})$ has a NE and possesses the finite improvement property.

Proof: In order to prove that the collaborative computation offloading game has a NE and possesses the finite improvement property (i.e., the game converges within a certain iterations.), the key step is to show that the game is a potential game [42]. For more details of the proof, please refer to the appendix. ∎

B. Analysis of Computational Complexity

As discussed above, OECCO in Algorithm 1 is simple and can give an optimal solution to the problem of CMOCO, but it has a very high computational complexity. The upper bound of its computational complexity is given as follows.

Proposition 4: The computational complexity of OECCO in Algorithm 1 is $O(3^n)$, where n is the number of computation tasks.

Proof: For OECCO in Algorithm 1, the running time of step 1 is $O(3^n)$, since there are three offloading choices for each computation task, i.e., computing locally, offloading to the MEC server, and offloading to the remote CCC server. In step 2 and step 3, in order to discard these offloading decision combinations that fail to meet the given latency and wireless channel constraints and to find an offloading decision combination with the total minimum energy consumption, we have to traverse all the offloading decision combinations, and the running time is also $O(3^n)$. Therefore, the computational complexity of OECCO can be calculated by adding them up, i.e., $O(3^n)$, where n is the number of computation tasks.

Compared to OECCO, although ACCO can only give a near-optimal solution to the problem of CMOCO, it has a much lower computation complexity and can be implemented with a large number of computation tasks in practice. In particular, the following conclusion can be easily proved for ACCO.

Proposition 5: The computational complexity of ACCO in Algorithm 2 is $O(n^2)$, where n is the number of computation tasks.

Proof: The running time of ACCO in Algorithm 2 mainly consists of three parts, i.e., the forall loop from step 2 to step 7, the forall loop from step 11 to step 16, and the while loop of steps 19–30. Specifically, the running time of steps 2–7 is $O(n)$. For steps of 11–16, the outer forall loop runs at most n times, and the running time of the inner procedure (i.e., Procedure 1) is $O(n)$. Thus, the computational complexity of steps 11–16 is $O(n \cdot n) = O(n^2)$. Moreover, for the while loop from step 19 to step 30, the outer while and forall loop runs at most n times, the running time of step 27 is $O(n)$, and thus the computational complexity of steps 19–30 is $O(n^2)$. Finally, the whole computational complexity of ACCO in Algorithm 2 can be calculated by adding them up, i.e., $O(n^2)$. ∎

As a centralized computation offloading scheme, centralized management is required for ACCO. To address this issue and provide a more general offloading scheme, we further proposed GT-CCO in Section IV-C3. For GT-CCO, the following conclusion can be easily obtained.

TABLE I
COMPARISONS AMONG OECCO, ACCO, AND GT-CCO, WHERE n AND C DENOTE THE NUMBER OF COMPUTATION TASKS AND THE ITERATIONS FOR THE CONVERGENCE OF GT-CCO, RESPECTIVELY

Algorithms	Computational complexity	Centralized or distributed?	Optimal solution or not?
OECCO	$O(3^n)$	Centralized	optimal
ACCO	$O(n^2)$	Centralized	Near-optimal
GT-CCO	$O(C \cdot n)$	Distributed	Near-optimal

Proposition 6: The computational complexity of GT-CCO in Algorithm 3 is $O(C \cdot n)$, where C is the number of iterations for the convergence of GT-CCO, and n is the number of computation tasks.

Proof: For GT-CCO in Algorithm 3, there are two-tier loops in total. According to Theorem 2, the outer for loop, which denotes the iteration process of the game players (i.e., the MDs in this paper) for reaching to the NE state, converges in finite times. We denote C as the number of iterations for the outer for loop in GT-CCO. Moreover, the inner forall loop of steps 3–6 runs at most n times, and the process from step 7 to step 15 takes constant time, i.e., $O(1)$. Finally, the total running time of GT-CCO can be calculated by multiplying them, i.e., $O(C \cdot n)$, where C is the number of iterations for the convergence of GT-CCO, and n is the number of computation tasks. ■

C. Discussions

In order to address the problem of collaborative computation offloading with coexistence of centralized cloud and distributed edge, we presented three solutions, i.e., OECCO, ACCO, and GT-CCO. Brief comparisons among them are summarized in Table I. Note that although it is assumed that there exists only one wireless BS and one MEC server in this paper, our final solution, i.e., the distributed GT-CCO, can also be applied to the scenarios with more than one BSs and MEC servers.

VI. PERFORMANCE EVALUATION

In this section, we presented some numerical results to evaluate our proposed collaborative computation offloading schemes over FiWi networks.

A. Parameter Settings

Without loss of generality, we adopt a network scenario with coexistence of centralized cloud and distributed MEC over FiWi networks, where there is an ONU-BS and a MEC server connected to the ONU-BS via a dedicated fiber link. The CCC server is assumed to be located 1000 km far away from the CO, and the wireless BS, which covers a range of 50 m, has a frequency resource of 50 orthogonal channels, which means that it can provide services for at most 50 MDs at a time. The wireless channel bandwidth is set to 40 MHz, and $N = 100$ MDs are randomly distributed in the coverage of the wireless BS. The uplink data rate of the optical fiber network is set to 1 Gbps, and the computing capacity of a MD and the MEC server is 0.8 GHz and 4 GHz, respectively. Moreover, the transmit power of a MD is set as 100 mW and the background noise σ is -100 dBm. The size of input computation data varies from 300 KB to 800 KB, and the number of CPU cycles required by a computation task T_i is set between 100 Megacycles and 1 Gigacycles. The maximum permissible processing time for a computation task T_i is assumed to be randomly distributed from 0.1 to 2 seconds. Table II lists the key parameters adopted in our simulation.

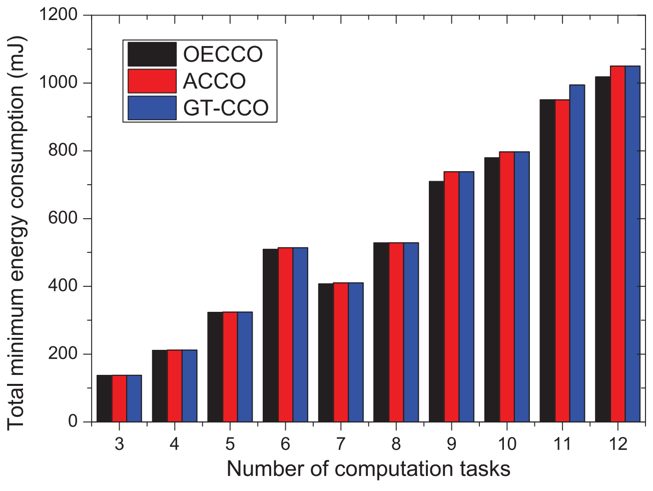
TABLE II
PARAMETER SETTINGS IN THE SIMULATION

notation	Description	Value and unit
N	Number of MDs	100
K	Number of wireless channels	50
ω	Wireless channel bandwidth	40 MHz
c	Uplink data rate of the optical network	1 Gbps
p_i	Transmit power of a MD	100 mW
σ	Background noise	-100 dBm
m_i	Input computation data size of T_i	300–800 KB
c_i	Number of CPU cycles required by T_i	100–1000 Megacycles
γ_i	Consumed energy per CPU cycle of MD i	500 mJ/Gigacycle
$t_i^{\max}$	Maximum permissible latency for accomplishing T_i	0.1–2 seconds

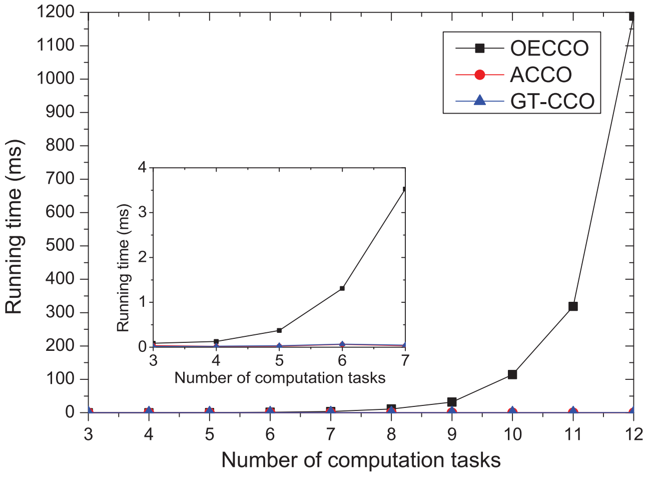
B. Numerical Results

In order to evaluate the effectiveness and the efficiency of our proposed ACCO and GT-CCO, we compare their obtained total minimum energy consumption and the running time with those of our benchmark, i.e., OECCO. Fig. 2 illustrates the comparison results with different number of computation tasks among OECCO, ACCO, and GT-CCO. Note that the number of computation tasks only ranges from 3 to 12 in consideration of OECCO's exponential computational complexity. From Fig. 2(a), we can see that, compared to our benchmark (OECCO), both ACCO and GT-CCO can find near-optimal solutions to the problem of CMOCO, i.e., near-optimal collaborative computation offloading decision profiles for all the computation tasks. Moreover, we also find that GT-CCO and ACCO nearly lead to the same results after many times of simulations. Furthermore, as depicted in Fig. 2(b), both ACCO and GT-CCO are confirmed to have very higher computational efficiency than that of OECCO. According to this group of simulations, our previously propositions (2)–(6) are validated.

Moreover, to further verify the necessity and the performance of our collaborative computation offloading strategy, we compare the obtained total minimum energy consumption by adopting different computation offloading strategies, i.e., local computing by all MDs, the distributed computation offloading algorithm (DCOA) in [11], centralized ACCO, and distributed GT-CCO. Fig. 3 shows the comparison results among them, where the X-axis denotes the number of computation tasks varying from 10 to 100, and the Y-axis is the obtained total



(a)



(b)

Fig. 2. Illustration of the numerical results of total minimum energy consumption and running time with different number of computation tasks: (a) comparisons of the total minimum energy consumption among OECCO, ACCO, and GT-CCO; (b) comparisons of the running time among OECCO, ACCO, and GT-CCO.

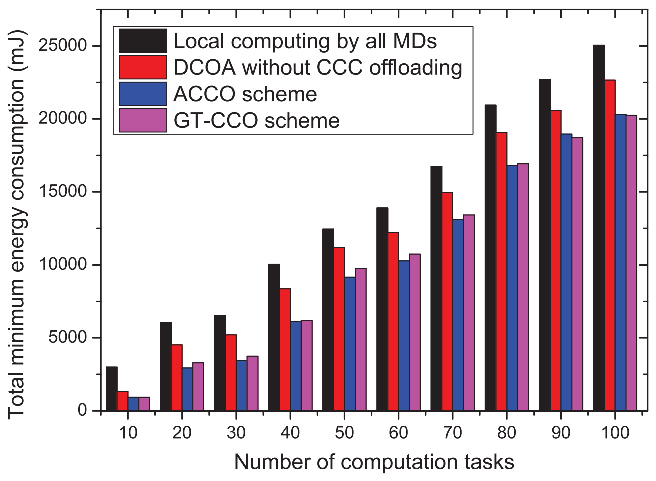


Fig. 3. Comparisons of the total minimum energy consumption by adopting different computation offloading schemes, i.e., local computing by all MDs, DCOA without CCC offloading, centralized ACCO scheme, and distributed GT-CCO scheme.

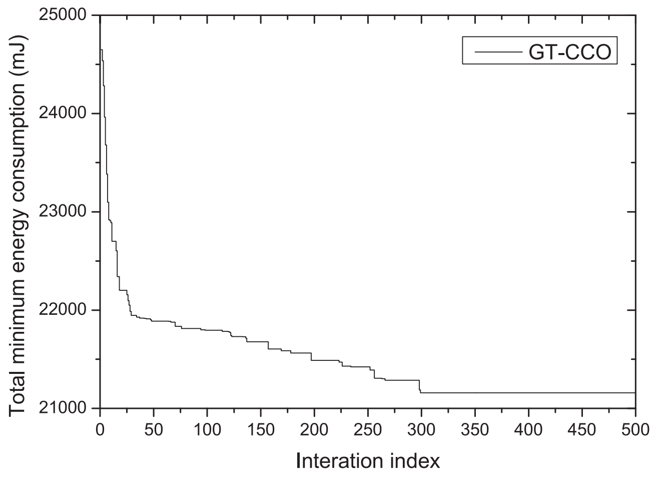


Fig. 4. Convergence behavior of GT-CCO in terms of the total minimum energy consumption.

minimum energy consumption, i.e., $E_{\min}$. From this figure, one can easily observe that, all of DCOA, ACCO, and GT-CCO can achieve lower total energy consumption than the scheme via computing locally by all the MDs, and these results show the necessity of computation offloading. Furthermore, after many times of simulations, we find that both of our proposed ACCO and GT-CCO schemes can achieve an average 20% performance improvement over the DCOA scheme without CCC offloading, where the necessity and the performance of introducing the collaborative computation offloading among the centralized cloud, distributed MEC servers, and the MDs are corroborated. This trend scales well as the number of computation tasks increases.

As stated above, compared to the centralized ACCO scheme, the distributed GT-CCO scheme is more practical and can be easily extended to the multi-MEC multi-user scenarios, since it does not need to collect the private information of the MDs and centralized control and management is not required in it. Nevertheless, as a game-theoretic method, the property of NE and convergence of GT-CCO should be ensured. Apart from the above theoretical proofs, we next evaluate the NE and convergence of GT-CCO from the view of simulation.

Fig. 4 shows the numerical results of the total minimum energy consumption by adopting GT-CCO with the increasing of iterations. We see that, our proposed GT-CCO can keep the decreasing of the total energy consumption and converge to an equilibrium. Besides, in order to test the convergence stability of GT-CCO, we run repeated experiments on the average iterations of GT-CCO with different number of computation tasks varying from 10 to 100. As depicted in Fig. 5, the average iterations for the convergence of GT-CCO almost show a linear growth trend as the number of computation tasks increases. This result demonstrates that the convergence of GT-CCO is fast and stable, as it scales well with the increasing of the number of computation tasks.

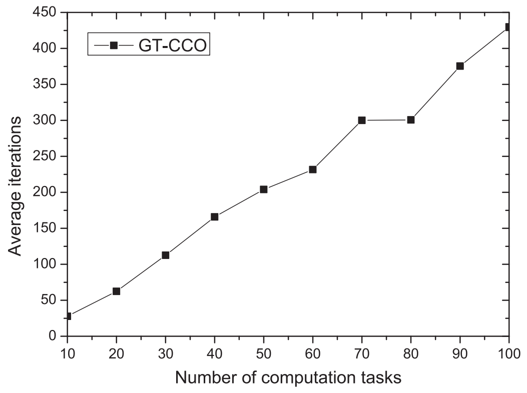


Fig. 5. Average iterations for the convergence of GT-CCO with different number of computation tasks.

VII. Conclusion

In this paper, we studied the problem of collaborative computation offloading with centralized cloud and multi-access edge computing. We first presented an architecture for collaborative computation offloading over FiWi networks, and defined the problem of cloud-MEC collaborative computation offloading. After that, the problem was formulated as a constrained optimization problem, with the objective of minimizing the MDs' energy consumption while satisfying MDs' computation execution time constraint and the total number of wireless channels. ACCO and GT-CCO were proposed step by step as our solutions. Numerical results corroborated the superior computation offloading performance of our collaborative schemes over those solutions without centralized cloud, and this advantage scales well as the number of computation tasks increases. For future work, it should be meaningful to study the collaborative computation offloading problem in heterogeneous networks with multiple MEC servers.

Appendix

Proof of Theorem 2

Proof: First, we construct a potential function as [11], [26]

$$\Phi(\lambda) = \frac{1}{2}\sum_{i\in\mathcal{N}}\sum_{j\in\mathcal{N}\setminus\{i\}} p_i g_i^s p_j g_j^s I_{\{\lambda_i=\lambda_j\}} I_{\{\lambda_i\neq 0\}} + \sum_{i\in\mathcal{N}} p_i g_i^s Q_i I_{\{\lambda_i=0\}}, \tag{13}$$

Here, $I_{\{\#\}}$ is an indicator function where $I_{\{\#\}}=1$ if the condition # is true, and otherwise. According to the condition $\min\{e_i^{mec}, e_i^{ccc}\} \leq e_i^{loc}$, the wireless interference threshold Q_i for choosing offloading decisions (i.e., $\sum_{j\in\mathcal{N}\setminus\{i\}:\lambda_j=\lambda_i} p_j g_j^s \leq Q_i$) can be calculated as

$$Q_i = \begin{cases} \dfrac{p_i g_i^s}{2^{\frac{cm_i p_i f_i^{mec}}{\omega(cc_i\gamma_i^{loc} f_i^{mec} - p_i m_i f_i^{mec} - cc_i p_i)}} - 1} - \sigma, & \text{if } e_i^{mec} \leq e_i^{ccc}, \\ \dfrac{p_i g_i^s}{2^{\frac{cm_i p_i}{\omega(cc_i\gamma_i^{loc} - np_i m_i - cp_i\tau)}} - 1} - \sigma, & \text{if } e_i^{ccc} < e_i^{mec}, \end{cases} \tag{14}$$

Then, referring to the similar proof shown in [11] and [26] and the equations in (13) and (14), we can prove that the collaborative computation offloading game $\Gamma = (\mathcal{N}, \{\Lambda_i\}_{i\in\mathcal{N}}, \{L_i\}_{i\in\mathcal{N}})$ is a potential game with the potential function in (13). That is, when a MD i updates its offloading decision λ_i to λ_i^* by adopting its current optimal offloading decision, i.e., $L_i(\lambda_i^*, \lambda_{-i}) \leq L_i(\lambda_i, \lambda_{-i})$, its potential function also decreases, i.e., $\Phi(\lambda_i^*, \lambda_{-i}) \leq \Phi(\lambda_i, \lambda_{-i})$. To prove this, we can simply partition the update process into three cases:

Case 1 ($\lambda_i = 1, \lambda_i^* = -1$ or $\lambda_i = -1, \lambda_i^* = 1$): In this case, according to the condition $L_i(\lambda_i^*, \lambda_{-i}) \leq L_i(\lambda_i, \lambda_{-i})$ and (4) and (11), we can obtain the result that

$$\sum_{j\in\mathcal{N}\setminus\{i\}:\lambda_j=\lambda_i^*} p_j g_j^s \leq \sum_{j\in\mathcal{N}\setminus\{i\}:\lambda_j=\lambda_i} p_j g_j^s. \tag{15}$$

Then, we have

$$\begin{aligned}
&\Phi(\lambda_i^*, \lambda_{-i}) - \Phi(\lambda_i, \lambda_{-i}) \\
&= \frac{1}{2}\sum_{i\in\mathcal{N}}\sum_{j\in\mathcal{N}\setminus\{i\}} p_i g_i^s p_j g_j^s I_{\{\lambda_i^*=\lambda_j\}} I_{\{\lambda_i^*\neq 0\}} \\
&\quad + \sum_{i\in\mathcal{N}} p_i g_i^s Q_i I_{\{\lambda_i^*=0\}} \\
&\quad - \frac{1}{2}\sum_{i\in\mathcal{N}}\sum_{j\in\mathcal{N}\setminus\{i\}} p_i g_i^s p_j g_j^s I_{\{\lambda_i=\lambda_j\}} I_{\{\lambda_i\neq 0\}} \\
&\quad - \sum_{i\in\mathcal{N}} p_i g_i^s Q_i I_{\{\lambda_i=0\}} \\
&= \frac{1}{2}\sum_{i\in\mathcal{N}}\sum_{j\in\mathcal{N}\setminus\{i\}} p_i g_i^s p_j g_j^s I_{\{\lambda_i^*=\lambda_j\}} \\
&\quad - \frac{1}{2}\sum_{i\in\mathcal{N}}\sum_{j\in\mathcal{N}\setminus\{i\}} p_i g_i^s p_j g_j^s I_{\{\lambda_i=\lambda_j\}} \\
&= \frac{1}{2}\sum_{i\in\mathcal{N}\setminus\{k\}} p_i g_i^s p_k g_k^s I_{\{\lambda_i^*=\lambda_k\}} \\
&\quad + \frac{1}{2}\sum_{j\in\mathcal{N}\setminus\{k\}} p_k g_k^s p_j g_j^s I_{\{\lambda_k=\lambda_j\}} \\
&\quad + \frac{1}{2}\sum_{i\in\mathcal{N}\setminus\{k\}}\sum_{j\in\mathcal{N}\setminus\{j,k\}} p_i g_i^s p_j g_j^s I_{\{\lambda_i^*=\lambda_j\}} \\
&\quad - \frac{1}{2}\sum_{i\in\mathcal{N}\setminus\{k\}} p_i g_i^s p_k g_k^s I_{\{\lambda_i=\lambda_k\}} \\
&\quad - \frac{1}{2}\sum_{j\in\mathcal{N}\setminus\{k\}} p_k g_k^s p_j g_j^s I_{\{\lambda_k=\lambda_j\}} \\
&\quad - \frac{1}{2}\sum_{i\in\mathcal{N}\setminus\{k\}}\sum_{j\in\mathcal{N}\setminus\{j,k\}} p_i g_i^s p_j g_j^s I_{\{\lambda_i=\lambda_j\}} \\
&= \sum_{i\in\mathcal{N}\setminus\{j\}} p_i g_i^s p_j g_j^s I_{\{\lambda_i^*=\lambda_j\}}
\end{aligned}$$

$$
\begin{aligned}
&- \sum_{i\in\mathcal{N}\setminus\{j\}} p_i g_i^s p_j g_j^s I_{\{\lambda_i=\lambda_j\}} \\
&= p_i g_i^s \left(\sum_{j\in\mathcal{N}\setminus\{k\}} p_j g_j^s I_{\{\lambda_i^*=\lambda_j\}} \right. \\
&\quad \left. - \sum_{j\in\mathcal{N}\setminus\{k\}} p_j g_j^s I_{\{\lambda_i=\lambda_j\}} \right) \\
&\le 0. \qquad (16)
\end{aligned}
$$

Case 2 ($\lambda_i = 0$, $\lambda_i^* = -1/1$)*:* In such case, we know that $\sum_{j\in\mathcal{N}\setminus\{i\}:\lambda_j=\lambda_i^*} p_j g_j^s \le Q_i$, and then we have

$$
\begin{aligned}
&\Phi(\lambda_i^*, \lambda_{-i}) - \Phi(\lambda_i, \lambda_{-i}) \\
&\quad = \frac{1}{2}\sum_{i\in\mathcal{N}}\sum_{j\in\mathcal{N}\setminus\{i\}} p_i g_i^s p_j g_j^s I_{\{\lambda_i^*=\lambda_j\}} I_{\{\lambda_i^*\ne 0\}} \\
&\qquad + \sum_{i\in\mathcal{N}} p_i g_i^s Q_i I_{\{\lambda_i^*=0\}} \\
&\qquad - \frac{1}{2}\sum_{i\in\mathcal{N}}\sum_{j\in\mathcal{N}\setminus\{i\}} p_i g_i^s p_j g_j^s I_{\{\lambda_i=\lambda_j\}} I_{\{\lambda_i\ne 0\}} \\
&\qquad - \sum_{i\in\mathcal{N}} p_i g_i^s Q_i I_{\{\lambda_i=0\}} \\
&\quad = \frac{1}{2}\sum_{i\in\mathcal{N}}\sum_{j\in\mathcal{N}\setminus\{i\}} p_i g_i^s p_j g_j^s I_{\{\lambda_i^*=\lambda_j\}} \\
&\qquad - \sum_{i\in\mathcal{N}} p_i g_i^s Q_i I_{\{\lambda_i=0\}} \\
&\quad = p_i g_i^s \sum_{j\in\mathcal{N}\setminus\{i\}} p_j g_j^s I_{\{\lambda_i^*=\lambda_j\}} \\
&\qquad - p_i g_i^s Q_i I_{\{\lambda_i=0\}} \\
&\qquad - \sum_{j\in\mathcal{N}\setminus\{i\}} p_j g_j^s Q_j I_{\{\lambda_j=0\}} \\
&\quad = p_i g_i^s \left(\sum_{j\in\mathcal{N}\setminus\{i\}} p_j g_j^s I_{\{\lambda_i^*=\lambda_j\}} - Q_i \right) \\
&\quad \le 0. \qquad (17)
\end{aligned}
$$

Case 3 ($\lambda_i = -1/1, \lambda_i^* = 0$)*:* For this case, we have $\sum_{j\in\mathcal{N}\setminus\{i\}:\lambda_j=\lambda_i^*} p_j g_j^s \ge Q_i$. Then, according to the condition $L_i(\lambda_i^*, \lambda_{-i}) \le L_i(\lambda_i, \lambda_{-i})$ and (15), it implies that

$$
\begin{aligned}
&\Phi(\lambda_i^*, \lambda_{-i}) - \Phi(\lambda_i, \lambda_{-i}) \\
&\quad = \frac{1}{2}\sum_{i\in\mathcal{N}}\sum_{j\in\mathcal{N}\setminus\{i\}} p_i g_i^s p_j g_j^s I_{\{\lambda_i^*=\lambda_j\}} I_{\{\lambda_i^*\ne 0\}} \\
&\qquad + \sum_{i\in\mathcal{N}} p_i g_i^s Q_i I_{\{\lambda_i^*=0\}} \\
&\qquad - \frac{1}{2}\sum_{i\in\mathcal{N}}\sum_{j\in\mathcal{N}\setminus\{i\}} p_i g_i^s p_j g_j^s I_{\{\lambda_i=\lambda_j\}} I_{\{\lambda_i\ne 0\}} \\
&\qquad - \sum_{i\in\mathcal{N}} p_i g_i^s Q_i I_{\{\lambda_i=0\}} \\
&\quad = \sum_{i\in\mathcal{N}} p_i g_i^s Q_i I_{\{\lambda_i^*=0\}} \\
&\qquad - \frac{1}{2}\sum_{i\in\mathcal{N}}\sum_{j\in\mathcal{N}\setminus\{i\}} p_i g_i^s p_j g_j^s I_{\{\lambda_i=\lambda_j\}} \\
&\quad = p_i g_i^s \left(Q_i - \sum_{j\in\mathcal{N}\setminus\{i\}} p_j g_j^s I_{\{\lambda_i=\lambda_j\}} \right) \\
&\quad \le p_i g_i^s \left(Q_i - \sum_{j\in\mathcal{N}\setminus\{i\}} p_j g_j^s I_{\{\lambda_i^*=\lambda_j\}} \right) \\
&\quad \le 0. \qquad (18)
\end{aligned}
$$

Therefore, the collaborative computation offloading game is a potential game. Finally, according to the potential game theory in [42], our proposed collaborative computation offloading game $\Gamma = (\mathcal{N}, \{\Lambda_i\}_{i\in\mathcal{N}}, \{L_i\}_{i\in\mathcal{N}})$ has a NE and possesses the finite improvement property. ■

REFERENCES

[1] B. P. Rimal, D. P. Van, and M. Maier, "Mobile edge computing empowered fiber-wireless access networks in the 5G era," *IEEE Commun. Mag.*, vol. 55, no. 2, pp. 192–200, Feb. 2017.

[2] R. Yu, Y. Zhang, S. Gjessing, W. Xia, and K. Yang, "Toward cloud-based vehicular networks with efficient resource management," *IEEE Netw.*, vol. 27, no. 5, pp. 48–55, Sep. 2013.

[3] T. Soyata, R. Muraleedharan, C. Funai, M. Kwon, and W. Heinzelman, "Cloud-vision: Real-time face recognition using a mobile-cloudlet-cloud acceleration architecture," in *Proc. IEEE Symp. Comput. Commun.*, 2012, pp. 000059–000066.

[4] C. C. Coskun, K. Davaslioglu, and E. Ayanoglu, "Three-stage resource allocation algorithm for energy-efficient heterogeneous networks," *IEEE Trans. Veh. Technol.*, vol. 66, no. 8, pp. 6942–6957, Aug. 2017.

[5] Y. Mao, J. Zhang, and K. B. Letaief, "Joint task offloading scheduling and transmit power allocation for mobile-edge computing systems," in *Proc. 2017 IEEE Wireless Commun. Netw. Conf.*, 2017, pp. 1–6.

[6] O. Munoz, A. Pascual-Iserte, and J. Vidal, "Optimization of radio and computational resources for energy efficiency in latency-constrained application offloading," *IEEE Trans. Veh. Technol.*, vol. 64, no. 10, pp. 4738–4755, Oct. 2015.

[7] Z. Su, Q. Xu, F. Hou, Q. Yang, and Q. Qi, "Edge caching for layered video contents in mobile social networks," *IEEE Trans. Multimedia*, vol. 19, no. 10, pp. 2210–2221, Oct. 2017.

[8] Z. Su, Q. Qi, Q. Xu, S. Guo, and X. Wang, "Incentive scheme for cyber physical social systems based on user behaviors," *IEEE Trans. Emerg. Topics Comput.*, to be published, doi: 10.1109/TETC.2017.2671843.

[9] ETSI, "Mobile edge computing—A key technology towards 5G," Eur. Telecommun. Standards Inst., Sophia Antipolis Cedex, France, White Paper 11, Sept. 2015.

[10] S. Wang, X. Zhang, Y. Zhang, L. Wang, J. Yang, and W. Wang, "A survey on mobile edge networks: Convergence of computing, caching and communications," *IEEE Access*, vol. 5, pp. 6757–6779, 2017.

[11] 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.

[12] J. Liu, Y. Mao, J. Zhang, and K. B. Letaief, "Delay-optimal computation task scheduling for mobile-edge computing systems," in *Proc. 2016 IEEE Int. Symp. Inf. Theory*, 2016, pp. 1451–1455.

[13] T. Q. Dinh, J. Tang, Q. D. La, and T. Q. S. Quek, "Offloading in mobile edge computing: Task allocation and computational frequency scaling," *IEEE Trans. Commun.*, vol. 65, no. 8, pp. 3571–3584, Aug. 2017.

[14] S. Jeong, O. Simeone, and J. Kang, "Mobile edge computing via a UAV-mounted cloudlet: Optimization of bit allocation and path planning," *IEEE Trans. Veh. Technol.*, vol. 67, no. 3, pp. 2049–2063, Mar. 2018.

[15] S. Parsaeefard, R. Dawadi, M. Derakhshani, T. Le-Ngoc, and M. Baghani, "Dynamic resource allocation for virtualized wireless networks in massive-MIMO-aided and front-haul-limited C-RAN," *IEEE Trans. Veh. Technol.*, vol. 66, no. 10, pp. 9512–9520, Oct. 2017, doi: 10.1109/TVT.2017.2712669.

[16] K. Zhang *et al.*, "Energy-efficient offloading for mobile edge computing in 5g heterogeneous networks," *IEEE Access*, vol. 4, pp. 5896–5907, 2016.

[17] T. G. Rodrigues, K. Suto, H. Nishiyama, and N. Kato, "Hybrid method for minimizing service delay in edge cloud computing through VM migration and transmission power control," *IEEE Trans. Comput.*, vol. 66, no. 5, pp. 810–819, May 2017.

[18] Y. Luo, F. Effenberger, and M. Sui, "Cloud computing provisioning over passive optical networks," in *Proc. 1st IEEE Int. Conf. Commun. China*, 2012, pp. 255–259.

[19] A. S. Reaz, V. Ramamurthi, M. Tornatore, and B. Mukherjee, "Cloud-integrated WOBAN: An offloading-enabled architecture for service-oriented access networks," *Comput. Netw.*, vol. 68, pp. 5–19, Aug. 2014.

[20] C. Wehner, "Multi-access edge computing," Artesyn Embedded Technol., White Paper, Feb. 2017.

[21] J. Liu, H. Guo, H. Nishiyama, H. Ujikawa, K. Suzuki, and N. Kato, "New Perspectives on future smart FiWi networks: scalability, reliability, and energy efficiency," *IEEE Commun. Surveys Tuts.*, vol. 18, no. 2, pp. 1045–1072, Apr.–Jun. 2016.

[22] J. Liu, H. Guo, Z. M. Fadlullah, and N. Kato, "Energy consumption minimization for FiWi enhanced LTE-A HetNets with UE connection constraint," *IEEE Commun. Mag.*, vol. 54, no. 11, pp. 56–62, Nov. 2016.

[23] Z. M. Fadlullah, H. Nishiyama, Y. Kawamoto, H. Ujikawa, K. I. Suzuki, and N. Yoshimoto, "Cooperative QoS control scheme based on scheduling information in FiWi access network," *IEEE Trans. Emerg. Topics Comput.*, vol. 1, no. 2, pp. 375–383, Dec. 2013.

[24] K. Saito, H. Nishiyama, N. Kato, H. Ujikawa, and K. I. Suzuki, "A MPCP-based centralized rate control method for mobile stations in FiWi access networks," *IEEE Wireless Commun. Lett.*, vol. 4, no. 2, pp. 205–208, Apr. 2015.

[25] B. P. Rimal, D. P. Van, and M. Maier, "Cloudlet enhanced fiber-wireless access networks for mobile-edge computing," *IEEE Trans. Wireless Commun.*, vol. 16, no. 6, pp. 3601–3618, Jun. 2017.

[26] J. Zheng, Y. Cai, Y. Wu, and X. S. Shen, "Stochastic computation offloading game for mobile cloud computing," in *Proc. 2016 IEEE/CIC Int. Conf. Commun. China*, 2016, pp. 1–6.

[27] Y. Mao, J. Zhang, and K. B. Letaief, "Dynamic computation offloading for mobile-edge computing with energy harvesting devices," *IEEE J. Sel. Areas Commun.*, vol. 34, no. 12, pp. 3590–3605, Dec. 2016.

[28] K. Sato and T. Fujii, "Radio environment aware computation offloading with multiple mobile edge computing servers," in *Proc. IEEE Wireless Commun. Netw. Conf. Workshops*, 2017.

[29] B. P. Rimal, D. P. Van, and M. Maier, "Mobile-Edge Computing vs. Centralized Cloud Computing in Fiber-Wireless Access Networks," in *Proc. INFOCOM WKSHPS*, 2016, pp. 991–996.

[30] S. Ten, "Ultra low-loss optical fiber technology," in *Proc. Opt. Fiber Commun. Conf. Exhib.*, 2016, pp. 1–3.

[31] M. Guinan, V. Diaz, and M. Edwards, "The super connected world: Optical fiber advances and next gen backbone, mobile backhaul, and access networks," Corning Inc., New York, NY, USA, White Paper WP6024, 2012.

[32] H. Guo, J. Liu, Z. Fadlullah, and N. Kato, "On minimizing energy consumption in FiWi enhanced LTE-A HetNets," *IEEE Trans. Emerg. Topics Comput.*, to be published, doi: 10.1109/TETC.2016.2598478.

[33] H. Guo, J. Liu, and L. Zhao, "Big data acquisition under failures in FiWi enhanced smart grid," *IEEE Trans. Emerg. Topics Comput.*, 2017, to be published, doi: 10.1109/TETC.2017.2675911.

[34] E. Cuervo *et al.*, "MAUI: Making smartphones last longer with code offload," in *Proc. 8th Int. Conf. Mobile Syst. Appl. Serv.*, 2010, pp. 49–62.

[35] B.-G. Chun, S. Ihm, P. Maniatis, M. Naik, and A. Patti, "CloneCloud: Elastic execution between mobile device and cloud," in *Proc. 6th Conf. Comput. Syst.*, 2011, pp. 301–304.

[36] A. Kansal and F. Zhao, "Fine-grained energy profiling for power-aware application design," *ACM SIGMETRICS Perform. Eval. Rev.*, vol. 36, no. 2, pp. 26–31, Aug. 2008.

[37] M. S. Kiaei, K. Fouli, M. Scheutzow, M. Maier, M. Reisslein, and C. Assi, "Delay analysis for ethernet long-reach passive optical networks," in *Proc. 2012 IEEE Int. Conf. Commun.*, 2012, pp. 3099–3104.

[38] K.-H. Loh, B. Golden, and E. Wasil, "Solving the maximum cardinality bin packing problem with a weight annealing-based algorithm," *Oper. Res. Cyber-Infrastruct.*, vol. 47, pp. 147–164, 2009.

[39] H. Yaiche, R. R. Mazumdar, and C. Rosenberg, "A game theoretic framework for bandwidth allocation and pricing in broadband networks," *IEEE/ACM Trans. Netw.*, vol. 8, no. 5, pp. 667–678, Oct. 2000.

[40] J. Zheng, Y. Cai, N. Lu, Y. Xu, and X. Shen, "Stochastic game-theoretic spectrum access in distributed and dynamic environment," *IEEE Trans. Veh. Technol.*, vol. 64, no. 10, pp. 4807–4820, Oct. 2015.

[41] Z. M. Fadlullah, C. Wei, Z. Shi, and N. Kato, "GT-QoSec: A game-theoretic joint optimization of QoS and security for differentiated services in next generation heterogeneous networks," *IEEE Trans. Wireless Commun.*, vol. 16, no. 2, pp. 1037–1050, Feb. 2017.

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

Hongzhi Guo (S'08–M'16) received the B.S. degree in computer science and technology from Harbin Institute of Technology, Harbin, China, in 2004, and the M.S. and Ph.D. degrees in computer application technology from Harbin Institute of Technology Shenzhen Graduate School, Shenzhen, China, in 2006 and 2011, respectively. He is currently a Lecturer with the School of Cyber Engineering, Xidian University, Xi'an, China. His research interests include MEC, UDN, 5G communications, and IoT.

Jiajia Liu (S'11–M'12–SM'15) is currently a Full Professor with the School of Cyber Engineering, Xidian University, Xi'an, China. His research interests include wireless mobile communications, FiWi, IoT, etc. He has authored or coauthored more than 50 peer-reviewed papers in many prestigious IEEE journals and conferences. He is currently an Associate Editor for the IEEE TRANSACTIONS ON COMPUTERS and the IEEE TRANSACTIONS ON VEHICULAR TECHNOLOGY, an Editor for the IEEE NETWORK, a Guest Editor for the IEEE TRANSACTIONS ON EMERGING TOPICS IN COMPUTING and the IEEE INTERNET OF THINGS JOURNAL. He is a Distinguished Lecturer with IEEE Communications Society.