COLLABORATIVE COMPUTATION OFFLOADING FOR MULTI-ACCESS EDGE COMPUTING OVER FIBER-WIRELESS NETWORKS

RESEARCH PAPER BY –
HONGZHI GUO, Member, IEEE
JIAJIA LIU, Senior Member, IEEE

INTRODUCTION AND MOTIVATION

Before introducing the topic as a whole it is very important to understand the keywords associated with the topic :

WHAT IS COMPUTATION OFFLOADING ?

Computation offloading is the transfer of resource intensive computational tasks to an external platform, such as a cluster, grid, or a cloud. Offloading may be necessary due to hardware limitations of a devices, such as limited computational power, storage, and energy. Offloading is mainly used in mobile cloud computing, edge computing, and fog computing.

WHAT IS MEC ?

Multi-access edge computing, formerly mobile edge computing, is a network architecture concept that enables cloud computing capabilities and an IT service environment at the edge of the cellular network and, more in general at the edge of any network. The basic idea behind MEC is that by running applications and performing related processing tasks closer to the cellular customer, network congestion is reduced and applications perform better. MEC technology is designed to be implemented at the cellular base stations or other edge nodes, and enables flexible and rapid deployment of new applications and services for customers.

WHAT ARE FIBER-WIRELESS NETWORKS ?

Fiber-wireless technology, known in short as “Fi-Wi,” is a combination of optical-fiber-based network and wireless network. Fi-Wi has recently come forward as one of the emerging future network technologies to provide telecommunication services to the clusters of end points which are geographically distant. Optical networks are designed to provide long-distance, high-bandwidth communications and the wireless networks provide ubiquitous, flexible communications in community areas. The high-capacity optical fiber is used to span the longest distances, and a lower cost wireless link carries the signal for the last mile to nearby users.

In the last decade or so the popularity of smart mobile devices (MDs) like smart phones, smart watches/bands, etc. has increased drastically. Furthermore, with the introduction of more and more new Internet-of-Things (IOT) devices like shared bike, and shared mobile power supply, lots of new applications have emerged . These applications e.g., face/fingerprint/iris recognition, natural language processing, and interactive gaming are all computation intensive and consume high energy. However, due to MDs' limited computing capacity and battery power , such applications can't be supported on MD's alone. This problem introduced the need for computation offloading.

Earlier, resource-rich cloud was used to offload MD's computation tasks . However, it was observed that this approach had the drawbacks of long latency and poor reliability caused by the transmission through the wide area network. Recently, MEC, with its ability to provide cloud computing capabilities closer to the mobile users has been proposed. Compared to earlier approach , MECO achieves lower latency and higher reliability.

However , existing multi-access edge computing offloading (MECO) research only considered the offloading of computation tasks of mobile devices to the edge servers (MEC servers) and ignored the huge computation resources in the centralized cloud computing centre(CCC). Due to the ever growing number of MD's and mobile applications , the computation load on the MEC servers has become more prominent. Moreover the stringent cloud processing capabilities provided by the centralized cloud can work great in tandem with the low latency, high reliability MECO scheme. These two technologies namely, MEC and cloud computing are complementary. Furthermore, the latency and the reliability in the WAN have been well improved with the introduction of the passive optical networks (PONs). PONs is a telecommunications technology used to provide fiber to the end consumer.

By using 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 considered to be a promising technology to support the co-existence of centralized cloud and distributed MEC services.

Toward this end, this research paper introduces FiWi networks to provide support for the coexistence of centralized cloud and multi-access edge computing, and study the problem of collaborative computation offloading.

In particular, this paper’s major contributions are summarized as follows:

• In order to provide supports for the coexistence of centralized cloud and multi-access edge computing, this paper presents a generic architecture by adopting the hybrid FiWi access networks.

• This paper studies the problem of collaborative computation offloading with centralized cloud and MEC in a multichannel interference environment, and formulates 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, this paper proposes 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, this paper further proposes a distributed collaborative computation offloading scheme by adopting the game theory, i.e., a game-theoretic collaborative computation offloading scheme.

MCO

- Stands for Mobile Computation Offloading
- Offloading of tasks directly from MD's directly to the cloud
- Poor reliability
- High latency

MECO

- Tasks offloaded from MD's to the edge
- Better reliability
- Lower latency
- Does not take into consideration offloading of centralized cloud computing (CCC)

Collaborative Computation Offloading

- Co-existence of centralized cloud and MEC servers facilitated using FiWi networks
- Both centralized cloud and MEC servers are used for offloading the computation tasks of the MD's
- Low latency, high reliability, good scalability, high mobility

RELATED WORKS

In the recent years, much research has been done on MECO. A number of MECO schemes have been proposed :

- X. Chen and others 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 in their paper "Efficient Multi-User Computation Offloading for Mobile-Edge Cloud Computing" , where better computation performance in terms of energy consumption and computation execution time was reported.
- J. Zheng and others studied the problem of multi-user computation offloading for mobile cloud computing with active/inactive users, and presented an efficient multi-agent stochastic learning algorithm consisting of dynamic offloading decision process.

In order to reduce MD's energy consumption, many researchers have focused on incorporating energy harvesting or dynamic voltage scaling technologies into MECO schemes.

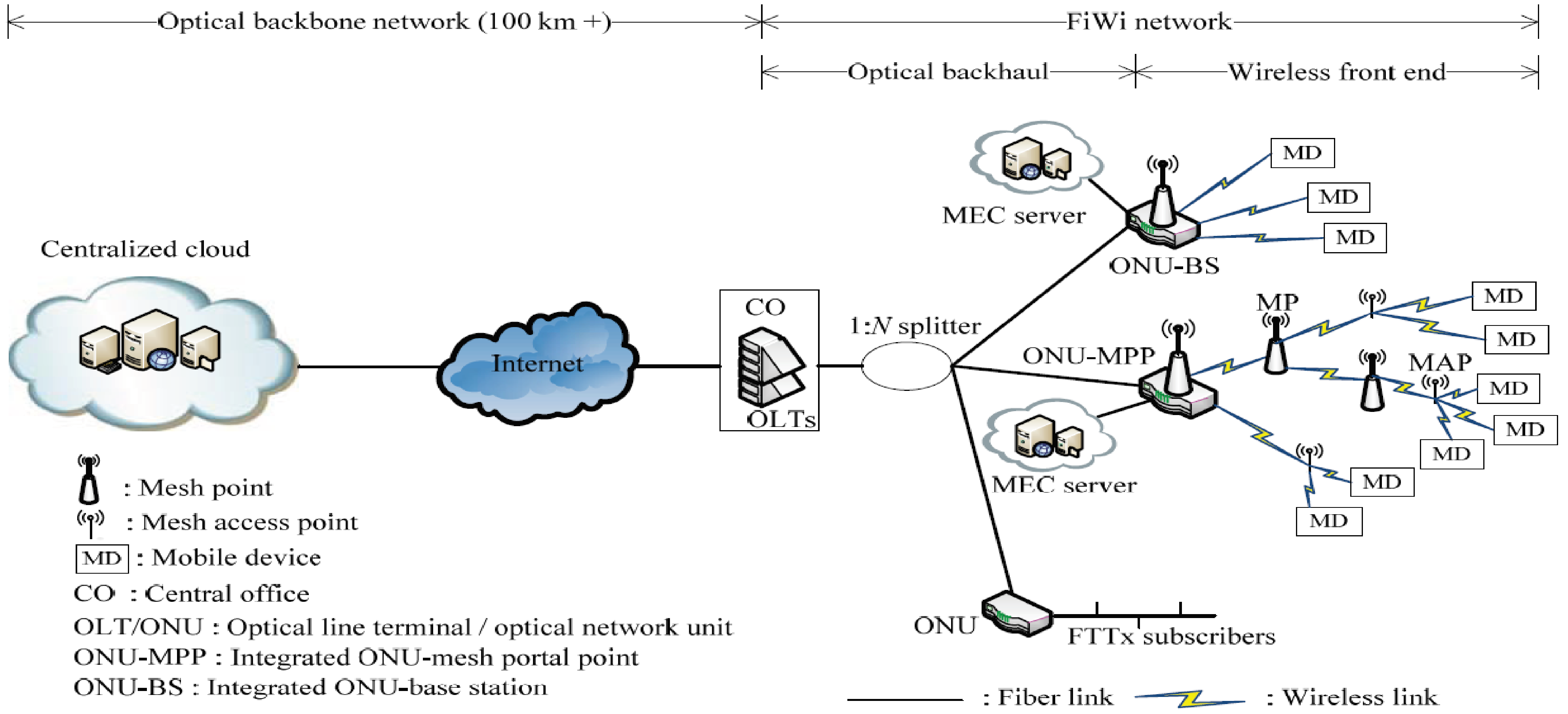
- Mao and others studied the dynamic MECO problem with energy harvesting devices, and proposed the Lyapunov optimization based dynamic computation offloading algorithm. It is a low-complexity online algorithm which jointly decides the offloading decision, the CPU-cycle frequencies for mobile execution, and the MDs' transmit power for computation offloading.
- 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, Dinh and others presented a computational framework by jointly optimizing the task allocation decision and the MD's CPU frequency.

Apart from this, there are some researchers focusing on introducing PONs to provide supports for cloud computing/edge computing services :

- A. S. Reaz and others 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.
- B. P. Rimal and others explored the possibilities of empowering integrated FiWi access networks to offer MEC capabilities in the 5G era. They also 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.

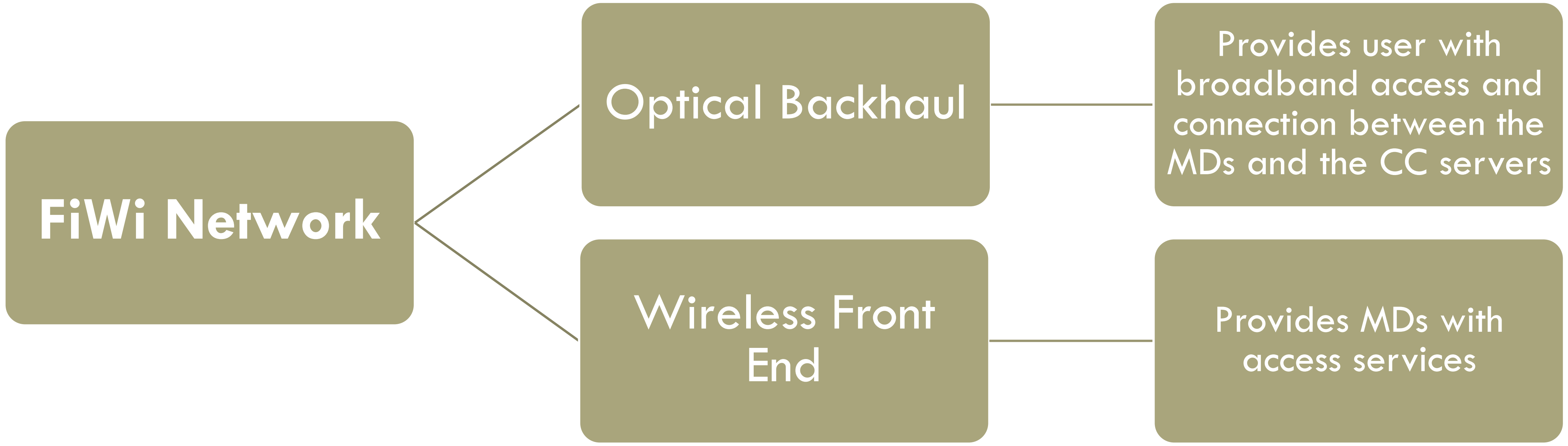
Existing research on MECO, however, only focused on the computation offloading between the MD's and the MEC servers and ignored the huge computation resources in the CCC center. This paper proposes a collaborative computation offloading scheme for centralized cloud and MEC servers by adopting hybrid FiWi access networks where the computation resources of the MD's, the MEC and CCC servers can be utilized adequately.

ARCHITECHTURE FOR COLLABORATIVE COMPUTATION OFFLOADING OVER FIWI NETWORKS



The figure that you saw in the previous slide clearly depicts a generic architecture with coexistence of centralized cloud and MEC over FiWi networks, which consists of optical backbone network connecting to CCC far away, and the FiWi Hosted Networks providing MEC services. By adopting ultra-low loss and low latency fibers we can combat the issue of long latency arising due to increasing distances.

Furthermore, the FiWi networks can be divided into two parts :


FiWi Network
Optical Backhaul
Provides user with broadband access and connection between the MDs and the CC servers
Wireless Front End
Provides MDs with access services

This paper has adopted cost-effective Ethernet Passive Optical Network (EPON)/ 10G-EPON in the optical backhaul. 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 to provide cloud computing resources.

There are three subsets of the Optical Network Units (ONUs) which connect to the OLT via a 1:N splitter. The CO is responsible for managing the information transmission between the MDs and their associated ONUs by adopting the OLTs located at it. The three subsets of ONUs :

- The first subset provides fiber-to-the-x (FTTx) services to the wired subscribers. E.g. – Fiber to the home (FTTH), Fiber to the Office (FTTO)

- The second subset of the ONUs are the collocated ONU-BS which are equipped with the cellular base station and serve as the interface between the optical backhaul and the wireless front end and provides the MDs with wireless access services.

The third subset of ONUs are equipped with mesh portal points (MPPs) – collocated ONU-MPPs. They connect wireless mesh network (WMN) in the wireless front end to the optical backhaul of the FiWi network. 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.

To access the internet a MD may use the ONU-BS or the multi-hop WMN network. 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. However, this research paper considers that only one MEC server is connected to a single ONU-BS/MPP.

By using this architecture, depending on the MDs' computation tasks, the computing ability of the MEC servers, 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.

PROPOSED MODEL

In this section, we will first introduce the system model, and then formulate the problem of collaborative computation offloading with centralized cloud and MEC. After that, we will present solutions to the problem as presented in the paper, i.e., an approximation collaborative computation offloading (ACCO) scheme, and a distributed game theoretic collaborative computation offloading (GT-CCO) scheme.

A. SYSTEM MODEL

Consider a set of collocated MDs denoted by $\mathcal{N} = \{1,2,\ldots\ldots N\}$. Let's 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.

The computation task T_i can be denoted as :

$$\mathbf{T_i = \{m_i, c_i, t_i^{max}\}}$$

where : $m_i \rightarrow$ size of computation input data like program codes, input data etc.

$c_i \rightarrow$ required CPU cycles to finish task T_i

$t_i^{max} \rightarrow$ maximum permissible latency for accomplishing task T_i

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.

Suppose that there are K orthogonal frequency channels for the wireless BS denoted as $\mathcal{K} = \{1,2,\ldots..,K\}$ and denote

$\Lambda_i \in \{0, 1, -1\}$ as the offloading decision of MD I, where :

1) Λ_i = 0 (Local Execution Model) : MD i decides to execute its computation task T_i locally.

Required CPU cycles to finish task Ti

$$t_i^{loc} = \frac{c_i}{f_i^{loc}}, \qquad (1)$$

Processing time of task T_i

Computing ability per CPU cycle of MD i

$$e_i^{loc} = c_i \cdot \gamma_i^{loc} \qquad (2)$$

Energy consumption of task T_i

Consumed energy per CPU cycle of MD i

2) Λ_i = 1 (Mobile-Edge Execution Model) : MD i chooses to offload its computation task T_i to the MEC server. In this case, the processing execution time not only includes the computation execution time but also the time cost for transmitting T_i via wireless access and dedicated fiber link. We can ignore the time cost for sending computation outcome since for most mobile applications computation output data is much smaller than computation input data.

This term represents processing time at MEC server

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

This term represents transmission time from MD to BS

This term represents transmission time from BS to MEC

Here, c : uplink data rate of the optical fiber network

f_i^{mec} : computing ability of the MEC server

r_i : uplink data rate of transmitting from MD i to the wireless BS

The corresponding energy consumption of MD i by offloading to the MEC server can be given by :

$$e_i^{mec} = p_i \cdot t_i^{mec} \tag{4}$$

Transmit power of MD i

2) Λ_i = -1 (Centralized Cloud Execution Model) : MD i will offload its computation task T_i to the CCC center thousands of miles away. Since centralized cloud such as Amazon AWS have sufficient cloud computation resources and very strong computing ability, we can ignore the execution time taken by the CCC server.

Uplink propagation delay of the optical backbone network

$$t_i^{ccc} = \frac{m_i}{r_i} + n\frac{m_i}{c} + \tau \qquad (5)$$

This term represents transmission time from MD to BS

This term represents transmission delay from the wireless BS to the CCC server

Here, n : number of optical amplifying from the ONU-BS to the CCC center thousands of miles away

$$e_i^{ccc} = p_i \cdot t_i^{ccc} \qquad (6)$$

B. PROBLEM FOUNDATION

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.

The problem of CMOCO can be formulated as :

$$\begin{aligned} &\min_{\{\lambda_i\}} \sum_{i=1}^{N} \lambda_i \cdot e_i \\ &s.t.\ C1: \lambda_i \cdot t_i \leq t_i^{max},\ \forall\, i \in \mathcal{N}, \\ &\quad C2: \sum_{i=1}^{N} |\lambda_i| \leq K, \quad \forall\, i \in \mathcal{N}, \\ &\quad C3: \lambda_i \in \{0, 1, -1\}, \quad \forall\, i \in \mathcal{N} \end{aligned} \tag{7}$$

According to equations (3), (4), (5), (6), e_i (consumed energy of task T_i) can be given as

$$e_i = \begin{cases} e_i^{loc}, & if\ \lambda_i = 0, \\ e_i^{mec}, & if\ \lambda_i = 1, \\ e_i^{ccc}, & if\ \lambda_i = -1. \end{cases} \tag{8}$$

Here, t_i : latency of task T_i

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

To solve the problem of CMOCO, we need to find an optimal computation offloading decision profile of all MDs, which has the minimum total energy consumption while satisfying all the aforementioned constraints.

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

Proof : An optimal solution to the CMOCO problem is a computation decision profile 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. 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.

WHAT IS THE MAXIMUM CARDINALITY BIN PROBLEM ?
N → number of items of varying sizes
M → number of bins of same capacity
Aim : To assign maximum number of items to the limited number of bins without violating the capacity constraint of each bin

C. SOLUTIONS

1) An optimal enumeration collaborative computation offloading solution (OECCO):

In order to solve the problem of CMOCO, 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. The details of OECCO are briefly described in Algorithm 1.

Algorithm 1 : OECCO algorithm

Input: $\mathcal{N}, \mathcal{K}, T_i; f_i^{loc}, f_i^{mec}; \mathcal{Y}_i^{loc}, p_i;$ c, $\mathcal{T}, \forall i \in N.$

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 Λi ;
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.
```

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

Proof. OECCO traverses the whole solution space and goes through all optional offloading decision combinations, so 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.

2) An approximation collaborative computation offloading (ACCO):

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

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 as they do not satisfy the latency constraint, 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.

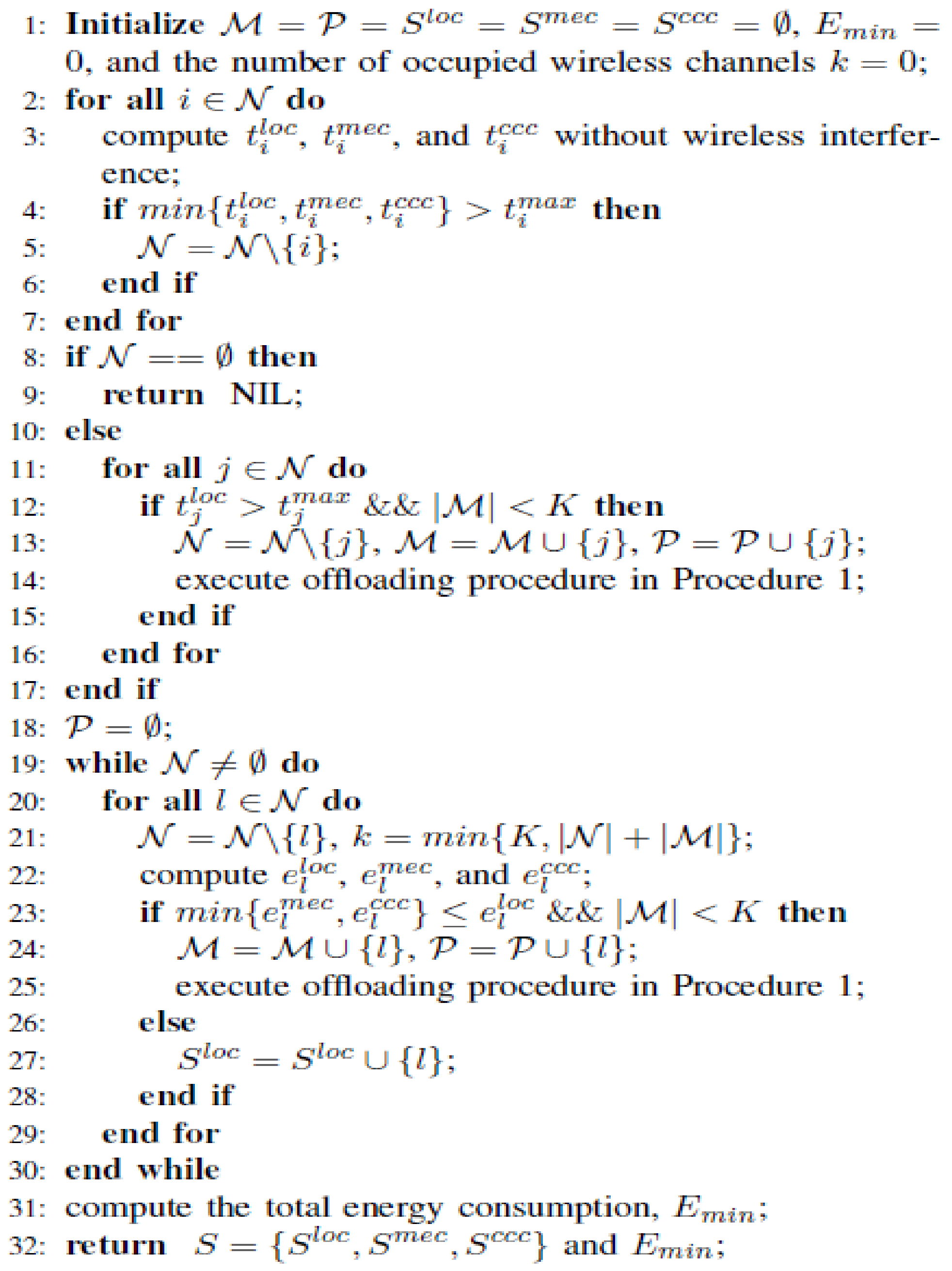
Finally, for the remaining computation tasks, which can be executed locally on the MDs, or be offloaded to the MEC server via wireless access, or be offloaded to the CCC server via the FiWi and optical backbone networks, we iteratively update the number of occupied wireless networks, and compute the consumed energy separately by adopting the offloading decisions *loc*, *mec*, and ccc. After that we choose the offloading decision with the lowest energy consumption.

Algorithm 2 : ACCO algorithm

Input: $\mathcal{N}$, $\mathcal{K}$, T_i; f_i^{loc}, f_i^{mec}; $\mathcal{Y}_i^{loc}$, p_i; c, $\mathcal{T}$, $\forall i \in N$.

Output: an optimal computation offloading decision set *S*,

and the total minimum energy consumption E_{min}; otherwise, NIL.



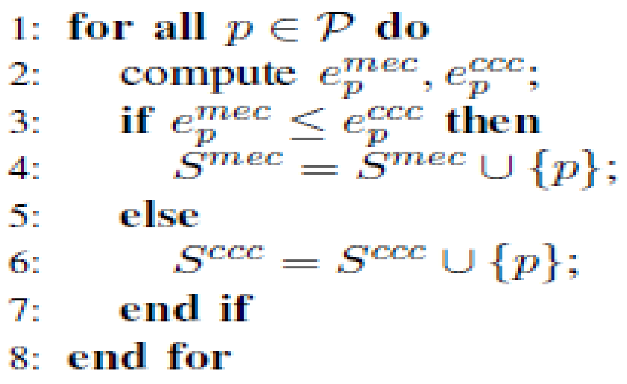
```
1: Initialize $\mathcal{M} = \mathcal{P} = S^{loc} = S^{mec} = S^{ccc} = \emptyset$, $E_{min} = 0$, and the number of occupied wireless channels $k = 0$;
2: for all $i \in \mathcal{N}$ do
3:     compute $t_i^{loc}$, $t_i^{mec}$, and $t_i^{ccc}$ without wireless interference;
4:     if $min\{t_i^{loc}, t_i^{mec}, t_i^{ccc}\} > t_i^{max}$ 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 $t_j^{loc} > t_j^{max}$ && $|\mathcal{M}| < K$ 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 $E_{min}$;
```

Here, $\mathcal{M}$: set of all tasks completed till now

$\mathcal{P}$: set of all tasks to be offloaded (processed in procedure 1)

Procedure 1 : Offloading decision procedure

In this procedure we are checking whether offloading to CCC or MEC server results in lesser energy consumption and then we take offloading decision accordingly



```
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
```

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

Proof. According to the for-all loop of steps 20 - 29 in Algorithm 2, ACCO adopts a greedy strategy to choose MD *i*'s offloading decision, by adopting which the consumed energy of MD *i* is minimum. However, considering that the total number of wireless channels is limited and we randomly choose a MD *i* from the set of remaining MDs *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 *i*.

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. 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.

Let $\Lambda_{-i} = \{\Lambda_1, \ldots, \Lambda_{i-1}, \Lambda_{i+1}, \ldots, \Lambda_n\}$ denote 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.,

Energy consumption function of MD i

$$\begin{aligned} &\min_{\lambda_i \in \{0,1,-1\}} \quad L_i(\lambda_i, \lambda_{-i}) \\ &s.t.\ \lambda_i \cdot t_i \leq t_i^{max},\ \forall\, i \in \mathcal{N}, \\ &\quad \sum_{i=1}^{N} |\lambda_i| \leq K, \quad \forall\, i \in \mathcal{N}, \\ &\quad \lambda_i \in \{0, 1, -1\}, \quad \forall\, i \in \mathcal{N}, \end{aligned} \tag{9}$$

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., $L_{i(}\Lambda_i^* ; \Lambda_{-i}) \leq L_i(\Lambda_i^* ; \Lambda_{-i})$;$\forall i \in N$.

$$L_i(\lambda_i, \lambda_{-i}) = \begin{cases} e_i^{loc}, & if\ \lambda_i = 0, \\ e_i^{mec}, & if\ \lambda_i = 1, \\ e_i^{ccc}, & if\ \lambda_i = -1. \end{cases} \tag{10}$$

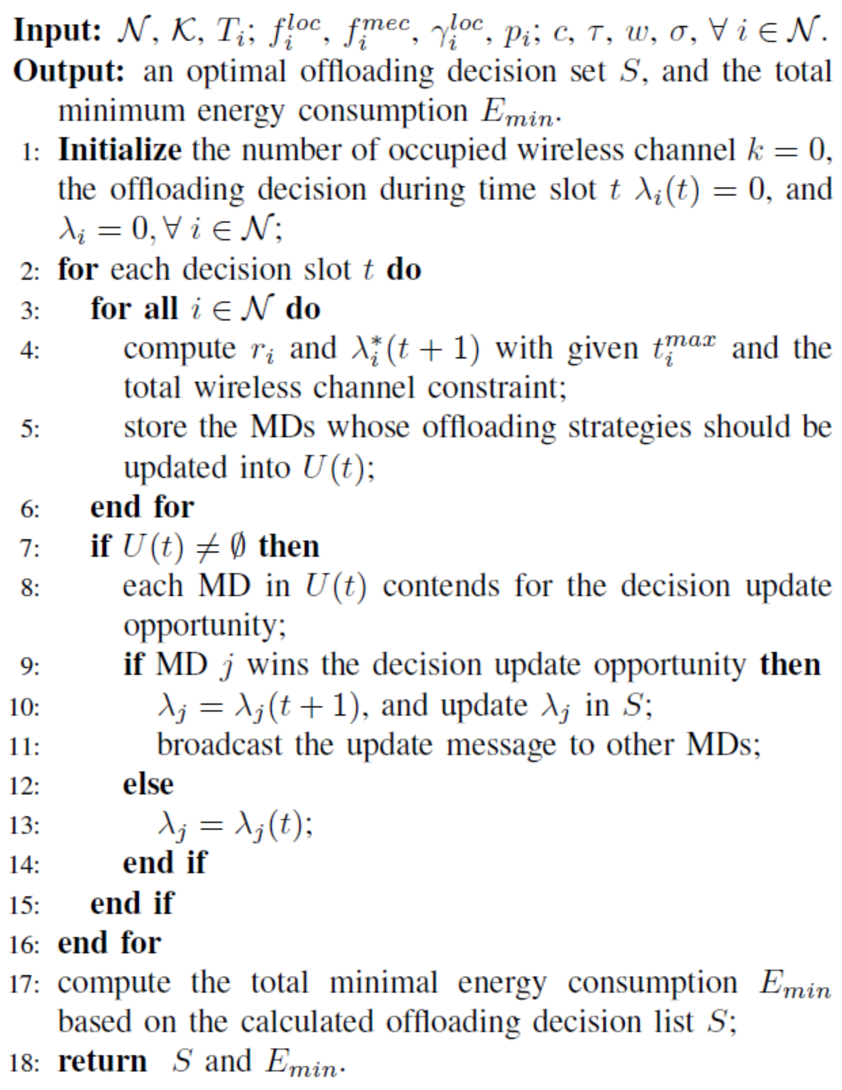
After that, we can formulate the problem of CMOCO as a strategic game $\mathscr{T} = (\mathscr{N}; \{\Lambda_i\}_{i\in N}; \{L_i\}_{i\in N})$, where the set of MDs N is the set of rational game players, i is the strategy set for player i, and the energy consumption function of MD i, i.e., $Li(\Lambda_i; \Lambda_{-i})$, 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.

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 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.

Algorithm 3 : GT-CCO algorithm



```
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 $E_{min}$.
```

ANALYSIS AND DISCUSSIONS

A. ANALYSIS OF NASH EQUILIBRIUM AND CONVERGENCE

Definition 3. *Nash equilibrium (NE):*

For a computation offloading decision profile $\Lambda^* = \{ \Lambda_1^* ; \Lambda_2^* ;\ldots\ldots; \Lambda_N^* \}; \Lambda^* \in \{\Lambda_i\}_{i \in N}$, 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.,

$$L_i(\lambda_i^*, \lambda_{-i}^*) \leq L_i(\lambda_i, \lambda_{-i}^*), \forall \lambda_i \in \Lambda_i, i \in \mathcal{N} \qquad (11)$$

For the collaborative computation offloading game, we have the following conclusion : *The collaborative computation offloading game $\mathcal{T} = (\mathcal{N}; \{\Lambda_i\}_{i \in N} ; \{L_i\}_{i \in N})$ has a NE and possesses the finite improvement property.*

B. ANALYSIS OF COMPUTATIONAL COMPLEXITY

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

Proof. The running time of step 1 in OECCO Algorithm (Algorithm 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.

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

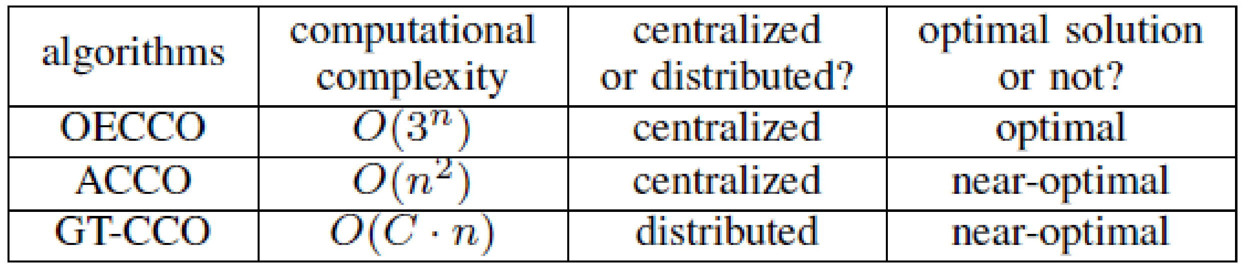
Proof. The running time of ACCO in Algorithm 2 mainly consists of three parts, i.e., the for-all loop from step 2 to step 7, the for-all loop from step 11 to step 16, and the while loop of steps 19 - 30. Specifically, the running time of steps 2 - 7 is O(*n*). For steps of 11 - 16, the outer for-all 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.n*) = O(n^2). Moreover, for the while loop from step 19 to step 30, the outer while and for-all 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).

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

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

C. DISCUSSIONS

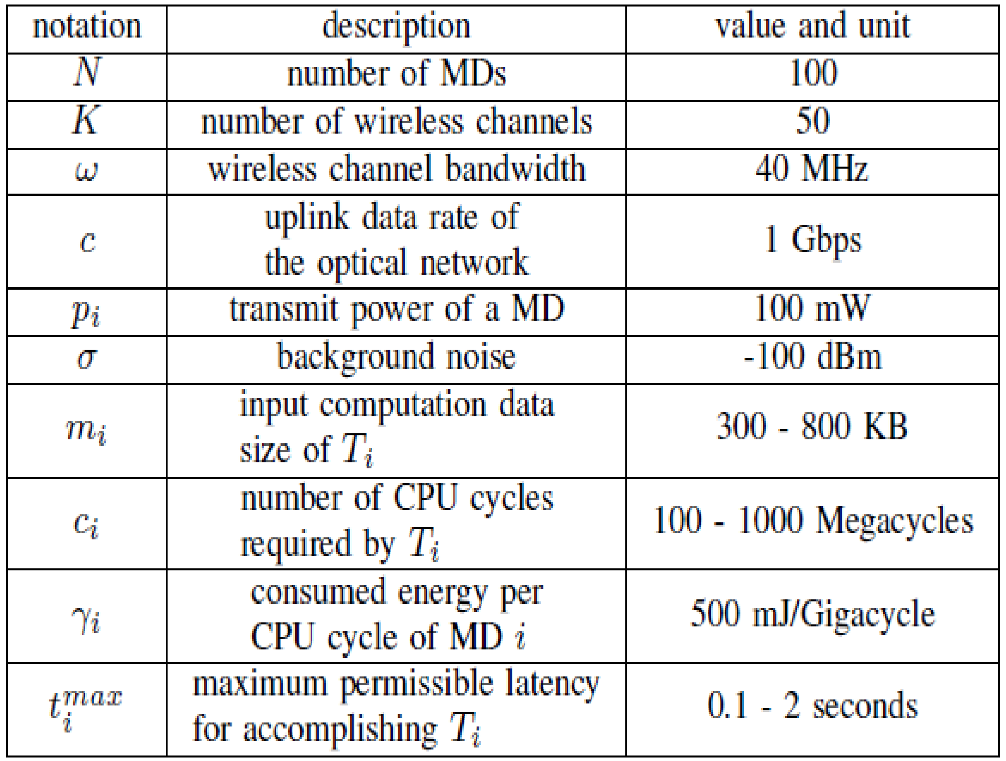
In order to address the problem of collaborative computation offloading with coexistence of centralized cloud and distributed edge, three solutions were presented, i.e., OECCO, ACCO, and GT-CCO. Brief comparisons among them are summarized in Table I. Although it is assumed that there exists only one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

PERFORMANCE EVALUATION

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. $N = 100$ 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. 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.

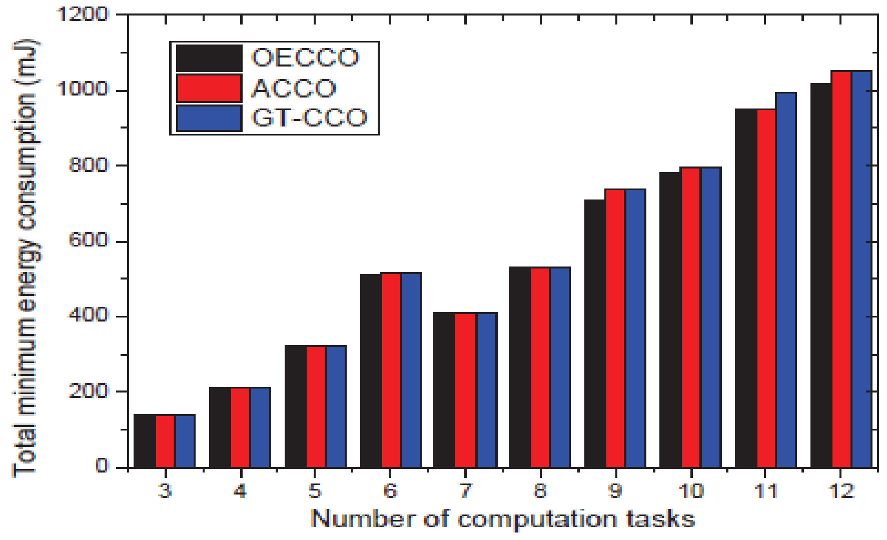


notation	description	value and unit
N	number of MDs	100
K	number of wireless channels	50
ω	wireless channel bandwidth	40 MHz
c	uplink data rate of the optical network	1 Gbps
p_i	transmit power of a MD	100 mW
σ	background noise	-100 dBm
m_i	input computation data size of T_i	300 - 800 KB
c_i	number of 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

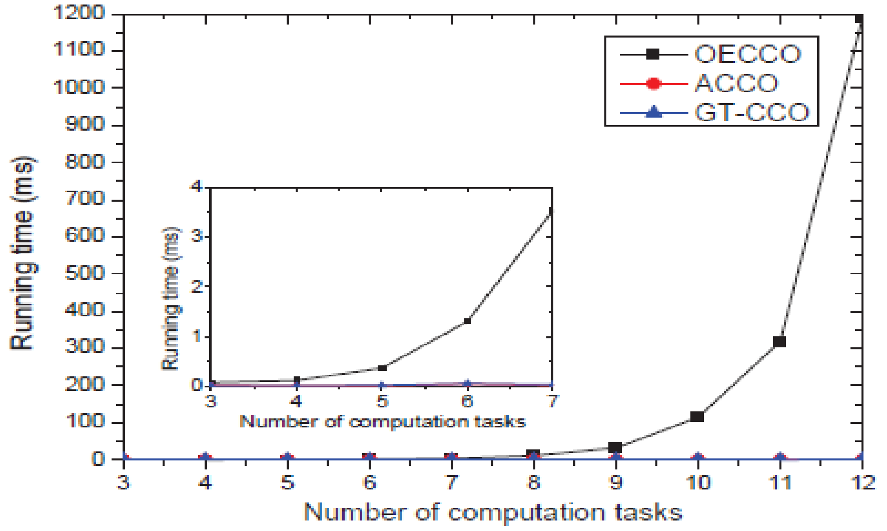
B. NUMERICAL RESULTS

We compare the obtained total minimum energy consumption and the running time of the proposed ACCO and GT-CCO algorithms with that of our benchmark, i.e., OECCO, in order to evaluate their effectiveness and efficiency. Fig. 2 illustrates the comparison results with different number of computation tasks among OECCO, ACCO, and GT-CCO. The number of computation tasks only ranges from 3 to 12 in consideration of OECCO's exponential computational complexity. The following results were observed:

- Compared to our benchmark (OECCO), both ACCO and GT-CCO can find near-optimal solutions to the problem of CMOCO. (Fig. 2(a))
- GT-CCO and ACCO nearly lead to the same results after many times of simulations.
- Both ACCO and GT-CCO are confirmed to have much higher computational efficiency than that of OECCO. (Fig. 2(b))



(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.

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), centralized ACCO, and distributed GT-CCO. From Fig. 3, it can easily be observed that:

- All of DCOA, ACCO, and GTCCO can achieve lower total energy consumption than the scheme via computing locally by all the MDs, and this shows the necessity of computation offloading.
- After many times of simulations, 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.

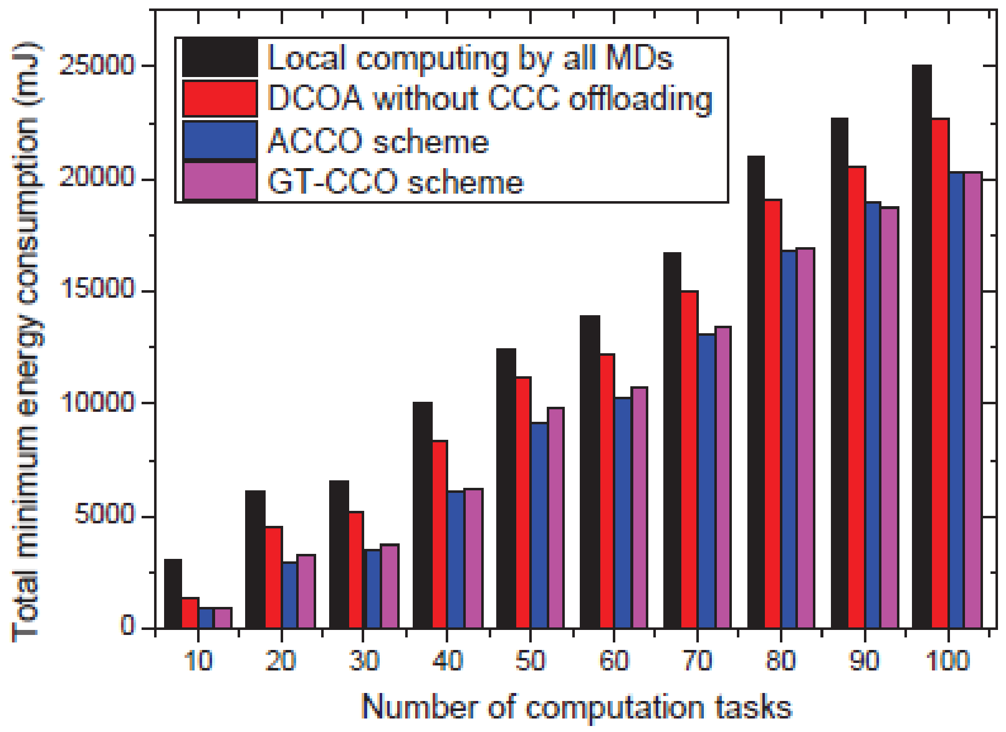


Fig. 3. Comparisons of the total minimum energy consumption by adopting different computation offloading schemes, i.e., local computing by all MDs, DCOA without CCC offloading, centralized ACCO scheme, and distributed GT-CCO scheme.

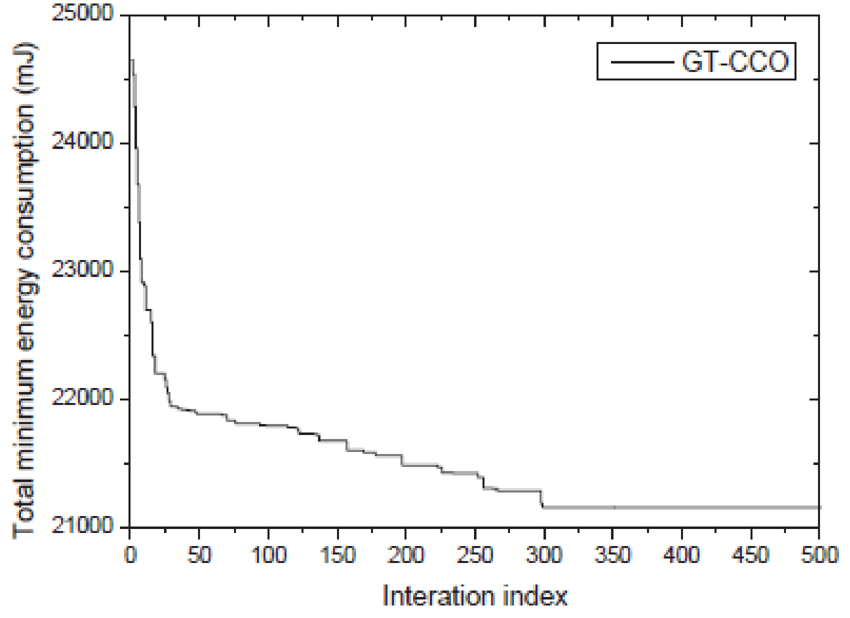


Fig. 4. Convergence behavior of GT-CCO in terms of the total minimum energy consumption.

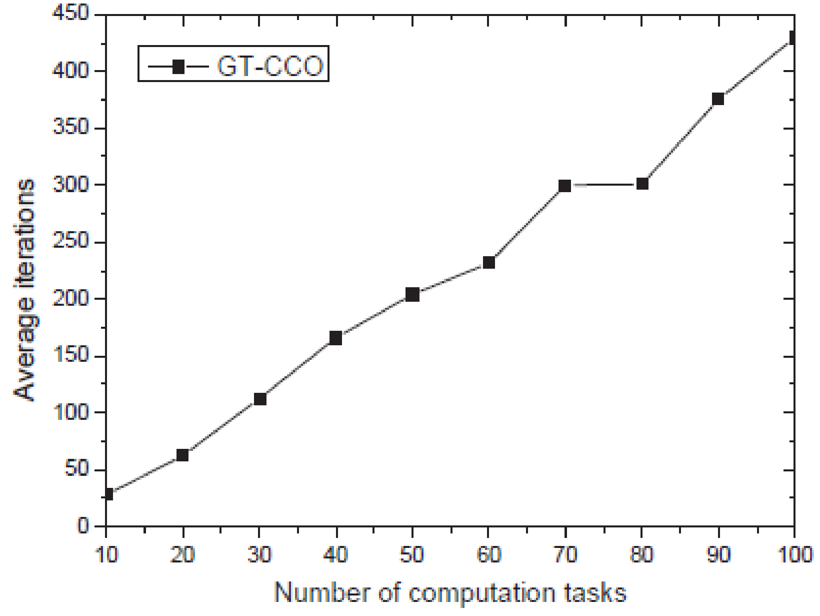


Fig. 5. Average iterations for the convergence of GT-CCO with different number of computation tasks.

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. 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.

CONCLUSION

In this paper, we studied the problem of collaborative computation offloading with centralized cloud and multi-access edge computing. First, an architecture for collaborative computation offloading over FiWi networks was presented, and the problem of cloud-MEC collaborative computation offloading was defined. 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.

SUBMITTED BY :
VATSAL GUPTA (17104060)
ARJAV JAIN (17104070)
ANANYA SHARMA (17104038)

UNDER THE GUIDANCE OF :
PROFESSOR VIKAS HASSIJA