Received November 20, 2017, accepted December 21, 2017, date of publication December 28, 2017, date of current version May 9, 2018.

Digital Object Identifier 10.1109/ACCESS.2017.2787737

Computation Offloading Based on Cooperations of Mobile Edge Computing-Enabled Base Stations

WENHAO FAN[1], (Member, IEEE), YUAN'AN LIU[1], (Member, IEEE), BIHUA TANG[1], FAN WU[1], AND ZHONGBAO WANG[2]

[1]School of Electronic Engineering and the Beijing Key Laboratory of Work Safety Intelligent Monitoring, Beijing University of Posts and Telecommunications, Beijing 100876, China

[2]School of Information Science and Technology, Dalian Maritime University, Dalian 116026, China

Corresponding author: Wenhao Fan (whfan@bupt.edu.cn)

This work was supported in part by the National Natural Science Foundation of China under Grant 61502050, in part by the China Postdoctoral Science Foundation under Grant 2015M580070 and Grant 2016T90066, and in part by the YangFan Innovative & Entrepreneurial Research Team Project of Guangdong Province.

ABSTRACT Mobile edge computing (MEC) can augment the computation capabilities of mobile terminals (MTs) through offloading the computational tasks from the MTs to the MEC-enabled base station (MEC-BS) covering them. However, the load of MEC-BS will rise as the increase of the scale of tasks. Existing schemes try to alleviate the load of MEC-BS through refusing, postponing, or queuing the offloading requests of the MTs; thus, the users' QoS will largely deteriorate due to service interruption and prolonged waiting and execution time. In this paper, we investigate the cooperations of multiple MEC-BSs and propose a novel scheme to enhance the computation offloading service of an MEC-BS through further offloading the extra tasks to other MEC-BSs connected to it. An optimization algorithm is proposed to efficiently solve the optimization problem which maximizes the total benefits of time and energy consumptions gained by all the MTs covered by the MEC-BS. A balance factor is used to flexibly adjust the bias of optimization between minimizations of time and energy consumption. Extensive simulations are carried out in eight different scenarios, and the results demonstrate that our scheme can largely enhance the system performance, and it outperforms the reference scheme in all scenarios.

INDEX TERMS Computation offloading, mobile edge computing, resource management, optimization.

I. INTRODUCTION

In recent years, with rapid development of mobile information industry, the wide use of mobile terminals (MTs) promotes constant emergence of the rich media applications, augmented reality, virtual reality, intelligent video acceleration and other new businesses on mobile platform. These new types of mobile applications (apps) put higher tendencies on high complexity, high energy consumption, and high time delay sensitivity, which bring a big challenge to the computation capabilities and battery capacities of MTs. The contradiction between the high resource occupation of apps and the low capabilities of MTs will exist for a long time, and become severer as the rapid increase of the apps' scales.

Mobile edge computing (MEC) is a new network architecture concept, which enables cloud computing capabilities and an IT service environment at the edge of the cellular networks [1]–[5]. Traditional base stations are updated to MEC-enabled base stations (MEC-BSs) by equipping computation functionality (such as MEC servers) on them, so these MEC-BSs can enable capability augmentation of MTs, more specifically, MEC-BSs can help MTs to process computational tasks, in order to speed up apps' executions and reduce MTs's energy consumptions.

MEC allows a MT to perform computation offloading to offload its computational tasks to the MEC-BS covering it. When the execution of a task at the MEC-BS is done, the MEC-BS will return the task's result to the MT. So, as shown in Fig. 1, the whole process of computation offloading includes 3 parts: 1) the MT sends an offloading request (including necessary information of the computational task) to the MEC-BS; 2) the MEC-BS executes the computational task; 3) the MEC-BS sends the offloading response (including the execution result) to the MT.

Because of limited computation resources, the MEC-BS can not provide endless computation offloading service for all tasks from the MTs under it coverage. Thus, how to manage the resources of MEC-BS efficiently is vital to the system performance maximization. However, the MEC-BS will be

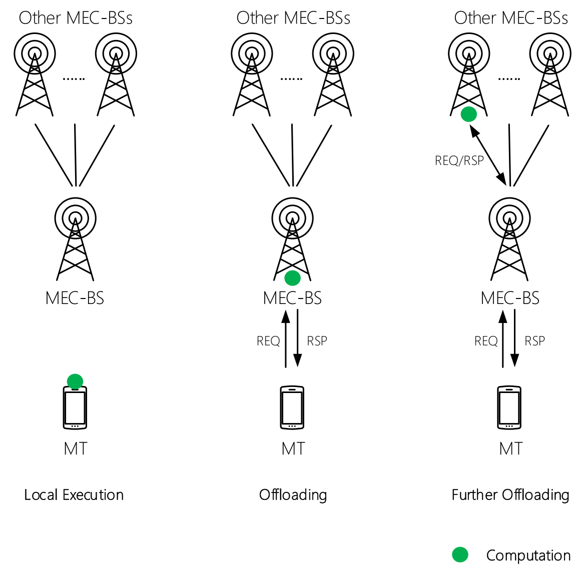

FIGURE 1. **Local execution, offloading, and further offloading.**

still overloaded if there are too many tasks offloaded from the MTs. In existing works, the schemes try to alleviate the load of MEC-BS through refusing, postponing or queuing the offloading requests of MTs. Correspondingly, the users' QoS will largely deteriorate due to service interruptions, and prolonged waiting and execution time.

In this paper, we investigate the cooperations of multiple MEC-BSs, and propose a novel scheme to enhance the computation offloading service of the MEC-BS through further offloading the extra tasks to other MEC-BSs connected to it. As shown in Fig. 1, to process a further offloaded task, there are 3 steps: 1) the original MEC-BS transmits the offloading request (including necessary information of the computational task) of the task to the destination MEC-BS via the connection between them; 2) the destination MEC-BS executes the task; 3) the destination MEC-BS transmits the offloading response (including the execution result) of the task to the original MEC-BS via the connection between them.

In this way, the heavy load of the original MEC-BS can be alleviated effectively through scheduling the tasks to multiple MEC-BSs. The execution time of tasks will decrease due to the load alleviation of the original MEC-BS, and the enlargement of total system capacity, however, meanwhile, extra data transmission time between the original MEC-BS and other MEC-BSs will involve. Our scheme optimizes task scheduling with the goal of maximizing the total benefit gained by all MTs covered by the original MEC-BS. The benefit gained by a MT includes the improvement of time and energy consumptions compared with the consumptions generated when computation offloading is disabled, that is, all tasks from the MT have to be processed at the MT locally. In the optimization algorithm of our scheme, a balance factor is used to adjust the bias between the benefits of time and energy consumptions, so our scheme can provide a very flexible optimization according to different QoS requirements.

In order to evaluate the performance of our scheme, extensive simulations are carried out under 8 different scenarios. The total benefits gained by our scheme and a reference scheme, which disables the cooperations of MEC-BSs, are compared using different criteria. The simulation results demonstrate that our scheme is always superior to the reference scheme in all scenarios, and it can largely alleviate the load of the original MEC-BS and enhance the system performance.

The rest of our paper is organized as follows. Section 2 discusses the related works on computation offloading technologies in MEC. Section 3 presents the system model of our scheme, including the computation, transmission and benefit models. Section 4 describes the cooperative computation offloading algorithm, consisting of the optimization problem and the parallelized processing. Finally, section 5 concludes our work.

II. RELATED WORKS

Mobile edge computing is a hot topic in recent years, and it absorbs high attentions in the design of future generation mobile communication system [2]. Existing works can be divided into 2 predominant categories: A. network architecture based schemes, which investigate the deployments, protocols, interactions in MEC implementation, etc.; B. algorithmic schemes, which manage the computation resources to optimize the system utilization.

A. NETWORK ARCHITECTURE BASED SCHEMES

Edge computing architecture designed in [6] envisages the interconnection of microinstallations at the network edge and data centers in a telcoąŕs central office. Active remote node (ARN) are placed at RAT cell aggregation cite to interface end-users and the core network; innovative distributed data centers consisting of micro-DCs are placed in selected core locations to accelerate the system service. Based on visualization technology, a middleware for MEC is proposed in [7], where MEC servers are located at the aggregation node of multiple RAT base stations and access points. FemtoClouds system proposed in [8] leverages the nearby unutilized mobile devices to serve compute as a service at the network edge. It aims at providing a dynamic and self-configuring multiple device mobile cloud system to scale the computation of Cloudlet by coordinating multiple mobile devices. REPLISOM architecture [9] enables MEC in LTE networks. It augments the evolved NodeB (eNB) with cloud computing resources at the edge that provide clone virtual machine, storage and network resource for specific IoT application.

The European Telecommunications Standards Institute (ETSI) has already published an architecture for Mobile Edge Computing (MEC), which specifically targets cellular networks [2]. In the proposed architecture, MEC servers can be deployed at multiple locations, such as at the LTE macro base station (eNodeB) site, at the 3G Radio Network Controller (RNC) site, at a multi-Radio Access Technology (RAT) cell aggregation site, and at an aggregation point (which may also be at the edge of the core network). The objective is an overarching framework for distributed computing, offloading of applications from mobile devices, and onboarding from third parties. Sabella *et al.* [10] investigate the MEC architecture proposed by ETSI to apply it to IoT services.

In the architectures designed in above works, MEC servers can be deployed at multiple places, such as mobile devices, base stations, RAT aggregation nodes, core networks, etc. Our proposed scheme is an instance of Algorithmic Schemes (category B), and it can be applied to the MEC architectures deploying MEC severs on base stations, to enhance the system performance. Base on a cooperative computation offloading algorithm, our scheme maximizes all MTs' total benefit, which reflexes the improvement of time and energy consumptions brought by computation offloading.

B. ALGORITHMIC SCHEMES

We focus on existing works which improve the performance of MTs via MEC technologies. The main aim of these works is to reduce the latency of task processing or prolonging the battery lives of MTs, through managing the resources of MTs and base stations efficiently using algorithms.

The scheme in [11] aims at minimizing both total tasks' execution latency and the MT's energy consumption by jointly optimizing the task allocation decision and the MT's central process unit (CPU) frequency. A linear relaxation-based approach and a semidefinite relaxation (SDR)-based approach for the fixed CPU frequency case of MT's CPU, and an exhaustive search-based approach and an SDR-based approach for the elastic CPU frequency case of MT's CPU, are proposed. An energy-efficient computation offloading mechanism for MEC in 5G heterogeneous networks is proposed in [12]. An optimization algorithm is designed to jointly optimize offloading and radio resource allocation to obtain the minimal energy consumption under the latency constraints. An computational task scheduling policy for MEC systems is proposed in [13]. By analyzing the average delay of each task and the average power consumption at the mobile device, the authors formulate a power-constrained delay minimization problem, and propose an efficient one-dimensional search algorithm to find the optimal task scheduling policy. Chen *et al.* [14] propose a distributed computation offloading algorithm to efficiently maximize the number of beneficial mobile users. They formulate a distributed computation offloading decision making problem among mobile device users as a multi-user computation offloading game. An integrated framework for computation offloading and interference management in wireless cellular networks with MEC is proposed in [15]. The authors formulate the computation offloading decision, physical resource block (PRB) allocation, and MEC computation resource allocation as optimization problems. The MEC server makes the offloading decision according to the local computation overhead estimated by all user equipments (UEs) and the offloading overhead estimated by the MEC server itself. Then, the MEC server performs the PRB allocation using the graph coloring method. The scheme in [16] investigates an energy efficiency computation offloading scheme with performance guaranteed problem in mobile-edge computing. KKT conditions are applied in order to solve the energy minimizing optimization problem which is determined by energy consumption and bandwidth capacity at each time slot. Sardellitti *et al.* [17] consider an MIMO multi-cell system where multiple mobile users (MUs) request computation offloading from a common cloud server. Algorithms are proposed to solve the joint optimization of the radio resources to minimize the overall users' energy consumption, while meeting latency constraints in single-user case and multi-user case.

In summary, the algorithmic schemes proposed in above works can efficiently solve the computation offloading problems under different constraints and scenarios, however, none of them consider the load alleviation problem for the MEC-BS when it is overloaded. Our proposed scheme aims at enhancing MTs' total benefit of time and energy consumptions, while alleviating the load of the original MEC-BS by scheduling the computational tasks to other MEC-BSs connected to the original MEC-BS. Thus, our scheme can efficiently optimize the system performance, especially when the system load is heavy.

III. SYSTEM MODEL

We consider a scenario consisting of a set $\mathcal{B}$ of B ($|\mathcal{B}| = B$) MEC-BSs. Each MEC-BS b_k ($k \in \mathcal{B} = \{1, 2, \ldots, B\}$) is equipped with a MEC server, so it is capable to provide computation offloading service for the MTs under its coverage.

As the 1st MEC-BS in $\mathcal{B}$, b_1 is connected to other MEC-BSs ($b_k, \forall k \in \mathcal{B} - \{1\} = \{2, 3, \ldots, B\}$) via wired connections, and it covers a set $\mathcal{M}$ of M ($|\mathcal{M}| = M$) MTs. We define as m_i the ith ($i \in \mathcal{M} = \{1, 2, \ldots, M\}$) MT covered by b_1.

A MT can run multiple mobile applications concurrently, and each application may contain multiple computation-sensitive tasks. We use a set $\mathcal{H}$ ($|\mathcal{H}| = H$) to include all types of these tasks of all MTs in $\mathcal{M}$, and express the jth type of tasks as h_j ($j \in \mathcal{H} = \{1, 2, \ldots, H\}$).

Each h_j is profiled by an ordered vector $< c_j, q_j, s_j >$, which is characterized by: 1) c_j, the amount of h_j's computation; 2) q_j, the size of the offloading request (including necessary description and parameters of h_j) for h_j sent by a MT to a MEC-BS; 3) s_j, the size of the offloading response (including the result of h_j's execution) for h_j received by a MT from a MEC-BS.

m_i has a probability $p_{i,j}$ ($p_{i,j} \in [0, 1]$) to generate a h_j during its running period. We express $h_{i,j}$ as a h_j generated by m_i. Note that, $p_{i,j}$ actually represents the proportion of $h_{i,j}$ in the tasks generated by m_i, so we have $\sum_{j \in \mathcal{H}} p_{i,j} = 1$. We assume the task generation of a MT satisfies the Poisson distribution. The task generation rate of m_i is defined as λ_i, that is, λ_i tasks are generated by m_i per second.

There are 2 ways to completing $h_{i,j}$: execute it locally or offload it remotely. If $h_{i,j}$ is executed at m_i, the efficiency may decrease due to low computation capability of m_i, which may cause time and energy consumptions by m_i executing $h_{i,j}$; if $h_{i,j}$ is offloaded to b_1, the efficiency may benefit from the b_1's powerful computation resources, but at the same time, it may suffer from the time consumption and energy consumption caused by data transmission between m_i and MEC-BSs.

When $h_{i,j}$ is offloaded to b_1, it can be executed at b_1, or be further offloaded to another MEC-BS through the connection between the 2 MEC-BSs, if b_1 is overloaded or under heavy load so that it can not guarantee the required QoS to $h_{i,j}$.

We define as $\boldsymbol{\alpha} = \{\alpha_{i,j,k} | i \in \mathcal{M}, j \in \mathcal{H}, k \in \mathcal{B}\}$ the selection probability set to express the probability that MT selects local execution, offloading, further offloading for each task in the scenario.

For $h_{i,j}$, given $\forall k \in \mathcal{B}$, the value of $\alpha_{i,j,k}$ represents: 1) the probability that $h_{i,j}$ is offloaded from m_i to b_1, if $k = 1$; 2) the probability that $h_{i,j}$ is offloaded from m_i to b_1, then is further offloaded to b_k, if $k \neq 1$. Obviously, $\alpha_{i,j,k} \in [0, 1]$. Note that, $\sum_{k\in\mathcal{B}} \alpha_{i,j,k} \in [0, 1]$, which represents the total probability that $h_{i,j}$ is offloaded to any one MEC-BS in $\mathcal{B}$, thus, the probability that the task is executed at m_i is $1 - \sum_{k\in\mathcal{B}} \alpha_{i,j,k}$.

A. COMPUTATION MODEL

When $h_{i,j}$ begins to be executed at m_i or a MEC-BS, it must wait in a queue, which includes all pending tasks arriving before $h_{i,j}$. According to queuing theory [18], we model the execution of $h_{i,j}$ as a M/M/1 queuing system.

1) EXECUTION AT MT

The computation resource of m_i are shared by all its tasks executed locally.

By defining as θ_i the service rate of m_i, if $h_{i,j}$ is selected to be executed at m_i, the time consumed by completing $h_{i,j}$ is

$$t_{i,j}^{\mathrm{MT}} = \frac{p_{i,j}\lambda_i c_{i,j}}{\theta_i - \sum_{j\in\mathcal{H}} \left((1 - \sum_{k\in\mathcal{B}} \alpha_{i,j,k}) p_{i,j}\lambda_i c_j\right)} \quad (1)$$

where the denominator is the stable processing speed [18] (amount of computation processed per second) of m_i. $\sum_{j\in\mathcal{H}} \left((1 - \sum_{k\in\mathcal{B}} \alpha_{i,j,k}) p_{i,j}\lambda_i c_{i,j}\right)$ is the total amount of computation of m_i's tasks executed locally per second. It can be observed that the processing speed of m_i decreases as the increase of m_i's tasks executed locally. Note that, $\theta_i - \sum_{j\in\mathcal{H}} \left((1 - \sum_{k\in\mathcal{B}} \alpha_{i,j,k}) p_{i,j}\lambda_i c_j\right) > 0$ is the hard constraint [18] of Formula (1), which means the tasks' arriving rate cannot exceed m_i's service rate.

The energy consumed by executing $h_{i,j}$ at m_i is given by

$$e_{i,j}^{\mathrm{MT}} = \zeta_i p_{i,j}\lambda_i c_j \quad (2)$$

where ζ_i is a factor denoting the energy consumed by executing per amount of computation at m_i.

2) EXECUTION AT MEC-BS

The computation resources of a MEC-BS are shared by the tasks offloaded to the MEC-BS.

By defining as μ_k the service rate of b_k, we have the time consumed by completing $h_{i,j}$ if it is selected to be offloaded to b_k

$$t_{i,j,k}^{\mathrm{BS}} = \frac{p_{i,j}\lambda_i c_{i,j}}{\mu_k - \sum_{i\in\mathcal{M}} \sum_{j\in\mathcal{H}} (\alpha_{i,j,k} p_{i,j}\lambda_i c_j)} \quad (3)$$

where the denominator is the stable processing speed [18] (amount of computation processed per second) of b_k. $\sum_{i\in\mathcal{M}} \sum_{j\in\mathcal{H}} (\alpha_{i,j,k} p_{i,j}\lambda_i c_j)$ sums up the amount of computation of each MT's tasks offloaded to b_k. It can be observed that the processing speed of b_k decreases as the increase of the tasks offloaded to b_k. Note that, $\mu_k - \sum_{i\in\mathcal{M}} \sum_{j\in\mathcal{H}} (\alpha_{i,j,k} p_{i,j}\lambda_i c_j) > 0$ is the hard constraint [18] of Formula (3), which means the tasks' arriving rate cannot exceed b_k's service rate.

$h_{i,j}$ is offloaded to and executed at MEC-BS side, thus, there are no energy consumptions generated at m_i.

B. TRANSMISSION MODEL

1) COMMUNICATIONS BETWEEN MTS AND b_1

The wireless resources provided by b_1 are shared by the MTs under its coverage. We ignore the impacts of inter-BS and intra-BS interferences caused by computation offloading, because the sizes of offloading requests and responses transmitted between the MT and MEC-BS are tiny.

We define as $r_i^{\mathrm{MT}\to\mathrm{BS1}}$ the uplink data transmission rate from m_i to b_1. Then we have the time consumed by sending the offloading request of $h_{i,j}$ from m_i to b_1, if $h_{i,j}$ is selected to be offloaded.

$$t_{i,j}^{\mathrm{MT}\to\mathrm{BS1}} = \frac{p_{i,j}\lambda_i q_j}{r_i^{\mathrm{MT}\to\mathrm{BS1}}} \quad (4)$$

Let ω_i be the transmit power used by m_i in the uplink data transmission from m_i to b_1. The energy consumption of m_i for the transmission is

$$e_{i,j}^{\mathrm{MT}\to\mathrm{BS1}} = w_i t_{i,j}^{\mathrm{MT}\to\mathrm{BS1}} \quad (5)$$

The downlink data transmission rate from b_1 to m_i is denoted by $r_i^{\mathrm{BS1}\to\mathrm{MT}}$. Then we have the time consumed by receiving the offloading response of $h_{i,j}$ from b_1 to m_i

$$t_{i,j}^{\mathrm{BS1}\to\mathrm{MT}} = \frac{p_{i,j}\lambda_i s_j}{r_i^{\mathrm{BS1}\to\mathrm{MT}}} \quad (6)$$

The energy consumption of m_i for the transmissions can be ignored, because the power used by the m_i to receive an offloading response is very low.

2) COMMUNICATIONS BETWEEN b_1 AND OTHER MEC-BSs

The wired connections between b_1 and other MEC-BSs are bi-directional. We define as $r_k^{\mathrm{BS1}\to\mathrm{BS}}$ the data transmission rate from b_1 to b_k through the wired connection between them. Reversely, $r_k^{\mathrm{BS}\to\mathrm{BS1}}$ represents the data transmission rate from b_k to b_1 through the connection. Note that, the transmissions between b_1 and b_k are not applicable if $k = 1$, so we set $r_1^{\text{BS1}\to\text{BS}} = \infty$ and $r_1^{\text{BS}\to\text{BS1}} = \infty$. Besides, $r_k^{\text{BS1}\to\text{BS}} = r_k^{\text{BS}\to\text{BS1}}$ if the connection between b_1 and b_k is symmetric, else $r_k^{\text{BS1}\to\text{BS}} \neq r_k^{\text{BS}\to\text{BS1}}$. We consider the data transmission rates between b_1 and other MEC-BSs are high, whereas, the sizes of offloading requests and responses are tiny, thus, the impact caused by concurrently transmitted tasks can be ignored.

The time consumed by transmitting the offloading request of $h_{i,j}$ from b_1 to b_k is expressed as

$$t_{i,j,k}^{\text{BS1}\to\text{BS}} = \frac{p_{i,j}\lambda_i q_j}{r_k^{\text{BS1}\to\text{BS}}} \tag{7}$$

Similarly, the time consumed by transmitting the offloading response of $h_{i,j}$ from b_k to b_1 is expressed as

$$t_{i,j,k}^{\text{BS}\to\text{BS1}} = \frac{p_{i,j}\lambda_i s_j}{r_k^{\text{BS}\to\text{BS1}}} \tag{8}$$

The energy consumptions of m_i for above transmissions are 0, since the transmissions only happen among MEC-BSs.

C. BENEFIT MODEL

The total time consumption for completing $h_{i,j}$ includes: 1) the time consumed by local execution, if $h_{i,j}$ is selected to be executed at m_i; 2) the time consumed by computation offloading, if $h_{i,j}$ is selected to be offloaded to b_1; 3) the time consumed by computation offloading, if $h_{i,j}$ is selected to be further offloaded to $b_k, \forall k \in \mathcal{B} - \{1\}$.

In 1), the time consumption is generated by executing $h_{i,j}$ at m_i, that is, $(1 - \sum_{k\in\mathcal{B}} \alpha_{i,j,k}) t_{i,j}^{\text{MT}}$.

In 2), the time consumption is generated by transmitting the offloading request of $h_{i,j}$ from m_i to b_1, executing $h_{i,j}$ at b_1, and transmitting the offloading response of $h_{i,j}$ from b_1 to m_i, that is, $\alpha_{i,j,1}(t_{i,j}^{\text{MT}\to\text{BS1}} + t_{i,j,1}^{\text{BS}} + t_{i,j}^{\text{BS1}\to\text{MT}})$.

In 3), the time consumption is generated by transmitting the offloading request of $h_{i,j}$ from m_i to b_1, then from b_1 to b_k, executing $h_{i,j}$ at b_1, and transmitting the offloading response of $h_{i,j}$ from b_k to b_1, then from b_1 to m_i, that is, $\alpha_{i,j,k}(t_{i,j}^{\text{MT}\to\text{BS1}} + t_{i,j,k}^{\text{BS1}\to\text{BS}} + t_{i,j,k}^{\text{BS}} + t_{i,j,k}^{\text{BS}\to\text{BS1}} + t_{i,j}^{\text{BS1}\to\text{MT}})$.

In summary, we have the total time consumption for completing $h_{i,j}$

$$\begin{aligned} t_{i,j} &= (1 - \sum_{k\in\mathcal{B}} \alpha_{i,j,k}) t_{i,j}^{\text{MT}} + \alpha_{i,j,1}(t_{i,j}^{\text{MT}\to\text{BS1}} + t_{i,j,1}^{\text{BS}} \\ &\quad + t_{i,j}^{\text{BS1}\to\text{MT}}) + \sum_{k\in\mathcal{B}-\{1\}} \Big(\alpha_{i,j,k}(t_{i,j}^{\text{MT}\to\text{BS1}} + t_{i,j,k}^{\text{BS1}\to\text{BS}} \\ &\quad + t_{i,j,k}^{\text{BS}} + t_{i,j,k}^{\text{BS}\to\text{BS1}} + t_{i,j}^{\text{BS1}\to\text{MT}})\Big) \\ &= (1 - \sum_{k\in\mathcal{B}} \alpha_{i,j,k}) t_{i,j}^{\text{MT}} + \sum_{k\in\mathcal{B}} \Big(\alpha_{i,j,k}(t_{i,j}^{\text{MT}\to\text{BS1}} + t_{i,j,k}^{\text{BS1}\to\text{BS}} \\ &\quad + t_{i,j,k}^{\text{BS}} + t_{i,j,k}^{\text{BS}\to\text{BS1}} + t_{i,j}^{\text{BS1}\to\text{MT}})\Big) \end{aligned} \tag{9}$$

Note that, $t_{i,j,1}^{\text{BS1}\to\text{BS}} = 0$ and $t_{i,j,k}^{\text{BS}\to\text{BS1}} = 0$ since $r_1^{\text{BS1}\to\text{BS}} = \infty$ and $r_1^{\text{BS}\to\text{BS1}} = \infty$, so the formulas of 2) and 3) can be combined together.

The total energy consumption on m_i for completing $h_{i,j}$ includes: 1) the energy consumed by local execution, if $h_{i,j}$ is selected to be executed at m_i; 2) the energy consumed by computation offloading, if $h_{i,j}$ is selected to be offloaded to b_1, or further offloaded to $b_k, k \in \mathcal{B} - \{1\}$.

In 1), the energy consumption on m_i is generated by executing $h_{i,j}$ at m_i, that is, $(1 - \sum_{k\in\mathcal{M}} \alpha_{i,j,k}) e_{i,j}^{\text{MT}}$.

In 2), the energy consumption on m_i is generated by transmitting the offloading request of $h_{i,j}$ from m_i to b_1, that is, $\alpha_{i,j,k} e_{i,j}^{\text{MT}\to\text{BS1}}$.

In summary, we have the total energy consumption for completing $h_{i,j}$

$$e_{i,j} = (1 - \sum_{k\in\mathcal{M}} \alpha_{i,j,k}) e_{i,j}^{\text{MT}} + \sum_{k\in\mathcal{B}} (\alpha_{i,j,k} e_{i,j}^{\text{MT}\to\text{BS1}}) \tag{10}$$

Therefore, the total time consumption and energy consumption of all MTs covered by b_1 can be formulated as $\sum_{i\in\mathcal{M}} \sum_{j\in\mathcal{H}} t_{i,j}$ and $\sum_{i\in\mathcal{M}} \sum_{j\in\mathcal{H}} e_{i,j}$, respectively.

In order to quantitatively measure the total benefit gained by all MTs in $\mathcal{M}$ through computation offloading, we define the benefit function for as

$$f = \tau \frac{\tilde{t} - \sum_{i\in\mathcal{M}} \sum_{j\in\mathcal{H}} t_{i,j}}{\tilde{t}} + (1-\tau) \frac{\tilde{e} - \sum_{i\in\mathcal{M}} \sum_{j\in\mathcal{H}} e_{i,j}}{\tilde{e}} \tag{11}$$

where $\tilde{t}$ and $\tilde{e}$ are the total time consumption and total energy consumption without computation offloading, respectively. Namely, $\tilde{t} = \sum_{i\in\mathcal{M}} \sum_{j\in\mathcal{H}} t_{i,j} | \alpha_{i,j,k} = 0$, $\tilde{e} = \sum_{i\in\mathcal{M}} \sum_{j\in\mathcal{H}} e_{i,j} | \alpha_{i,j,k} = 0, \forall i \in \mathcal{M}, \forall i \in \mathcal{M}, \forall j \in \mathcal{H}, \forall k \in \mathcal{B}$.

The benefit function reflexes the improvement of time and energy consumption via computation offloading, with consideration of the trade-off between the benefit of time consumption and the benefit of energy consumption. $\tau_{i,k,l} \in [0, 1]$ is a balance factor configuring the proportions of the two benefits.

IV. COOPERATIVE COMPUTATION OFFLOADING ALGORITHM

A. OPTIMIZATION PROBLEM

The aim of our algorithm is to maximize the total benefit f gained by all MTs in $\mathcal{M}$, while ensuring all constraints are not violated. The aim is equivalent to minimizing $-f$, thus the corresponding optimization problem can be formulated as

$$\underset{\alpha}{\text{minimize}} \quad -f \tag{12}$$

$$\text{subject to } \alpha_{i,j,k} \in [0, 1], \quad \forall i \in \mathcal{M}, \forall j \in \mathcal{H}, \forall k \in \mathcal{B} \tag{13}$$

$$\sum_{k\in\mathcal{B}} \alpha_{i,j,k} \in [0, 1], \quad \forall i \in \mathcal{M}, \forall j \in \mathcal{H} \tag{14}$$

$$\theta_i - \sum_{j\in\mathcal{H}} \Big((1 - \sum_{k\in\mathcal{B}} \alpha_{i,j,k}) p_{i,j}\lambda_i c_j)\Big) > 0, \quad \forall i \in \mathcal{M} \tag{15}$$

$$\mu_k - \sum_{i\in\mathcal{M}} \sum_{j\in\mathcal{H}} (\alpha_{i,j,k} p_{i,j}\lambda_i c_j) > 0, \quad \forall k \in \mathcal{B} \tag{16}$$

where constraint (13) is the value range of each $\alpha_{i,j,k}$. As aforementioned, Constraint (14) is the value range of the total probability that $h_{i,j}$ is offloaded. Constraint (15) and (16) are hard constraints of the queuing systems of each MT in $\mathcal{M}$ and b_1, respectively.

We expand $\tilde{t}$ as

$$\tilde{t} = \sum_{i\in\mathcal{M}}\sum_{j\in\mathcal{H}} t_{i,j}^{\mathrm{MT}}|_{\alpha_{i,j,k}=0} = \sum_{i\in\mathcal{M}}\sum_{j\in\mathcal{H}}\left(\frac{p_{i,j}\lambda_i c_{i,j}}{\theta_i - \sum_{j\in\mathcal{H}}(p_{i,j}\lambda_i c_j)}\right) \tag{17}$$

It can be observed that the sufficient condition of $\tilde{t}$ is that $\forall i \in \mathcal{M}$, $\theta_i - \sum_{j\in\mathcal{H}}(p_{i,j}\lambda_i c_j) > 0$ must hold. By comparing the condition with constraint (15), $\forall i \in \mathcal{M}$, we have

$$\theta_i - \sum_{j\in\mathcal{H}}\Big((1-\sum_{k\in\mathcal{B}}\alpha_{i,j,k})p_{i,j}\lambda_i c_j)\Big) \geq \theta_i - \sum_{j\in\mathcal{H}}(p_{i,j}\lambda_i c_j) > 0 \tag{18}$$

Thus, constraint (15) always holds.

B. CONVEXITY OF OPTIMIZATION PROBLEM

Substituting relevant formulas, we can expand the target function (12) as Formula (19), as shown at the bottom of this page, which is a linear combination of nonlinear function f_1 and f_2, and linear function f_3 and f_4. The convexity of f_1 and f_2 are proved in Appendix A and Appendix B, respectively. Constraint (13), (14) and (16) are all linear functions. Thus, the optimization problem is convex, and it has a global minimum [19].

C. OPTIMIZATION ALGORITHM USING INTERIOR POINT METHOD

We design an algorithm to solve the optimization problem using the interior point method [19] with logarithmic barrier function. By making constraint (13), (14) and (16) implicit in the objection function f, we rewrite the optimization problem as

$$\begin{aligned}
&\underset{\boldsymbol{\alpha}}{\text{minimize}}\ \big(-f+\phi B(\boldsymbol{\alpha})\big)\\
&= \underset{\boldsymbol{\alpha}}{\text{minimize}}\ \Bigg(-f+\phi\Bigg(\sum_{i\in\mathcal{M}}\sum_{j\in\mathcal{H}}\sum_{k\in\mathcal{B}}(-\log\alpha_{i,j,k})\\
&\quad+\sum_{i\in\mathcal{M}}\sum_{j\in\mathcal{H}}\sum_{k\in\mathcal{B}}\log(\alpha_{i,j,k}-1)\\
&\quad+\sum_{i\in\mathcal{M}}\sum_{j\in\mathcal{H}}\big(-\log(\sum_{k\in\mathcal{B}}\alpha_{i,j,k})\big)\\
&\quad+\sum_{i\in\mathcal{M}}\sum_{j\in\mathcal{H}}(\log(\sum_{k\in\mathcal{B}}\alpha_{i,j,k}-1)\\
&\quad+\sum_{k\in\mathcal{B}}\Big(\log\Big(\sum_{i\in\mathcal{M}}\sum_{j\in\mathcal{H}}(\alpha_{i,j,k}p_{i,j}\lambda_i c_j)-\mu_k\Big)\Big)\Bigg)\Bigg)
\end{aligned} \tag{20}$$

$$\text{subject to } \boldsymbol{\alpha} \in \mathbf{O} \tag{21}$$

where $\mathbf{O}$ is the feasible region of the problem, which is a space described by constraint (13), (14) and (16). We use $\boldsymbol{\alpha}^{(0)} = 0$ as the initial point since the interior point method must begin in $\mathbf{O}$, and $\boldsymbol{\alpha}^{(0)}$ is a strictly feasible point in $\mathbf{O}$.

Algorithm 1 Algorithm Solving Optimization Problem

```
Given $\boldsymbol{\alpha}^{(0)} = 0, \epsilon, \phi^{(0)}, \pi$;
$k = 0$;
loop
  Solve the problem (20) using Newton's method with $\boldsymbol{\alpha}^{(k)}$ as initial point, and $\phi^{(k)}$ as the value of $\phi$, obtain the solution $\boldsymbol{\alpha}^{(k)}$;
  if $\phi^{(k)}B(\boldsymbol{\alpha}^{(k)}) < \epsilon$
    break;
  end if
  $\boldsymbol{\alpha}^{(k+1)} <= \boldsymbol{\alpha}^{(k)}$;
  $\phi^{(k+1)} <= \pi\phi^{(k)}$;
  $k <= k+1$;
end loop
Optimal solution $\boldsymbol{\alpha}^{*} <= \boldsymbol{\alpha}^{(k)}$;
```

Now we can give the optimization algorithm using interior point method, which is shown in **Algorithm 1**.

$$\begin{aligned}
-f &= -1 + \underbrace{\frac{\tau}{\tilde{t}}\sum_{i\in\mathcal{M}}\left(\frac{\sum_{j\in\mathcal{H}}\big((1-\sum_{k\in\mathcal{B}}\alpha_{i,j,k})p_{i,j}\lambda_i c_j\big)}{\theta_i - \sum_{j\in\mathcal{H}}\big((1-\sum_{k\in\mathcal{B}}\alpha_{i,j,k})p_{i,j}\lambda_i c_j\big)}\right)}_{f_1} + \underbrace{\frac{\tau}{\tilde{t}}\sum_{k\in\mathcal{B}}\left(\frac{\sum_{i\in\mathcal{M}}\sum_{j\in\mathcal{H}}(\alpha_{i,j,k}p_{i,j}\lambda_i c_j)}{\mu_k - \sum_{i\in\mathcal{M}}\sum_{j\in\mathcal{H}}(\alpha_{i,j,k}p_{i,j}\lambda_i c_j)}\right)}_{f_2}\\
&\quad+\underbrace{\frac{\tau}{\tilde{t}}\sum_{i\in\mathcal{M}}\sum_{j\in\mathcal{H}}\sum_{k\in\mathcal{B}}\left(\alpha_{i,j,k}p_{i,j}\lambda_i\Big(\frac{q_j}{r_i^{\mathrm{MT}\to\mathrm{BS1}}}+\frac{q_j}{r_k^{\mathrm{BS1}\to\mathrm{BS}}}+\frac{s_j}{r_k^{\mathrm{BS}\to\mathrm{BS1}}}+\frac{s_j}{r_k^{\mathrm{BS1}\to\mathrm{MT}}}\Big)\right)}_{f_3}\\
&\quad+\underbrace{\frac{1-\tau}{\tilde{e}}\sum_{i\in\mathcal{M}}\sum_{j\in\mathcal{H}}\left((1-\sum_{k\in\mathcal{B}}\alpha_{i,j,k})\zeta_i p_{i,j}\lambda_i c_j + \sum_{k\in\mathcal{B}}\Big(\frac{\alpha_{i,j,k}\omega_i p_{i,j}\lambda_i q_j}{r_i^{\mathrm{MT}\to\mathrm{BS1}}}\Big)\right)}_{f_4}
\end{aligned} \tag{19}$$

D. COOPERATIVE COMPUTATION OFFLOADING ALGORITHM

The computation offloading for each MT in $\mathcal{M}$ is managed by b_1 using cooperative computation offloading algorithm, which is in charge of collecting and monitoring the information (offloading requests and responses, parameters of MTs and MEC-BSs) sent by the MTs and other MEC-BSs, running the optimization algorithm, and sending the optimization result to each MT.

Local execution, offloading or further offloading for a certain task are decided by the selection probability in the optimization result sent from b_1 to the MT. For $h_{i,j}$, $1-\sum_{k\in\mathcal{B}}\alpha_{i,j,k}$ represents the probability of local execution at m_i; $\alpha_{i,j,1}$ represents the probability of offloading to b_1; $\alpha_{i,j,k}, \forall k \in \mathcal{B}-\{1\}$ represents the probability of further offloading from b_1 to b_k.

At initialization of the system, all MTs in $\mathcal{M}$ upload their required parameters, including $\forall i \in \mathcal{M}, \forall j \in \mathcal{H}, p_{i,j}, \lambda_i, \theta_i, \zeta_i, \omega_i$ and the profiles of all its tasks (As aforementioned, $\forall i \in \mathcal{M}, \forall j \in \mathcal{H}, < c_j, q_j, s_j >$), to b_1. b_1 also collects the required parameters of all MEC-BSs, including its parameters μ_1, $r_1^{\mathrm{MT}\rightarrow\mathrm{BS1}}$, $r_1^{\mathrm{BS1}\rightarrow\mathrm{MT}}$ and $\forall k \in \mathcal{M}-\{1\}, r_k^{\mathrm{BS1}\rightarrow\mathrm{BS}}$, and other MEC-BSs' parameters $\forall k \in \mathcal{M}-\{1\}, \mu_k$ and $r_k^{\mathrm{BS}\rightarrow\mathrm{BS1}}$.

During the running period of the system, for a certain MT or a certain MEC-BS except b_1, if the values of any its parameters change, the MT or the MEC-BS only needs to send the new values to b_1, in order to update the corresponding parameters collected by b_1. b_1 also updates its parameters if their values change.

b_1 monitors the value changes of all parameters periodically. If value changes happen in a period, b_1 will run **Algorithm 1** in the next period. The periodical mechanism balances the frequency of the algorithm's execution and the timeliness of the optimization result.

The detail of the cooperative computation offloading algorithm is described in **Algorithm 2** for MT side, **Algorithm 3** for other MEC-BSs side, and **Algorithm 4** for b_1 side. The 3 algorithms is composed of 8 loops, *L1-L8*. During the running period of the system, these loops are deployed into separate processes, and are executed in parallel.

(*L1*): if m_i receives $\alpha_{i,j,k}$ from b_1, it will update the value of $\alpha_{i,j,k}$ stored in $\boldsymbol{\alpha}_i$;

(*L2*): if any one of $p_{i,j}$, λ_i, θ_i, ζ_i, ω_i and the profile of any task of m_i changes, m_i will send the new value of the parameter to b_1;

(*L3*): when m_i has a new $h_{i,j}$, firstly, it generates a random number $o \in [0, 1]$ based on Uniform distribution. Then, it divides $[0, 1]$ into several intervals by accumulating the value of each $\alpha_{i,j,k}$ from $k = 1$ to B. If o falls into the interval representing k, then m_i will offload $h_{i,j}$ to b_k by sending offloading request and receiving offloading response. Else, o belongs to none of these intervals, then, m_i will execute $h_{i,j}$ locally.

(*L4*): if any one of μ_k and $r_k^{\mathrm{BS}\rightarrow\mathrm{BS1}}$ changes, b_k will send the new value of the parameter to b_1;

Algorithm 2 Cooperative Computation Offloading Algorithm Algorithm at m_i Side

```
Initial: set $\boldsymbol{\alpha}_i = \{\alpha_{i,j,k} | \forall j \in \mathcal{H}, \forall k \in \mathcal{B}\} = 0$, send $p_{i,j}$, $\lambda_i$, $\theta_i$, $\zeta_i$, $\omega_i$ and the profiles of all its tasks to $b_i$.
L1:
if $m_i$ receives $\alpha_{i,j,k}$ from $b_1$ then
  $m_i$ updates the value of $\alpha_{i,j,k}$ stored in $\boldsymbol{\alpha}_i$;
end if

L2:
if any one of $p_{i,j}$, $\lambda_i$, $\theta_i$, $\zeta_i$, $\omega_i$ and the profile of any task of $m_i$ changes then
  $m_i$ sends the new value of the parameter to $b_1$;
end if

L3:
if $m_i$ has a new $h_{i,j}$ then
  $m_i$ generates a random number $o \in [0, 1]$ based on Uniform distribution;
  $z = 0, v = 0$;
  for $n = 1$ to $k$ do
    if $o \le (z + \alpha_{i,j,n})$ then
      $v = n$;
      break;
    end if
    $z = z + \alpha_{i,j,n}$;
  end for
  if $v == 0$ then    // $o \in (z, 1] = (\sum_{k\in\mathcal{B}}\alpha_{i,j,k}, 1]$
    $m_i$ executes $h_{i,j}$ locally;
  else
    $m_i$ offloads $h_{i,j}$ to $b_v$ by sending offloading request and receiving offloading response;
  end if
end if
```

Algorithm 3 Cooperative Computation Offloading Algorithm at b_k Side ($k \in \mathcal{B}-\{1\}$)

```
Initial: send $\mu_k$ and $r_k^{\mathrm{BS}\rightarrow\mathrm{BS1}}$ to $b_1$.
L4:
if any one of $\mu_k$ and $r_k^{\mathrm{BS}\rightarrow\mathrm{BS1}}$ changes then
  $b_k$ sends the new value of the parameter to $b_1$;
end if

L5:
if $b_k$ receives the offloading request of $h_{i,j}$ forwarded from $b_1$ then
  $b_k$ executes $h_{i,j}$;
  $b_k$ return the offloading response of $h_{i,j}$ to $b_1$;
end if
```

(*L5*): if b_k receives the offloading request of $h_{i,j}$ forwarded from b_1, b_k will execute $h_{i,j}$, then return the offloading response of $h_{i,j}$ to b_1;

(*L6*): L6 runs periodically. If b_1 receives any new values of the required parameters in last period, b_1 will run

Algorithm 4 Cooperative Computation Offloading Algorithm at b_1 Side

```
Initial: collect required parameters from MTs, MEC-BSs, and itself, then store their values.
L6 (runs periodically):
if $b_1$ receives any new values of the required parameters then
  $b_1$ runs Algorithm 1 and obtains the optimal solution $\boldsymbol{\alpha}^*$;
  for each $\alpha_{i,j,k}$ in $\boldsymbol{\alpha}^*$ do
    if the value of $\alpha_{i,j,k}$ is updated then
      $b_1$ sends the new value of $\alpha_{i,j,k}$ to $m_i$;
    end if
  end for
end if

L7:
if $b_1$ receives the offloading request of $h_{i,j}$ offloaded from $m_i$ then
  if $h_{i,j}$ goes to $b_1$ then
    $b_1$ executes $h_{i,j}$;
    $b_1$ sends the offloading response of $h_{i,j}$ to $m_i$;
  else
    $b_1$ forwards the offloading request of $h_{i,j}$ to the target MEC-BS;
end if

L8:
if $b_1$ receives the offloading response of $h_{i,j}$ sent from $b_k$ ($\forall k \in \mathcal{B} - \{1\}$) then
  $b_1$ forwards the offloading response to $m_i$;
end if
```

Algorithm 1 and obtains the optimal solution $\boldsymbol{\alpha}^*$. Then, b_1 checks each element in $\boldsymbol{\alpha}^*$. If its value is updated, b_1 sends the new value to the corresponding MT.

(*L7*): if b_1 receives the offloading request of $h_{i,j}$ offloaded from m_i, first, b_1 will check the request. If $h_{i,j}$ goes to b_1, b_1 will execute $h_{i,j}$, then send the offloading response of $h_{i,j}$ to m_i; if $h_{i,j}$ goes to another MEC-BS, b_1 will forward the offloading request of $h_{i,j}$ to the target MEC-BS.

(*L8*): if b_1 receives the offloading response of $h_{i,j}$ sent from b_k ($\forall k \in \mathcal{B} - \{1\}$), b_1 will forward the offloading response to m_i.

V. SIMULATIONS AND PERFORMANCE EVALUATIONS

Extensive simulations are carried out to evaluate the performance of our proposed scheme. The simulation results demonstrate that our scheme can reach the maximum total benefit for all MTs in the scenario. Moreover, our scheme outperformed the reference - the common computation offloading scheme, which is disables the cooperations of MEC-BSs, that is, the further offloading is not supported. Our scheme always gains a higher total benefit than the reference scheme in the scenarios with different criteria. Note that, there is no benefit if computation offloading is disabled, so we didn't compare such scheme with our proposed scheme.

TABLE 1. The parameters used in simulations and their default values.

Name	Value	Name	Value
M	[5, 200]	H	[4, 16]
B	[2, 5]	p_{ij}	[0, 1]
λ_i	[0.1, 1] /s	$c_{i,j}$	[100, 500] MI
q_j	[2.5, 5] MB	s_j	[0.5, 2.5] MB
$\mu_k, k \in \mathcal{B}$	$[0.4, 1.6]\times 10^4$ MIPS	$r_i^{\mathrm{MT}\to\mathrm{BS1}}$	[1.5,3.5] MB/s
$r_i^{\mathrm{BS1}\to\mathrm{MT}}$	[3.5,5.5] MB/s	$r_k^{\mathrm{BS1}\to\mathrm{BS}}$	$[10, 20]$ MB/s
$r_k^{\mathrm{BS}\to\mathrm{BS1}}$	$[10, 20]$ MB/s	ζ_i	$[3, 6] \times 10^{-3}$ J/MI
ω_i	[70,130] mW	θ_i	$[1.1\delta_i, 2\delta_i]$ MIPS
τ	$[0, 1]$		

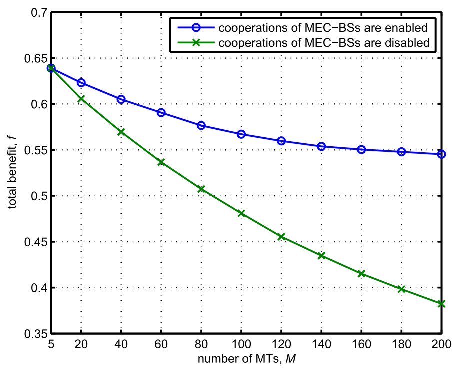

FIGURE 2. f gained by the two schemes with the increase of M.

The settings of the parameters used in simulations are shown in Table 1, where $\delta_i = \sum_{j\in\mathcal{H}}(p_{i,j}\lambda_i c_j)$. Note that, $\theta_i > \delta_i$ ensures constraint (15) holds. Parameter values are configured based on the table unless stated clearly.

In simulations, we evaluate the total benefits (f, as shown in Formula (19)) gained by our proposed scheme and the reference scheme in the scenarios with 6 different criteria, which are **1)** M, the number of MTs; **2)** H, the number of MTs' tasks; **3)** B, the number of MEC-BSs; **4)**, μ_1, the service rate of b_1; **5)** m_k, the service rate of $b_k, k \in \mathcal{B} - \{1\}$; **6)** $r_k^{\mathrm{BS1}\to\mathrm{BS}}$ and $r_k^{\mathrm{BS}\to\mathrm{BS1}}, \forall k \in \mathcal{B} - \{1\}$, data transmission rate between b_1 and other MEC-BSs; **7)** $\theta_i, \forall i \in \mathcal{M}$, the service rate of each MT; **8)** τ, the balance factor of time and energy benefits. The result in each of the 8 scenarios is obtained from 50 repeated simulations using different random seeds.

A. DIFFERENT NUMBERS OF MTs

As shown in Fig. 2, we measured the total benefits gained via the two schemes with different numbers of MTs. In simulations, we increase M from 5 to 200, while other settings are listed in Table 1, except fixed values $H = 10$, $B = 3$, $\theta_i = 1.5\psi_i$ MIPS, $\mu_1 = 1 \times 10^4$ MIPS, $\mu_k = 1 \times 10^4$ MIPS, $\forall k \in \mathcal{B} - \{1\}$, $\tau = 0.5$.

M influences the sum of the tasks generated by the MTs, further, it dominates the total system load. When M is high,

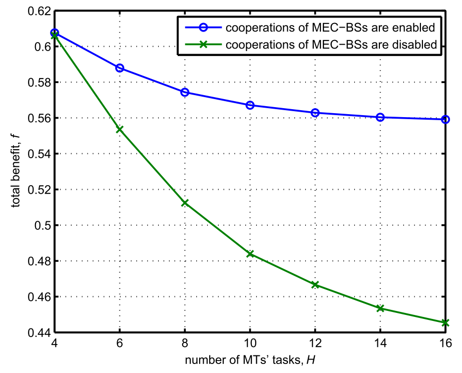

FIGURE 3. *f* **gained by the two schemes with the increase of** ***H*****.**

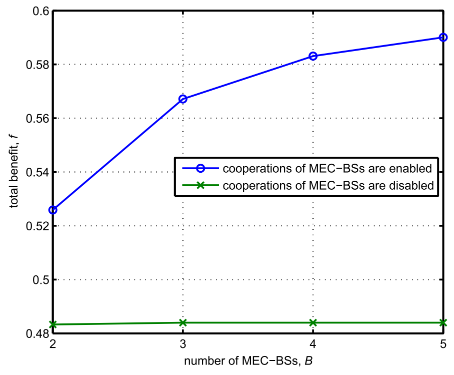

FIGURE 4. *f* **gained by the two schemes with the increase of** ***B*****.**

the computation required by the tasks can not be fully satisfied by MEC-BSs, so the total benefits gained will decrease. Therefore, in Fig. 2, both of the curves of the two schemes drop as the increase of M.

Our scheme always outperforms the reference scheme. When $M = 5$, the total benefits gained by the two schemes are 0.6390 and 0.6389, respectively; when $M = 200$, the benefits are 0.5452 and 0.3824, respectively. The gap between the curves of the two schemes becomes larger as M increases, because when M is low, b_1 itself can provide computation offloading with a high quality, whereas, when M is high, in our scheme, the extra tasks are further offloaded to other MEC-BSs, so the system performance deterioration of our scheme is less than that of the reference scheme with further offloading disabled.

B. DIFFERENT NUMBERS OF MTs' TASKS

As shown in Fig. 3, we measured the total benefits gained via the two schemes with different numbers of MTs' tasks. In simulations, we increase H from 4 to 16, while other settings are listed in Table 1, except fixed values $M = 100$, $B = 3$, $\theta_i = 1.5\psi_i$ MIPS, $\mu_1 = 1 \times 10^4$ MIPS, $\mu_k = 1 \times 10^4$ MIPS, $\forall k \in \mathcal{B} - \{1\}$, $\tau = 0.5$.

H impacts the scale of the tasks generated by the MTs, further, it determines the total system load. When H is high, the computation resource of the system can not provide adequate offloading service for the tasks, so the total performance will decrease. Thus, as shown in Fig. 3, both of the two schemes' curves drop as H increases.

Our scheme is always superior to the reference scheme. When $H = 4$, the total benefits of the two schemes are 0.6076 and 0.6071, respectively; when $H = 16$, the benefits are 0.5592 and 0.4454, respectively. The gap between the curves of the two schemes becomes wider as the increase of H, because when H is low, the required computation capability can be satisfied by b_1 solely, so the further offloading is less used. Whereas, when H is high, the computation resource of b_1 too limited to ensure the-same-level performance. In this case, further offloading is used frequently to alleviate the system load of b_1 in our scheme, but in the reference scheme, further offloading is disabled.

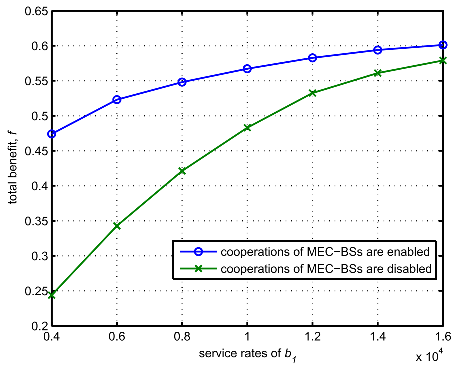

FIGURE 5. *f* **gained by the two schemes with the increase of** μ_1**.**

C. DIFFERENT NUMBERS OF MEC-BSs

As shown in Fig. 4, we measured the total benefits gained via the two schemes with different numbers of MEC-BSs. In simulations, we increase B from 2 to 5, while other settings are listed in Table 1, except fixed values $M = 100$, $H = 10$, $\theta_i = 1.5\psi_i$ MIPS, $\mu_1 = 1 \times 10^4$ MIPS, $\mu_k = 1 \times 10^4$ MIPS, $\forall k \in \mathcal{B} - \{1\}$, $\tau = 0.5$.

B reflexes the number of MEC-BSs assisting to carry out further offloading. The system can accept more tasks and provide better performance through further offloading if B is high, else, the most of the system load will centralize to b_1 only. Thus, the curve of our scheme rises as the increase of B, whereas, the curve of the reference scheme is flat since the further offloading is disabled, so B has no effect on the reference scheme.

D. DIFFERENT SERVICE RATES OF b_1

As shown in Fig. 5, we measured the total benefits gained via the two schemes with different service rates of b_1. In simulations, we increase μ_1 from 0.4×10^4 to 1.6×10^4 MIPS, while other settings are listed in Table 1, except fixed values $M = 100$, $H = 10$, $B = 3$, $\theta_i = 1.5\psi_i$ MIPS, $\mu_k = 1 \times 10^4$ MIPS, $\forall k \in \mathcal{B} - \{1\}$, $\tau = 0.5$.

μ_1 represents the computation capability of b_1. b_1 can accept more tasks or provide shorter time consumption on computation in computation offloading, if it has a high μ_1.

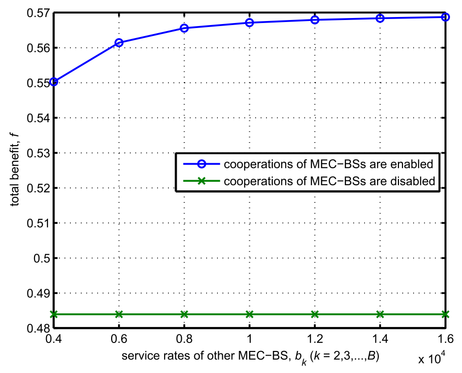

FIGURE 6. ***f* gained by the two schemes with the increase of μ_k, $k = 2, \ldots, B$.**

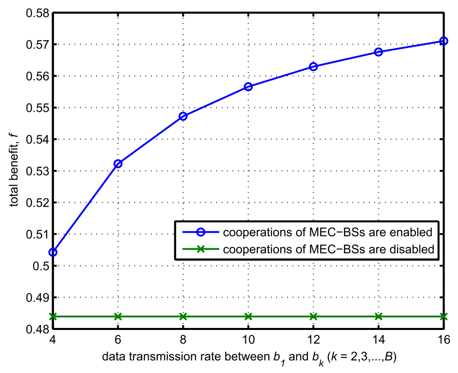

FIGURE 7. ***f* gained by the two schemes with the increase of $r_k^{\mathrm{BS}\to\mathrm{BS1}}$ and $r_k^{\mathrm{BS1}\to\mathrm{BS}}$, $k = 2, \ldots, B$.**

Thus, in Fig. 5, it can be observed that both of the curves of the two schemes rise as the increase of μ_1.

Although, the total benefit gained by our scheme is always higher than that by the reference scheme. At $\mu_1 = 0.4 \times 10^4$ MIPS, the total benefits of the two schemes are 0.4741 and 0.2439, respectively; at $\mu_1 = 1.6 \times 10^4$ MIPS, the two benefits are 0.6012 and 0.579, respectively. It can be seen that the gap between the two curves decreases as the increase of μ_1. The main reason lies in that, in our scheme, b_1 is assisted by other MEC-BSs so the system can gain more benefits through further offloading the tasks to other MEC-BSs especially when μ_1 is low which represents b_1 is lack of computation resources. Whereas, when m_1 is high, the further offloading is less used, since b_1 is powerful enough to execute most of the tasks.

E. DIFFERENT SERVICE RATES OF OTHER MEC-BSs

As shown in Fig. 6, we measured the total benefits gained via the two schemes with different service rates of other MEC-BSs. In simulations, we increase $\mu_k, \forall k \in \mathcal{B}-\{1\}$ from 0.4×10^4 to 1.6×10^4 MIPS, while other settings are listed in Table 1, except fixed values $M = 100$, $H = 10$, $B = 3$, $\theta_i = 1.5\psi_i$ MIPS, $\mu_1 = 1 \times 10^4$ MIPS, $\tau = 0.5$.

The service rates of other MEC-BSs determines the computation capabilities of these MEC-BSs, thus, they impacts the time consumptions of the execution processes in further offloading. As $m_k, \forall k \in \mathcal{B}-\{1\}$ increases, the time consumptions decreases, so the benefits gained by further offloading grows. In Fig. 6, the curve of our scheme rises as $\mu_k, \forall k \in \mathcal{B}-\{1\}$ increases, whereas, the curve of the reference scheme is flat, because our scheme enables further offloading and the reference scheme disables it.

F. DIFFERENT DATA TRANSMISSION RATES BETWEEN b_1 AND OTHER MEC-BSs

As shown in Fig. 7, we measured the total benefits gained via the two schemes with different data transmission rates between b_1 and other MEC-BSs. In simulations, we increase the $r_k^{\mathrm{BS1}\to\mathrm{BS}}$ and $r_k^{\mathrm{BS}\to\mathrm{BS1}}$, $\forall k \in \mathcal{B}-\{1\}$ from 4 to 16 MB/s, while other settings are listed in Table 1, except fixed values $M = 100$, $H = 10$, $B = 3$, $\theta_i = 1.5\psi_i$ MIPS, $\mu_1 = 1 \times 10^4$ MIPS, $\mu_k = 1 \times 10^4$ MIPS, $\forall k \in \mathcal{B}-\{1\}$, $\tau = 0.5$.

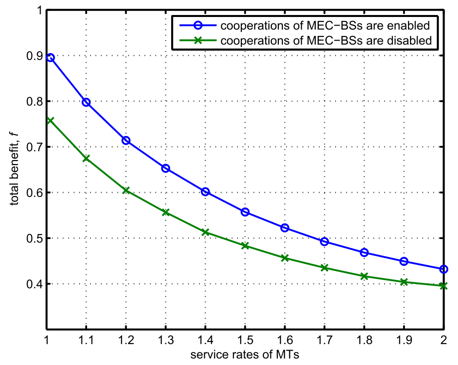

FIGURE 8. ***f* gained by the two schemes with the increase of θ_i, $i \in \mathcal{M}$.**

The data transmission rates between b_1 and other MEC-BSs impact the time consumptions of transmission processes in further offloading. As the transmission rates increase, the time consumptions decrease, so the benefits gained by further offloading grows. As shown Fig. 7, the curve of our scheme rises as $r_k^{\mathrm{BS1}\to\mathrm{BS}}$ and $r_k^{\mathrm{BS}\to\mathrm{BS1}}$, $\forall k \in \mathcal{B}-\{1\}$ increase, whereas, the curve of the reference scheme is unchanged since further offloading is disabled.

G. DIFFERENT SERVICE RATES OF MTs

As shown in Fig. 8, we measured the total benefits gained via the two schemes with different service rates of MTs. In simulations, we increase $theta_i$, $\forall i \in \mathcal{M}$ from $1.01\delta_i$ to $2\delta_i$, while other settings are listed in Table 1, except fixed values $M = 100$, $H = 10$, $B = 3$, $\mu_1 = 1 \times 10^4$ MIPS, $\mu_k = 1 \times 10^4$ MIPS, $\forall k \in \mathcal{B}-\{1\}$, $\tau = 0.5$.

θ_i represents the computation capability of m_i. m_i will need less computation offloading service if it has a high θ_i, so its tasks can be efficiently executed at m_i, otherwise, m_i will require the service more frequently. Generally, both of the curves of the two schemes drops as the increase of θ_i, $\forall i \in \mathcal{M}$, because as θ_i grows, m_i becomes more and more powerful to bear the fast execution of its tasks.

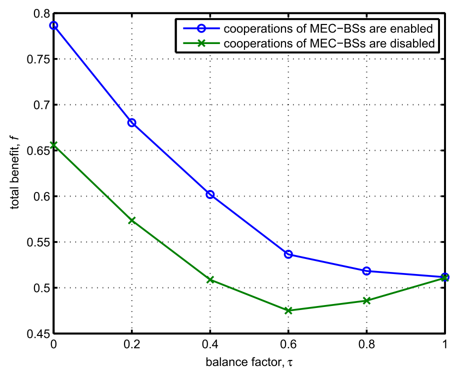

FIGURE 9. ***f* gained by the two schemes with the increase of τ.**

However, for those tasks which are still offloaded to MEC-BSs, our scheme serves better (0.8954 vs. 0.7572 at $\theta_i = 1.01\delta_i, \forall i \in \mathcal{M}$, 0.4319 vs. 0.3954 at $\theta_i = 2\delta_i, \forall i \in \mathcal{M}$) than the reference scheme since b_1 is assisted by multiple MEC-BSs in our scheme. The gap between the 2 scheme narrows as the increase of $\theta_i, \forall i \in \mathcal{M}$, because more tasks are executed locally when $\theta_i, \forall i \in \mathcal{M}$ is high, so the superiority of our scheme drops as the scale of the objects in optimizations shrinks.

H. DIFFERENT VALUES OF BALANCE FACTOR

As shown in Fig. 9, we measured the total benefits gained via the two schemes with different values of balance factor. In simulations, we increase τ from 0 to 1, while other settings are listed in Table 1, except fixed values $M = 100$, $H = 10$, $B = 3$, $\theta_i = 1.5\psi_i$ MIPS, $\mu_1 = 1 \times 10^4$ MIPS, $\mu_k = 1 \times 10^4$ MIPS, $\forall k \in \mathcal{B} - \{1\}$.

The balance factor τ trades off the weight of benefit between time consumption and energy consumption. In Fig. 9, as τ increases, the curve of our scheme drops nonlinearly with a decreasing scale, whereas, the curve of the reference scheme first drops from 0 to 0.6 and then rises from 0.6 to 1. It means the two schemes can gain more benefits when time consumption is considered only ($\tau = 0$) or with a higher weight. Conversely, energy consumption can not be improved by further offloading, so the benefit gain by energy consumption improvement ($\tau = 1$) is lower than the former. Around $\tau = 0.6$, the curve of the reference scheme gets the minimum because $\tau = 0.6$ almost reaches the 1:1 balance of time and energy consumptions improvements.

The total benefit gained by our scheme is always higher (e.g., 0.7865 vs. 0.6558 at $\tau = 0$) than that by the reference scheme except the case that $\tau = 1$. When $\tau = 1$, the total benefits via the two schemes are both 0.5108, since further offloading provided by our scheme has no effect on energy consumption improvement.

VI. CONCLUSION

A novel computation offloading scheme based on the cooperations of multiple MEC-enabled base stations is proposed in this paper, in order to reduce the time consumptions and energy consumptions of the MTs covered by the MEC-BS. In our scheme, the MEC-BS can schedule extra tasks to other MEC-BSs connected to it, so that to alleviate the load on the MEC-BS and increase the capacity of the total computation resources provided to MTs. A cooperative computation offloading algorithm is proposed to solve the optimization problem, which maximizes the total benefit of time and energy consumptions gained by all MTs. A balance factor is used to adjust the bias between time and energy consumptions. The algorithm consists of multiple loops that runs separately and in parallel. Extensive simulations are carried out to evaluate the performance of our scheme, and compare our scheme with a reference scheme which disables the cooperations of MEC-BSs. Our scheme can largely improve the system performance, and the superiority of our scheme is demonstrated in all 8 scenarios with different criteria.

APPENDIX A
PROOF OF THE CONVEXITY OF FUNCTION f_1

Let $\eta_i = \sum_{j\in\mathcal{H}} \left((\sum_{k\in\mathcal{B}} \alpha_{i,j,k}) p_{i,j}\lambda_i c_j \right)$ and $\rho_i = \sum_{j\in\mathcal{H}} (p_{i,j}\lambda_i c_j)$, then function f_1 can be rewritten as

$$f_1 = \sum_{i\in\mathcal{M}} \left(\frac{\rho_i - \eta_i}{\theta_i - \rho_i + \eta_i} \right) \tag{22}$$

Function (22) is a decreasing function on each η_i, $i \in \mathcal{M}$, and it is twice differentiable. We have its Hessian matrix, $\mathbf{X}$, as shown at the bottom of this page.

The determinant of $\mathbf{X}$ is $|\mathbf{X}| = \prod_{i\in\mathcal{M}} \frac{2\theta_i}{(\theta_i-\rho_i+\eta_i)^3} \cdot \frac{2\theta_i}{(\theta_i-\rho_i+\eta_i)^3} > 0, \forall i \in \mathcal{M}$ since $\theta_i > 0$ and $\theta_i - \rho_i + \eta_i \geq 0$ (see Formula (18)). Thus, $|\mathbf{X}| \geq 0$.

Based on above analysis, $\mathbf{X}$ is a M-order symmetric matrix, and each sequential principal minor determinant of $\mathbf{X}$ is greater than or equal to 0 ($|D_k(\mathbf{X})| = \prod_{i=1}^{k} \frac{2\theta_i}{(\theta_i-\rho_i+\eta_i)^3}$, $k = 1, 2, \ldots, M$). Thus, $\mathbf{X}$ is positive semi-definite, function (22) decreases as the increase of each η_i, $i \in \mathcal{M}$, and all $\eta_i, \forall i \in \mathcal{M}$ are linear functions with respect to $\boldsymbol{\alpha}$, so the function f_1 is convex [19].

$$\mathbf{X} = \begin{pmatrix} \frac{2\theta_1}{(\theta_1-\rho_1+\eta_1)^3} & 0 & \cdots & 0 \\ 0 & \frac{2\theta_2}{(\theta_2-\rho_2+\eta_2)^3} & \cdots & 0 \\ \cdots & \cdots & \cdots & \cdots \\ 0 & 0 & \cdots & \frac{2\theta_M}{(\theta_M-\rho_M+\eta_M)^3} \end{pmatrix}$$

APPENDIX B
PROOF OF THE CONVEXITY OF FUNCTION f_2

Let $\eta_k = \sum_{i\in\mathcal{M}}\sum_{j\in\mathcal{H}}(\alpha_{i,j,k}p_{i,j}\lambda_i c_j)$, then function f_2 can be rewritten as

$$f_2 = \sum_{k\in\mathcal{B}}\left(\frac{\eta_k}{\mu_k - \eta_k}\right) \tag{23}$$

Function (23) is a increasing function on η_k, and it is twice differentiable. We have its Hessian matrix

$$\mathbf{X} = \begin{pmatrix} \frac{2\mu_1}{(\mu_1-\eta_1)^3} & 0 & \cdots & 0 \\ 0 & \frac{2\mu_2}{(\mu_2-\eta_2)^3} & \cdots & 0 \\ \cdots & \cdots & \cdots & \cdots \\ 0 & 0 & \cdots & \frac{2\mu_B}{(\mu_B-\eta_B)^3} \end{pmatrix}$$

Obviously, the determinant of $\mathbf{X}$ is $|\mathbf{X}| = \prod_{k\in\mathcal{B}}\frac{2\mu_k}{(\mu_k-\eta_k)^3} > 0$ since $\mu > 0$ and $\mu - \eta \geq 0$ (See constraint (16)).

Based on above analysis, $\mathbf{X}$ is positive definite, function (23) increases as the increase of η_k, and η_k is a linear functions with respect to $\boldsymbol{\alpha}$, so the function f_2 is convex [19].

REFERENCES

[1] N. Fernando, S. W. Loke, and W. Rahayu, "Mobile cloud computing: A survey," *Future Generat. Comput. Syst.*, vol. 29, no. 1, pp. 84–106, 2013.

[2] Y. C. Hu, M. Patel, D. Sabella, N. Sprecher, and V. Young, "Mobile Edge Computing—A key technology towards 5G," Eur. Telecommun. Standards Inst., Sophia Antipolis, France, ETSI White Paper 11, 2015, pp. 1–16.

[3] M. T. Beck, M. Werner, S. Feld, and T. Schimper, "Mobile edge computing: A taxonomy," in *Proc. 6th Int. Conf. Adv. Future Internet*, 2014, pp. 1–7.

[4] A. Ahmed and E. Ahmed, "A survey on mobile edge computing," in *Proc. IEEE 10th Int. Conf. Intell. Syst. Control (ISCO)*, Jan. 2016, pp. 1–8.

[5] Y. Mao, C. You, J. Zhang, K. Huang, and K. B. Letaief, "A survey on mobile edge computing: The communication perspective," *IEEE Commun. Surveys Tuts.*, vol. 19, no. 4, pp. 2322–2358, 4th Quart., 2017.

[6] L. Velasco *et al.*, "A service-oriented hybrid access network and clouds architecture," *IEEE Commun. Mag.*, vol. 53, no. 4, pp. 159–165, Apr. 2015.

[7] A. Carrega, M. Repetto, P. Gouvas, and A. Zafeiropoulos, "A middleware for mobile edge computing," *IEEE Cloud Comput.*, vol. 4, no. 4, pp. 26–37, Jul./Aug. 2017.

[8] K. Habak, M. Ammar, K. A. Harras, and E. Zegura, "Femto clouds: Leveraging mobile devices to provide cloud service at the edge," in *Proc. IEEE 8th Int. Conf. Cloud Comput.*, Jun. 2015, pp. 9–16.

[9] S. Abdelwahab, B. Hamdaoui, M. Guizani, and T. Znati, "Replisom: Disciplined tiny memory replication for massive IoT devices in LTE edge cloud," *IEEE Internet Things J.*, vol. 3, no. 3, pp. 327–338, Jun. 2016.

[10] D. Sabella, A. Vaillant, P. Kuure, U. Rauschenbach, and F. Giust, "Mobile-edge computing architecture: The role of MEC in the Internet of Things," *IEEE Consum. Electron. Mag.*, vol. 5, no. 4, pp. 84–91, Oct. 2016.

[11] 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.

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

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

[14] 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.

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

[16] X. Tao, K. Ota, M. Dong, H. Qi, and K. Li, "Performance guaranteed computation offloading for mobile-edge cloud computing," *IEEE Wireless Commun. Lett.*, vol. 6, no. 6, pp. 774–777, Dec. 2017.

[17] S. Sardellitti, G. Scutari, and S. Barbarossa, "Joint optimization of radio and computational resources for multicell mobile-edge computing," *IEEE Trans. Signal Inf. Process. Over Netw.*, vol. 1, no. 2, pp. 89–103, Jun. 2015.

[18] D. Gross, *Fundamentals of Queueing Theory*. New York, NY, USA: Wiley, 2008.

[19] S. P. Boyd and L. Vandenberghe, *Convex Optimization*. Cambridge, U.K.: Cambridge Univ. Press, 2004.

WENHAO FAN received the B.E. and Ph.D. degrees from the Beijing University of Posts and Telecommunications (BUPT), Beijing, China, in 2008 and 2013, respectively. He is currently an Assistant Professor with the School of Electronic Engineering, BUPT.

His main research interests include mobile cloud computing, parallel computing and transmission, information security for mobile smartphones, and software engineering for mobile Internet.

YUAN'AN LIU received the B.E., M.Eng., and Ph.D. degrees in electrical engineering from the University of Electronic Science and Technology, Chengdu, China, in 1984, 1989, and 1992, respectively. He is currently a Professor with the School of Electronic Engineering, Beijing University of Posts and Telecommunications, Beijing, China.

His main research interests include pervasive computing, wireless communications, and electromagnetic compatibility.

Dr. Liu is a fellow of the Institution of Engineering and Technology, U.K, and a Senior Member of the Electronic Institute of China. He is the Vice Chairman of the Electromagnetic Environment and Safety of the China Communication Standards Association and the Vice Director of the Wireless and Mobile Communication Committee, Communication Institute of China.

BIHUA TANG received the B.E. degree from Sichuan University, Chengdu, China, in 1984, and the M.Eng. degree from the University of Electronic Science and Technology, Chengdu, in 1989. She is currently a Professor with the School of Electronic Engineering, Beijing University of Posts and Telecommunications, Beijing, China.

Her main research interests include mobile computing, wireless sensor networks, and wireless networks. She has authored or co-authored over 50 papers.

FAN WU received the B.E. degree from the University of Electronic Science and Technology of China, Chengdu, China, in 2004, and the Ph.D. degree from the Beijing University of Posts and Telecommunications (BUPT), Beijing, China, in 2009. She is currently an Associate Professor with the School of Electronic Engineering, BUPT.

Her main research interests include wireless sensor networks, mobile computing, and hardware engineering for sensors.

ZHONGBAO WANG received the Ph.D. degree in communication and information systems from Dalian Maritime University (DLMU), China, in 2012. He is currently an Associate Professor with the School of Information Science and Technology, DLMU.

His current research interests include wireless communications and microwave technology. He has authored or co-authored over 50 papers in journals and conferences.