Joint Multi-User Computation Offloading and Data Caching for Hybrid Mobile Cloud/Edge Computing

Xiaolong Yang, Zesong Fei, Jianchao Zheng, Ning Zhang, and Alagan Anpalagan

***Abstract*—In this paper, we investigate a hybrid mobile cloud/edge computing system with coexistence of centralized cloud and mobile edge computing, which enables computation offloading and data caching to improve the performance of users. Computation offloading and data caching decisions are jointly optimized to minimize the total execution delay at the mobile user side, while satisfying the constrains in terms of the maximum tolerable energy consumption of each user, the computation capability of each MEC server, and the cache capacity of each access point (AP). The formulated problem is non-convex and challenging because of the highly coupled decision variables. To address such an untractable problem, we first transform the original problem into an equivalent convex one by McCormick envelopes and introducing auxiliary variables. To the end, we propose a distributed algorithm based on the alternating direction method of multipliers (ADMM), which can achieve near optimal computation offloading and data caching decisions. The proposed algorithm has lower computational complexity compared to the centralized algorithm. Simulation results are presented to verify that the proposed algorithm can effectively reduce computing delay for end users while ensuring the performance of each user.**

***Index Terms*—Hybrid mobile cloud/edge computing, multi-user computation offloading, data caching, McCormick envelopes, ADMM.**

I. Introduction

DRIVEN by the roaring Internet of Things (IoT), the explosive growth of various mobile applications such as remote healthcare systems, surveillance and security monitoring systems, and assisted driving call for stringent requirements on ultra-low latency and high reliability [1], [2]. As a result, the volume of data traffic and energy consumption used to compute and deliver the data increase astronomically, imposing a heavy processing burden to the IoT devices. Although offloading the computing tasks to the cloud servers is a promising approach to address this challenge, it may experience higher energy consumption and longer latency in networks. By deploying the mobile edge computing (MEC) servers at the access points (APs), computation-intensive tasks can be offloaded to the MEC servers instead of the remote cloud servers. MEC serves as a potential solution to supplement cloud computing has brought about great concern [3], [4]. However, MEC ignored the huge computation resources in the cloud servers. Therefore, making full use of powerful resources at both cloud and edge in the hybrid cloud/edge computation systems is particularly necessary and important.

At the same time, caching also shows attractive advantages in dealing with the proliferation of mobile data traffic [5]–[10]. Moreover, considerable research efforts have been dedicated to investigating the advantages of joint design of computation offloading and caching in MEC systems, which alleviates the load of the backhaul links and increases the computation capacity provided to users [11]–[15]. Particularly, authors in [11], [12] investigated joint design of computation offloading and caching strategy in a single-cell network. However, these works seldom considered the effects of users' execution strategy in multi-user computation offloading systems, and the relationship between computation offloading and the related input data. Although [13] proposed an integrated framework to dynamically orchestrate the networking, caching and computing resources, this work only focused on the mobile scenarios, and it was difficult to cache the associated input computation data in a fixed place. Meanwhile, [14] studied the joint content caching and computing in software defined networks, yet without considering the computation offloading and doing the corresponding operations at the computation inputting data. Furthermore, their proposed scheme was not applicable to the MEC networks. Then, the authors in [15] proposed an optimal offloading scheme with caching computation results in MEC networks. However, the authors overlooked the huge computation and caching resources in the centralized cloud computing center.

The existing research on the joint computation offloading and caching scheme in MEC networks just considered the computation offloading between users and APs, and neglected the

Manuscript received January 7, 2019; revised April 18, 2019 and July 1, 2019; accepted September 8, 2019. Date of publication September 19, 2019; date of current version November 12, 2019. This work was supported in part by the National Natural Science Foundation of China under Grants 61421001, 61871032, and 61801505, in part by China National S&T Major Projects 2017ZX03001017 and MCM20170101, in part by 111 Project of China under Grant B14010, in part by Beijing Natural Science Foundation under Grants 4152047 and L182038, in part by the Jiangsu Provincial Nature Science Foundation of China under Grant BK20170755, and in part by the National Postdoctoral Program for Innovative Talents of China under Grant BX201700109. The review of this article was coordinated by Prof. J.-M. Chung. *(Corresponding authors: Zesong Fei; Jianchao Zheng.)*

X. Yang and Z. Fei are with the School of Information and Electronics, Beijing Institute of Technology, Beijing 100081, China (e-mail: 3120160381@bit.edu.cn; feizesong@bit.edu.cn).

J. Zheng is with the National Innovation Institute of Defense Technology Academy of Military Sciences PLA, Beijing 100010, China, and also with the College of Communications Engineering, Army Engineering University of PLA, Nanjing 210007, China (e-mail: longxingren.zjc.s@163.com).

N. Zhang is with the Department of Computing Sciences, Texas A&M University-Corpus Christi, Corpus Christi, TX 78412 USA (e-mail: ning.zhang@tamucc.edu).

A. Anpalagan is with the Department of Electrical and Computer Engineering, Ryerson University, Toronto, ON M5B 2K3, Canada (e-mail: alagan@ee.ryerson.ca).

Digital Object Identifier 10.1109/TVT.2019.2942334

huge computation and caching resources in the cloud servers as well as users' individual execution strategy, which highly degrades users' performance. With the number of users and applications increasing, the computing and caching capacity's bottlenecks of MEC servers have hindered the way to improving users' performance. Meanwhile, although offloading the computing tasks to the cloud servers leads to long execution time for users, the cloud servers can allocate rich computing and caching resources to process users' tasks. MEC with caching and offloading to cloud servers is complementary. Moreover, to reduce the total execution delay for the users within the hybrid mobile cloud/edge computation systems, it is necessary to consider the delay caused by uplink/downlink transmission, which greatly impacts the design of the computation offloading and data caching strategy.

Therefore, in this paper, we investigate the joint optimization of computation offloading and caching decisions in a hybrid mobile cloud/edge computation system for minimizing the sum computing latency of all the users subject to user's battery, each AP' storage and computing capacity constraints. Since the offloading and caching decisions variables are binary values, the formulated joint optimization problem is a discrete and bilinear nonconvex optimization problem, which is intractable to address. Consequently, we decouple the bilinear discrete variables using the method of McCormick envelopes and introducing auxiliary variables. Then the formulated problem is converted into a linear optimization problem by relaxing the discrete variables into continuous value. Eventually, a low computation complexity algorithm based on alternating direction method of multipliers (ADMM) is applied, which is used to provide near optimal solutions.

The main contributions of this paper are summarized as follows:

- The joint optimization of computation offloading with data caching strategy is studied in a hybrid mobile cloud/edge computation system by considering the limited computing and caching resources of MEC servers, and the users' finite battery capacity. The joint optimization problem in hybrid mobile cloud/edge computation systems to minimize the total execution delay of all the users has not been reported in the published literature.
- In order to decrease the unpredictable network latency and utilize network bandwidth fully, the associated computation inputting data is partitioned into two parts, including the data collected by users and the related databases/libraries, and then we consider them separately. To our knowledge, it is the first attempt to investigate how to cache the data from the related databases/libraries, how to offload the users' collecting data, and where to execute the computing tasks for hybrid mobile cloud/edge computation systems.
- The studied joint optimization problem is a bilinear discrete program problem. Using the McCormick envelopes and binary variables relaxation, the original nonconvex optimization problem is transformed into a linear program problem. Furthermore, we discuss the convexity of the transformed problem. To overcome the drawback of the centralized algorithm, a distributed computation offloading with data caching scheme based on ADMM is proposed.

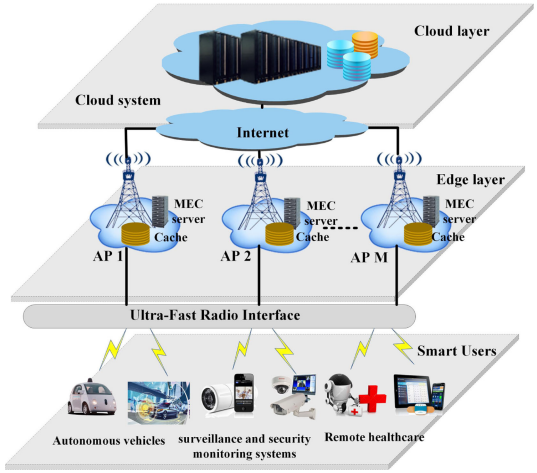


Fig. 1. The hybrid mobile cloud/edge computation systems.

The rest of this paper is organized as follows. In Section II, the system model and problem formulation are described. Then, Section III introduces the joint computation offloading and caching scheme. Simulation results are given in Section IV. Finally, the conclusion of the paper is provided in Section V.

II. System Model and Problem Formulation

In this section, we first present a system model for hybrid mobile cloud/edge computation systems, including network, communication, and computation, as well as caching models. Then the problem of jointly optimizing the offloading and caching decision is formulated in details.

A. Network Model

As shown in Fig. 1, we consider a scenario consisting of M APs, I mobile users with single antenna, and remote cloud servers. Let $\mathcal{M} = \{1, \ldots, M\}$ and $\mathcal{I} = \{1, \ldots, I\}$ denote the set of APs and mobile users, respectively. Each mobile user is connected to its associated AP through wireless links, while APs and the cloud servers are connected via the fiber wired links. Each AP $m \in \mathcal{M}$ is equipped with a MEC server and hence can provide computation offloading service to the mobile users under its radio coverage. The cloud servers can be regarded as the computation and data center. The computational capability (CPU cycles/s) of APs and the cloud servers assigned to mobile user i, $i \in \mathcal{I}$ are denoted by f_i^M and f_i^C, respectively [16]. For each mobile user i, $i \in \mathcal{I}$, the locally computational capability (CPU cycles/s) is denoted by f_i^L (CPU cycles/s), which can be different for various users. Each mobile user has only one computation-intensive task to be either computed locally or offloaded for remote processing within one time slot, and each computation task is atomic and unsplit [17].

Due to the different data types, the computation input data is composed of two components, namely, the data collected by mobile users and the database of the computation tasks. Each AP $m \in \mathcal{M}$ has a storage space C_m to store data (e.g. libraries and

TABLE I
MAIN PARAMETERS IN THIS PAPER

Notation	Description
$\mathcal{M}$	The set of APs
$\mathcal{I}$	The set of users
$G_{i,m}$	The channel power gain from user i to AP m
$d_{i,m}$	The distance from user i to AP m
p_i	The uplink transmit power of user i
p_m	The uplink transmit power of AP m
B	The available spectrum bandwidth
$R^U_{i,m}$, $R^D_{i,m}$	The uplink and downlink data rate between user i and AP m
S_i	Input data size of task H_i
W_i	The required CPU cycles to accomplish task H_i
D_i	Input data of task H_i collected by users
U_i	The corresponding database of task H_i
$\mathbf{a}_i$, $\mathbf{b}_i$	Offloading decision vectors of user i
$\mathbf{c}_i$	Caching decision profile of task H_i
r^{MC}	The transmission rate between the cloud servers and APs
E_i^{max}	The total battery capacity of user i
f_i^C, f_i^M, f_i^L	Processing ability of user i of local/MEC/cloud servers processing
T_i^L, $T_{i,m}^M$, $T_{i,m}^C$	Computing delay for user i in local/MEC/cloud servers processing
E_i^L, $E_{i,m}^M$, $E_{i,m}^C$	Energy consumption for user i in local/MEC/cloud servers processing

databases) associated with specific computation tasks. Each AP can push its cached data to the mobile users which request the computation data, while the uncached data have to be delivered to the users via the backhaul link from the cloud servers. Our network model can reflect or be applicable to many practical scenarios, such as autonomous vehicle application, surveillance and security monitoring systems, and the remote healthcare system, where a large number of sensors or cameras have a mount of data to be transmitted to the MEC servers on the APs or the cloud servers for further analysis and computation. Mobile user mobility may make the high quality channel state information unavailable for achieving reliable data communication and improving the offloading performance [18], [19]. However, we do not consider these issues in this work and will discuss them in our future work. The main parameters used in this paper are summarized in Table I.

B. Communication Model

The uplink data rate is first introduced when user offloads the data collected by itself onto AP. Furthermore, we consider that one user only access one AP within a time slot for the data transmission. Let $g_{i,m}$ and $d_{i,m}$ denote the coefficient of the effective channel power gain and the distance from user i to AP m, respectively. In this paper, we consider the scenario that the users move very slowly during the data offloading, so $g_{i,m}$ can be seen as a constant in a time slot and can change over different time slots (i.e., block fading channel). Then, the corresponding channel power gain can be given as

$$G_{i,m} = g_{i,m} d_{i,m}^{-\alpha}, \tag{1}$$

where α is the path loss factor. According to [20], [21], the uplink data rate of a user that chooses to offload its collecting data to the AP via a wireless link can be expressed as

$$R^U_{i,m} = B \log_2 \left(1 + \frac{G_{i,m} p_i}{\sum_{j \in \mathcal{I} \setminus \{i\}} G_{j,m} p_j + \sigma^2}\right), i \in \mathcal{I}, m \in \mathcal{M}, \tag{2}$$

where B, p_i and σ^2 are the available spectrum bandwidth, the uplink transmit power of user i and the noise power, respectively. Note that multiple users share the same spectrum resource in the wireless interference model.

Similarly, since AP may send the caching associated data (e.g., libraries and databases) required for computing the task to the corresponding user, this may reduce the heavy traffic burden at the backhaul link. Denote the downlink transmit power of AP m as p_m, then the achievable transmit rate of user i at the downlink is given by:

$$R^D_{i,m} = B \log_2 \left(1 + \frac{G_{i,m} p_m}{\sum_{n \in \mathcal{M} \setminus \{m\}} G_{i,n} p_n + \sigma^2}\right), i \in \mathcal{I}, m \in \mathcal{M}. \tag{3}$$

Note that since the size of input computational data is much larger than the size of computation results, so the latency of transmitting computation results is neglected in this work.

C. Computation Model

In the computation model, we consider that multiple tasks can be computed sequentially [22]. According to [23], the CPU computing capacity remains constant within one time slot. Thus, we focus on the widely used task model $H_i \triangleq (S_i, W_i)$, where S_i and W_i stand for the size of computation input data and the required CPU cycles to accomplish the task H_i, respectively. In this paper, we consider that the input computation data S_i can be divided into two parts, namely, the data collected by users and the corresponding database of computation task, which are denoted as D_i and U_i, respectively. Taking positioning on autonomous vehicles as an example [24], [25], the data to be dealt with is composed of the sensor data and vehicle-to-infrastructure data. The sensor data is collected from on-board vehicle sensors, which includes the information about the environment and other neighboring vehicles. The vehicle-to-infrastructure data includes weather conditions, traffic flow, etc., which is cached in APs or the cloud servers within one time slot.

In general, a computation task can be executed locally, at AP, or on the more powerful cloud servers. Let $a_{i,m} \in \{0, 1\}$ denote whether data D_i is offloaded to AP m or not, and the corresponding offloading decision of user i is denoted by a series of binary variables, namely, $\mathbf{a}_i = \{a_{i,1}, a_{i,2}, ..., a_{i,M}\}$. Similarly, we define $b_{i,m} \in \{0, 1\}$ to indicate whether the task H_i is processed on the cloud servers or not, and the corresponding decision is expressed as $\mathbf{b}_i = \{b_{i,1}, b_{i,2}, ..., b_{i,M}\}$. Moreover,

since the computation capacity of each MEC server is limited, the following constraint must be held,

$$\sum_{i\in\mathcal{I}} a_{i,m}W_i \le O_m, m \in \mathcal{M}, \tag{4}$$

where O_m denotes the maximum computation capacity of the MEC server m. Correspondingly, each user only adopts one offloading decision to process the computation task. Thus, the offloading decisions of user i are constrained by

$$\sum_{m\in\mathcal{M}} (a_{i,m} + b_{i,m}) \le 1, i \in \mathcal{I}, m \in \mathcal{M}. \tag{5}$$

The constraint in (5) implies that only one offloading decision can be chosen for each computation task H_i.

D. Caching Model

The users may request a particular task, which requires the caching data, such as the data from libraries and databases [26]. For each AP, one can determine whether to cache the data from database, according to the popularity information of the data. The data U_i caching placement decision can be denoted by a binary indicator variable $c_{i,m} \in \{0, 1\}$. Here, $c_{i,m} = 1$ means that data U_i is cached at the MEC server m; otherwise $c_{i,m} = 0$. Therefore, we give $\mathbf{c}_i = \{c_{i,1}, c_{i,2}, ..., c_{i,M}\}$ as the caching decision profile. However, due to the limited storage capability of AP, not all data from the remote database can be cached at the same time. Therefore, the sum size of all the cached data cannot exceed the storage capability C_m of the AP m, which can be expressed as

$$\sum_{i\in\mathcal{I}} (a_{i,m}D_i + c_{i,m}U_i) \le C_m, m \in \mathcal{M}. \tag{6}$$

Moreover, AP can fetch the uncaching associated data from the cloud servers via backhual fiber link, where the cloud servers store all the computation database.

E. Problem Formulation

For each task H_i, it can be computed locally by user i, the AP, or cloud servers. According to different computation and caching strategies, the latency and energy consumption of the task H_i may be different. The processing time of task H_i by computing locally can be expressed by

$$T_i^L = c_{i,m}\frac{U_i}{R_{i,m}^D} + (1 - c_{i,m})\left(\frac{U_i}{R_{i,m}^D} + \frac{U_i}{r^{MC}}\right) + \frac{W_i}{f_i^L} \tag{7a}$$

$$= \frac{U_i}{R_{i,m}^D} + (1 - c_{i,m})\frac{U_i}{r^{MC}} + \frac{W_i}{f_i^L}, \forall i, m, \tag{7b}$$

where $\frac{U_i}{R_{i,m}^D}$ and $\frac{W_i}{f_i^L}$ indicate that the downlink transmission time between user i and AP m, and the computation execution time of task H_i completed locally by user i, respectively. In addition, $\frac{U_i}{r^{MC}}$ is the transmission delay of the associated data transmitting from the cloud servers to the AP m, where r^{MC} is the transmission rate between the cloud servers and AP. We assume that the rate r^{MC} is a constant regardless of the number of APs and users. Then, the energy consumption of task H_i by accomplishing locally can be calculated as

$$E_i^L = \varepsilon(f_i^L)^2 W_i, \forall i, \tag{8}$$

where ε is the energy parameter, which is set as $\varepsilon = 10^{-11}$, according to the work in [27].

In the following, we present the processing time and energy consumption by computing the task i at AP or cloud servers. The execution time of the MEC server for calculating task H_i can be given as

$$T_{i,m}^M = c_{i,m}\frac{D_i}{R_{i,m}^U} + (1 - c_{i,m})\left(\frac{D_i}{R_{i,m}^U} + \frac{U_i}{r^{MC}}\right) + \frac{W_i}{f_i^M} \tag{9a}$$

$$= \frac{D_i}{R_{i,m}^U} + (1 - c_{i,m})\frac{U_i}{r^{MC}} + \frac{W_i}{f_i^M}, \forall i, m, \tag{9b}$$

where $\frac{D_i}{R_{i,m}^U}$ is the uplink transmission time for transmitting the data D_i between the user i and AP m, and $\frac{W_i}{f_i^M}$ is the computing delay for executing H_i at AP m. Also, we can calculate the energy of user i for offloading the data D_i as,

$$E_{i,m}^M = p_i\frac{D_i}{R_{i,m}^U}, \forall i, m. \tag{10}$$

The cloud servers computing time of task H_i can be expressed as follows,

$$T_{i,m}^C = \frac{D_i}{R_{i,m}^U} + \frac{D_i}{r^{MC}} + \frac{W_i}{f_i^C}, \forall i, m, \tag{11}$$

where $\frac{W_i}{f_i^C}$ stands for the latency for processing the task H_i on the cloud server. Moreover, it is easy to observe that energy consumption $E_{i,m}^C$ of user i for the cloud computing is the same as the MEC servers.

In order to reduce the time consumption of all the users while satisfying the limited battery capacity of users, the computation capability constraint of each MEC server, and the cache capacity constraint of each AP, we jointly optimize the data caching and offloading decisions. The corresponding optimization problem can be formulated as follows,

$$\min_{\mathbf{a}_i,\mathbf{b}_i,\mathbf{c}_i} \sum_{m\in\mathcal{M}}\sum_{i\in\mathcal{I}} [(1 - a_{i,m} - b_{i,m})T_i^L + a_{i,m}T_{i,m}^M + b_{i,m}T_{i,m}^C] \tag{12a}$$

$$\text{s.t.} \quad (4), (5), (6),$$

$$(1 - a_{i,m} - b_{i,m})E_i^L \le \rho_i E_i^{\max} \tag{12b}$$

$$a_{i,m}E_{i,m}^M \le \rho_i E_i^{\max} \tag{12c}$$

$$b_{i,m}E_{i,m}^C \le \rho_i E_i^{\max} \tag{12d}$$

$$a_{i,m}, b_{i,m}, c_{i,m} \in \{0, 1\}, \quad \forall i, m, \tag{12e}$$

where the objective function (12a) is to minimize the total computing task delay. The constraint (12b) states the energy consumption of local processing cannot exceed the residual battery capacity of user i, where ρ_i is a factor denoting the weight on the remainding energy consumption relative to the

total battery capacity $E_i^{\max}$ of the user i. The constraints (12c) and (12d) indicate that the energy computation for the MEC server processing and cloud server computing are limited by the battery of user, respectively.

Note that optimization problem defined in (12) is a nonlinear combinational programming problem involving integer and binary variables. It is obvious that the constraints (4)–(6) and (12b)–(12e) are nonlinear and discrete. In addition, due to the coupling between the offloading decision variable and the caching decision variable (i.e., $b_{i,m}c_{i,m}$), the objective function in problem (12) is discrete and nonconvex. Therefore, it is challenging and intractable to find the optimal solutions to this problem within the polynomial time complexity. In the following subsection, we will show that the nonlinear combinational optimization problem can be converted into a linear programming problem by adopting McCormick envelopes and relaxation, and propose a method to solve the transformed problem using a distributed algorithm.

III. Joint Computation Offloading and Caching Scheme

In this section, we first decouple the product relationship $b_{i,m}c_{i,m}$ in problem (12) with the aid of McCormick envelopes. After the decomposition, we reformulate the optimization problem (12) by relaxing the binary variables of the offloading and caching decision, and then propose a distributed algorithm for solving the transformed problem via consensus ADMM. Together, the corresponding convergence results and complexity are provided. Moreover, a method of recovering the binary variables of the offloading and caching decisions from the continuous variables is also presented.

A. Problem Reformulation

Note that the bilinear product $b_{i,m}c_{i,m}$ makes the problem (12) intractable. In the following, by means of defining $z_{i,m} = b_{i,m}(1-c_{i,m})$, the problem (12) can be transformed as,

$$\min_{\mathbf{a}_i,\mathbf{b}_i,\mathbf{c}_i,\mathbf{z}} \sum_{m\in\mathcal{M}}\sum_{i\in\mathcal{I}}\Bigg[\frac{U_i}{R_{i,m}^D}+\frac{W_i}{f_i^L}+\frac{U_i}{r^{MC}} + a_{i,m}\left(\frac{D_i}{R_{i,m}^U}+\frac{W_i}{f_i^M}-\frac{U_i}{R_{i,m}^D}-\frac{W_i}{f_i^L}\right) + b_{i,m}\left(\frac{D_i}{R_{i,m}^U}+\frac{D_i}{r^{MC}}+\frac{W_i}{f_i^C}-\frac{U_i}{R_{i,m}^D} - \frac{W_i}{f_i^L}\right) - c_{i,m}\frac{U_i}{r^{MC}} - z_{i,m}\frac{U_i}{r^{MC}}\Bigg] \quad (13a)$$

$$\text{s.t.} \quad (4),(5),(6),$$

$$(12b)-(12e),$$

$$a_{i,m}, b_{i,m}, c_{i,m} \in \{0,1\}, \quad \forall i,m, \quad (13b)$$

$$z_{i,m} = b_{i,m}(1-c_{i,m}), i\in\mathcal{I}, m\in\mathcal{M}. \quad (13c)$$

To obtain the desired convex relaxations for constraint (13c), we use the McCormick relaxation method [28] to get the convex relaxation constraints, which are obtained by,

$$z_{i,m} \le b_{i,m}, i\in\mathcal{I}, m\in\mathcal{M}, \quad (14a)$$

$$z_{i,m} \le 1-c_{i,m}, i\in\mathcal{I}, m\in\mathcal{M}, \quad (14b)$$

$$z_{i,m} \ge 0, i\in\mathcal{I}, m\in\mathcal{M}, \quad (14c)$$

$$z_{i,m} \ge b_{i,m}-c_{i,m}, i\in\mathcal{I}, m\in\mathcal{M}, \quad (14d)$$

In particular, according to [29] and [30], it is not difficult to verify that the constraint (13c) is equivalent strictly to the constraints (14a)–(14d).

However, even after decoupling the bilinear variables, the problem (13) remains non-convex due to the binary variables $a_{i,m}, b_{i,m}, c_{i,m}, z_{i,m}$. To deal with this issue, the binary variables of $a_{i,m}, b_{i,m}, c_{i,m}, z_{i,m}$ are relaxed to the continuous decision variables as $0\le a_{i,m}\le 1$, $0\le b_{i,m}\le 1$, $0\le c_{i,m}\le 1$, and $0\le z_{i,m}\le 1$. The relaxed variables can be interpreted as the fraction of the associated computing task data which can be offloaded to MEC servers or remote cloud servers, or cached in the MEC servers. Thus, the problem (13) can be rewritten as

$$\min_{\mathbf{a}_i,\mathbf{b}_i,\mathbf{c}_i,\mathbf{z}} \sum_{m\in\mathcal{M}}\sum_{i\in\mathcal{I}}\Bigg[\frac{U_i}{R_{i,m}^D}+\frac{W_i}{f_i^L}+\frac{U_i}{r^{MC}} + a_{i,m}\left(\frac{D_i}{R_{i,m}^U}+\frac{W_i}{f_i^M}-\frac{U_i}{R_{i,m}^D}-\frac{W_i}{f_i^L}\right) + b_{i,m}\left(\frac{D_i}{R_{i,m}^U}+\frac{D_i}{r^{MC}}+\frac{W_i}{f_i^C}-\frac{U_i}{R_{i,m}^D} - \frac{W_i}{f_i^L}\right) - c_{i,m}\frac{U_i}{r^{MC}} - z_{i,m}\frac{U_i}{r^{MC}}\Bigg] \quad (15a)$$

$$\text{s.t.} \quad \sum_{i\in\mathcal{I}} a_{i,m}W_i \le O_m, m\in\mathcal{M}, \quad (15b)$$

$$\sum_{m\in\mathcal{M}}(a_{i,m}+b_{i,m}) \le 1, i\in\mathcal{I}, m\in\mathcal{M}, \quad (15c)$$

$$\sum_{i\in\mathcal{I}}(a_{i,m}D_i + c_{i,m}U_i) \le C_m, m\in\mathcal{M}, \quad (15d)$$

$$(1-a_{i,m}-b_{i,m})E_i^L \le \rho_i E_i^{\max}, i\in\mathcal{I}, m\in\mathcal{M}, \quad (15e)$$

$$a_{i,m}E_{i,m}^M \le \rho_i E_i^{\max}, i\in\mathcal{I}, m\in\mathcal{M}, \quad (15f)$$

$$b_{i,m}E_{i,m}^C \le \rho_i E_i^{\max}, i\in\mathcal{I}, m\in\mathcal{M}, \quad (15g)$$

$$z_{i,m} \le b_{i,m}, i\in\mathcal{I}, m\in\mathcal{M}, \quad (15h)$$

$$z_{i,m} \le 1-c_{i,m}, i\in\mathcal{I}, m\in\mathcal{M}, \quad (15i)$$

$$z_{i,m} \ge 0, i\in\mathcal{I}, m\in\mathcal{M}, \quad (15j)$$

$$z_{i,m} \le b_{i,m}-c_{i,m}, i\in\mathcal{I}, m\in\mathcal{M}, \quad (15k)$$

$$a_{i,m}, b_{i,m}, c_{i,m}, z_{i,m} \in [0,1], \quad \forall i,m. \quad (15l)$$

Then, we will show that problem (15) is convex through Proposition 1.

Proposition 1: If problem (15) is feasible, it is a convex optimization problem with respect to the optimization variables $\mathbf{a}_i, \mathbf{b}_i, \mathbf{c}_i$, and $\mathbf{z}$.

Proof: With the aid of McCormick envelopes and the relaxation of the binary variables, the objective function of problem (15) is a linear combination of variables $\mathbf{a}_i, \mathbf{b}_i, \mathbf{c}_i$, and $\mathbf{z}$. Then, it is easy to find that this objective function is convex. Meanwhile, all the constraints of problem (15) are linear. In this way, Proposition 1 is proved. ■

As shown, problem (15) is a convex problem. Many convex optimization methods can be used to handle this linear programming problem, such as dual decomposition and interior-point methods [31]. However, the centralized algorithm design becomes challenging because global network state information is prohibitively large, which renders impractically heavy computation load to the network. Especially, when the number of APs and users are excessive, the situation will become worse. Instead, it is preferable and practical to achieve the optimal offloading and caching decisions in a distributed way.

B. Problem Solutions via ADMM

To devise a distributed way to solve problem (15), we introduce local copies $\tilde{a}^m_{i,k}$, $\tilde{b}^m_{i,k}$, and $\tilde{z}^m_{i,k}$ of the variables $a_{i,m}$, $b_{i,m}$, and $z_{i,m}$ at MEC server m corresponding to each MEC server $k \in \mathcal{M}$, respectively. With all the local variables $\{\tilde{a}^m_{i,k}, \tilde{b}^m_{i,k}, \tilde{z}^m_{i,k}, c_{i,m}\}_{m\in\mathcal{M}}$, we use $\tilde{\mathbf{a}}^m = \{\tilde{a}^m_{i,k}\}_{i\in\mathcal{I},k\in\mathcal{M}}$, $\tilde{\mathbf{b}}^m = \{\tilde{b}^m_{i,k}\}_{i\in\mathcal{I},k\in\mathcal{M}}$, $\tilde{\mathbf{z}}^m = \{\tilde{z}^m_{i,k}\}_{i\in\mathcal{I},k\in\mathcal{M}}$, $\mathbf{c}_m = \{c_{i,m}\}_{i\in\mathcal{I},m\in\mathcal{M}}$ to denote the vectors of the copies corresponding to the station m, respectively. Thus, the feasible set of local variables with respect to AP m is given as follows,

$$\chi_m = \left\{ \begin{array}{c|c} & \sum_{i\in\mathcal{I}} \tilde{a}^m_{i,k} W_i \le O_k, \forall k \\ & \sum_{k\in\mathcal{M}} (\tilde{a}^m_{i,k} + \tilde{b}^m_{i,k}) \le 1, \forall i \\ \tilde{\mathbf{a}}^m & \sum_{i\in\mathcal{I}} (\tilde{a}^m_{i,k} D_i + c_{i,k} U_i) \le C_k, \forall k \\ & (1 - \tilde{a}^m_{i,k} - \tilde{b}^m_{i,k}) E^L_i \le \rho_i E^{\max}_i, \quad \forall i,k \\ \tilde{\mathbf{b}}^m & \tilde{a}^m_{i,k} E^M_{i,k} \le \rho_i E^{\max}_i, \quad \forall i,k \\ & \tilde{b}^m_{i,k} E^C_{i,k} \le \rho_i E^{\max}_i, \quad \forall i,k \\ \tilde{\mathbf{z}}^m & \tilde{z}^m_{i,k} \le \tilde{b}^m_{i,k}, \quad \forall i,k \\ & \tilde{z}^m_{i,k} \le 1 - c_{i,m}, \quad \forall i,k \\ \mathbf{c}_m & \tilde{z}^m_{i,k} \ge 0, \quad \forall i,k \\ & \tilde{z}^m_{i,k} \le \tilde{b}^m_{i,k} - c_{i,m}, \quad \forall i,k \end{array} \right\}. \tag{16}$$

Moreover, the objective function (15a) can be expressed as $\min \sum_{m\in\mathcal{M}} \sum_{i\in\mathcal{I}} f(\mathbf{a}_i, \mathbf{b}_i, \mathbf{c}_i, \mathbf{z})$. For each station m, the associated local function can be formulated as

$$g_m = \begin{cases} \sum_{i\in\mathcal{I}} f(\mathbf{e}), \mathbf{e} \in \chi_m; \\ 0, \qquad\qquad \text{otherwise}, \end{cases} \tag{17}$$

where $\mathbf{e}$ denotes $\{\tilde{\mathbf{a}}^m, \tilde{\mathbf{b}}^m, \tilde{\mathbf{z}}^m, \mathbf{c}_m\}_{m\in\mathcal{M}}$.

From the above analysis, the corresponding global consensus problem [14] of the problem (15) through introducing consensus variables can be shown as,

$$\begin{aligned} \min \ & \sum_{m\in\mathcal{M}} g_m(\mathbf{e}) \\ \text{s.t.} \ & \tilde{a}^m_{i,k} = a_{i,m}, \tilde{b}^m_{i,k} = b_{i,m}, \tilde{z}^m_{i,k} = z_{i,m}, \quad \forall i,m,k. \end{aligned} \tag{18}$$

Now, we can use the Alternating Direction Method of Multipliers (ADMM) algorithm [32] to decompose problem (18) effectively. Moreover, it is easy to observe that the objective function of (18) is separable with respect to each station, and the consensus constraints in problem (18) hold that the local variables should be equal for each AP. Next, according to [33], the augmented Lagrangian for problem (18) can be expressed as,

$$\begin{aligned} & L_\rho(\mathbf{e}, \mathbf{s}, \{\boldsymbol{\sigma}^m, \boldsymbol{\delta}^m, \boldsymbol{\eta}^m\}) \\ & = \sum_{m\in\mathcal{M}} g_m(\mathbf{e}) + \sum_{m\in\mathcal{M}} \sum_{\substack{k\in\mathcal{M} \\ i\in\mathcal{I}}} \sigma^m_{i,k} (\tilde{a}^m_{i,k} - a_{i,k}) \\ & \quad + \sum_{m\in\mathcal{M}} \sum_{\substack{k\in\mathcal{M} \\ i\in\mathcal{I}}} \delta^m_{i,k} (\tilde{b}^m_{i,k} - b_{i,k}) + \sum_{m\in\mathcal{M}} \sum_{\substack{k\in\mathcal{M} \\ i\in\mathcal{I}}} \eta^m_{i,k} (\tilde{z}^m_{i,k} - z_{i,k}) \\ & \quad + \frac{\rho}{2} \sum_{m\in\mathcal{M}} \sum_{\substack{k\in\mathcal{M} \\ i\in\mathcal{I}}} \|\tilde{a}^m_{i,k} - a_{i,k}\|^2 + \frac{\rho}{2} \sum_{m\in\mathcal{M}} \sum_{\substack{k\in\mathcal{M} \\ i\in\mathcal{I}}} \|\tilde{b}^m_{i,k} - b_{i,k}\|^2 \\ & \quad + \frac{\rho}{2} \sum_{m\in\mathcal{M}} \sum_{\substack{k\in\mathcal{M} \\ i\in\mathcal{I}}} \|\tilde{z}^m_{i,k} - z_{i,k}\|^2, \end{aligned} \tag{19}$$

where $\boldsymbol{\sigma}^m = \{\sigma^m_{i,k}\}, \boldsymbol{\delta}^m = \{\delta^m_{i,k}\}, \boldsymbol{\eta}^m = \{\eta^m_{i,k}\}$ are the associated Lagrangian multipliers with respect to problem (18) and ρ is a positive penalty parameter, which is a enormously important value for effecting the property of the iterative method. Moreover, $\mathbf{s} = \{\mathbf{a}_i, \mathbf{b}_i, \mathbf{z}\}$. In order to optimize the target variables conveniently, the linear and quadratic terms of the equality constraints are combined through scaling the Lagrange multipliers with respect to (18) and the augmented Lagrangian function (19) can be rewritten as follows,

$$\begin{aligned} L_\rho(\mathbf{e}, \mathbf{s}, \{\boldsymbol{u}^m, \boldsymbol{v}^m, \boldsymbol{w}^m\}) & = \sum_{m\in\mathcal{M}} g_m(\mathbf{e}) \\ & + \frac{\rho}{2} \sum_{m\in\mathcal{M}} \sum_{\substack{k\in\mathcal{M} \\ i\in\mathcal{I}}} \|\tilde{a}^m_{i,k} - a_{i,k} + u^m_{i,k}\|^2 \\ & + \frac{\rho}{2} \sum_{m\in\mathcal{M}} \sum_{\substack{k\in\mathcal{M} \\ i\in\mathcal{I}}} \|\tilde{b}^m_{i,k} - b_{i,k} + v^m_{i,k}\|^2 \\ & + \frac{\rho}{2} \sum_{m\in\mathcal{M}} \sum_{\substack{k\in\mathcal{M} \\ i\in\mathcal{I}}} \|\tilde{z}^m_{i,k} - z_{i,k} + w^m_{i,k}\|^2, \end{aligned} \tag{20}$$

where $u^m_{i,k} = \frac{\sigma^m_{i,k}}{\rho}$, $v^m_{i,k} = \frac{\delta^m_{i,k}}{\rho}$, and $w^m_{i,k} = \frac{\eta^m_{i,k}}{\rho}$. Moreover, $\boldsymbol{u}^m = \{u^m_{i,k}\}, \boldsymbol{v}^m = \{v^m_{i,k}\}$, and $\boldsymbol{w}^m = \{w^m_{i,k}\}$ are the new dual variables after scaling the corresponding Lagrange multipliers in (19). It is easy to obtain the dual problem of (19), i.e.,

$$\max_{\boldsymbol{\theta}} d(\boldsymbol{\theta}) \tag{21}$$

in which the dual function $d(\boldsymbol{\theta})$ is expressed by

$$d(\boldsymbol{\theta}) = \min_{\mathbf{e},\mathbf{s}} L_\rho(\mathbf{e}, \mathbf{s}, \boldsymbol{\theta}) \tag{22}$$

where $\boldsymbol{\theta} = \{\boldsymbol{u}^m, \boldsymbol{v}^m, \boldsymbol{w}^m\}$.

The ADMM algorithm is applied to solving the dual problem (21) by iteratively updating $\mathbf{e}, \mathbf{s}$, and $\boldsymbol{\theta}$. The values $\{\mathbf{e}^{(t)}, \mathbf{s}^{(t)}, \boldsymbol{\theta}^{(t)}\}$ stand for the optimization values at the t-th iteration. Then, we sequentially update of the dual variable $\boldsymbol{\theta}$ and the primal variables $\{\mathbf{e}, \mathbf{s}\}$ in the $(t+1)$-th iteration, which is presented as follows:

Step 1. Local variables updating: In this step, given $\{\mathbf{s}^{(t)}, \boldsymbol{\theta}^{(t)}\}$, we first minimize L_ρ with respect to local variables $\mathbf{e}$, i.e.,

$$\mathbf{e}^{(t+1)} = \arg\min_{\mathbf{e}} L_\rho(\mathbf{e}, \mathbf{s}^{(t)}, \boldsymbol{\theta}^{(t)}). \tag{23}$$

The problem (23) can be decomposed into N parallel subproblems, and each subproblem can be solved separately at each AP according to the following formulation:

$$\begin{aligned}\mathbf{e}^{(t+1)} = & \arg\min_{\{\tilde{a}^m_{i,k}, \tilde{b}^m_{i,k}, \tilde{z}^m_{i,k}, c_{i,m}\}} \\ & \times \Bigg[g_m(\mathbf{e}) + \frac{\rho}{2} \sum_{\substack{k\in\mathcal{M} \\ i\in\mathcal{I}}} \|\tilde{a}^m_{i,k} - a^{(t)}_{i,k} + u^{m(t)}_{i,k}\|^2 \\ & + \frac{\rho}{2} \sum_{\substack{k\in\mathcal{M} \\ i\in\mathcal{I}}} \|\tilde{b}^m_{i,k} - b^{(t)}_{i,k} + v^{m(t)}_{i,k}\|^2 \\ & + \frac{\rho}{2} \sum_{\substack{k\in\mathcal{M} \\ i\in\mathcal{I}}} \|\tilde{z}^m_{i,k} - z^{(t)}_{i,k} + w^{m(t)}_{i,k}\|^2 \Bigg]. \end{aligned} \tag{24}$$

Accordingly, each station $m \in \mathcal{M}$ solves the following equivalent optimization problem at the $(t+1)$-th iteration:

$$\begin{aligned} \min_{\{\tilde{a}^m_{i,k}, \tilde{b}^m_{i,k}, \tilde{z}^m_{i,k}, c_{i,m}\}} & \Bigg[g_m(\mathbf{e}) + \frac{\rho}{2} \sum_{\substack{k\in\mathcal{M} \\ i\in\mathcal{I}}} \|\tilde{a}^m_{i,k} - a^{(t)}_{i,k} + u^{m(t)}_{i,k}\|^2 \\ & + \frac{\rho}{2} \sum_{\substack{k\in\mathcal{M} \\ i\in\mathcal{I}}} \|\tilde{b}^m_{i,k} - b^{(t)}_{i,k} + v^{m(t)}_{i,k}\|^2 \\ & + \frac{\rho}{2} \sum_{\substack{k\in\mathcal{M} \\ i\in\mathcal{I}}} \|\tilde{z}^m_{i,k} - z^{(t)}_{i,k} + w^{m(t)}_{i,k}\|^2 \Bigg], \\ \text{s.t.} \quad & \mathbf{e} \in \chi_m. \end{aligned} \tag{25}$$

It is easy to observe that problem (25) is a convex problem with a quadric objective function and convex constraint. Thus, the corresponding optimal solution can be found by using the primal-dual interior-point algorithm [31] or standard software, such as CVX, CPLEX.

Step 2. Global variables updating: In this step, we focus on the global variables updating. Given $\mathbf{e}^{(t+1)}$, we minimize L_ρ with respect to global variables $\mathbf{s}$. Accordingly, the global variables $\mathbf{s}$ can be updated according to the following formulations,

$$\begin{aligned}\{\mathbf{a}_i\}^{(t+1)} = & \arg\min_{\{a_{i,k}\}} \\ & \times \left[\frac{\rho}{2} \sum_{m\in\mathcal{M}} \sum_{\substack{k\in\mathcal{M} \\ i\in\mathcal{I}}} \|\tilde{a}^{m(t+1)}_{i,k} - a_{i,k} + u^{m(t)}_{i,k}\|^2 \right], \end{aligned} \tag{26}$$

$$\begin{aligned}\{\mathbf{b}_i\}^{(t+1)} = & \arg\min_{\{a_{i,k}\}} \\ & \times \left[\frac{\rho}{2} \sum_{m\in\mathcal{M}} \sum_{\substack{k\in\mathcal{M} \\ i\in\mathcal{I}}} \|\tilde{b}^{m(t+1)}_{i,k} - b_{i,k} + v^{m(t)}_{i,k}\|^2 \right], \end{aligned} \tag{27}$$

$$\begin{aligned}\{\mathbf{z}\}^{(t+1)} = & \arg\min_{\{z_{i,k}\}} \\ & \times \left[\frac{\rho}{2} \sum_{m\in\mathcal{M}} \sum_{\substack{k\in\mathcal{M} \\ i\in\mathcal{I}}} \|\tilde{z}^{m(t+1)}_{i,k} - z_{i,k} + w^{m(t)}_{i,k}\|^2 \right]. \end{aligned} \tag{28}$$

Since problem (26), (27), (28) are unconstrained quadratic convex problems, we differentiate them with respect to $\mathbf{a}_i, \mathbf{b}_i, \mathbf{z}$ and obtain the following equations,

$$\rho \sum_{m\in\mathcal{M}} (\tilde{a}^{m(t+1)}_{i,k} - a_{i,k} + u^{m(t)}_{i,k}) = 0, \quad \forall i,k, \tag{29}$$

$$\rho \sum_{m\in\mathcal{M}} (\tilde{b}^{m(t+1)}_{i,k} - b_{i,k} + v^{m(t)}_{i,k}) = 0, \quad \forall i,k, \tag{30}$$

$$\rho \sum_{m\in\mathcal{M}} (\tilde{z}^{m(t+1)}_{i,k} - z_{i,k} + w^{m(t)}_{i,k}) = 0, \quad \forall i,k. \tag{31}$$

Then the global solution w.r.t. $\mathbf{a}_i, \mathbf{b}_i, \mathbf{z}$ can be obtained as follows,

$$a^{(t+1)}_{i,k} = \frac{1}{M} \sum_{m\in\mathcal{M}} (\tilde{a}^{m(t+1)}_{i,k} + u^{m(t)}_{i,k}), \quad \forall i,k, \tag{32}$$

$$b^{(t+1)}_{i,k} = \frac{1}{M} \sum_{m\in\mathcal{M}} (\tilde{b}^{m(t+1)}_{i,k} + v^{m(t)}_{i,k}), \quad \forall i,k, \tag{33}$$

$$z^{(t+1)}_{i,k} = \frac{1}{M} \sum_{m\in\mathcal{M}} (\tilde{z}^{m(t+1)}_{i,k} + w^{m(t)}_{i,k}), \quad \forall i,k. \tag{34}$$

By initializing the Lagrange multipliers of equations (32), (33), (34) as zero at the t-th iteration, the formulations can be reduced to

$$a^{(t+1)}_{i,k} = \frac{1}{M} \sum_{m\in\mathcal{M}} \tilde{a}^{m(t+1)}_{i,k}, \quad \forall i,k, \tag{35}$$

$$b^{(t+1)}_{i,k} = \frac{1}{M} \sum_{m\in\mathcal{M}} \tilde{b}^{m(t+1)}_{i,k}, \quad \forall i,k, \tag{36}$$

$$z^{(t+1)}_{i,k} = \frac{1}{M} \sum_{m\in\mathcal{M}} \tilde{z}^{m(t+1)}_{i,k}, \quad \forall i,k, \tag{37}$$

Algorithm 1: Binary Variables Recovery Algorithm.

Set $\mathcal{M}' := \emptyset$;
for each $k \in \mathcal{M}$ **do**
  set $a^*_{\tilde{i},k} := 1$ with $\tilde{i} = \arg\max_{i\in\mathcal{I}} a^*_{i,k}$ and $a^*_{i,k} := 0$ for all $i \in \mathcal{I}\backslash\{\tilde{i}\}$;
  set $\mathcal{M}' := \mathcal{M}' \cup \{k\}$;
  If any of the constraints (4)–(6) and (12b)–(12e) does not satisfy,
  Then Break
end for
Output the recovered binary variables $a^*_{i,k}$, $\forall i \in \mathcal{I}, k \in \mathcal{M}$.

which imply that the global variables $\mathbf{a}_i, \mathbf{b}_i, \mathbf{z}$ are updated at the $(t+1)$-th iteration by averaging the total local copies at all the corresponding stations.

Step 3. Lagrange multipliers updating: The Lagrange multipliers updating in this step can be represented as follows,

$$\boldsymbol{u}^{m(t+1)} = \boldsymbol{u}^{m(t)} + \tilde{\mathbf{a}}^{m(t+1)} - \mathbf{a}_i^{(t+1)}, \tag{38}$$

$$\boldsymbol{v}^{m(t+1)} = \boldsymbol{v}^{m(t)} + \tilde{\mathbf{b}}^{m(t+1)} - \mathbf{b}_i^{(t+1)}, \tag{39}$$

$$\boldsymbol{w}^{m(t+1)} = \boldsymbol{w}^{m(t)} + \tilde{\mathbf{z}}^{m(t+1)} - \mathbf{z}^{(t+1)}. \tag{40}$$

Since the cloud servers receive all the updating local variables from each MEC server, the Lagrange multipliers updating using equations (38)–(40) would be calculated at the cloud servers.

Step 4. Algorithm Stopping Criterion: The stopping criterion in [33] is adopted. The primal residuals for each MEC server under the feasible condition must be small, such as

$$\|\tilde{\mathbf{a}}^{m(t+1)} - \mathbf{a}_i^{(t+1)}\|_2 \le \epsilon_{pri}, \quad \forall m, \tag{41}$$

$$\|\tilde{\mathbf{b}}^{m(t+1)} - \mathbf{b}_i^{(t+1)}\|_2 \le \epsilon_{pri}, \quad \forall m, \tag{42}$$

$$\|\tilde{\mathbf{z}}^{m(t+1)} - \mathbf{z}^{(t+1)}\|_2 \le \epsilon_{pri}, \quad \forall m. \tag{43}$$

In the same spirit, the dual residuals under dual feasible condition are expressed as follows,

$$\|\mathbf{a}_i^{(t+1)} - \mathbf{a}_i^{(t)}\|_2 \le \epsilon_{dual}, \quad \forall m, \tag{44}$$

$$\|\mathbf{b}_i^{(t+1)} - \mathbf{b}_i^{(t)}\|_2 \le \epsilon_{dual}, \quad \forall m, \tag{45}$$

$$\|\mathbf{z}^{(t+1)} - \mathbf{z}^{(t)}\|_2 \le \epsilon_{dual}, \quad \forall m. \tag{46}$$

$\epsilon_{pri} > 0$ and $\epsilon_{dual} > 0$ express the feasible tolerances for the primal and dual feasibility conditions, respectively.

Step 5. Binary variables recovery: Due to the continuous relaxation in Subsection III-A, the obtained continuous values $\mathbf{a}_i, \mathbf{b}_i, \mathbf{c}_i, \mathbf{z}$ must be mapped to the binary values. According to [34], the binary values can be recovered by using the method as follows,

$$a^*_{\tilde{i},k} = \begin{cases} 1, & \text{if } \tilde{i} = \arg\max_{i\in\mathcal{I}} a^*_{i,k}; \\ 0, & \text{otherwise.} \end{cases} \tag{47}$$

Here, $a^*_{\tilde{i},k}$ denotes the recovered binary values. The details of the binary variables recovery algorithm are given in Algorithm 1. In the same way, the other three continuous relaxation values are converted into the binary values.

Algorithm 2: Proposed Distributed ADMM-Based Computation Offloading and Data Caching Algorithm.

Input: The number of APs M, ρ_i, $E_i^{\max}$, and α
Output: $\mathbf{a}_i$, $\mathbf{b}_i$, and $\mathbf{c}_i$, $i \in \mathcal{I}$
Initialization: Set the stopping criterion values ϵ_{pri} and ϵ_{dual},
and initialize the feasible set $(\mathbf{a}_i{}^{(t)}, \mathbf{b}_i{}^{(t)}, \mathbf{c}_i{}^{(t)})$, $i \in \mathcal{I}$,
and the scaling Lagrange multipliers vectors $\boldsymbol{u}^m$, $\boldsymbol{v}^m$, and $\boldsymbol{w}^m$, $t = 0$;
repeat
  for each AP m, $m \in \mathcal{M}$ **do**
    Update local variables $\tilde{\mathbf{a}}^{m(t+1)}$, $\tilde{\mathbf{b}}^{m(t+1)}$, $\tilde{\mathbf{z}}^{m(t+1)}$, $\mathbf{c}_m^{(t+1)}$
    by solving problem (25);
  end for
  Update global variables $\mathbf{a}_i^{(t+1)}$, $\mathbf{b}_i^{(t+1)}$, and $\mathbf{z}^{(t+1)}$ using (35), (36), and (37);
  Update multipliers $\boldsymbol{u}^{m(t+1)}$, $\boldsymbol{v}^{m(t+1)}$, and $\boldsymbol{w}^{m(t+1)}$ using (38), (39), and (40);
  $t = t + 1$;
until $\|\mathbf{a}_i^{(t+1)} - \mathbf{a}_i^{(t)}\|_2 \le \epsilon_{dual}$, $\|\mathbf{b}_i^{(t+1)} - \mathbf{b}_i^{(t)}\|_2 \le \epsilon_{dual}$, and $\|\mathbf{z}^{(t+1)} - \mathbf{z}^{(t)}\|_2 \le \epsilon_{dual}$, $\forall i$.
return $\mathbf{a}_i^{(t+1)}$, $\mathbf{b}_i^{(t+1)}$, and $\mathbf{c}_i^{(t+1)}$, $i \in \mathcal{I}$ to problem (15);
Map the continuous values to the binary values using Algorithm 1;
Output the recovered binary variables $\mathbf{a}_i^*$, $\mathbf{b}_i^*$, and $\mathbf{c}_i^*$, $i \in \mathcal{I}$.

The details of the proposed distributed ADMM-based computation offloading and data caching algorithm is given in Algorithm 2.

C. Algorithm Convergence

According to Proposition 1, problem (15) is convex. Besides, since problem (18) is the equivalent global consensus version of problem (15) by introducing local copies of the global variables into each AP and $g_m(\mathbf{e})$ is linear with respect to $\mathbf{e}$, problem (18) is also convex [35]. According to [36], [37], all the variables and the objective function of problem (18) are bounded. Then, it can be easily verified that problem (18) converges to its optimal point. In addition, since the feasible set of local variables of AP $m \in \mathbf{M}$ and the constraints of problem (18) are affine in respect to the optimization variables [33], the optimal duality gap of problem (18) is zero. Based on the appendix of [33], the objective function of problem (18) is convex, proper and closed, and its corresponding Lagrangian function (20) has a saddle point. Therefore, by applying the above updating rules with the ADMM method, the proposed ADMM algorithm is guaranteed to find the optimal point of problem (15).

D. Complexity Analysis

It is worthwhile to compare the complexity of the proposed ADMM-based distributed algorithm with other four algorithms, such as the centralized algorithm, optimal offloading and caching at AP without offloading to cloud servers (OOCWC) [38], optimal offloading to APs or cloud servers without caching (OMCWC) [39], and task caching and offloading (TCO) [12]. For the centralized algorithm in [40], its complexity is $\mathcal{O}(I^3(M+1)^3)$. Since [38] addresses its corresponding problem using distributed ADMM method in a single-cell scenario, OOCWC applies the consensus ADMM to solve the problem of the computation offloading and caching at AP without offloading to the cloud servers in the multi-cell scenario. Thus, the complexity of OOCWC is $\mathcal{O}(I^3)$ at each iteration. As stated in [39], the complexity of OMCWC at each iteration is $\mathcal{O}(I)$. For TCO, its complexity is $\mathcal{O}(I^3)$ in each iteration [40]. For the proposed ADMM-based distributed algorithm, in the local variables updating step, the computation complexity is $\mathcal{O}(I^3)$. Then, for the global variables updating, the computation complexity is $\mathcal{O}(I(M+1))$. Besides, the computation complexity of Lagrange multipliers updating is $\mathcal{O}(I)$ [35], and the per-iteration computational complexity of the proposed Algorithm 1 is $\mathcal{O}(I^3)+\mathcal{O}(I(M+1))+\mathcal{O}(I)\approx\mathcal{O}(I^3)$. Therefore, the major computational complexity for the proposed distributed algorithm is $K\mathcal{O}(I^3)$, where K denotes the number of iterations to solve problem (18). Clearly, Algorithm 1 has much lower computation complexity compared to the centralized algorithm. Although the complexity of the proposed algorithm is the same or less than the other three algorithms, the performance of the proposed algorithm is better than the other algorithms as demonstrated in simulation.

IV. Simulation Results

In this section, the extensive simulation results are presented to verify the performance of the proposed distributed algorithm via ADMM. Simulation is performed on Matlab R2018a. In this simulation, the scenario that the wireless channels remain unchanged during the processing of the mobile users computing tasks is considered [16]. This is applicable to a relatively static or low-speed movement scenario. Consider the scenario consisting of five APs, and the radius of each cell is set to 250 m. Moreover, each user is randomly distributed in the coverage area of 5 APs and it can associate with the corresponding AP through only one wireless channel. The wireless channel gain $G_{i,m}$ is modeled as $g_{i,m}d_{i,m}^{-\alpha}$ [41], where $g_{i,m}$ can be set as a constant, $d_{i,m}$ is the distance between user and its associating AP, and the path loss coefficient is $\alpha=4$. The uplink transmission power of each user is set equal to 0.01 W. Besides, the downlink transmission power of each AP is 2 W. The total channel bandwidth is set to be 20 MHz, and the noise power is $\sigma^2=2\times10^{-13}$ W.

For the computation task, we consider the autonomous vehicle applications [25], where the size of computation inputting data is 5500 KB and the correspondingly required number of CPU cycles to accomplish the task is 1000 M. The computational capability f_i^L is set equally for each user, e.g., $f_i^L=$ 512 M cycles/sec, $\forall i\in\mathcal{I}$, and the computational capability allocated for each user i at AP is 10G cycles/sec [42]. Furthermore, the cloud servers allocating the computational capability to the each user i is 30G cycles/sec. Each AP has the same storage ability to cache the associated input databases, and the storage ability is 10 GB.

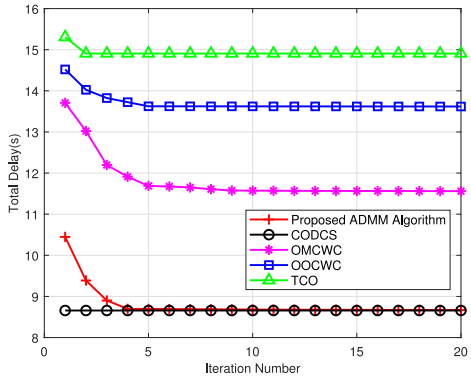


Fig. 2. The convergence of the proposed algorithm.

A. Comparison With Other Methods

To evaluate the proposed distributed algorithm, we compare our optimal joint offloading and caching strategy with the following four benchmark strategies, namely:

- *Optimal offloading and caching at AP without offloading to cloud servers (OOCWC)*: Each user only chooses to offload the computing task to AP, and AP cache the correspondingly inputting computation data [38].
- *Optimal offloading to APs or cloud servers without caching (OMCWC)*: Each user may offload its computation task to APs or cloud servers to process these tasks, and the corresponding computation data is not cached at APs [39].
- *Centralized optimal offloading and caching strategy (CODCS)*: A centralized data offloading and caching decision is found for all BSs and cloud servers to minimize the hybrid three-layer network latency.
- *Task caching and offloading (TCO)*: [12] designs the task caching and offloading based on the block coodinate descent method, which gives the offloading strategy first and then designs the caching strategy iteratively.

B. Performance Comparison

Fig. 2 shows the convergence behavior of the proposed distributed algorithm and OMCWC, OOCWC, CODCS and TCO for comparison. In this figure, the number of users is set as 10. It can be easily seen that the proposed algorithm is greatly improved within the convergence time. Moreover, our proposed algorithm can finally converge to the near optimal value, and the gap between proposed algorithm and CODCS is narrow and negligible. Moreover, our proposed algorithm uses the global variable consensus optimization method which avoids the local optimum at the expense of consuming more time to explore the near optimal solution. Also we can see that our proposed algorithm converges to a stable point within about 10 iterations.

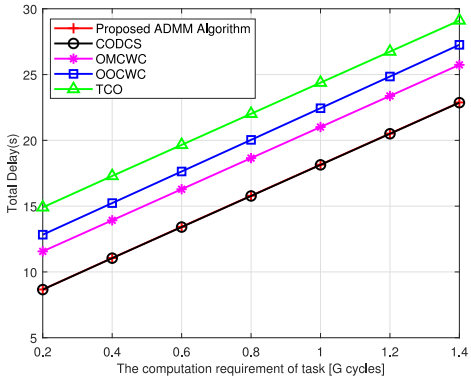


Fig. 3. Performance comparison of different schemes versus the computation requirement of tasks.

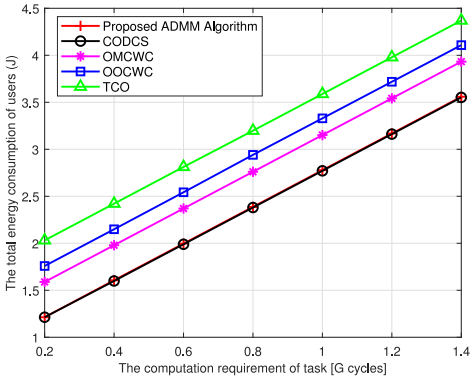


Fig. 4. The performance on the weighted sum of energy consumption in terms of different computation requirement of tasks.

In Fig. 3, we compare the performance of different offloading schemes for different computation requirements of tasks. As can be seen, the sum delay of all the users increases linearly with the computation requirements of the tasks. The number of users here is 10. Compared with OMCWC, OOCWC and TCO scheme, the total delay of all the users for the proposed ADMM algorithm can be reduced by 11.01%, 16.07% and 21.39%, respectively. Moreover, the performance of the proposed ADMM algorithm is very close to CODCS with the increasing computation requirement of tasks. This is because APs cache more of the most popularly associative databases. As a result, when the user computes the task locally or offloads data to APs, it does not need to fetch the corresponding databases directly from the cloud servers which incurs the backhaul delay. Besides, our proposed ADMM algorithm can make full use of cloud servers computation capacity. Thus, the proposed algorithm can greatly reduce the total user computing delay as the computation requirement of tasks increases.

Fig. 4 depicts the weighted sum of energy consumption for all users with the increasing computation requirement of tasks. We consider 10 users in this figure. The larger computation requirement of tasks leads to higher computation load in the networks, which can result in more energy consumption of the users. It is shown that the total energy consumption increases with increasing computation requirement when the number of

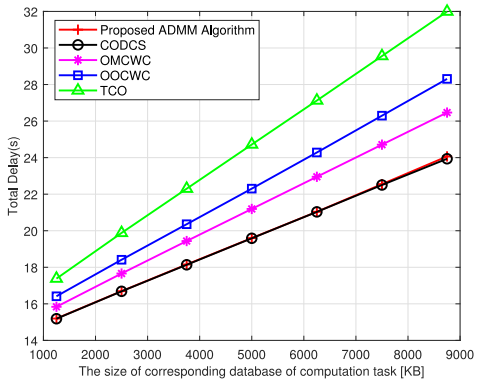


Fig. 5. The total delay versus different size of corresponding database of computation task.

users is constant. The reason is that when increasing the computation requirement of the computation intensive task, more users compute their own tasks at APs or cloud servers, which can produce more transmission energy consumption. In addition, our proposed ADMM algorithm has better network performance. This is because the proposed scheme can take full advantage of caching resources and the computation resources of APs and cloud servers. Comparing with OMCWC, OOCWC and TCO scheme, the proposed ADMM scheme reduces about 9.50%, 13.37% and 18.55% of the total users' energy consumption with different computation requirements of tasks. Besides, the curve of the proposed algorithm almost coincides with that of the scheme CODCS. This indicates that the proposed scheme can achieve the optimal solution as the centralized algorithm.

Fig. 5 compares the proposed ADMM algorithm with other four schemes in terms of the total users' computing task delay versus different sizes of corresponding database of computation task. In this figure, the number of users is set as 10. As can be observed from Fig. 5, in the regime of low size of corresponding database, the performance gap between the proposed ADMM algorithm and the other four schemes is small, since the transmission delay of the data collected by users and computation delay play a major role in the total users' computation delay. However, in the high regime size of corresponding database, the performance gap between our proposed algorithm and OMCWC, OOCWC and TCO scheme becomes large as the size of corresponding database increases, since the transmission delay of corresponding database makes a great contribution to the total user computing delay. Specifically, it is easy to observe that the gap between the proposed ADMM algorithm and CODCS scheme is fairly small. Compared with other schemes, the proposed ADMM algorithm reduces about 9.07%, 14.98%, 24.77% respectively in the total user computing delay with different database of computation task. The simulation results also illustrate that the proposed ADMM algorithm outperforms all other schemes in terms of the total user computing delay under the condition of ensuring almost the same performance as the CODCS scheme.

In order to further study the proposed ADMM algorithm, the total energy consumption of all the users with varying the size of the data collected by users among different schemes is

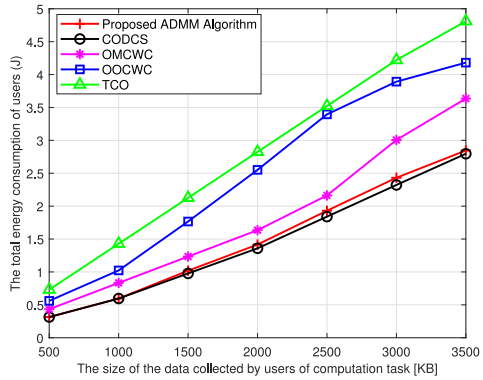


Fig. 6. The total energy consumption versus the size of the data collected by users of computation task.

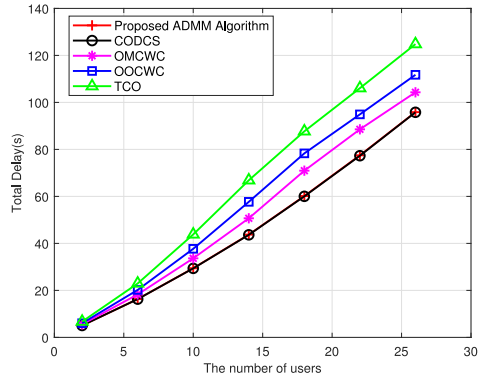


Fig. 7. The total user delay versus different number of users.

shown in Fig. 6. In Fig. 6, the number of users is set to 10. On the whole, the total users energy consumption increases in the size of the data collected by users of computation task. From Fig. 6, the proposed ADMM algorithm achieves lower total users' energy consumption than all other schemes. In the low regime of the data collected by users, the performance gap between the proposed ADMM algorithm and the other four schemes is small. The reason is that most users compute their tasks at APs or cloud servers in the low regime of the data collected by users. As the data collected by users increases, the incremental energy consumption is mainly from the data collected from users in transmission periods. Moreover, when the data collected by users is too large, most users may switch to local execution instead of offloading the tasks to APs or cloud servers, which would result in more energy consumption. The simulation results also demonstrate that the proposed ADMM algorithm can reduce the total user energy consumption by up to 21.73%, 31.94%, 40.85% respectively as compared to the OMCWC, OOCWC and TCO schemes. This is because our proposed ADMM algorithm takes full advantage of APs or cloud server computation offloading and users have more flexibility for offloading.

In Fig. 7, we compare the performance of different algorithms with the number of users varying from 2 to 26. Mobile users are randomly distributed in the coverage area of 5 APs. It can be seen that our proposed scheme outperforms the schemes of OMCWC, OOCWC and TCO. Besides, the gap between our proposed scheme and the scheme CODCS is quite small. This is due to fact that the proposed ADMM algorithm can provide more optional offloading and caching strategies, and lower total user delay by jointly optimizing the APs and cloud servers offloading and caching. In addition, the total user delay increases as the number of users increases. This is because more users will have more computing tasks, which can lead to increasing the total user delay. Moreover, when the number of users exceeds the computing or caching ability, some users may execute their tasks locally or further offload the tasks to cloud servers while causing more energy consumption. The simulation results also show that the proposed scheme can reduce the total user computing delay by up to 8.09%, 14.22%, 23.19% more than the schemes OMCWC, OOCWC and TCO, respectively.

V. Conclusion

In this paper, we investigated joint optimization of computation offloading and data caching strategy in a hybrid mobile cloud/edge computation system, and formulated it as a constrained optimization problem, with the objective to minimize the total users delay while satisfying the constraints of the users' energy consumption, APs' storage and computation capabilities. To tackle the bilinear discrete program problem, we first transformed the original nonconvex optimization problem into a linear program using the McCormick envelopes and binary variables relaxation. After that, in order to overcome the disadvantages of centralized algorithm, a distributed joint computation offloading with data caching scheme based on ADMM was proposed. Finally, numerical results have demonstrated that the proposed scheme can achieve better performance than the others. For the future work, we will study the mobility-aware service migration and computation offloading for vehicular edge computing networks.

References

[1] B. Lorenzo, J. Garcia-Rois, X. Li, J. Gonzalez-Castano, and Y. Fang, "A robust dynamic edge network architecture for the Internet of Things," *IEEE Netw.*, vol. 32, no. 1, pp. 8–15, Jan. 2018.

[2] R. Yu, G. Xue, V. T. Kilari, and X. Zhang, "The fog of things paradigm: Road toward on-demand Internet of Things," *IEEE Commun. Mag.*, vol. 56, no. 9, pp. 48–54, Sep. 2018.

[3] J. Zheng, Y. Cai, Y. Wu, and X. Shen, "Dynamic computation offloading for mobile cloud computing: A stochastic game-theoretic approach," *IEEE Trans. Mobile Comput.*, vol. 18, no. 4, pp. 771–786, Apr. 2019.

[4] B. Li, Z. Fei, and Y. Zhang, "UAV communications for 5G and beyond: Recent advances and future trends," *IEEE Internet Things J.*, vol. 6, no. 2, pp. 2241–2263, Apr. 2019.

[5] J. Liao, K. Wong, Y. Zhang, Z. Zheng, and K. Yang, "Coding, multicast, and cooperation for cache-enabled heterogeneous small cell networks," *IEEE Trans. Wireless Commun.*, vol. 16, no. 10, pp. 6838–6853, Oct. 2017.

[6] X. Li, X. Wang, K. Li, Z. Han, and V. C. M. Leung, "Collaborative multi-tier caching in heterogeneous networks: Modeling, analysis, and design," *IEEE Trans. Wireless Commun.*, vol. 16, no. 10, pp. 6926–6939, Oct. 2017.

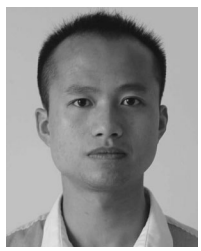
[7] Q. Li, W. Shi, X. Ge, and Z. Niu, "Cooperative edge caching in software-defined hyper-cellular networks," *IEEE J. Sel. Areas Commun.*, vol. 35, no. 11, pp. 2596–2605, Nov. 2017.

[8] W. Li, S. M. A. Oteafy, and H. S. Hassanein, "Rate-selective caching for adaptive streaming over information-centric networks," *IEEE Trans. Comput.*, vol. 66, no. 9, pp. 1613–1628, Sep. 2017.

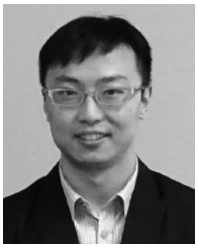
[9] K. N. Doan, T. Van Nguyen, T. Q. S. Quek, and H. Shin, "Content-aware proactive caching for backhaul offloading in cellular network," *IEEE Trans. Wireless Commun.*, vol. 17, no. 5, pp. 3128–3140, May 2018.
[10] X. Yang, J. Zheng, Z. Fei, and B. Li, "Optimal file dissemination and beamforming for cache-enabled C-RANs," *IEEE Access*, vol. 6, pp. 6390–6399, Mar. 2018.
[11] W. Fan, Y. Liu, B. Tang, F. Wu, and H. Zhang, "TerminalBooster: Collaborative computation offloading and data caching via smart basestations," *IEEE Wireless Commun. Lett.*, vol. 5, no. 6, pp. 612–615, Dec. 2016.
[12] Y. Hao, M. Chen, L. Hu, M. S. Hossain, and A. Ghoneim, "Energy efficient task caching and offloading for mobile edge computing," *IEEE Access*, vol. 6, pp. 11365–11373, Mar. 2018.
[13] Y. He, N. Zhao, and H. Yin, "Integrated networking, caching, and computing for connected vehicles: A deep reinforcement learning approach," *IEEE Trans. Veh. Technol.*, vol. 67, no. 1, pp. 44–55, Jan. 2018.
[14] Q. Chen, F. R. Yu, T. Huang, R. Xie, J. Liu, and Y. Liu, "Joint resource allocation for software-defined networking, caching, and computing," *IEEE/ACM Trans. Netw.*, vol. 26, no. 1, pp. 274–287, Feb. 2018.
[15] S. Yu, R. Langar, X. Fu, L. Wang, and Z. Han, "Computation offloading with data caching enhancement for mobile edge computing," *IEEE Trans. Veh. Technol.*, vol. 67, no. 11, pp. 11098–11112, Nov. 2018.
[16] M. Chen, B. Liang, and M. Dong, "Multi-user multi-task offloading and resource allocation in mobile cloud systems," *IEEE Trans. Wireless Commun.*, vol. 17, no. 10, pp. 6790–6805, Oct. 2018.
[17] J. Ren, G. Yu, Y. Cai, and Y. He, "Latency optimization for resource allocation in mobile-edge computation offloading," *IEEE Trans. Wireless Commun.*, vol. 17, no. 8, pp. 5506–5519, Aug. 2018.
[18] T. Ouyang, Z. Zhou, and X. Chen, "Follow me at the edge: Mobility-aware dynamic service placement for mobile edge computing," *IEEE J. Sel. Areas Commun.*, vol. 36, no. 10, pp. 2333–2345, Oct. 2018.
[19] J. Mei, K. Zheng, L. Zhao, Y. Teng, and X. Wang, "A latency and reliability guaranteed resource allocation scheme for LTE V2V communication systems," *IEEE Trans. Wireless Commun.*, vol. 17, no. 6, pp. 3850–3860, Jun. 2018.
[20] J. Zheng, Y. Wu, N. Zhang, H. Zhou, Y. Cai, and X. Shen, "Optimal power control in ultra-dense small cell networks: A game-theoretic approach," *IEEE Trans. Wireless Commun.*, vol. 16, no. 7, pp. 4139–4150, Jul. 2017.
[21] Y. Wu, L. P. Qian, J. Zheng, H. Zhou, and X. S. Shen, "Green-oriented traffic offloading through dual connectivity in future heterogeneous small cell networks," *IEEE Commun. Mag.*, vol. 56, no. 5, pp. 140–147, May 2018.
[22] T. Frederic, D. Yanakiev, and J. Berg, "Control method for autonomous vehicles," U.S. Patent Application No. 15/697,368, Sep. 2017.
[23] F. Wang, J. Xu, X. Wang, and S. Cui, "Joint offloading and computing optimization in wireless powered mobile-edge computing systems," *IEEE Trans. Wireless Commun.*, vol. 17, no. 3, pp. 1784–1797, Mar. 2018.
[24] M. Chen, Y. Qian, Y. Hao, Y. Li, and J. Song, "Data-driven computing and caching in 5G networks: Architecture and delay analysis," *IEEE Wireless Commun.*, vol. 25, no. 1, pp. 70–75, Feb. 2018.
[25] S. Kuutti, S. Fallah, K. Katsaros, M. Dianati, F. Mccullough, and A. Mouzakitis, "A survey of the state-of-the-art localization techniques and their potentials for autonomous vehicle applications," *IEEE Internet Things J.*, vol. 5, no. 2, pp. 829–846, Apr. 2018.
[26] W. Wen, Y. Cui, F. Zheng, S. Jin, and Y. Jiang, "Enhancing performance of random caching in large-scale heterogeneous wireless networks with random discontinuous transmission," *IEEE Trans. Commun.*, vol. 66, no. 12, pp. 6287–6303, Dec. 2018.
[27] A. P. Miettinen, and J. K. Nurminen, "Energy efficiency of mobile clients in cloud computing," in *Proc. 2nd USENIX Conf. Hot Topics Cloud Comput.*, Boston, CA, USA, Jun. 2010, pp. 1–4.
[28] P. M. Castro, "Tightening piecewise McCormick relaxations for bilinear problems," *Comput. Chem. Eng.*, vol. 72, pp. 300–311, 2015.
[29] Y. Wang, X. Tao, X. Zhang, and G. Mao, "Joint caching placement and user association for minimizing user download delay," *IEEE Access*, vol. 4, pp. 8625–8633, 2016.
[30] H. Nagarajan, M. Lu, E. Yamangil, and R. Bent, "Tightening McCormick relaxations for nonlinear programs via dynamic multivariate partitioning," in *Proc. Int. Conf. Princ. Pract. Constraint Program.*, Toulouse, France, Sep. 2016, pp. 369–387.
[31] C. Chi, W. Li, and C. Lin, *Convex Optimization for Signal Processing and Communications: From Fundamentals to Applications*. Boca Raton, FL, USA: CRC Press, Feb. 2017.
[32] Z. Wu and Z. Fei, "Precoder design in downlink CoMP-JT MIMO network via WMMSE and asynchronous ADMM," *Sci. China Inf. Sci.*, vol. 61, no. 8, pp. 082306:1–082306:13, Aug. 2018.
[33] S. Boyd, N. Parikh, E. Chu, B. Peleato, and J. Eckstein, "Distributed optimization and statistical learning via the alternating direction method of multipliers," *Found. Trends Mach. Learn.*, vol. 3, no. 1, pp. 1122, 2011.
[34] P. Luong, F. Gagnon, C. Despins, and L. Tran, "Optimal joint remote radio head selection and beamforming design for limited fronthaul C-RAN," *IEEE Trans. Signal Process.*, vol. 65, no. 21, pp. 5605–5620, Nov. 2017.
[35] L. Majzoobi, F. Lahouti, and V. Shah-Mansouri, "Analysis of distributed ADMM algorithm for consensus optimization in presence of node error," *IEEE Trans. Signal Process.*, vol. 67, no. 7, pp. 1774–1784, Apr. 2019.
[36] M. Liu, R. Yu, Y. Teng, V. C. M. Leung, and M. Song, "Computation offloading and content caching in wireless blockchain networks with mobile edge computing," *IEEE Trans. Veh. Technol.*, vol. 67, no. 11, pp. 11008–11021, Nov. 2018.
[37] X. Cao and K. J. R. Liu, "Distributed linearized ADMM for network cost minimization," *IEEE Trans. Signal Inf. Process. Over Netw.*, vol. 4, no. 3, pp. 626–638, Sep. 2018.
[38] S. Bi and Y. J. Zhang, "Computation rate maximization for wireless powered mobile-edge computing with binary computation offloading," *IEEE Trans. Wireless Commun.*, vol. 17, no. 6, pp. 4177–4190, Jun. 2018.
[39] H. Guo and J. Liu, "Collaborative computation offloading for multiaccess edge computing over fiberwireless networks," *IEEE Trans. Veh. Technol.*, vol. 67, no. 5, pp. 4514–4526, May 2018.
[40] A. Tal and A. Nemirovski, *Lectures on Modern Convex Optimization: Analysis, Algorithms, and Engineering Applications*. Philadelphia, PA, USA: Soc. Ind. Appl. Math., Aug. 2001.
[41] J. Zheng, Y. Cai, Y. Liu, Y. Xu, B. Duan, and X. Shen, "Optimal power allocation and user scheduling in multicell networks: Base station cooperation using a game-theoretic approach," *IEEE Trans. Wireless Commun.*, vol. 13, no. 12, pp. 6928–6942, Dec. 2014.
[42] 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.

Xiaolong Yang received the M.S. degree in communication and information systems from HeBei University, Baoding, China, in 2016. He is currently working toward the Ph.D. degree with the School of Information and Electronics, Beijing Institute of Technology, Beijing, China. Since December 2018, he has been a Visiting Student with Singapore University of Technology and Design, Singapore. His research interests include wireless caching, mobile edge computing, and resource allocation.

Zesong Fei received the B.Eng. and Ph.D. degrees in electronic engineering from the Beijing Institute of Technology (BIT), Beijing, China, in 1999 and 2004, respectively. He is currently a Professor with the Research Institute of Communication Technology, BIT. From 2012 to 2013, he was a Visiting Scholar with the University of Hong Kong, Hong Kong. His current research interests focus on mobile edge computing, physical-layer security, and error control coding.



Jianchao Zheng received the B.S. degree in communications engineering and the Ph.D. degree in communications and information systems from the College of Communications Engineering, PLA University of Science and Technology, Nanjing, China, in 2010 and 2016, respectively. From 2015 to 2016, he was a Visiting Scholar with the Broad-band Communications Research Group, Department of Electrical and Computer Engineering, University of Waterloo, Waterloo, ON, Canada. He is currently an Assistant Researcher with the National Innovation Institute of Defense Technology, Academy of Military Sciences of PLA, Beijing, China. He has authored or coauthored several papers in international conferences and reputed journals in his research area. His research interests focus on interference mitigation techniques, green communications and computing networks, game theory, learning theory, and optimization techniques.



Ning Zhang received the Ph.D. degree from the University of Waterloo, Waterloo, ON, Canada, in 2015. After that, he was a Postdoc Research Fellow with the University of Waterloo and University of Toronto, Canada, respectively. He is currently an Assistant Professor with Texas A&M University-Corpus Christi, USA. His current research interests include next-generation mobile networks, mobile edge computing, and security. He serves/served as an Associate Editor for the IEEE TRANSACTIONS ON COGNITIVE COMMUNICATIONS AND NETWORKING, IEEE INTERNET OF THING JOURNAL, IEEE ACCESS, and *IET Communications*, and a Guest Editor of several international journals, such as IEEE WIRELESS COMMUNICATIONS and IEEE TRANSACTIONS ON COGNITIVE COMMUNICATIONS AND NETWORKING. He has also served as the workshop Chair for MobiEdge'18 (in conjunction with IEEE WiMob 2018) and CoopEdge'18 (in conjunction with IEEE EDGE 2018), and 5G and NTN'19 (in conjunction with IEEE EDGE 2019). He was the recipient of several best paper awards from IEEE Globecom in 2014, IEEE WCSP in 2015, *Journal of Communications and Information Networks* in 2018, IEEE ICC, IEEE ICCC, and IEEE TECHNICAL COMMITTEE ON TRANSMISSION ACCESS AND OPTICAL SYSTEMS in 2019, respectively.

Alagan Anpalagan received the B.A.Sc., M.A.Sc., and Ph.D. degrees, all in electrical engineering from the University of Toronto, Toronto, ON, Canada. He was with the ELCE Department, Ryerson University, Canada, in 2001, and was promoted to Full Professor in 2010. He has served in many department in administrative positions such as Associate Chair, Program Director for electrical engineering, and Graduate Program Director. During his sabbatical, he was a Visiting Professor with the Asian Institute of Technology, and Visiting Researcher at Kyoto University. His industrial experience include working for three years with Bell Mobility, Nortel Networks, and IBM. He directs a research group working on radio resource management and radio access and networking areas within the WINCORE Laboratory. He has coauthored four edited books and two books in wireless communication and networking areas. He served as an Editor for the IEEE COMMUNICATIONS SURVEYS AND TUTORIALS (2012–2014), IEEE COMMUNICATIONS LETTERS (2010–2013), and *EURASIP Journal of Wireless Communications and Networking* (2004–2009). He has also served as a Guest Editor for six Special Issues published in IEEE, IET, and ACM. He served as TPC Co-Chair, IEEE VTC Fall 2017, TPC Co-Chair, IEEE INFOCOM16: Workshop on Green and Sustainable Networking and Computing, IEEE Globecom15: SAC Green Communication and Computing, IEEE PIMRC11: Cognitive Radio and Spectrum Management. He has served as a Vice Chair, IEEE SIG on Green and Sustainable Networking and Computing with Cognition and Cooperation (2015–2018), IEEE Canada Central Area Chair (2012–2014), IEEE Toronto Section Chair (2006–2007), ComSoc Toronto Chapter Chair (2004–2005), and IEEE Canada Professional Activities Committee Chair (2009–2011). He was the recipient of the IEEE Canada J.M. Ham Outstanding Engineering Educator Award (2018), YSGS Outstanding Contribution to Graduate Education Award (2017), Deans Teaching Award (2011), Faculty Scholastic, Research, and Creativity Award thrice from the Ryerson University. He was also the recipient of IEEE M.B. Broughton Central Canada Service Award (2016), Exemplary Editor Award from IEEE ComSoc (2013), Editor-in-Chief Top10 Choice Award in Transactions on Emerging Telecommunications Technology (2012), and a co-author of a paper that received IEEE SPS Young Author Best Paper Award (2015). He is a Registered Professional Engineer in the province of Ontario, Canada, Fellow of the Institution of Engineering and Technology, and Fellow of the Engineering Institute of Canada.