This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TWC.2021.3081991, IEEE Transactions on Wireless Communications

Optimizing AI Service Placement and Resource Allocation in Mobile Edge Intelligence Systems

Zehong Lin, *Student Member, IEEE*, Suzhi Bi, *Senior Member, IEEE*, and Ying-Jun Angela Zhang, *Fellow, IEEE*

***Abstract*—Leveraging recent advances on mobile edge computing (MEC), edge intelligence has emerged as a promising paradigm to support mobile artificial intelligence (AI) applications at the network edge. In this paper, we consider the AI service placement problem in a multi-user MEC system, where the access point (AP) places the most up-to-date AI program at user devices to enable local computing/task execution at the user side. To fully utilize the stringent wireless spectrum and edge computing resources, the AP sends the AI service program to a user only when enabling local computing at the user yields a better system performance. We formulate a mixed-integer non-linear programming (MINLP) problem to minimize the total computation time and energy consumption of all users by jointly optimizing the service placement (i.e., which users to receive the program) and resource allocation (on local CPU frequencies, uplink bandwidth, and edge CPU frequency). To tackle the MINLP problem, we derive analytical expressions to calculate the optimal resource allocation decisions with low complexity. This allows us to efficiently obtain the optimal service placement solution by search-based algorithms such as meta-heuristic or greedy search algorithms. To enhance the algorithm scalability in large-sized networks, we further propose an ADMM (alternating direction method of multipliers) based method to decompose the optimization problem into parallel tractable MINLP subproblems. The ADMM method eliminates the need of searching in a high-dimensional space for service placement decisions and thus has a low computational complexity that grows linearly with the number of users. Simulation results show that the proposed algorithms perform extremely close to the optimum and significantly outperform the other representative benchmark algorithms.**

***Index Terms*—Edge intelligence, mobile edge computing, service placement, resource allocation.**

I. Introduction

WITH the rapid development of Internet of Things (IoT), tens of billions of mobile devices, like smartphones, wearable devices, and sensors, are connected to the Internet, generating unprecedented volumes of data, such as social media contents, mobile payment statistics, and users' geo-location information, at the network edge. This triggers the proliferation of various mobile artificial intelligence (AI) applications (e.g., augmented reality, autonomous driving, and intelligent personal assistants) to fully unleash the potential of mobile big data. Nonetheless, the intensive computational demand for training and inference of AI applications far exceeds the computation and energy capacity of mobile devices.

Edge intelligence (EI) [2]–[5], the integration of mobile edge computing (MEC) and AI technologies, has recently emerged as a promising paradigm to support computation-intensive AI applications at the network edge. Specifically, the edge servers of mobile networks, e.g., cellular base stations (BSs) and wireless access points (APs) [6], [7], can provide cloud-like computing capabilities, greatly complementing the limited capacity of resource-constrained mobile devices. As the edge servers are in close proximity to the mobile devices and data sources, MEC avoids moving big data across the backhaul network compared with the conventional mobile cloud computing (MCC), and thus achieves lower latency and better privacy protection. With the aid of MEC, EI can push the computationally intensive training and inference processes of the AI models to the edge servers, making the mobile AI applications much more efficient. Meanwhile, recent advances in mobile AI chips, such as the neural processing units (NPU) integrated in HiSilicon's Kirin 970 chips and Apple's A11 bionic chips, equip the latest models of mobile devices with AI computation capabilities. With well-trained models, these advanced mobile devices can choose to run AI inference locally or at the edge servers [3]–[5] following the two basic computation offloading models of MEC, i.e., binary offloading and partial offloading [6].

In recent years, joint optimization of computation offloading and system-level resource allocation (e.g., radio spectrum, computing power and transmit power) for MEC systems has attracted significant research interests [8]–[16]. Most of the works implicitly assume that the required service programs for task computation are already available at both the edge servers and mobile devices. This assumption, however, is not true in AI services since the underlying AI models typically require continuous re-training. In particular, the AI models are trained using historical data and applied to the inference for future unseen data sampled from the same underlying distribution. Nonetheless, the environments are often nonstationary, and the data distribution can change over time. For instance, in an online shopping application, the customers'

Manuscript received December 10, 2020; revised March 24, 2021; accepted May 12, 2021. This work was supported in part by the National Key Research and Development Program under Project 2019YFB1803305, the National Natural Science Foundation of China (Project 61871271), the General Research Fund established by the Research Grants Council of Hong Kong under Project 14208017 and Project 14201920, the Guangdong Province Pearl River Scholar Funding Scheme 2018, the Key Project of Department of Education of Guangdong Province (No. 2020ZDZX3050), the Foundation of Shenzhen City (Project JCYJ20170818101824392, JCYJ20190808120415286). This article was presented in part at the IEEE Global Communications Conference (GLOBECOM), Taipei, Taiwan, December 2020 [1]. The associate editor coordinating the review of this article and approving it for publication was L. Yang. (*Corresponding author: Suzhi Bi.*)

Zehong Lin and Ying-Jun Angela Zhang are with the Department of Information Engineering, The Chinese University of Hong Kong, Hong Kong (e-mail: lz018@ie.cuhk.edu.hk; yjzhang@ie.cuhk.edu.hk).

Suzhi Bi is with the College of Electronics and Information Engineering, Shenzhen University, Shenzhen, China, 518060 (e-mail: bsz@szu.edu.cn). Suzhi Bi is also with the Peng Cheng Laboratory, Shenzhen, China, 518066.

© 2021 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See https://www.ieee.org/publications/rights/index.html for more information.

buying preferences may vary with time, depending on many factors, including the date, the availability of alternatives, etc. The changes of the underlying data distribution may result in concept drift problems [17]–[19], degrading AI inference performance. Therefore, to avoid model degradation over time, an AI service program must be updated either periodically or upon significant changes of the environment by re-training the AI model with newly collected data [20], [21]. The updated AI service program is then selectively disseminated to the edge servers and/or mobile devices. Notice that a server or a device can execute an AI task only when the updated service program is placed at it. Otherwise, its computation tasks must be offloaded to other devices where the service program is available. In this regard, [22] studied service placement in an MEC network with multiple edge servers to maximize the number of served computation offloading requests under edge storage, computation, and communication constraints. To cope with the unknown and fluctuating service demand, [23] proposed an online learning algorithm to optimize spatial-temporal dynamic service placement decisions among multiple edge servers to minimize the computation delay. Considering parallel computing at both cloud and edge servers, [24] and [25] studied collaborative service placement and computation offloading to minimize the computation latency.

The above works [22]–[25] assumed that mobile devices offload all their computation tasks for remote execution, and mainly focused on optimizing the service placement at the edge servers to ease the burden of the cloud. Nonetheless, due to the time-varying characteristic of wireless channels and limited edge computing capability, offloading all the computation tasks to the edge server is not always the optimal choice. To make efficient use of idle local computing power, it is more advantageous to opportunistically offload computation tasks and allow the mobile devices to execute some tasks locally [8]–[16]. Noticeably, placing service programs at the mobile devices incurs additional program transmission delay. Recent work in [26] takes such delay into account and jointly optimizes the service placement and computation offloading decisions in a single-user MEC system. The optimal design becomes much more complicated in a general multi-user scenario with heterogeneous wireless channel conditions and hardware configurations, where the users share limited system resources including the computing power of the server and the uplink channel spectrum for task offloading. In this case, the system resource allocation, computation offloading, and service placement decisions are closely correlated. For instance, whether placing the service at a user depends on the delay of obtaining the program, the user's local computing capability, the computing capability of the server, and the allocated bandwidth for task offloading. Therefore, we need to jointly optimize these coupling factors to achieve the optimal system computing performance.

In this paper, we consider the AI service placement problem in a multi-user MEC system, as shown in Fig. 1. Upon the update of an AI model, the edge server selectively transmits the program of the AI model to a subset of users via a broadcast channel. In particular, the AP sends the AI service program to a user only when enabling local computing at the user yields a better system performance in terms of the total time and energy consumption (TEC). We are interested in minimizing the total TEC of all users. The main contributions of this paper are summarized as follows.

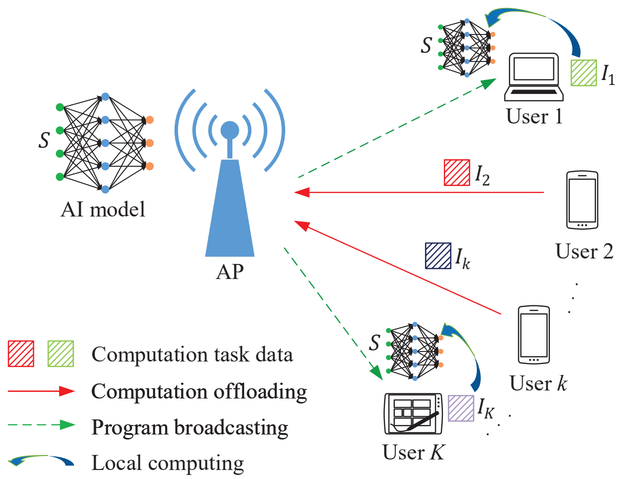


Fig. 1: The considered MEC system with AI service placement.

- We formulate a mixed-integer non-linear programming (MINLP) problem for joint optimization of service placement, computational and radio resource allocation to minimize the total TEC of the users. The problem is challenging to solve due to the combinatorial service placement decision and its strong coupling with the communication and computational resource allocation decisions.
- We derive analytical expressions to efficiently calculate the optimal resource allocation decisions, including the local CPU frequencies, the edge CPU frequency, and the uplink bandwidth allocation, given the service placement decision. The analysis allows us to search for the optimal service placement solution at a low computational complexity via, e.g., greedy search or meta-heuristic methods.
- To avoid high-dimensional search when the network size is large, we propose an ADMM (alternating direction method of multipliers) based algorithm that decomposes the original problem into parallel and tractable subproblems, one for each user. As such, the total computational complexity of the ADMM-based algorithm increases linearly with the number of users, and is much more scalable than the search-based algorithms especially when the network size is large.

Simulation results show that the proposed algorithms achieve a close-to-optimal performance and significantly reduce the total computation delay and energy consumption compared with various benchmark algorithms. Moreover, we observe that the proposed search-based algorithms and the ADMM algorithm have their respective advantages. In particular, the search-based algorithms achieve lower computational complexity when the network size is small (e.g., ≤ 8 users) due to the analytical expressions for calculating the optimal resource allocation decisions. On the other hand, the ADMM algorithm is preferred when the network size becomes large.

The rest of the paper is organized as follows. We introduce the system model and formulate the joint optimization problem in Section II. In Section III, we derive analytical expressions to calculate the optimal computational and communication resource allocation decisions. Based on the analysis, we optimize the service placement decision via search-based algorithms. In Section IV, we propose an ADMM-based algorithm to enhance the algorithm scalability in large-sized networks. In Section V, we evaluate the proposed algorithms via extensive simulations. Finally, we conclude the paper in Section VI.

II. System Model and Problem Formulation

A. System Model

As shown in Fig. 1, we consider a multi-user MEC system consisting of K single-antenna users, denoted by the set $\mathcal{K} = \{1, \cdots, K\}$, and a single-antenna AP co-located with an edge server.[1] Suppose that each user has a certain amount of local data (e.g., personal images) to be processed by a common AI service (e.g., an image recognition program). The edge server periodically re-trains the AI model based on the latest data to avoid model degradation. It selectively disseminates the updated model to the mobile devices based on the optimal service placement decision. A mobile device can either offload its local data to the edge server for remote processing (e.g., AI inference) or process its data locally if it receives the service program from the edge server. One example application of on-device AI inference is skin cancer detection [27], which deploys a pre-trained convolutional neural network (CNN) on a mobile device. For a new skin image provided by the mobile device, the AI model is used to classify skin lesions locally. Another application is smart classrooms in [28], where three pre-trained deep neural network (DNN) models, for object, text, and voice recognition, are embedded in a mobile app to facilitate on-device DNN inference to control different classroom devices via an IoT micro-server.

To reduce the communication overhead incurred by periodic service placement, we assume that the AP disseminates the service program via downlink broadcasting. Let $\mathcal{K}_1 \subseteq \mathcal{K}$ denote the subset of users the AP chooses to transmit the service program to, and $\mathcal{K}_0 = \mathcal{K} \setminus \mathcal{K}_1$ denote the set of remaining users. Notice that only the users in $\mathcal{K}_1$ are able to compute their tasks locally, while the users in $\mathcal{K}_0$ have to offload all the computation to the edge server for remote execution. We can show that the AP sends the program to a user only when the user is going to perform local computing. Otherwise, it may cause an unnecessary increase in service placement delay. Therefore, we suppose that all users in $\mathcal{K}_1$ perform local computing hereafter.

In this paper, we consider a frequency division duplexing (FDD) operating mode where the downlink program broadcasting and the uplink computation offloading are operated simultaneously over orthogonal frequency bands, denoted by W_D and W_U, respectively. Besides, the users in $\mathcal{K}_0$ share the uplink bandwidth using frequency division multiple access (FDMA). That is, user k occupies a bandwidth of $a_k W_\text{U}$, where $a_k \in [0, 1]$ and $\sum_{k \in \mathcal{K}_0} a_k \leq 1$. Likewise, let h_k and g_k denote the wireless channel gains between the AP and user k in the downlink and uplink, respectively. In this paper, we consider an offline model that the AP is assumed to have non-causal knowledge of users' channel state information (CSI) [2] and computation requirements.

B. Downlink Service Placement Model

Suppose that the AP disseminates a service program of size S to the users in the set $\mathcal{K}_1$ by broadcasting in the downlink channel with power p_0. To ensure correct decoding of all users in $\mathcal{K}_1$, the AP adapts its broadcasting rate r_0 according to the worst-case user in $\mathcal{K}_1$. That is,

$$r_0 = W_\text{D} \log_2 \left(1 + \frac{p_0 h_\text{min}}{W_\text{D} N_0} \right), \tag{1}$$

where $h_\text{min} = \min\{h_k | k \in \mathcal{K}_1\}$ denotes the smallest downlink channel gain in the user set $\mathcal{K}_1$ and N_0 denotes the noise power spectral density. Then, the time consumed on broadcasting the service program is $\tau_0 = \frac{S}{r_0}$.

Let p_k^r denote the circuit power consumption at the receiver of user k. The energy consumed for receiving the service program at user k is

$$e_k^r = p_k^r \tau_0. \tag{2}$$

C. Computation Model

Suppose that user k has I_k-bit task data to be computed by the AI service program.[3] Besides, let L_k denote the computing workload in terms of the total number of CPU cycles required for completing the task of user k. Depending on the local availability of the program, we describe the details of the local computing and edge computing models in the following.

1) Local Computing: A user k conducts local computing when it is in $\mathcal{K}_1$, i.e., having the service program placed locally. Let f_k^l denote the local CPU frequency of user k, which is limited by a maximum value F_k, i.e., $f_k^l \leq F_k$. Then, the time consumed on local computing by user k is

$$\tau_k^l = \frac{L_k}{f_k^l}. \tag{3}$$

The corresponding energy consumption is [9]

$$e_k^l = \kappa_k (f_k^l)^3 \tau_k^l = \kappa_k (f_k^l)^2 L_k, \tag{4}$$

where $\kappa_k > 0$ denotes the computing energy efficiency coefficient.

[1] In the remainder of this paper, we use AP and edge server interchangeably.

[2] Similar to conventional wireless communication systems, the CSI can be obtained by channel estimation using pilot signals. If the AP suffers from CSI estimation errors, the performance of the proposed algorithms may degrade. In this case, some robust optimization techniques are needed to maintain the performance, which, however, is out of the scope of this paper.

[3] The task data and service program are determined by the specific AI application. For the deep learning based image recognition application in [29] that was written in C++, the task data can be personal images and the service program is the executable file compiled from the C++ code.

2) Edge Computing: A user k offloads its computation tasks to the edge server when it is in $\mathcal{K}_0$. Let p_k denote the transmit power of user k. Then, the uplink data rate of user k is

$$r_k^u = a_k W_{\mathrm{U}} \log_2\left(1 + \frac{p_k g_k}{a_k W_{\mathrm{U}} N_0}\right). \tag{5}$$

The time spent on offloading the task data of user k is

$$\tau_k^u = \frac{I_k}{r_k^u}, \tag{6}$$

and the corresponding energy consumption is

$$e_k^u = p_k \tau_k^u = p_k \frac{I_k}{r_k^u}. \tag{7}$$

Suppose that the edge server assigns a CPU frequency f_k^c to compute the task of user k. Then, the task processing time of user k at the edge server is

$$\tau_k^c = \frac{L_k}{f_k^c}, \quad \forall k \in \mathcal{K}_0. \tag{8}$$

Due to the limitation of the computation capability of the edge server, the following edge CPU frequency constraint holds:

$$\sum_{k \in \mathcal{K}_0} f_k^c \leq F^c, \tag{9}$$

where F^c is the maximum CPU frequency of the edge server.

In practice, the transmit power of the AP is much stronger than the users. Meanwhile, the size of the computation result is usually much smaller than the input data size. Thus, we can neglect the time spent on downloading the computing results from the AP to the users (as in [8], [11]–[13], [15]).

D. Problem Formulation

From the above discussion, we can calculate the total time consumption of user k as

$$T_k = \begin{cases} \tau_0 + \tau_k^l, & \text{if } k \in \mathcal{K}_1, \\ \tau_k^u + \tau_k^c, & \text{if } k \in \mathcal{K}_0, \end{cases} \tag{10}$$

and the total energy consumption of user k

$$E_k = \begin{cases} e_k^r + e_k^l, & \text{if } k \in \mathcal{K}_1, \\ e_k^u, & \text{if } k \in \mathcal{K}_0. \end{cases} \tag{11}$$

In particular, the total computation time of a user in $\mathcal{K}_1$ consists of the downlink service deployment delay and the local computing time. Likewise, the total computation time of a user in $\mathcal{K}_0$ consists of the task offloading time and the edge computing time. Define TEC_k of user k as the weighted sum of the computation time and energy consumption, i.e., $\mathrm{TEC}_k = \beta_k^T T_k + \beta_k^E E_k$, where $\beta_k^T \geq 0$ and $\beta_k^E \geq 0$ are the weighting factors that satisfy $\beta_k^E = 1 - \beta_k^T$ [26], [30]. We are interested in minimizing the total TEC, given by $\sum_{k\in\mathcal{K}} \mathrm{TEC}_k$, by jointly optimizing the service placement decision $\mathcal{K}_1 \subseteq \mathcal{K}$, the local CPU frequencies $\mathbf{f}^l \triangleq \{f_k^l\}$, the uplink bandwidth allocation $\mathbf{a} \triangleq \{a_k\}$, and the edge CPU frequency allocation $\mathbf{f}^c \triangleq \{f_k^c\}$. Mathematically, the total TEC minimization problem is formulated as

$$\begin{aligned}
\text{(P1)}: \min_{\mathcal{K}_1, \mathbf{f}^l, \mathbf{a}, \mathbf{f}^c} \; & V(\mathcal{K}_1, \mathbf{f}^l, \mathbf{a}, \mathbf{f}^c) \triangleq \sum_{k\in\mathcal{K}_1} \left[\beta_k^T \left(\tau_0 + \frac{L_k}{f_k^l}\right)\right. \\
& \left. + \beta_k^E \left(p_k^r \tau_0 + \kappa_k (f_k^l)^2 L_k\right)\right] \\
& + \sum_{k\in\mathcal{K}_0} \left[\beta_k^T \left(\tau_k^u + \frac{L_k}{f_k^c}\right) + \beta_k^E p_k \tau_k^u\right] && \text{(12a)} \\
\text{s.t.} \quad & \sum_{k\in\mathcal{K}_0} a_k \leq 1, && \text{(12b)} \\
& \sum_{k\in\mathcal{K}_0} f_k^c \leq F^c, && \text{(12c)} \\
& 0 \leq f_k^l \leq F_k, \quad \forall k \in \mathcal{K}_1, && \text{(12d)} \\
& a_k \geq 0, \; f_k^c \geq 0, \quad \forall k \in \mathcal{K}_0, && \text{(12e)} \\
& \mathcal{K}_1 \subseteq \mathcal{K}, \; \mathcal{K}_0 = \mathcal{K} \setminus \mathcal{K}_1. && \text{(12f)}
\end{aligned}$$

Define $V^*(\mathcal{K}_1) = \min_{\mathbf{f}^l, \mathbf{a}, \mathbf{f}^c} V(\mathcal{K}_1, \mathbf{f}^l, \mathbf{a}, \mathbf{f}^c)$ as the optimal objective function value of (P1) given $\mathcal{K}_1$. Here, (12b) and (12c) correspond to the bandwidth allocation constraint and the edge CPU frequency allocation constraint, respectively. (12d) is due to the local CPU frequency constraints.

Problem (P1) is a mixed-integer non-linear programming (MINLP) problem, which is in general non-convex. In Section III and IV, we propose low-complexity algorithms to address the problem.

III. Optimal Computational and Communication Resource Allocation

We observe that (P1) is jointly convex in $(\mathbf{f}^l, \mathbf{a}, \mathbf{f}^c)$ once $\mathcal{K}_1$ is given. In this section, we derive analytical expressions to efficiently calculate the optimal computational and communication resource allocation $\{(\mathbf{f}^l)^*, \mathbf{a}^*, (\mathbf{f}^c)^*\}$ for a given service placement decision $\mathcal{K}_1$. Based on the analysis, search-based algorithms such as meta-heuristic methods (e.g., Gibbs sampling, particle swarm optimization, etc.) can be conducted to optimize $\mathcal{K}_1$ with low-complexity.

A. Optimal Resource Allocation for Given $\mathcal{K}_1$

Suppose that $\mathcal{K}_1$ is given. We can accordingly obtain $\mathcal{K}_0 = \mathcal{K} \setminus \mathcal{K}_1$. From (5) and (6), we see that τ_k^u is uniquely determined by a_k. Therefore, it is equivalent to regard τ_k^u's as the optimization variables of (P1) and introduce the following constraints on τ_k^u's to (P1):

$$\frac{I_k}{\tau_k^u} \leq a_k W_{\mathrm{U}} \log_2\left(1 + \frac{p_k g_k}{a_k W_{\mathrm{U}} N_0}\right), \quad \forall k \in \mathcal{K}_0, \tag{13}$$

$$\tau_k^u \geq 0, \quad \forall k \in \mathcal{K}_0. \tag{14}$$

Notice that (P1) can be separately optimized for the users in $\mathcal{K}_1$ and the users in $\mathcal{K}_0$. In particular, each user $k \in \mathcal{K}_1$ independently optimizes its local CPU frequency f_k^l by solving

$$\min_{0 \leq f_k^l \leq F_k} \quad \beta_k^T \left(\tau_0 + \frac{L_k}{f_k^l}\right) + \beta_k^E \left(p_k^r \tau_0 + \kappa_k (f_k^l)^2 L_k\right). \tag{15}$$

This article has been accepted for publication in IEEE Transactions on Wireless Communications. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TWC.2021.3081991

Since (15) is a convex optimization problem, we can obtain the optimal solution $(f_k^l)^*$ by finding the stationary point and considering the boundary condition:

$$(f_k^l)^* = \min\left\{F_k, \sqrt[3]{\frac{\beta_k^T}{2\beta_k^E \kappa_k}}\right\}. \tag{16}$$

From (16), we observe that $(f_k^l)^*$ increases with the ratio $\frac{\beta_k^T}{\beta_k^E \kappa_k}$. Noticeably, a smaller ratio indicates more emphasis on minimizing the energy consumption at user k. Thus, the optimal local CPU frequency of user k is reduced.

On the other hand, for the users in $\mathcal{K}_0$, we need to jointly optimize the uplink time allocation $\{\tau_k^u\}$, the uplink bandwidth allocation $\mathbf{a}$, and the edge CPU frequency allocation $\mathbf{f}^c$ by solving

$$\min_{\{\tau_k^u\},\mathbf{a},\mathbf{f}^c} \sum_{k\in\mathcal{K}_0} \left[\beta_k^T\left(\tau_k^u + \frac{L_k}{f_k^c}\right) + \beta_k^E p_k \tau_k^u\right] \tag{17a}$$

$$\text{s.t.} \quad \frac{I_k}{\tau_k^u} \le a_k W_{\mathrm{U}} \log_2\left(1 + \frac{p_k g_k}{a_k W_{\mathrm{U}} N_0}\right), \quad \forall k \in \mathcal{K}_0, \tag{17b}$$

$$\sum_{k\in\mathcal{K}_0} a_k \le 1, \tag{17c}$$

$$\sum_{k\in\mathcal{K}_0} f_k^c \le F^c, \tag{17d}$$

$$\tau_k^u \ge 0,\ a_k \ge 0,\ f_k^c \ge 0, \quad \forall k \in \mathcal{K}_0. \tag{17e}$$

We can express the partial Lagrangian as

$$\begin{aligned}\mathcal{L}(\{\tau_k^u\}, \mathbf{a}, \mathbf{f}^c, \boldsymbol{\lambda}, \mu, \nu) &= \sum_{k\in\mathcal{K}_0} \left[\beta_k^T\left(\tau_k^u + \frac{L_k}{f_k^c}\right) + \beta_k^E p_k \tau_k^u\right] \\ &+ \sum_{k\in\mathcal{K}_0} \lambda_k \left[\frac{I_k}{\tau_k^u} - a_k W_{\mathrm{U}} \log_2\left(1 + \frac{p_k g_k}{a_k W_{\mathrm{U}} N_0}\right)\right] \\ &+ \mu\left(\sum_{k\in\mathcal{K}_0} a_k - 1\right) + \nu\left(\sum_{k\in\mathcal{K}_0} f_k^c - F^c\right),\end{aligned} \tag{18}$$

where $\boldsymbol{\lambda} \triangleq \{\lambda_k\} \ge 0$ denotes the dual variables associated with the constraints in (17b). $\mu \ge 0$ and $\nu \ge 0$ are the dual variables associated with the constraints in (17c) and (17d), respectively. Accordingly, the dual function is

$$\begin{aligned} g(\boldsymbol{\lambda}, \mu, \nu) = &\min_{\{\tau_k^u\},\mathbf{a},\mathbf{f}^c} \ \mathcal{L}(\{\tau_k^u\}, \mathbf{a}, \mathbf{f}^c, \boldsymbol{\lambda}, \mu, \nu) \\ &\quad \text{s.t.} \quad \tau_k^u \ge 0,\ a_k \ge 0,\ f_k^c \ge 0, \quad \forall k \in \mathcal{K}_0, \end{aligned} \tag{19}$$

and the corresponding dual problem is

$$\max_{\boldsymbol{\lambda}\ge 0, \mu \ge 0, \nu \ge 0} \ g(\boldsymbol{\lambda}, \mu, \nu). \tag{20}$$

(17) is a convex problem, and thus strong duality holds between (17) and (20). Therefore, we can equivalently solve (17) by solving (20).

Let $\{\lambda_k^*, \mu^*, \nu^*\}$ denote the optimal dual variables. Then, the closed-form expressions of the optimal solution $\{(\tau_k^u)^*, a_k^*, (f_k^c)^*\}$ to (17) are given in the following propositions.

Proposition 3.1: The optimal offloading time allocation $(\tau_k^u)^*$ is given by

$$(\tau_k^u)^* = \sqrt{\frac{\lambda_k^* I_k}{\beta_k^T + \beta_k^E p_k}}, \quad \forall k \in \mathcal{K}_0. \tag{21}$$

Proof: The partial derivative of $\mathcal{L}(\{\tau_k^u\}, \mathbf{a}, \mathbf{f}^c, \boldsymbol{\lambda}, \mu, \nu)$ with respect to τ_k^u is

$$\frac{\partial \mathcal{L}(\{\tau_k^u\}, \mathbf{a}, \mathbf{f}^c, \boldsymbol{\lambda}, \mu, \nu)}{\partial \tau_k^u} = \beta_k^T + \beta_k^E p_k - \lambda_k \frac{I_k}{(\tau_k^u)^2}. \tag{22}$$

By setting $\frac{\partial \mathcal{L}(\{\tau_k^u\}, \mathbf{a}, \mathbf{f}^c, \boldsymbol{\lambda}, \mu, \nu)}{\partial \tau_k^u} = 0$ at the minimum point, we have

$$\tau_k^u = \sqrt{\frac{\lambda_k I_k}{\beta_k^T + \beta_k^E p_k}}, \tag{23}$$

which completes the proof. ■

From Proposition 3.1, we observe that a smaller value of $(\beta_k^T + \beta_k^E p_k)$ and/or a larger task data size I_k leads to a longer offloading delay. Besides, $\lambda_k^* > 0$ must hold, because otherwise $(\tau_k^u)^* = 0$ and $r_k^u = \frac{I_k}{(\tau_k^u)^*} \to \infty$, which is not achievable.

Proposition 3.2: The optimal uplink bandwidth allocation a_k^* is given by

$$a_k^* = \frac{\frac{p_k g_k}{W_{\mathrm{U}} N_0}}{-\left(W\left(-\frac{1}{\exp\left(\frac{\mu^* \ln 2}{\lambda_k^* W_{\mathrm{U}}} + 1\right)}\right)\right)^{-1} - 1}, \quad \forall k \in \mathcal{K}_0, \tag{24}$$

where $W(x)$ denotes the Lambert-W function, which is the inverse function of $z\exp(z) = x$, i.e., $z = W(x)$.

Proof: The partial derivative of $\mathcal{L}(\{\tau_k^u\}, \mathbf{a}, \mathbf{f}^c, \boldsymbol{\lambda}, \mu, \nu)$ with respect to a_k is

$$\begin{aligned}&\frac{\partial \mathcal{L}(\{\tau_k^u\}, \mathbf{a}, \mathbf{f}^c, \boldsymbol{\lambda}, \mu, \nu)}{\partial a_k} \\ &= -\frac{\lambda_k W_{\mathrm{U}}}{\ln 2}\left[\ln\left(1 + \frac{p_k g_k}{a_k W_{\mathrm{U}} N_0}\right) - \frac{p_k g_k}{a_k W_{\mathrm{U}} N_0 + p_k g_k}\right] + \mu.\end{aligned} \tag{25}$$

By setting $\frac{\partial \mathcal{L}(\{\tau_k^u\}, \mathbf{a}, \mathbf{f}^c, \boldsymbol{\lambda}, \mu, \nu)}{\partial a_k} = 0$ at the minimum point, we have

$$\ln\left(1 + \frac{p_k g_k}{a_k W_{\mathrm{U}} N_0}\right) = \frac{\mu \ln 2}{\lambda_k W_{\mathrm{U}}} + 1 - \frac{1}{1 + \frac{p_k g_k}{a_k W_{\mathrm{U}} N_0}}. \tag{26}$$

By taking a natural exponential operation at both sides, we have

$$\left(1 + \frac{p_k g_k}{a_k W_{\mathrm{U}} N_0}\right) \exp\left(\frac{1}{1 + \frac{p_k g_k}{a_k W_{\mathrm{U}} N_0}}\right) = \exp\left(\frac{\mu \ln 2}{\lambda_k W_{\mathrm{U}}} + 1\right). \tag{27}$$

Consider two positive values u and v that satisfy $\frac{1}{u}\exp(u) = v$, it holds that

$$-u\exp(-u) = -\frac{1}{v}. \tag{28}$$

Therefore, we have $u = -W(-\frac{1}{v})$, where $W(x)$ is the Lambert-W function, which is the inverse function of

© 2021 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See https://www.ieee.org/publications/rights/index.html for more information.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TWC.2021.3081991, IEEE Transactions on Wireless Communications

$z\exp(z) = x$, i.e., $z = W(x)$. Comparing (27) and (28), we can infer that

$$\frac{1}{1+\frac{p_k g_k}{a_k W_{\mathrm{U}} N_0}} = -W\left(-\frac{1}{\exp\left(\frac{\mu \ln 2}{\lambda_k W_{\mathrm{U}}}+1\right)}\right)$$

$$\Rightarrow\ a_k = \frac{\frac{p_k g_k}{W_{\mathrm{U}} N_0}}{-\left(W\left(-\frac{1}{\exp\left(\frac{\mu \ln 2}{\lambda_k W_{\mathrm{U}}}+1\right)}\right)\right)^{-1}-1}, \tag{29}$$

which completes the proof. ■

Proposition 3.2 indicates that $\mu^* > 0$ must hold. Otherwise, $a_k^* \to \infty$, which is evidently not true at the optimum because $a_k^* \leq 1$ must hold. Since $\mu^* > 0$, and $\lambda_k^* > 0$ by Proposition 3.1, we have $-\frac{1}{\exp\left(\frac{\mu^* \ln 2}{\lambda_k^* W_{\mathrm{U}}}+1\right)} \in (-1/e, 0)$. Moreover, we have $W(x) \in (-1, 0)$ when $x \in (-1/e, 0)$. Thus, the right-hand side of (24) is strictly positive, i.e., $a_k^* > 0$.

Proposition 3.3: The optimal edge CPU frequency allocation $(f_k^c)^*$ is given by

$$(f_k^c)^* = \sqrt{\frac{\beta_k^T L_k}{\nu^*}}, \quad \forall k \in \mathcal{K}_0. \tag{30}$$

Proof: The partial derivative of $\mathcal{L}(\{\tau_k^u\}, \mathbf{a}, \mathbf{f}^c, \boldsymbol{\lambda}, \mu, \nu)$ with respect to f_k^c is

$$\frac{\partial \mathcal{L}(\{\tau_k^u\}, \mathbf{a}, \mathbf{f}^c, \boldsymbol{\lambda}, \mu, \nu)}{\partial f_k^c} = -\beta_k^T \frac{L_k}{(f_k^c)^2} + \nu. \tag{31}$$

By setting $\frac{\partial \mathcal{L}(\{\tau_k^u\}, \mathbf{a}, \mathbf{f}^c, \boldsymbol{\lambda}, \mu, \nu)}{\partial f_k^c} = 0$ at the minimum point, we have

$$f_k^c = \sqrt{\frac{\beta_k^T L_k}{\nu}}, \tag{32}$$

which completes the proof. ■

Proposition 3.3 indicates that $\nu^* > 0$ must hold to ensure a finite $(f_k^c)^*$. Meanwhile, we observe that $(f_k^c)^*$ increases with the weighting factor β_k^T and the computing workload L_k. This means that the edge server allocates more computing power to speed up the computation of the users that have heavier computing workload (larger L_k) or emphasize more on the computation delay (larger β_k^T).

Based on Propositions 3.1-3.3, we can apply the ellipsoid method [31] to obtain the optimal $\{\boldsymbol{\lambda}^*, \mu^*, \nu^*\}$. The basic idea of the ellipsoid method is to iteratively generate a sequence of ellipsoids with decreasing volumes from an initial ellipsoid $\mathcal{E}^{(0)}$ that contains $\{\boldsymbol{\lambda}^*, \mu^*, \nu^*\}$ [31]. Specifically, we can use any $\{\boldsymbol{\lambda}, \mu, \nu\} \geq 0$ as the center of $\mathcal{E}^{(0)}$ and set the volume to be sufficiently large to contain $\{\boldsymbol{\lambda}^*, \mu^*, \nu^*\}$. At each iteration t, we update the dual variables $\{\boldsymbol{\lambda}, \mu, \nu\}$ using the following subgradients:

$$\Delta\lambda_k = \frac{I_k}{\tau_k^u} - a_k W_{\mathrm{U}} \log_2\left(1+\frac{p_k g_k}{a_k W_{\mathrm{U}} N_0}\right), \quad \forall k \in \mathcal{K}_0, \tag{33}$$

$$\Delta\mu = \sum_{k\in\mathcal{K}_0} a_k - 1, \tag{34}$$

$$\Delta\nu = \sum_{k\in\mathcal{K}_0} f_k^c - F^c, \tag{35}$$

and generate a new ellipsoid $\mathcal{E}^{(t)}$ of reduced volume that contains the corresponding half-space of $\mathcal{E}^{(t-1)}$ [31]. The update of $\{\boldsymbol{\lambda}, \mu, \nu\}$ repeats until the specified stopping criterion [31] is met. Since (17) is convex, the ellipsoid method guarantees to converge to the optimal solution.

Algorithm 1 Optimal Resource Allocation for Problem (P1) With Given $\mathcal{K}_1$

```
input: service placement decision $\mathcal{K}_1$.
initialization: $\mathcal{K}_0 = \mathcal{K} \setminus \mathcal{K}_1$, $\{\boldsymbol{\lambda}, \mu, \nu\} \geq 0$ for $\mathcal{K}_0$;
for each $k \in \mathcal{K}_1$ do
    Calculate $(f_k^l)^*$ using (16).
end for
repeat
    for each $k \in \mathcal{K}_0$ do
        Calculate $a_k^*$ using (24), and $(f_k^c)^*$ using (30);
    end for
    Update $\{\boldsymbol{\lambda}, \mu, \nu\}$ by the ellipsoid method using the
    subgradients defined in (33)-(35).
until $\{\boldsymbol{\lambda}, \mu, \nu\} \geq 0$ converge to a prescribed accuracy.
return the optimal $\{(\mathbf{f}^l)^*, \mathbf{a}^*, (\mathbf{f}^c)^*\}$ to (P1) given $\mathcal{K}_1$.
```

Algorithm 1 illustrates the pseudo code of the algorithm to solve (P1) given $\mathcal{K}_1$. In this algorithm, the complexity of obtaining the optimal local CPU frequencies $(\mathbf{f}^l)^*$ is $O(|\mathcal{K}_1|)$, because we can directly calculate the optimal $(f_k^l)^*$ in closed-form for the users in $\mathcal{K}_1$. In addition, the ellipsoid method requires $O(m^2)$ iterations to converge, where m is the number of dual variables [32]. In this paper, $m = |\mathcal{K}_0| + 2$. Since we can calculate the optimal primal variables $\{(\tau_k^u)^*, a_k^*, (f_k^c)^*\}$ in closed-form, the complexity of each iteration in the ellipsoid method is proportional to the number of users in $\mathcal{K}_0$, i.e., $O(|\mathcal{K}_0|)$. Therefore, the overall computational complexity is $O\big(m^2|\mathcal{K}_0| + |\mathcal{K}_1|\big)$. Since $|\mathcal{K}_0| \leq K$ and $|\mathcal{K}_1| = K - |\mathcal{K}_0|$, we can conclude that the overall computational complexity of Algorithm 1 is upper bounded by $O\big(K^3\big)$.

B. Optimization of Service Placement Decision

The analysis in the previous subsection allows us to efficiently obtain the optimal resource allocation solutions $\{(\mathbf{f}^l)^*, \mathbf{a}^*, (\mathbf{f}^c)^*\}$ with given $\mathcal{K}_1$. This facilitates low-complexity implementation of search-based algorithms, such as the meta-heuristic algorithms including Gibbs sampling [33], particle swarm optimization [34], etc. to obtain the optimal $\mathcal{K}_1$. In particular, meta-heuristic algorithms strategically sample a subset of all feasible solutions of $\mathcal{K}_1$. With the closed-form expressions derived in Propositions 3.1-3.3, we can quickly calculate the objective function value associated with each sampled $\mathcal{K}_1$ by Algorithm 1, thus significantly expediting the meta-heuristic algorithms.

To further reduce the complexity, we discuss below an iterative greedy search algorithm. Let $\mathcal{K}_1^{(n)}$ denote the service placement decision at iteration n. Likewise, we have $\mathcal{K}_0^{(n)} = \mathcal{K} \setminus \mathcal{K}_1^{(n)}$. Correspondingly, the optimal objective value of (P1) at iteration n is $V^*\big(\mathcal{K}_1^{(n)}\big)$. We initially set $\mathcal{K}_0^{(0)} = \mathcal{K}$ and $\mathcal{K}_1^{(0)} = \emptyset$. Then, in each iteration $n \geq 1$, we find the best user in $\mathcal{K}_0^{(n-1)}$ such that once the user is removed from $\mathcal{K}_0^{(n-1)}$ and assigned to $\mathcal{K}_1^{(n-1)}$, the optimal total TEC $V^*\big(\mathcal{K}_1^{(n-1)}\big)$ drops

© 2021 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See https://www.ieee.org/publications/rights/index.html for more information.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TWC.2021.3081991, IEEE Transactions on Wireless Communications

the most significantly. The process repeats until we cannot further decrease the total TEC by moving a user from $\mathcal{K}_0^{(n-1)}$ to $\mathcal{K}_1^{(n-1)}$, or $\mathcal{K}_0^{(n-1)} = \emptyset$.

There are at most K iterations in the greedy search algorithm. In the n-th iteration, the algorithm needs to search over $(K - n + 1)$ users in the set $\mathcal{K}_0^{(n-1)}$ and solves the corresponding optimization problem in (P1). Thus, a total of $\sum_{n=1}^{K} K - n + 1 = \frac{K^2+K}{2}$ optimization problems need to be solved in the worst case. As discussed, the complexity of Algorithm 1 is upper bounded by $O\big(K^3\big)$. Thus, the overall complexity of the greedy search algorithm is upper bounded by $O\big(K^5\big)$, implying that the algorithm can find the solution in polynomial time.

C. A Homogeneous Special Case

In this subsection, we study a special case where the users differ only by their wireless channel gains g_k's and h_k's. In this case, the weighting factors, local computing capability, computing energy efficiency, transmit power, circuit power consumption, and task parameters (I_k, L_k) are identical for all users, i.e., $\beta_k^T = \beta$, $\beta_k^E = 1 - \beta$, $F_k = F$, $\kappa_k = \kappa$, $p_k = p$, $p_k^r = p^r$, $I_k = I$, and $L_k = L, \forall k \in \mathcal{K}$. In this case, $(f_k^l)^*$'s in (16) are equal at the optimum, i.e., $(f_k^l)^* = \min\left\{F, \sqrt[3]{\frac{\beta}{2(1-\beta)\kappa}}\right\}$, for the users in $\mathcal{K}_1$. Consequently, the local computing delay and energy consumption are equal for all users in $\mathcal{K}_1$. Likewise, $(f_k^c)^*$'s are equal at the optimum for all users in $\mathcal{K}_0$. Since $\sum_{k\in\mathcal{K}_0} f_k^c = F^c$ must hold at the optimum, the total edge CPU frequency is equally allocated to the users in $\mathcal{K}_0$.

On the other hand, we have the following proposition on the relation between g_k, $(\tau_k^u)^*$, the uplink spectral efficiency $\log_2\big(1+\frac{pg_k}{a_k^* W_{\mathrm{U}} N_0}\big)$ and the offloading data rate $a_k^* W_{\mathrm{U}} \log_2\big(1+\frac{pg_k}{a_k^* W_{\mathrm{U}} N_0}\big)$.

Proposition 3.4: For a user in $\mathcal{K}_0$, a worse uplink channel gain g_k results in a longer offloading time $(\tau_k^u)^*$, a lower spectral efficiency in the uplink, i.e., $\log_2\big(1+\frac{pg_k}{a_k^* W_{\mathrm{U}} N_0}\big)$, and a smaller offloading data rate $a_k^* W_{\mathrm{U}} \log_2\big(1+\frac{pg_k}{a_k^* W_{\mathrm{U}} N_0}\big)$.

Before proving Proposition 3.4, we first prove the following Corollary on the relation between the optimal λ_k^* and g_k.

Corollary 3.1: The optimal λ_k^* is a non-increasing function of g_k.

Proof: We prove Corollary 3.1 by contradiction. Suppose that λ_k^* increases with g_k. According to the KKT condition $\lambda_k^*\Big[\frac{I}{(\tau_k^u)^*} - a_k^* W_{\mathrm{U}} \log_2\big(1+\frac{pg_k}{a_k^* W_{\mathrm{U}} N_0}\big)\Big] = 0$ and $\lambda_k^* > 0$ from Proposition 3.1, we have $\frac{I}{(\tau_k^u)^*} - a_k^* W_{\mathrm{U}} \log_2\big(1+\frac{pg_k}{a_k^* W_{\mathrm{U}} N_0}\big) = 0$ at the optimum for all $k \in \mathcal{K}_0$. Note that $(\tau_k^u)^*$ increases with λ_k^* according to (21), and thus it also increases with g_k. In addition, since $W(x)$ is an increasing function when $x \in (-1/e, 0)$, we can infer from (24) that a_k^* increases with both g_k and λ_k^*. Therefore, $\frac{I}{(\tau_k^u)^*} - a_k^* W_{\mathrm{U}} \log_2\big(1+\frac{pg_k}{a_k^* W_{\mathrm{U}} N_0}\big)$ decreases with g_k. Thus, the condition $\frac{I}{(\tau_k^u)^*} - a_k^* W_{\mathrm{U}} \log_2\big(1+\frac{pg_k}{a_k^* W_{\mathrm{U}} N_0}\big) = 0$ cannot be simultaneously satisfied for all $k \in \mathcal{K}_0$. This contradiction implies that the assumption must be false, which leads to the proof. ■

Proof of Proposition 3.4: (21) and Corollary 3.1 indicate that a user with a worse uplink channel condition consumes a longer time $(\tau_k^u)^*$ to offload its computation task at the optimum. Since the task data size is identical for all users in the homogeneous case, the offloading data rate $a_k^* W_{\mathrm{U}} \log_2\big(1+\frac{pg_k}{a_k^* W_{\mathrm{U}} N_0}\big)$ decreases when g_k becomes worse according to (5) and (6).

In the following, we prove that the optimal uplink spectral efficiency also drops when g_k is poor. Indeed, by substituting (24) into (5), we can express the optimal uplink spectral efficiency of user k as

$$\log_2\left(1+\frac{pg_k}{a_k^* W_{\mathrm{U}} N_0}\right) = \log_2\big(\chi_k(\lambda_k^*, \mu^*)\big), \tag{36}$$

where

$$\chi_k(\lambda_k^*, \mu^*) = -\left(W\left(-\frac{1}{\exp\big(\frac{\mu^* \ln 2}{\lambda_k^* W_{\mathrm{U}}} + 1\big)}\right)\right)^{-1} \tag{37}$$

is a decreasing function in λ_k^*. We infer from (36) that the spectral efficiency of the uplink from user k to the AP decreases with λ_k^*, and thus increases with g_k. This completes the proof. ■

The above proposition shows that not only the uplink task offloading rate, but also the uplink spectral efficiency drops when g_k becomes worse. Interestingly, we observe in Fig. 3(c) in the simulation section that the optimal uplink bandwidth allocation a_k^* also increases when g_k is smaller. This interesting phenomenon and Proposition 3.4 imply that the users with worse uplink channels will be allocated with more uplink bandwidth but still result in a lower uplink data rate. In other words, it is spectrally inefficient to let users with poor channel gains offload their tasks for edge computing.

Inspired by the above analysis and observation, we design the following uplink-based heuristic algorithm for the homogeneous special case. In particular, we sort all users according to the ascending order of g_k, and initialize $\mathcal{K}_1 = \emptyset$. At iteration l, we select user k_l with the l-th smallest channel gain. The user k_l is added into $\mathcal{K}_1$, i.e., $\mathcal{K}_1 = \mathcal{K}_1 \cup \{k_l\}$, if it reduces the objective value $V^*(\mathcal{K}_1)$. The process repeats until $l = K$. This algorithm solves (P1) K times. The complexity is significantly lower than the greedy search algorithm proposed in Section III-B, which needs to solve (P1) $O(K^2)$ times. We will later show in Section V-A that this heuristic algorithm performs extremely close to the optimal one in the homogeneous special case.

IV. Joint Optimization Using ADMM-Based Algorithm

The complexity of the aforementioned search-based algorithms becomes high when K grows large. In this section, we propose an ADMM-based algorithm to decompose (P1) into K parallel MINLP problems, one for each user. As such, the overall complexity grows much more slowly when K increases.

We introduce binary decision variables b_k's to denote the service placement decisions, where $b_k = 1$ if user $k \in \mathcal{K}_1$ and $b_k = 0$ if user $k \in \mathcal{K}_0$. Denote $\mathbf{b} \triangleq \{b_k\}$. In (P1) τ_0 introduces strong coupling among the users, as it is determined by the worst channel gain among the users in $\mathcal{K}_1$. To facilitate the

© 2021 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See https://www.ieee.org/publications/rights/index.html for more information.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TWC.2021.3081991, IEEE Transactions on Wireless Communications

decomposition, we regard τ_0 as an optimization variable and reformulate (P1) as

$$\text{(P2):} \min_{\mathbf{b},\mathbf{f}^l,\mathbf{a},\mathbf{f}^c,\tau_0} \sum_{k=1}^{K} \Bigg\{ b_k \Bigg[\beta_k^T \left(\tau_0 + \frac{L_k}{f_k^l} \right) + \beta_k^E \left(p_k^r \tau_0 + \kappa_k (f_k^l)^2 L_k \right) \Bigg] + (1-b_k) \left[\beta_k^T \left(\frac{I_k}{r_k^u} + \frac{L_k}{f_k^c} \right) + \beta_k^E p_k \frac{I_k}{r_k^u} \right] \Bigg\} \tag{38a}$$

$$\text{s.t.} \quad \sum_{k=1}^{K} a_k \leq 1, \tag{38b}$$

$$\sum_{k=1}^{K} f_k^c \leq F^c, \tag{38c}$$

$$0 \leq f_k^l \leq F_k, \quad \forall k \in \mathcal{K}, \tag{38d}$$

$$a_k \geq 0, \ f_k^c \geq 0, \quad \forall k \in \mathcal{K}, \tag{38e}$$

$$b_k \in \{0,1\}, \quad \forall k \in \mathcal{K}, \tag{38f}$$

$$\tau_0 \geq b_k \frac{S}{W_{\mathrm{D}} \log_2 \left(1 + \frac{p_0 h_k}{W_{\mathrm{D}} N_0} \right)}, \quad \forall k \in \mathcal{K}. \tag{38g}$$

Note that the optimization variables $\mathbf{a}$, $\mathbf{f}^c$ and τ_0 are coupled among the users in the constraints (38b), (38c) and (38g), respectively. To decompose (P2), we introduce the local copies of the variables $\mathbf{a}$, $\mathbf{f}^c$ and τ_0 as $\mathbf{x} \triangleq \{x_k\}$, $\mathbf{y} \triangleq \{y_k\}$ and $\mathbf{z} \triangleq \{z_k\}$, respectively. Then, we reformulate (P2) as

$$\min_{\mathbf{b},\mathbf{f}^l,\mathbf{a},\mathbf{f}^c,\tau_0,\mathbf{x},\mathbf{y},\mathbf{z}} \sum_{k=1}^{K} q_k(b_k, f_k^l, x_k, y_k, z_k) + g(\mathbf{a}, \mathbf{f}^c, \tau_0) \tag{39a}$$

$$\text{s.t.} \quad \text{(38d), (38f)},$$

$$z_k \geq b_k \frac{S}{W_{\mathrm{D}} \log_2 \left(1 + \frac{p_0 h_k}{W_{\mathrm{D}} N_0} \right)}, \quad \forall k \in \mathcal{K}, \tag{39b}$$

$$x_k = a_k, \quad \forall k \in \mathcal{K}, \tag{39c}$$

$$y_k = f_k^c, \quad \forall k \in \mathcal{K}, \tag{39d}$$

$$z_k = \tau_0, \quad \forall k \in \mathcal{K}, \tag{39e}$$

$$x_k \geq 0, \ y_k \geq 0, \quad \forall k \in \mathcal{K}, \tag{39f}$$

where

$$q_k(b_k, f_k^l, x_k, y_k, z_k) = b_k \left[\beta_k^T \left(z_k + \frac{L_k}{f_k^l} \right) + \beta_k^E \left(p_k^r z_k + \kappa_k (f_k^l)^2 L_k \right) \right] + (1-b_k) \left[\beta_k^T \left(\frac{I_k}{r_k^{u\prime}} + \frac{L_k}{y_k} \right) + \beta_k^E \frac{p_k I_k}{r_k^{u\prime}} \right], \tag{40}$$

and $r_k^{u\prime} = x_k W_{\mathrm{U}} \log_2 \left(1 + \frac{p_k g_k}{x_k W_{\mathrm{U}} N_0} \right)$. Besides,

$$g(\mathbf{a}, \mathbf{f}^c, \tau_0) = \begin{cases} 0, & \text{if } (\mathbf{a}, \mathbf{f}^c, \tau_0) \in \mathcal{G}, \\ +\infty, & \text{otherwise}, \end{cases} \tag{41}$$

where

$$\mathcal{G} = \left\{ (\mathbf{a}, \mathbf{f}^c, \tau_0) \middle| \sum_{k=1}^{K} a_k \leq 1, \sum_{k=1}^{K} f_k^c \leq F^c, a_k \geq 0, f_k^c \geq 0, k \in \mathcal{K}; \tau_0 \geq 0 \right\}.$$

Now, we can apply the ADMM technique [35] to decompose Problem (39). By introducing multipliers to the constraints in (39c)-(39e), the augmented Lagrangian of (39) is

$$\mathcal{L}(\mathbf{u}, \mathbf{v}, \boldsymbol{\theta}) = \sum_{k=1}^{K} q_k(\mathbf{u}) + g(\mathbf{v}) + \sum_{k=1}^{K} \rho_k (x_k - a_k) + \sum_{k=1}^{K} \phi_k (y_k - f_k^c) + \sum_{k=1}^{K} \varphi_k (z_k - \tau_0) + \frac{c}{2} \sum_{k=1}^{K} (x_k - a_k)^2 + \frac{c}{2} \sum_{k=1}^{K} (y_k - f_k^c)^2 + \frac{c}{2} \sum_{k=1}^{K} (z_k - \tau_0)^2, \tag{42}$$

where $\mathbf{u} = \{\mathbf{b}, \mathbf{f}^l, \mathbf{x}, \mathbf{y}, \mathbf{z}\}$, $\mathbf{v} = \{\mathbf{a}, \mathbf{f}^c, \tau_0\}$, $\boldsymbol{\theta} = \{\boldsymbol{\rho}, \boldsymbol{\phi}, \boldsymbol{\varphi}\}$, and $c > 0$ is a fixed step size. Accordingly, the dual function is

$$d(\boldsymbol{\theta}) = \min_{\mathbf{u},\mathbf{v}} \ \mathcal{L}(\mathbf{u}, \mathbf{v}, \boldsymbol{\theta}) \quad \text{s.t.} \ \text{(38d), (38f), (39b), (39f)}, \tag{43}$$

and the dual problem is

$$\max_{\boldsymbol{\theta}} \ d(\boldsymbol{\theta}). \tag{44}$$

The ADMM method solves the dual problem (44) by iteratively updating $\mathbf{u}$, $\mathbf{v}$, and $\boldsymbol{\theta}$. We denote the values in the i-th iteration as $\{\mathbf{u}^i, \mathbf{v}^i, \boldsymbol{\theta}^i\}$. Then, in the $(i+1)$-th iteration, the variables are updated sequentially as follows:

1) Step 1: In this step, we update the local variables $\mathbf{u}$ as

$$\mathbf{u}^{i+1} = \operatorname*{argmin}_{\mathbf{u}} \mathcal{L}(\mathbf{u}, \mathbf{v}^i, \boldsymbol{\theta}^i). \tag{45}$$

Notice that the minimization problem in (45) can be decomposed into K parallel subproblems. Each subproblem solves two optimization problems, one for $b_k = 0$ and another one for $b_k = 1$. In particular, the optimization problem for $b_k = 0$ is

$$\min_{x_k, y_k, z_k \geq 0} \ \beta_k^T \left(\frac{I_k}{r_k^{u\prime}} + \frac{L_k}{y_k} \right) + \beta_k^E \frac{p_k I_k}{r_k^{u\prime}} + \rho_k^i x_k + \phi_k^i y_k + \varphi_k^i z_k + \frac{c}{2} \left(x_k - a_k^i \right)^2 + \frac{c}{2} (y_k - (f_k^c)^i)^2 + \frac{c}{2} (z_k - \tau_0^i)^2, \tag{46}$$

and the optimization problem for $b_k = 1$ is

$$\min_{f_k^l, x_k, y_k, z_k} \ \beta_k^T \left(z_k + \frac{L_k}{f_k^l} \right) + \beta_k^E \left(p_k^r z_k + \kappa_k (f_k^l)^2 L_k \right) + \rho_k^i x_k + \phi_k^i y_k + \varphi_k^i z_k + \frac{c}{2} \left(x_k - a_k^i \right)^2 + \frac{c}{2} (y_k - (f_k^c)^i)^2 + \frac{c}{2} (z_k - \tau_0^i)^2$$

$$\text{s.t.} \quad 0 \leq f_k^l \leq F_k,$$

© 2021 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See https://www.ieee.org/publications/rights/index.html for more information.

$$\begin{aligned} & z_k \geq \frac{S}{W_{\mathrm{D}} \log_2\left(1+\frac{p_0 h_k}{W_{\mathrm{D}} N_0}\right)}, \\ & x_k \geq 0,\ y_k \geq 0. \end{aligned} \tag{47}$$

Note that both (46) and (47) are strictly convex problems that can be solved using general convex optimization algorithm, e.g., projected Newton's method [32]. Therefore, we can simply select $b_k = 0$ or 1 that yields a smaller objective value as b_k^{i+1}, and the corresponding optimal solution as $\{(f_k^l)^{i+1}, x_k^{i+1}, y_k^{i+1}, z_k^{i+1}\}$. After solving the K subproblems, the optimal solution to (45) is given by $\mathbf{u}^{i+1} = \{\mathbf{b}^{i+1}, (\mathbf{f}^l)^{i+1}, \mathbf{x}^{i+1}, \mathbf{y}^{i+1}, \mathbf{z}^{i+1}\}$. Therefore, the overall computational complexity of Step 1 is $O(K)$. Notice that the K subproblems can be solved in parallel, thus the computational time of Step 1 is constant when we conduct parallel computing.

2) Step 2: Having obtained $\mathbf{u}^{i+1}$, we update the global variables $\mathbf{v}$ as

$$\mathbf{v}^{i+1} = \underset{\mathbf{v}}{\operatorname{argmin}}\, \mathcal{L}(\mathbf{u}^{i+1}, \mathbf{v}, \boldsymbol{\theta}^i). \tag{48}$$

By the definition of $g(\mathbf{v})$ in (41), $\mathbf{v}^{i+1} \in \mathcal{G}$ must hold at the optimum. Accordingly, the minimization problem in (48) is equivalent to the following convex optimization problem

$$\begin{aligned} \mathbf{v}^{i+1} = \underset{\mathbf{a}, \mathbf{f}^c, \tau_0}{\operatorname{argmin}} & \sum_{k=1}^K \rho_k^i (x_k^{i+1} - a_k) + \sum_{k=1}^K \phi_k^i (y_k^{i+1} - f_k^c) \\ & + \sum_{k=1}^K \varphi_k^i (z_k^{i+1} - \tau_0) + \frac{c}{2} \sum_{k=1}^K (x_k^{i+1} - a_k)^2 \\ & + \frac{c}{2} \sum_{k=1}^K (y_k^{i+1} - f_k^c)^2 + \frac{c}{2} \sum_{k=1}^K (z_k^{i+1} - \tau_0)^2 \\ \text{s.t.} \quad & \sum_{k=1}^K a_k \leq 1, \\ & \sum_{k=1}^K f_k^c \leq F^c, \\ & a_k \geq 0,\ f_k^c \geq 0, \quad \forall k \in \mathcal{K}, \\ & \tau_0 \geq 0. \end{aligned} \tag{49}$$

Instead of applying standard convex optimization tools to solve (49), we propose a low-complexity algorithm in the following.

Let ψ and γ denote the Lagrangian multipliers associated with constraints $\sum_{k=1}^K a_k \leq 1$ and $\sum_{k=1}^K f_k^c \leq F^c$, respectively. Then, we can obtain the optimal $\{\mathbf{a}^*, (\mathbf{f}^c)^*, \tau_0^*\}$ in closed-form as

$$a_k^* = \left(x_k^{i+1} + \frac{\rho_k^i - \psi^*}{c}\right)^+, \quad \forall k \in \mathcal{K}, \tag{50}$$

$$(f_k^c)^* = \left(y_k^{i+1} + \frac{\phi_k^i - \gamma^*}{c}\right)^+, \quad \forall k \in \mathcal{K}, \tag{51}$$

and

$$\tau_0^* = \left(\frac{\sum_{k=1}^K z_k^{i+1}}{K} + \frac{\sum_{k=1}^K \varphi_k^i}{cK}\right)^+, \tag{52}$$

where $(\cdot)^+ = \max\{\cdot, 0\}$. As a_k^* is non-increasing with $\psi^* \geq 0$, we can obtain the optimal ψ^* by bisection search over $\psi^* \in (0, \bar{\psi})$, where $\bar{\psi}$ is a sufficiently large value, until $\sum_{k=1}^K a_k^* = 1$ is satisfied. Likewise, $(f_k^c)^*$ is non-increasing with $\gamma^* \geq 0$. Therefore, we can obtain the optimal γ^* by bisection search over $\gamma^* \in (0, \bar{\gamma})$, where $\bar{\gamma}$ is a sufficiently large value. The pseudo-code of the algorithm is shown in Algorithm 2. Given an error tolerance ϵ_1 for ψ^*, the bisection search for ψ^* terminates within $\log_2(\frac{\bar{\psi}}{\epsilon_1})$ iterations. Likewise, the bisection search for γ^* terminates within $\log_2(\frac{\bar{\gamma}}{\epsilon_2})$ iterations, where ϵ_2 is the error tolerance for γ^*. Overall, the computational complexity of Step 2 is $O(K)$. In addition, we can calculate a_k^*'s and $(f_k^c)^*$'s for the K users in parallel under given dual variables ψ and γ, thus the computational time of Step 2 is constant when we conduct parallel computing.

Algorithm 2 Bisection Search Algorithm for Solving Problem (49)

```
input: local variables $\mathbf{u}^{i+1}$.
initialization: $\varepsilon_1 = 10^{-4}$; $\varepsilon_2 = 10^{-4}$; $\bar{\psi} \leftarrow$ sufficiently large value; $\bar{\gamma} \leftarrow$ sufficiently large value; $\psi^{\mathrm{UB}} = \bar{\psi}$, $\psi^{\mathrm{LB}} = 0$; $\gamma^{\mathrm{UB}} = \bar{\gamma}$, $\gamma^{\mathrm{LB}} = 0$;
repeat
    Set $\psi = \frac{\psi^{\mathrm{UB}} + \psi^{\mathrm{LB}}}{2}$, $\gamma = \frac{\gamma^{\mathrm{UB}} + \gamma^{\mathrm{LB}}}{2}$;
    for each $k \in \mathcal{K}$ do
        Calculate $a_k^*$ using (50);
        Calculate $(f_k^c)^*$ using (51);
    end for
    if $\sum_{k=1}^K a_k^* < 1$ then
        $\psi^{\mathrm{UB}} = \psi$
    else
        $\psi^{\mathrm{LB}} = \psi$
    end if
    if $\sum_{k=1}^K (f_k^c)^* < F^c$ then
        $\gamma^{\mathrm{UB}} = \gamma$
    else
        $\gamma^{\mathrm{LB}} = \gamma$
    end if
until $|\psi^{\mathrm{UB}} - \psi^{\mathrm{LB}}| < \varepsilon_1$ and $|\gamma^{\mathrm{UB}} - \gamma^{\mathrm{LB}}| < \varepsilon_2$.
Calculate $\tau_0^*$ using (52);
return $\{\mathbf{a}^*, (\mathbf{f}^c)^*, \tau_0^*\}$.
```

3) Step 3: Having obtained the local and global variables $\{\mathbf{u}^{i+1}, \mathbf{v}^{i+1}\}$, we update the multipliers $\boldsymbol{\theta}^i = \{\rho_k^i, \phi_k^i, \varphi_k^i\}$ as

$$\begin{aligned} \rho_k^{i+1} &= \rho_k^i + c\big(x_k^{i+1} - a_k^{i+1}\big), \quad \forall k \in \mathcal{K}, \\ \phi_k^{i+1} &= \phi_k^i + c\big(y_k^{i+1} - (f_k^c)^{i+1}\big), \quad \forall k \in \mathcal{K}, \\ \varphi_k^{i+1} &= \varphi_k^i + c\big(z_k^{i+1} - \tau_0^{i+1}\big), \quad \forall k \in \mathcal{K}. \end{aligned} \tag{53}$$

The computational complexity of Step 3 is also $O(K)$. Likewise, since (53) can be updated in parallel for the K users, the computational time of Step 3 is also constant when we perform parallel computing.

We repeat the above three sequential steps until a specified stopping criterion is met. In general, the stopping criterion is specified by two thresholds: namely, an absolute tolerance $\sum_{k=1}^K \left(|x_k^i - a_k^i| + |y_k^i - (f_k^c)^i| + |z_k^i - \tau_0^i|\right)$ and a relative tolerance $|\tau_0^i - \tau_0^{i-1}| + \sum_{k=1}^K \left(|a_k^i - a_k^{i-1}| + |(f_k^c)^i - (f_k^c)^{i-1}|\right)$ [12], [35]. The pseudo-code of the ADMM-based algorithm is presented in Algorithm 3. The convergence of the proposed

This article has been accepted for publication in IEEE Transactions on Wireless Communications. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TWC.2021.3081991

Algorithm 3 ADMM-Based Joint Service Placement and Resource Allocation Algorithm

```
initialization: $i = 0$; $\{\boldsymbol{\rho}^i, \boldsymbol{\phi}^i, \boldsymbol{\varphi}^i\} = 0$; $b_k^i = 1$, $a_k^i = \frac{1}{K}, \forall k \in \mathcal{K}$; $h_{\min} = \min\{h_k | k \in \mathcal{K}\}$, $\tau_0^i = \frac{S}{W_{\rm D} \log_2\left(1+\frac{p_0 h_{\min}}{N_0 W_{\rm D}}\right)}$; $c = 2$; $\sigma_1 = 0.0005K$;
repeat
    for each user $k \in \mathcal{K}$ do
        Update $\{b_k^{i+1}, (f_k^l)^{i+1}, x_k^{i+1}, y_k^{i+1}, z_k^{i+1}\}$ by solving (46) and (47);
    end for
    Update global variables $\{\mathbf{a}^{i+1}, (\mathbf{f}^c)^{i+1}, \tau_0^{i+1}\}$ by solving (49) with Algorithm 2;
    Update multipliers $\{\boldsymbol{\rho}^{i+1}, \boldsymbol{\phi}^{i+1}, \boldsymbol{\varphi}^{i+1}\}$ using (53);
    $i = i + 1$;
until $\sum_{k=1}^{K} \left(|x_k^i - a_k^i| + |y_k^i - (f_k^c)^i| + |z_k^i - \tau_0^i|\right) < 3\sigma_1$ and $|\tau_0^i - \tau_0^{i-1}| + \sum_{k=1}^{K} \left(|a_k^i - a_k^{i-1}| + |(f_k^c)^i - (f_k^c)^{i-1}|\right) < 2\sigma_1$;
return $\{\mathbf{b}^i, (\mathbf{f}^l)^i, \mathbf{a}^i, (\mathbf{f}^c)^i, \tau_0^i\}$ as an approximation solution to (P2).
```

algorithm is guaranteed as the dual problem (44) is convex in $\boldsymbol{\theta} = \{\boldsymbol{\rho}, \boldsymbol{\phi}, \boldsymbol{\varphi}\}$. Besides, the convergence of the algorithm is insensitive to the choice of the step size c [12], [36]. Without loss of generality, we simply set $c = 2$. As the complexity of each of the three steps is $O(K)$, the overall complexity of one ADMM iteration is $O(K)$. In addition, when conducting parallel computing, the computational time of the three steps are constant, thus the computational time of one ADMM iteration is constant. Therefore, the ADMM-based algorithm has excellent scalability in large-sized networks. Due to the non-convexity of (P2), a duality gap may exist and the ADMM-based algorithm may not exactly converge to the primal optimal solution of (P2). However, as we will show in the simulations, the gap between the obtained performance and the optimal one is extremely small.

V. Simulation Results

In this section, we evaluate the performance of the proposed algorithms via extensive simulations. Unless otherwise stated, we set the system uplink and downlink bandwidth as $W_{\rm U} = W_{\rm D} = 2$ MHz, and the noise power spectral density as $N_0 = -174$ dBm/Hz. We assume that the average channel gain $\bar{g}_k$ follows the free-space path loss model $\bar{g}_k = G\left(\frac{3\cdot 10^8}{4\pi f_0 d_k}\right)^{d_e}, \forall k \in \mathcal{K}$, where $G = 4.11$ denotes the antenna gain, $f_0 = 915$ MHz denotes the carrier frequency, d_k denotes the distance between user k and the AP, and $d_e = 3.4$ denotes the path loss exponent. The uplink channel g_k follows a Rayleigh fading channel model such that $g_k = \bar{g}_k \alpha$, where α denotes an independent exponential random variable with unit mean. Besides, we assume that the downlink channel h_k is correlated with the uplink channel g_k and the correlation coefficient is set as 0.75 [37]. Suppose that the K users are located at an equal distance of 150 meters from the AP. Without loss of generality, we assume that the weighting factors are identical for all the users, i.e., $\beta_k^T = \beta$ and $\beta_k^E = 1 - \beta, \forall k \in \mathcal{K}$. Besides, we set equal computing energy efficiency coefficient $\kappa_k = 10^{-28}, \forall k \in \mathcal{K}$. We also assume that the computing workload is proportional to the data size, i.e., $L_k = C I_k$ [38], where C denotes the number of CPU cycles for computing one bit of task data. Unless otherwise stated, we set $K = 10$, $S = 32$ Mbits, $C = 1000$, $F^c = 20$ GHz, $p_0 = 1$ W, $F_k = 1$ GHz, $p_k = 0.1$ W, and $p_k^r = 0.01$ W [39], $\forall k \in \mathcal{K}$. In addition, all curves in the figures are plotted based on the average of 100 independent simulation runs, each corresponding to an independent Rayleigh fading realization.

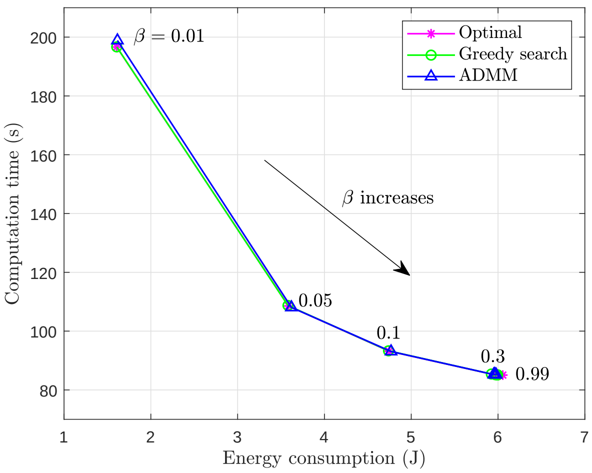


Fig. 2: The optimal energy-delay tradeoff under different values of β.

For performance comparison, we consider the following three representative benchmarks:

1) Optimal: the global optimal solution to (P1).
2) Independent optimization: each user minimizes its own TEC independently. Specifically, the edge CPU frequency, the uplink bandwidth and downlink bandwidth are equally allocated to all the users, and each user determines whether to download the program from the AP via unicast.
3) All edge computing: all the users offload their tasks to the AP for edge computing.

A. TEC Performance Evaluation under Equal Task Size

We first consider a special case where $I_k = I$ for all the users and the K users differ only by the channel gains g_k's and h_k's. We set $I = 8$ Mbits unless otherwise stated. In Fig. 2, we study the performance tradeoff between the total computation time and the total energy consumption when the weighting parameter β varies. We can see that the performance tradeoff curves achieved by the proposed greedy search and ADMM-based algorithms are close to the optimum. Besides, we observe that as β increases, the total computation time decreases and the total energy consumption increases. In particular, the total computation time decreases quickly with β when β is small and converges to a constant when $\beta \geq 0.3$. In the following simulations, we set $\beta = 0.1$ without loss of generality.

In Fig. 3, we study some interesting properties of the optimal solution to Problem (P1). In particular, we sort g_k's in

© 2021 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See https://www.ieee.org/publications/rights/index.html for more information.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TWC.2021.3081991, IEEE Transactions on Wireless Communications

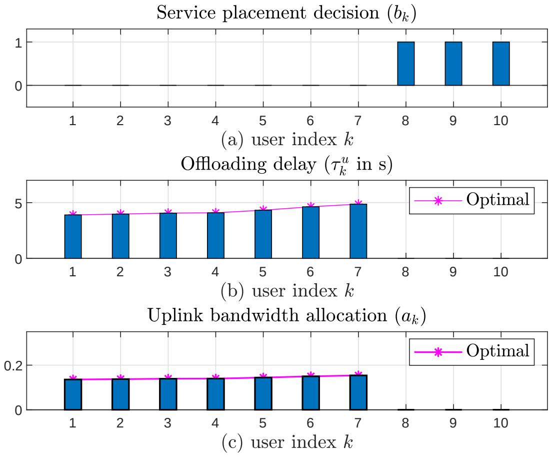


Fig. 3: Optimal solution.

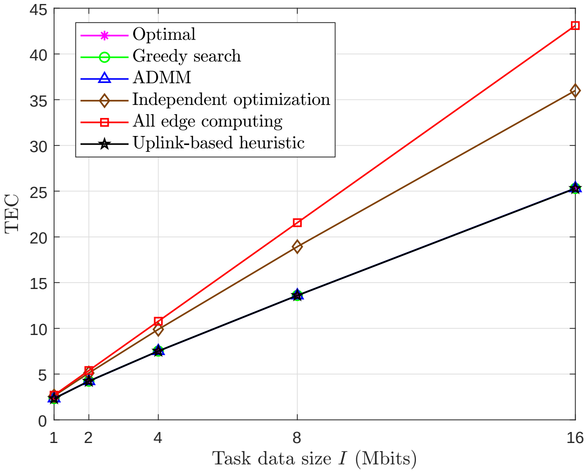


Fig. 5: Total TEC versus the task data size I.

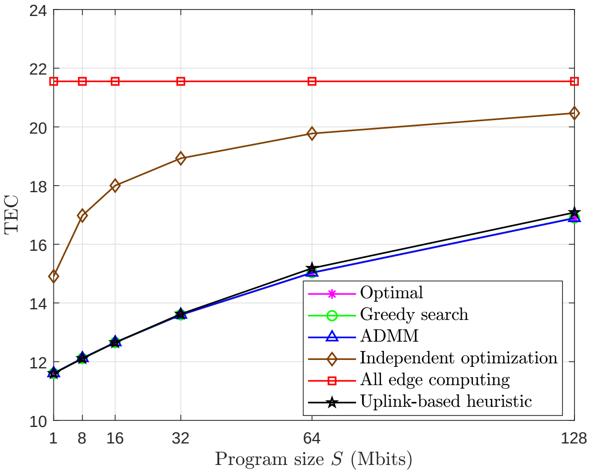


Fig. 4: Total TEC versus the program size S.

descending order such that the uplink channel gain decreases from g_1 to g_{10}. We can see from Fig. 3(a) that the three users with the smallest channel gains prefer to download the service program and conduct local computing. The other users with better channels offload their tasks for edge computing. Besides, for the users in $\mathcal{K}_0$, a smaller g_k leads to a longer offloading delay τ_k^u, as shown in Fig. 3(b). The above observations verify our analysis in Section III-C. In Fig. 3(c), we observe that the optimal uplink bandwidth allocation a_k increases when g_k decreases. This indicates that more uplink bandwidth should be allocated to the users with worse channels to achieve the minimum total offloading delay. Therefore, in the following simulations in Fig. 4-6, we evaluate the performance achieved by the uplink-based heuristic scheme devised in Section III-C. In this scheme, $\mathcal{K}_1$ is obtained by selecting the users in the ascending order of g_k.

In Fig. 4, we compare the TEC performance achieved by different schemes when the program size S varies. Besides, we present the TEC performance comparison when the task data size I varies in Fig. 5. From both figures, we observe that the TEC performance achieved by the proposed greedy search and ADMM-based methods are extremely close to the optimal scheme. The three curves are on top of each other. As expected, the uplink-based heuristic scheme also achieves a close-to-optimal performance. This further confirms that users with worse uplink channels tend to conduct local computing. In addition, the proposed algorithms achieve lower total TEC than the other representative benchmarks for all values of S and I. This demonstrates the advantage of jointly optimizing the service placement, computation offloading, and resource allocation for all the users. The total TEC increases with S and I for all the schemes except that the program size S has no impact on the total TEC of the all-edge-computing scheme. This is because the users in the all-edge-computing scheme offload all the tasks to the edge server without downloading the service program. Besides, we observe that the proposed algorithms tend to converge to the all-edge-computing scheme when S is large or I is small, e.g., $I \leq 1$ Mbits. This indicates that the users tend to offload all the computation tasks to the AP when the overhead of downloading the program outweighs the gain of local computing or the offloading latency is low.

In Fig. 6, we further study the impact of the computing workload C on the TEC performance when C varies from 1 to 2500. Likewise, it can be seen that the proposed algorithms and the uplink-based heuristic scheme both achieve a close-to-optimal performance for all values of C. Besides, the proposed algorithms significantly outperform the other two benchmark schemes when C is small, but all the schemes converge to the all-edge-computing scheme as C increases. This is because when the tasks become computationally intensive, the users tend to offload the tasks to the edge server that has much more powerful computing capability.

B. TEC Performance Evaluation under Heterogeneous Task Size

In this subsection, we evaluate the performance of the proposed algorithms in a heterogenous case, where the users have different task data sizes I_k's. In particular, the task data size of each user follows a uniform distribution with $I_k \in [1, 12]$ Mbits.

© 2021 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See https://www.ieee.org/publications/rights/index.html for more information.

This article has been accepted for publication in IEEE Transactions on Wireless Communications. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TWC.2021.3081991
This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TWC.2021.3081991, IEEE Transactions on Wireless Communications
12

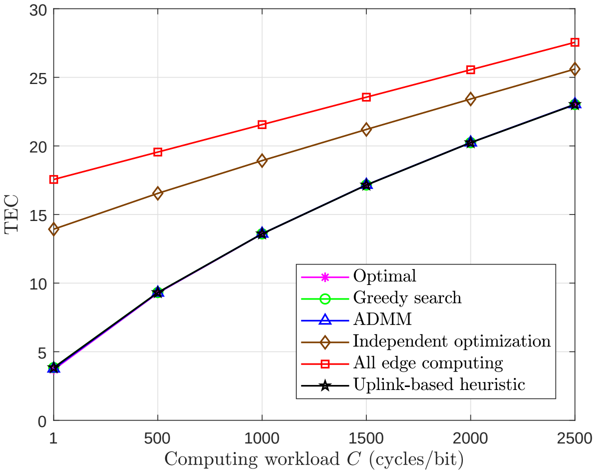


Fig. 6: Total TEC versus the computing workload C.

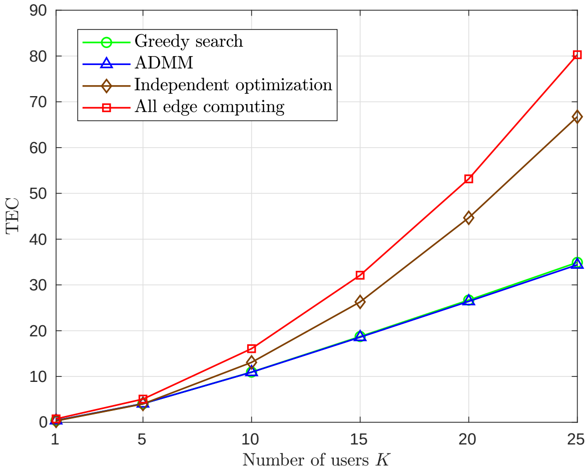


Fig. 8: Total TEC versus the number of users K.

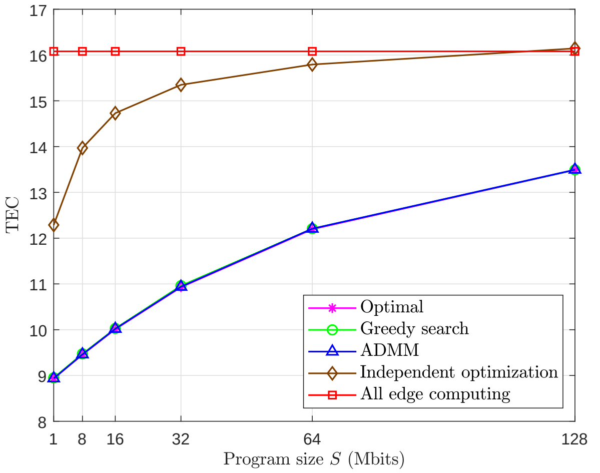


Fig. 7: Total TEC versus the program size S.

In Fig. 7, we plot the TEC performance achieved by different schemes when the program size S increases. We observe that the performance comparison among the schemes is almost consistent with Fig. 4. The proposed ADMM-based and greedy search algorithms and the optimal scheme have almost identical performance. When S increases, the high downloading overhead encourages task offloading, and all the schemes converge to the all-edge-computing scheme. We can also see that the independent optimization scheme outperforms the all-edge-computing scheme when $S \leq 64$ Mbits. However, it becomes slightly worse than the all-edge-computing scheme when $S = 128$ Mbits, due to the performance loss resulted from the equal resource allocation.

In Fig. 8, we plot the TEC performance when the number of users K varies from 1 to 25. Here, we do not plot the global optimal performance due to the prohibitively high computational complexity to obtain the global optimal solutions when K is large. We observe that the proposed ADMM-based and greedy search algorithms have almost identical performance and outperform the benchmark schemes. The all-edge-computing scheme is the worst for all K since the local computing capabilities are not utilized. When $K = 1$, the independent optimization scheme performs as well as the proposed algorithms because the single user can occupy all the resources in these three schemes. As K increases, the proposed algorithms achieve increasingly lower total TEC than the two benchmarks. The reason is that the edge CPU frequency and uplink/downlink bandwidth allocated to each user becomes smaller as K increases in these two benchmark schemes. Thus, users need more time to download the program or complete edge computing.

C. Evaluation of Computational Complexity

In Fig. 9, we investigate the computational complexity of the proposed greedy search and ADMM-based algorithms under the same setting of Fig. 8. Here, we plot the average number of iterations consumed by the proposed greedy search method and ADMM-based method, respectively. Specifically, for each iteration in the greedy search method, the service placement decision is fixed, and Algorithm 1 is executed to solve the corresponding optimization problem in (P1). In Fig. 9(a), we observe that the number of iterations in the greedy search method increases with K at a polynomial growth rate. We also fit the number of iterations in the greedy search method to a quadratic curve, where the R-square value is 0.999962. Since each iteration corresponds to an execution of Algorithm 1, we can conclude that the number of executions of Algorithm 1 in the greedy search method scales as $O(K^2)$. Moreover, because the computational complexity of Algorithm 1 is upper bounded by $O(K^3)$, the overall computational complexity of the greedy search method is upper bounded by $O(K^5)$, which verifies our analysis in Section III-B. In Fig. 9(b), we observe that the number of iterations consumed by the ADMM-based algorithm increases with K when K is small, i.e., $K \leq 10$. This is because the edge CPU frequency and uplink bandwidth allocation among the users and the service placement decisions become more flexible when K increases. Thus, the ADMM-based algorithm needs more iterations to optimize the solution. When K further increases, the ADMM-based algorithm takes almost a constant number of iterations,

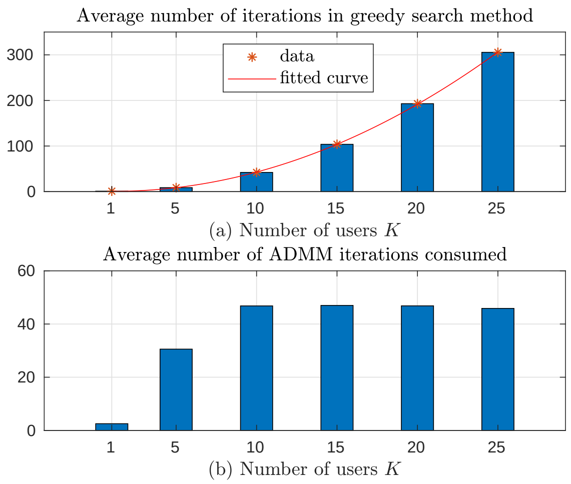


Fig. 9: Computational complexity comparison between the greedy search and ADMM-based methods when K varies.

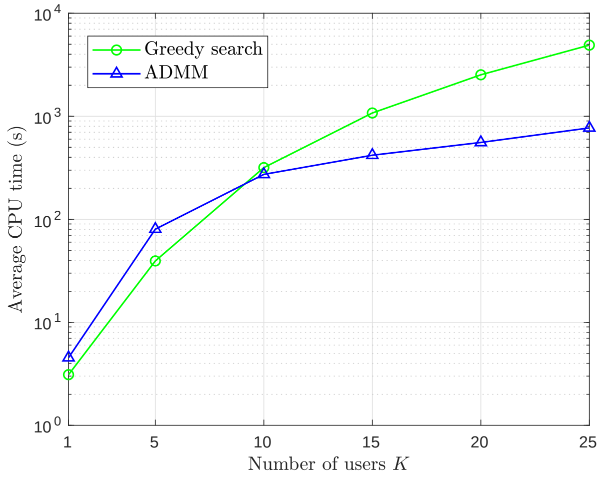


Fig. 10: Average CPU time comparison between the greedy search and ADMM-based methods when K varies.

i.e., the number of ADMM iterations scales as $O(1)$. This shows that the ADMM-based algorithm can converge within a few tens of iterations [35]. Besides, as presented in Algorithm 3, we set the convergence threshold as $\sigma_1 = 0.0005K$, which increases with K and facilitates the convergence. Furthermore, since the complexity of each iteration is $O(K)$, the overall computational complexity of the ADMM-based algorithm is $O(K)$. The above results show that the computational complexity of the ADMM-based algorithm increases much more slowly than the greedy search algorithm. In addition, we compare the average CPU time of the proposed greedy search and ADMM-based methods in Fig. 10. In particular, we conduct parallel computing to implement the ADMM-based algorithm. Notice that the computational time of the ADMM-based algorithm still increases slightly with K when $K \geq 10$ since the increase of data dimension incurs additional time for processing the data. We see that the average CPU time of the ADMM-based method is longer than the greedy search method when K is small (e.g., $K \leq 8$) but increases much more slowly than the greedy search method as K increases. When K is large, the average CPU time of the greedy search method is much longer than the ADMM-based method. This implies that the ADMM-based algorithm is more scalable in large-sized networks.

VI. Conclusions and Future Work

In this paper, we studied the AI service placement problem for achieving EI in a multi-user MEC system, where the edge server selectively places the AI service program at a subset of users to make use of their local computing capabilities. To minimize the computation time and energy consumption of all the users, we formulated the problem as a joint optimization of service placement, computational and radio resource allocation (on local CPU frequencies, uplink bandwidth and edge CPU frequency). We derived analytical expressions to efficiently calculate the optimal resource allocations decisions for a given service placement decision, based on which we applied search-based methods to optimize the service placement decision. We further proposed an ADMM-based algorithm to avoid high-dimensional search in large-sized networks by decomposing the original problem into parallel and tractable subproblems, one for each user. Extensive simulations showed that the proposed algorithms can achieve a close-to-optimal performance and significantly outperform various benchmark methods. By exploiting the idle local computing power, the proposed schemes significantly reduce the total computation delay and energy consumption compared with those that offload all tasks to the edge server. In particular, a larger program size leads to higher downloading delay and encourages the users to offload more tasks for edge computing. Besides, the performance advantage of the proposed algorithms becomes increasingly significant as the task data size, the weighting factor of computation time and the number of users increase, but becomes increasingly marginal as the computing workload increases.

For a special case where the users differ only by the wireless channel gains, we observed an interesting phenomenon that the edge server tends to place the service program at the users that suffer poor channel conditions, so that the limited spectrum can be used more efficiently by the other users to offload their computation tasks. In this case, we designed a heuristic scheme based on the ascending order of uplink channel gains. Simulation results showed that the uplink-based heuristic scheme achieves a close-to-optimal performance under various system setups in such a homogeneous special case.

For practical implementation, the search-based algorithms require simple calculations for the optimal resource allocation decisions with the derived analytical expressions. However, the computational complexity of ADMM-based algorithm increases linearly with the network size, which has a much smaller growth rate than the search-based algorithms. Besides, with parallel computing, the computational time of the ADMM-based algorithm increases much more slowly with the network size. Therefore, the search-based algorithms are preferred when network size is small and the proposed ADMM-based algorithm is much more scalable in large-sized networks. The proposed optimization framework in this paper can be leveraged to achieve communication-efficient edge-device inference, which is an important part of the vision on 6G communications, i.e., to support ubiquitous AI services with limited communication, computation, hardware, and energy resources [40].

Finally, we conclude the paper with some interesting future directions. First, it is interesting to consider service placement and resource allocation in a general multi-AP MEC system, where multiple APs collaboratively serve users to provide better and more reliable system computing performance. In this case, different communication protocols, e.g., orthogonal frequency-division multiple access (OFDMA), nonorthogonal multiple access (NOMA), and time division multiple access (TDMA), can be utilized to address the new challenges of AP-user association and interference management. Second, it is also promising to extend the single-service setup to a multi-service one. The system resources can be shared by all services or split into separate parts through virtualization techniques, one for each service. Meanwhile, the edge servers can achieve load balance by adaptively allocating the computing resources. Moreover, we assumed in this paper that the channel conditions and computation requirements are known in advance and studied an offline optimization problem. In practice, users are dynamic and may request the AI service at different time and with different frequencies. Hence, an online design is needed to apply to time-varying channel conditions and computation requirements.

References

[1] Z. Lin, S. Bi, and Y. J. Zhang, “Optimizing AI service placement and computation offloading in mobile edge intelligence systems,” in *Proc. IEEE GLOBECOM*, Dec. 2020, pp. 1–7.

[2] J. Park, S. Samarakoon, M. Bennis, and M. Debbah, “Wireless network intelligence at the edge,” *Proc. IEEE*, vol. 107, no. 11, pp. 2204–2239, Nov. 2019.

[3] Z. Zhou, X. Chen, E. Li, L. Zeng, K. Luo, and J. Zhang, “Edge intelligence: Paving the last mile of artificial intelligence with edge computing,” *Proc. IEEE*, vol. 107, no. 8, pp. 1738–1762, Aug. 2019.

[4] X. Wang, Y. Han, V. C. M. Leung, D. Niyato, X. Yan, and X. Chen, “Convergence of edge computing and deep learning: A comprehensive survey,” *IEEE Commun. Surveys Tuts.*, vol. 22, no. 2, pp. 869–904, 2nd Quart. 2020.

[5] Y. Shi, K. Yang, T. Jiang, J. Zhang, and K. B. Letaief, “Communication-efficient edge AI: Algorithms and systems,” *IEEE Commun. Surveys Tuts.*, vol. 22, no. 4, pp. 2167–2191, 4th Quart. 2020.

[6] 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.

[7] W. Shi, J. Cao, Q. Zhang, Y. Li, and L. Xu, “Edge computing: Vision and challenges,” *IEEE Internet Things J.*, vol. 3, no. 5, pp. 637–646, Oct. 2016.

[8] W. Zhang, Y. Wen, K. Guan, D. Kilper, H. Luo, and D. O. Wu, “Energy-optimal mobile cloud computing under stochastic wireless channel,” *IEEE Trans. Wireless Commun.*, vol. 12, no. 9, pp. 4569–4581, Sep. 2013.

[9] Y. Wang, M. Sheng, X. Wang, L. Wang, and J. Li, “Mobile-edge computing: Partial computation offloading using dynamic voltage scaling,” *IEEE Trans. Commun.*, vol. 64, no. 10, pp. 4268–4282, Oct. 2016.

[10] M. Chen, B. Liang, and M. Dong, “Joint offloading decision and resource allocation for multi-user multi-task mobile cloud,” in *Proc. IEEE ICC*, May 2016, pp. 1–6.

[11] C. You, K. Huang, H. Chae, and B. Kim, “Energy-efficient resource allocation for mobile-edge computation offloading,” *IEEE Trans. Wireless Commun.*, vol. 16, no. 3, pp. 1397–1411, Mar. 2017.

[12] 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.

[13] L. Huang, S. Bi, and Y. J. Zhang, “Deep reinforcement learning for online computation offloading in wireless powered mobile-edge computing networks,” *IEEE Trans. Mobile Comput.*, Jul. 2019.

[14] S. Bi, L. Huang, H. Wang, and Y. J. Zhang, “Lyapunov-guided deep reinforcement learning for stable online computation offloading in mobile-edge computing networks.” [Online]. Available: https://arxiv.org/abs/2010.01370.

[15] 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.

[16] J. Yan, S. Bi, Y. J. Zhang, and M. Tao, “Optimal task offloading and resource allocation in mobile-edge computing with inter-user task dependency,” *IEEE Trans. Wireless Commun.*, vol. 19, no. 1, pp. 235–250, Jan. 2020.

[17] A. Tsymbal, “The problem of concept drift: Definitions and related work,” Trinity College, Dublin, Ireland, Tech. Rep. TCD-CS-2004-15, 2004.

[18] J. Gama, I. Žliobaitė, A. Bifet, M. Pechenizkiy, and A. Bouchachia, “A survey on concept drift adaptation,” *ACM Comput. Surv.*, vol. 46, no. 4, Mar. 2014.

[19] G. Ditzler, M. Roveri, C. Alippi, and R. Polikar, “Learning in non-stationary environments: A survey,” *IEEE Computational Intelligence Magazine*, vol. 10, no. 4, pp. 12–25, 2015.

[20] M. S. H. Abad, E. Ozfatura, D. Gunduz, and O. Ercetin, “Hierarchical federated learning across heterogeneous cellular networks.” [Online]. Available: https://arxiv.org/abs/1909.02362.

[21] X. Cai, X. Mo, J. Chen, and J. Xu, “D2D-enabled data sharing for distributed machine learning at wireless network edge,” *IEEE Wireless Commun. Lett.*, vol. 9, no. 9, pp. 1457–1461, 2020.

[22] K. Poularakis, J. Llorca, A. M. Tulino, I. Taylor, and L. Tassiulas, “Joint service placement and request routing in multi-cell mobile edge computing networks,” in *Proc. IEEE INFOCOM*, Apr. 2019, pp. 10–18.

[23] L. Chen, J. Xu, S. Ren, and P. Zhou, “Spatio-temporal edge service placement: A bandit learning approach,” *IEEE Trans. Wireless Commun.*, vol. 17, no. 12, pp. 8388–8401, Dec. 2018.

[24] J. Xu, L. Chen, and P. Zhou, “Joint service caching and task offloading for mobile edge computing in dense networks,” in *Proc. IEEE INFOCOM*, Apr. 2018, pp. 207–215.

[25] L. Chen, C. Shen, P. Zhou, and J. Xu, “Collaborative service placement for edge computing in dense small cell networks,” *IEEE Trans. Mobile Comput.*, pp. 1–1, 2019.

[26] S. Bi, L. Huang, and Y. J. Zhang, “Joint optimization of service caching placement and computation offloading in mobile edge computing systems,” *IEEE Trans. Wireless Commun.*, vol. 19, no. 7, pp. 4947–4963, Jul. 2020.

[27] X. Dai, I. Spasić, B. Meyer, S. Chapman, and F. Andres, “Machine learning on mobile: An on-device inference app for skin cancer detection,” in *Proc. 4th International Conference on Fog and Mobile Edge Computing (FMEC)*, 2019, pp. 301–305.

[28] A. Pacheco, E. Flores, R. Sánchez, and S. Almanza-García, “Smart classrooms aided by deep neural networks inference on mobile devices,” in *Proc. IEEE International Conference on Electro/Information Technology (EIT)*, 2018, pp. 0605–0609.

[29] A. Hall and U. Ramachandran, “An execution model for serverless functions at the edge,” in *Proc. ACM IoTDI*, Apr. 2019, pp. 225–236.

[30] 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.

[31] S. Boyd and C. Barratt, *Linear Controller Design: Limits of Performance*. Upper Saddle River, NJ, USA: Prentice-Hall, 1991.

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

[33] L. P. Qian, Y. J. Zhang, J. Huang, and Y. Wu, “Demand response management via real-time electricity price control in smart grids,” *IEEE J. Sel. Areas Commun.*, vol. 31, no. 7, pp. 1268–1280, 2013.

[34] J. Kennedy and R. Eberhart, “Particle swarm optimization,” in *Proc. IEEE Int. Conf. Neural Netw.*, vol. 4, 1995, pp. 1942–1948 vol.4.

[35] 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. Learning*, vol. 3, no. 1, pp. 1–122, Jan. 2011.

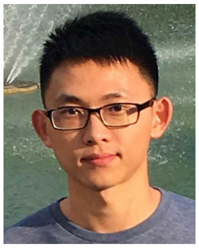
[36] E. Ghadimi, A. Teixeira, I. Shames, and M. Johansson, “Optimal parameter selection for the alternating direction method of multipliers (ADMM): Quadratic problems,” *IEEE Trans. Autom. Control*, vol. 60, no. 3, pp. 644–658, Mar. 2015.

[37] E. Perahia and D. C. Cox, “Shadow fading correlation between uplink and downlink,” in *Proc. IEEE VTS 53rd Veh. Technol. Conf.*, vol. 1, 2001, pp. 308–312 vol.1.

[38] T. Hao, J. Zhan, K. Hwang, W. Gao, and X. Wen, “AI-oriented medical workload allocation for hierarchical cloud/edge/device computing.” [Online]. Available: https://arxiv.org/abs/2002.03493.

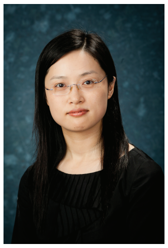
[39] Z. Zhou, S. Zhou, J. Cui, and S. Cui, “Energy-efficient cooperative communication based on power control and selective single-relay in wireless sensor networks,” *IEEE Trans. Wireless Commun.*, vol. 7, no. 8, pp. 3066–3078, 2008.

[40] K. B. Letaief, W. Chen, Y. Shi, J. Zhang, and Y. J. Zhang, “The roadmap to 6G: AI empowered wireless networks,” *IEEE Commun. Mag.*, vol. 57, no. 8, pp. 84–90, 2019.



Zehong Lin (S’17) received the B.Eng. degree from the School of Electronic and Information Engineering, South China University of Technology, Guangzhou, China, in 2017. He is currently pursuing the Ph.D. degree in information engineering with The Chinese University of Hong Kong, Hong Kong. His research interests include wireless communications and networking, edge intelligence, distributed machine learning and optimization, and mobile edge computing.

Suzhi Bi (S’10-M’14-SM’19) received the B.Eng. degree in communications engineering from Zhejiang University in 2009, and the Ph.D. degree in information engineering from The Chinese University of Hong Kong in 2013. From 2013 to 2015, he was a post-doctoral research fellow with the Department of Electrical and Computer Engineering, National University of Singapore. Since 2015, he has been with the College of Electronics and Information Engineering, Shenzhen University, China, where he is currently an Associate Professor. His research interests mainly involve in the optimizations in wireless information and power transfer, mobile computing, and smart power grid communications. He received the IEEE ComSoc Asia-Pacific Outstanding Young Researcher Award (2019), Guangdong Province “Pearl River Young Scholar” award (2018), two times Shenzhen University Outstanding Young Faculty award, and was a corecipient of the IEEE SmartGridComm 2013 Best Paper Award. He is an Associate Editor of IEEE Wireless Communications Letters.



Ying-Jun Angela Zhang (S’00-M’05-SM’10-F’20) is a Professor at Department of Information Engineering, The Chinese University of Hong Kong. Her research interests are in optimization and learning in wireless communication systems and smart power grids. She is now an Associate Editor-in-Chief of IEEE Open Journals of the Communication Society and a Member of IEEE ComSoc Fellow Evaluation Committee. Previously, she has served as the Chair of the Executive Editorial Committee of IEEE Transactions on Wireless Communications and a Founding Chair of IEEE ComSoc Smart Grid Communications Technical Committee. She is the co-recipient of the 2011 Marconi Prize Paper Award in Wireless Communications, the 2013 SmartGridComm Best Paper Award, and the 2014 IEEE ComSoc Asia Pacific Board Outstanding Paper Award. She won the 2006 Hong Kong Young Scientist Award as the only winner from engineering science. She is a Fellow of IEEE and a Distinguished Lecturer of IEEE ComSoc.