Joint Task Assignment and Resource Allocation for D2D-Enabled Mobile-Edge Computing

Hong Xing, Liang Liu, Jie Xu, and Arumugam Nallanathan

Abstract

With the proliferation of computation-extensive and latency-critical applications in the 5G and beyond networks, mobile-edge computing (MEC) or fog computing, which provides cloud-clone computation and/or storage capabilities at the network edge, is envisioned to reduce computation latency as well as conserve energy for wireless devices (WDs). This paper studies a novel device-to-device (D2D)-enabled multi-helper MEC system, in which a local user offloads its computation tasks to multiple helpers for cooperative computation. We assume a time division multiple access (TDMA) transmission protocol, under which the local user offloads the tasks to multiple helpers and downloads the results from them over orthogonal pre-scheduled time slots. Under this setup, we minimize the computation latency by optimizing the local user's task assignment jointly with the time and rate for task offloading and results downloading, as well as the computation frequency for task execution, subject to individual energy and computation capacity constraints at the local user and the helpers. However, the formulated problem is a mixed-integer non-linear program (MINLP) that is difficult to solve. To tackle this challenge, we propose an efficient algorithm by first relaxing the original problem into a convex one, and then constructing suboptimal task assignment based on the obtained optimal solution. Next, we consider a benchmark scheme that endows the WDs with their maximum computation capacities. To further reduce the implementation complexity, we also develop a heuristic scheme based on the greedy task assignment. Finally, numerical results validate the effectiveness of our proposed algorithm, as compared against the heuristic scheme and other benchmark ones without joint optimization of radio and computation resources or without task assignment design.

Part of this paper was accepted by IEEE International Conference on Communications (ICC), Kansas City, MO, USA, May, 2018 [1].

H. Xing, and A. Nallanathan are with the School of Electronic Engineering and Computer Science, Queen Mary University of London, London, E1 4NS, U.K. (e-mails: h.xing@qmul.ac.uk, nallanathan@ieee.org).

L. Liu is with the Department of Electrical and Computer Engineering, National University of Singapore, Singapore (e-mail: eleliu@nus.edu.sg).

J. Xu is with the School of Information Engineering, Guangdong University of Technology, Guangzhou, China (e-mail: jiexu@gdut.edu.cn). J. Xu is also the corresponding author of this paper.

Index Terms

Mobile-edge computing (MEC), fog computing, computation offloading, task assignment, resource allocation.

I. Introduction

It is envisioned that by the year of 2020, around 50 billions of interconnected Internet of things (IoT) devices will surge in wireless networks, featuring new applications such as video stream analysis, augmented reality, and autonomous driving. The unprecedented growth of these latency-critical services requires extensive real-time computation, which, however, is hardly affordable by conventional mobile-cloud computing (MCC) systems that usually deploy mobile-cloud servers far away from end users [2]. Compared with MCC, mobile-edge computing (MEC) endows cloud-computing capabilities within the radio access network (RAN) such that the users can offload the computation tasks to the edge server in their proximity for remote execution and then collect the results with enhanced energy efficiency and reduced latency (see [3] and the references therein). Meanwhile, in industry, technical specifications and standard regulations are also being developed by, e.g., the European Telecommunications Standard Institute (ETSI) [4], Cisco [5], and the 3rd Generation Partnership Project (3GPP) [6].

Various efforts have been dedicated to addressing technical challenges against different computation offloading models. In general, computation task models fall into two major categories: partial offloading and binary offloading. Tasks that cannot be partitioned for execution belong to the former category, while the latter category supports fine-grained partition of the tasks. On the other hand, there are various types of MEC system architectures, such as single-user single-server [7–9], multi-user single-server [10–18], as well as single/multi-user multi-server [19–22] To achieve satisfactory trade-offs between energy consumption and computing latency under different setups, it is critical to jointly optimize the radio and computation resources. Under a single-user single-server setup, [7] jointly optimized the task offloading scheduling and the transmitting power allocation to minimize the weighted sum of the execution delay and the system energy consumption. The optimal communication strategy as well as computational load distribution was obtained in [8] in terms of the trade-offs of energy consumption versus latency for a multi-antenna single-user single-server system. In a single-user single-helper single-server system, [9] proposed a novel user cooperation in both computation and communication to

improve the energy efficiency for latency-constrained computation. In a multiple-input multiple-output (MIMO) multicell system, [10] jointly optimized the precoding matrices of multiple wireless devices (WDs) and the CPU frequency assigned to each device with fixed binary offloading decisions, in order to minimize the overall users' energy consumption. A multi-user MCC system with a computing access point (CAP), which can serve as both a network gateway connecting to the cloud and an edge cloudlet, was studied in [11] to find the binary offloading decisions. Joint optimization of (partial) offloaded data length and offloading time/subcarrier allocation was studied in a multi-user single-server MEC system based on time-division multiple access (TDMA) and orthogonal frequency-division multiple access (OFDMA), respectively, in [13]. Furthermore, both binary and partial offloading were considered in a multi-user single-server MEC system exploiting multi-antenna non-orthogonal multiple access (NOMA) in [14]. [15] and [16] leveraged the inherent collaborative properties of augmented reality (AR) applications across multiple WDs to minimize the sum users' energy expenditure.

In the above works, the edge servers are mostly assumed to be one integrated server. However, considering multi-user multi-server systems where more than one CAPs are distributed over the network, it becomes non-trivial to answer the fundamental questions such as how to distribute the tasks among multiple servers, and how to schedule multiple tasks on one single server [19–23]. Computational resource sharing among WDs with intermittent connectivity was considered as early as in [19], in which a greedy task dissemination algorithm was developed to minimize task completion time. A polynomial-time task assignment scheme for tasks with inter-dependency was developed in [21] to achieve guaranteed latency-energy trade-offs. However, this line of works often assumed communication conditions (e.g., transmission rate and multiple access schemes) and computation capacities (e.g., execution rate) to be fixed or estimable by some service profiling, but ignored the potential performance improvement brought by dynamic management over such resources (e.g., transmitting power, bandwidth, and computation frequency).

In this paper, we study a device-to-device (D2D)-enabled multi-helper MEC system, in which a local user offloads a number of independent computation tasks to multiple nearby WDs serving as helpers, such as smart wearable devices, cellphones, tablets, and laptops, via direct D2D links. The motivation for us to study efficient task assignment and wireless resource allocation algorithms to facilitate D2D-enabled MEC are two-fold. First, as WDs constantly improve their capabilities (e.g., battery capacity, processing speed, spectral efficiency), heterogeneity of radio and computation resources among WDs can be exploited to support various demanding services

while achieving mutual benefits [20]. Second, with the proliferation of WDs, some of them may be prohibited from accessing to BSs. In this case, they can entrust some virtual central controller managed by network operators to collect their task features, and assist in D2D-enabled MEC by pooling and sharing their resources among each other. Assuming that the tasks cannot be further partitioned, we consider a TDMA communication protocol, under which the local user offloads tasks to different helpers and downloads computation results from them over orthogonal pre-scheduled time slots. We aim for minimizing the overall latency subject to individual energy and computation capacity constraints at the local user and the helpers.

A task offloading framework, *D2D fogging*, was proposed in [24], where WDs could share the computation and communication resources among each other via the assistance of network operators, and dynamic task offloading decisions were made to minimize the time-averaged total energy consumption. From the perspective of system model, ours and [24] employ different communications protocol. We adopt a three-phase TDMA protocol, under which task assignment becomes a very crucial issue because the tasks offloaded to different helpers are in couple with each other, thus making it a general NP-hard problem. By contrast, under OFDMA, all tasks can be executed independent of each other subject to a common deadline constraint in [24]. Furthermore, as each WD is assumed to be assigned with more than one task in this paper, the efficient matching-based algorithm that forms the building block of the online task assignment scheme in [24] cannot be applied any more. It is also worth noting that the major difference between this paper and the earlier conference version [1] is that instead of fixing the processing capacities, we consider controllable computing frequency exploiting dynamic voltage and frequency scaling (DVFS) [25] in this paper to achieve improved overall latency. To our best knowledge, this paper is among the earliest works investigating TDMA-based joint binary task offloading and wireless resource allocation for multiple tasks in a single-user multi-helper MEC system.

The contributions of our paper are summarized as follows. 1) First, we transform the computation latency minimization problem with complex objective function into an equivalent one by investigating the optimal structure of the solution. 2) We jointly optimize the tasks assignment, the task offloading time/rate, the local and remote task execution time/computation frequency, and the results downloading time/rate, subject to individual energy and computation frequency constraints at the local user and the helpers. However, since the formulated problem is a mixed-integer non-linear program (MINLP) that is difficult to solve in general, we propose an efficient

algorithm to obtain a high-quality sub-optimal solution by relaxing the binary task assignment variables into continuous ones, and then constructing suboptimal task assignment based on the optimal solution to the relaxed problem. 3) Next, to further reduce the implementation complexity, we also design a greedy task assignment based joint optimization scheme. 4) Finally, we evaluate the performance of the proposed convex-relaxation-based algorithm, as compared against the heuristic one and other benchmark schemes without joint optimization of radio and computation resources or without task assignment design.

The remainder of this paper is organized as follows. The system model is presented in Section II. The joint task assignment and wireless resource allocation problem is formulated in Section III. The convex-relaxation-based joint task assignment and wireless resource allocation algorithm is proposed in Section IV, while two low-complexity benchmark schemes are proposed in Section V. Numerical results are provided in Section VI, with concluding remarks drawn in Section VII.

Notation—We use upper-case boldface letters for matrices and lower-case boldface letters for vectors. "Independent and identically distributed" is simplified as $i.i.d.$, and $\triangleq$ means "denoted by". A circularly symmetric complex Gaussian (CSCG) distributed random variable (RV) y with mean u and variance σ^2 is denoted by $y \sim \mathcal{CN}(u, \sigma^2)$. A continuous RV z uniformly distributed over $[a, b]$ is denoted by $z \sim \mathcal{U}[a, b]$. $\mathbb{R}^{M\times N}$ and $\mathbb{R}^N$ stand for the sets of real matrices of dimension $M \times N$ and real vectors of dimension N, respectively. The cardinality of a set is represented by $|\cdot|$. In addition, $\mathcal{P}(N)$ means an N-degree polynomial.

II. System Model

We consider a multi-user cooperative MEC system that consists of one local user, and K nearby helpers denoted by the set $\mathcal{K} = \{1, \ldots, K\}$, all equipped with single antenna. For convenience, we define the local user as the $(K + 1)$-th WD. Suppose that the local user has L independent tasks[1] to be executed, denoted by the set $\mathcal{L} = \{1, \ldots, L\}$, and the input/output data length of each task is denoted by T_l/R_l in bits. In the considered MEC system, each task can be either computed locally, or offloaded to one of the K helpers for remote execution. Let $\mathbf{\Pi} \in \mathbb{R}^{L\times(K+1)}$

[1]In this paper, we do not consider interdependency among tasks enabling data transmission from one helper to another as in [19, 21], since even under this simple task model, it becomes clear later that task assignment among multiple D2D helpers over pre-scheduled TDMA slots has already been very demanding to solve.

denote the task assignment matrix, whose (l,k)-th entry, denoted by $\pi(l,k) \in \{0,1\}$, $l \in \mathcal{L}$, $k \in \mathcal{K} \cup \{K+1\}$, is given by

$$\pi(l,k) = \begin{cases} 1, & \text{if the } l\text{th task is assigned to the } k\text{th WD,} \\ 0, & \text{otherwise.} \end{cases}$$

Also, define $\mathcal{L}^{(k)} = \{l \in \mathcal{L} : \pi(l,k) = 1\}$ as the set of tasks that are assigned to WD k, $k \in \mathcal{K} \cup \{K+1\}$. It is worthy of noting that we assume $|\mathcal{L}^{(k)}| \geq 1$, $k \in \mathcal{K} \cup \{K+1\}$. That is, each WD including the local user should be assigned with at least one task, i.e., $L \geq K+1$[2]. Define by C_l in cycles the number of CPU cycles required for computing the lth task, $l \in \mathcal{L}$ [20, 22]. Also, denote the CPU frequency in cycles per second (Hz) at the kth WD as f_k, $k \in \mathcal{K} \cup \{K+1\}$.

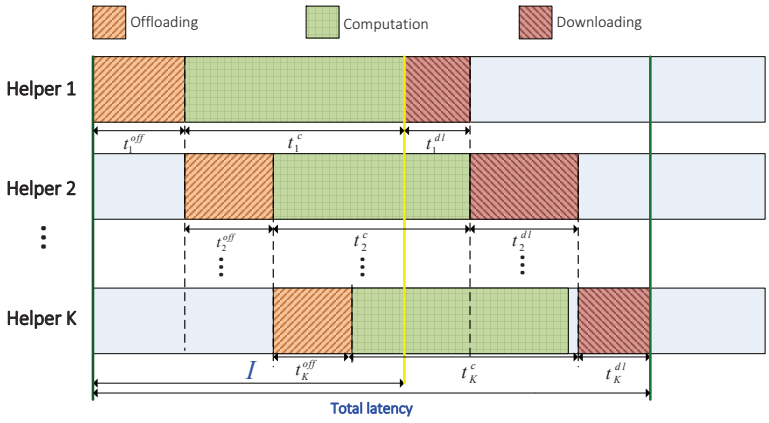


Fig. 1: The TDMA-based frame structure for the proposed MEC protocol.

A. Local Computing

The tasks in the set $\mathcal{L}^{(K+1)}$ are executed locally with the local computation frequency in cycles per second given as [17]

$$f_0 = \frac{\sum_{l=1}^{L} \pi(l, K+1) C_l}{t_0^c}, \tag{1}$$

[2]In practice, when $L \leq K$, a group of K' helpers are required to be selected *a priori* such that $L \geq K'+1$. However, detailed design regarding such selection mechanism is out of the discussion of this paper, and is left as our future work.

where t_0^c denotes the associated local computation time, and f_0 is subject to the maximum frequency constraint, i.e., $f_0 \leq f_0^{\max}$. The corresponding computation energy consumed by the local user is given by [25]

$$E_0^c = \kappa_0 \sum_{l=1}^{L} \pi(l, K+1) C_l f_0^2, \tag{2}$$

where κ_0 is a constant denoting the effective capacitance coefficient that is decided by the chip architecture of the local user. Replacing f_0 in (2) with (1), E_0^c can thus be expressed in terms of t_0^c as follows:

$$E_0^c = \kappa_0 \frac{\left(\sum_{l=1}^{L} \pi(l, K+1) C_l\right)^3}{(t_0^c)^2}. \tag{3}$$

B. Remote Computing at Helpers

The tasks assigned in $\mathcal{L}^{(k)}$ is offloaded to the kth helper, $k \in \mathcal{K}$, for remote execution. In this paper, we consider a three-phase TDMA communication protocol. As shown in Fig. 1, the local user first offloads the tasks in the set $\mathcal{L}^{(k)}$ to the kth helper, $k \in \mathcal{K}$, in a pre-scheduled order[3] via TDMA in the *task offloading* phase. Then the helpers execute their assigned computation tasks in the *task execution* phase. At last, in the *results downloading* phase, the helpers send the results back to the local user in the same order as in the task offloading phase also via TDMA. Note that at each TDMA time slot during the task offloading phase, the local user only offloads tasks to one helper. Similarly, during the results downloading phase, only one helper can transmit over each time slot. In the following, we introduce the three-phase protocol in detail.

1) Phase I: Task Offloading First, the tasks are offloaded to the helpers via TDMA. For simplicity, in this paper we assume that the local user offloads the tasks to the helpers with a fixed order of $1, 2, \ldots, K$ as in Fig. 1. In other words, the local user offloads tasks $\mathcal{L}^{(1)}$ to the 1st helper, then $\mathcal{L}^{(2)}$ to the 2nd helper, until $\mathcal{L}^{(K)}$ to the Kth helper.

Let h_k denote the channel power gain from the local user to the kth helper for offloading, $k \in \mathcal{K}$. The achievable rate in bits per second from at the kth helper is given by

$$r_k^{off} = B \log_2 \left(1 + \frac{p_k^{off} h_k}{\sigma_k^2}\right), \tag{4}$$

[3]Since frequent change of TDMA scheduling policy incurs large amount of signalling overhead, we assume a fixed-order TDMA protocol in this paper, which is practically reasonable, and also commonly adopted in the literature, e.g., [13, 17].

where B in Hz denotes the available transmission bandwidth, p_k^{off} is the transmitting power at the local user for offloading tasks to the kth helper, and σ_k^2 is the power of additive white Gaussian noise (AWGN) at the kth helper. Then, the time spent in offloading tasks to the kth helper is given by

$$t_k^{off} = \frac{\sum_{l=1}^{L} \pi(l,k) T_l}{r_k^{off}}. \tag{5}$$

According to (4) and (5), p_k^{off} is expressed in terms of t_k^{off} as

$$p_k^{off} = \frac{1}{\bar{h}_k} f\left(\frac{\sum_{l=1}^{L} \pi(l,k) T_l}{t_k^{off}}\right), \tag{6}$$

where $\bar{h}_k = h_k/\sigma_k^2$ is the normalized channel power gain, and $f(x)$ is a function defined as $f(x) \triangleq 2^{\frac{x}{B}} - 1$. The total energy consumed by the local user for offloading all the tasks to the helpers is thus expressed as

$$E_0^{off} = \sum_{k=1}^{K} \frac{1}{\bar{h}_k} f\left(\frac{\sum_{l=1}^{L} \pi(l,k) T_l}{t_k^{off}}\right) t_k^{off}. \tag{7}$$

2) Phase II: Task Execution After receiving the assigned tasks $\mathcal{L}^{(k)}$, $k \in \mathcal{K}$, the kth helper proceeds with the computation frequency given by

$$f_k = \frac{\sum_{l=1}^{L} \pi(l,k) C_l}{t_k^c}, \tag{8}$$

where t_k^c's is the remote computation time spent by the kth helper. Similarly, helper k's remote computing frequency given by (8) is also constrained by its maximum frequency, i.e.,$f_k \leq f_k^{\max}$. In addition, its computation energy is expressed as

$$E_k^c = \kappa_k \frac{\left(\sum_{l=1}^{L} \pi(l,k) C_l\right)^3}{(t_k^c)^2}, \tag{9}$$

where κ_k is the corresponding capacitance constant of the kth helper.

3) Phase III: Results Downloading After computing all the assigned tasks, the helpers begin transmitting computation results back to the local user via TDMA. Similar to the task offloading phase, we assume that the helpers transmit their respective results in the fixed order of $1, \ldots, K$. Let g_k denote the channel power gain from helper k to the local user for downloading. The achievable downloading rate from the kth helper is then given by

$$r_k^{dl} = B \log_2\left(1 + \frac{p_k^{dl} g_k}{\sigma_0^2}\right), \tag{10}$$

where p_k^{dl} denotes the transmitting power of the kth helper, and σ_0^2 denotes the power of AWGN at the local user. The corresponding downloading time is thus given by

$$t_k^{dl} = \frac{\sum_{l=1}^{L} \pi(l,k) R_l}{r_k^{dl}}. \tag{11}$$

Combining (10) and (11), the transmitting power of the kth helper is expressed as

$$p_k^{dl} = \frac{1}{\bar{g}_k} f\left(\frac{\sum_{l=1}^{L} \pi(l,k) R_l}{t_k^{dl}}\right), \tag{12}$$

where $\bar{g}_k = g_k/\sigma_0^2$ denotes the normalized channel power gain from the kth helper to the local user. The communication energy consumed by the kth helper is thus given by

$$E_k^{dl} = \frac{1}{\bar{g}_k} f\left(\frac{\sum_{l=1}^{L} \pi(l,k) R_l}{t_k^{dl}}\right) t_k^{dl}. \tag{13}$$

C. Total Latency

Since TDMA is used in both Phase I and Phase III, each helper has to wait until it is scheduled. Specifically, the first scheduled helper, i.e., helper 1, can transmit its task results to the local user only when the following two conditions are satisfied: first, its computation has been completed; and second, task offloading from the local user to all of the K helpers are completed such that the wireless channels begin available for data downloading. As a result, helper 1 starts transmitting its results after a period of waiting time given by

$$I_1 = \max\{t_1^{off} + t_1^c, \sum_{k=1}^{K} t_k^{off}\}, \tag{14}$$

where t_1^c is the task execution time at helper 1.

Moreover, for each of the other $K-1$ helpers, it can transmit the results to the local user only when: first, its computation has been completed; second, the $(k-1)$th helper scheduled preceding to it has finished transmitting. Consequently, denoting the waiting time for helper k ($k \geq 2$) to transmit the results as I_k, I_k is expressed as

$$I_k = \max\{\sum_{j=1}^{k} t_j^{off} + t_k^c, I_{k-1} + t_{k-1}^{dl}\}. \tag{15}$$

Accordingly, the completion time for all the results to finish being downloaded is expressed as

$$T = I_K + t_K^{dl}. \tag{16}$$

To sum up, taking local computing into account as well, the total latency for executing all of the L tasks is given by

$$T^{\text{total}} = \max\{t_0^c, T\}. \tag{17}$$

III. Problem Formulation

In this paper, we aim at minimizing the total latency for local/remote computing of all the tasks by optimizing the task assignment strategy ($\pi(l,k)$'s), the task offloading time (t_k^{off}'s), the task execution time (t_k^c's), and the results downloading time (t_k^{dl}'s), subject to the individual energy and computation frequency constraints at the local user as well as the K helpers. Specifically, we are interested in the following problem:

$$(\mathrm{P0}): \underset{\mathbf{\Pi},\{t_k^{off},t_k^{dl},t_k^c\},t_0^c}{\texttt{Minimize}} \quad T^{\mathrm{total}}$$

$$\texttt{Subject to}$$

$$\kappa_0 \frac{\left(\sum_{l=1}^{L}\pi(l,K+1)C_l\right)^3}{(t_0^c)^2} + \sum_{k=1}^{K}\frac{1}{\bar{h}_k} f\left(\frac{\sum_{l=1}^{L}\pi(l,k)T_l}{t_k^{off}}\right) t_k^{off} \le E_0, \tag{18a}$$

$$\kappa_k \frac{\left(\sum_{l=1}^{L}\pi(l,k)C_l\right)^3}{(t_k^c)^2} + \frac{1}{\bar{g}_k} f\left(\frac{\sum_{l=1}^{L}\pi(l,k)R_l}{t_k^{dl}}\right) t_k^{dl} \le E_k,\ \forall k \in \mathcal{K}, \tag{18b}$$

$$\frac{\sum_{l=1}^{L}\pi(l,K+1)C_l}{f_0^{\max}} \le t_0^c, \tag{18c}$$

$$\frac{\sum_{l=1}^{L}\pi(l,k)C_l}{f_k^{\max}} \le t_k^c,\ \forall k \in \mathcal{K}, \tag{18d}$$

$$\sum_{k=1}^{K+1}\pi(l,k) = 1,\ \forall l \in \mathcal{L}, \tag{18e}$$

$$\sum_{l=1}^{L}\pi(l,k) \ge 1,\ \forall k \in \mathcal{K}\cup\{K+1\}, \tag{18f}$$

$$\pi(l,k) \in \{0,1\},\ \forall l \in \mathcal{L},\ k \in \mathcal{K}\cup\{K+1\}, \tag{18g}$$

$$t_k^{off} \ge 0,\ t_k^{dl} \ge 0,\ \forall k \in \mathcal{K}. \tag{18h}$$

In the above problem, the objective function T^{total} is given by (17). The constraints given by (18a) and (18b) state that the sum energy consumption of computation and transmission for the local user and the kth helper cannot exceed E_0 and E_k's, respectively. In (18a), E_0^c and E_0^{off} are replaced with (2) and (7), respectively, while (18b) is obtained by substituting (9) and (13) for E_k^c's and E_k^{dl}'s, respectively. (18c) and (18d) guarantee that the computation frequency of the local users (c.f. (1)) and the helpers (c.f. (8)) stay below their respective limits. (18e) guarantees that each task must be and only assigned to one WD; and (18f) ensures that each of the local user

and the helpers is assigned with at least one task. Finally, (18g) imposes the binary offloading constraints.

A. Problem Reformulation

Note that T_{total} (c.f. (17)) is a complicated function involving accumulative $\max(\cdot)$ mainly due to recursive expression of I_k (c.f. (15)) for $k \geq 2$. Hence, to obtain an explicit objective function in terms of the optimization variables, we need to simplify T_{total} exploiting the following proposition.

Proposition 3.1: Problem (P0) can be recast into an equivalent problem as follows.

$$
\begin{aligned}
(\text{P0-Eqv}): \underset{\boldsymbol{\Pi},\{t_k^{off},t_k^{dl},t_k^c\},t_0^c}{\texttt{Minimize}} \quad & t_1^{off}+t_1^c+\sum_{k=1}^{K} t_k^{dl} \\
\texttt{Subject to} \quad & (18a)-(18h), \\
& \sum_{k=1}^{K} t_k^{off} \leq t_1^{off}+t_1^c, && (19a) \\
& t_k^c \leq t_1^{off}+t_1^c+\sum_{j=1}^{k-1} t_j^{dl}-\sum_{j=1}^{k} t_j^{off},\ \forall k \in \mathcal{K}\setminus\{1\}, && (19b) \\
& t_0^c \leq t_1^{off}+t_1^c+\sum_{k=1}^{K} t_k^{dl}. && (19c)
\end{aligned}
$$

Proof: A brief idea of the proof is given as follows. To remove $\max(\cdot)$ in I_k's, we first need to narrow down different cases leveraging the property of the optimal solution. Then, based on the simplified case, we recursively derive I_k's for $k \geq 2$. Finally, we arrive at the clear objective function of (P1-Eqv) subject to all the optimality conditions given by (19a)-(19c). Please refer to Appendix A for the proof in detail. ■

B. Suboptimal Design

The transformed Problem (P0-Eqv) is seen as an MINLP due to the integer constraints given by (18g), and is thus in general NP-hard. Although the optimal solution to (P0-Eqv) can be obtained by exhaustive search, it is computationally too expensive (approx. $O((K+1)^L)$ times of search) to implement in practice. Therefore, we solicit two approaches for suboptimal solution to (P0-Eqv) in the following sections. The first approach is to relax the binary variables into

continuous ones while the second approach aims for decoupling the task assignment and wireless resource allocation.

For the first approach, first, we relax (18g) into continuous constraints expressed as

$$\pi(l,k) \in [0,1],\ \forall l \in \mathcal{L},\ k \in \mathcal{K} \cup \{K+1\}. \tag{20}$$

Therefore, the relaxed problem is expressed as:

$$(\mathrm{P1}):\ \underset{\mathbf{\Pi},\{t_k^{off},t_k^{dl},t_k^c\},t_0^c}{\texttt{Minimize}} \quad t_1^{off}+t_1^c+\sum_{k=1}^{K} t_k^{dl}$$
$$\texttt{Subject to} \quad (18a)-(18f),\ (18h),\ (19a)-(19c),\ (20).$$

It is worthy of noting that, since E_0^c (c.f. (3)) and E_0^{off} (c.f. (7)) are composed perspective of convex functions $(\sum_l \pi(l,K+1)C_l)^3$ and $f(\sum_l \pi(l,k)T_l)$ w.r.t. the variables t_0^c and t_k^{off}'s, respectively, they are also convex functions. So are E_k^c and E_k^{dl}, $\forall k \in \mathcal{K}$. Therefore, (P1) is a convex problem. Next, we need to round the continuous $\pi(l,k)$'s into the binary one such that (18e) and (18f) are satisfied. The details of the proposed joint task assignment and wireless resource allocation scheme will be discussed in Section IV. In addition, we also provide a brief discussion regarding one special case of this approach in Section V-A, by which computation frequency of all the WDs are fixed to be their maximum serving as a benchmark scheme without computation resource allocation.

For the second approach, first, it is easy to verify that given $\mathbf{\Pi}$ fixed, (P0-Eqv) reduces to be a convex problem shown as below:

$$(\mathrm{P2}):\ \underset{\{t_k^{off},t_k^{dl},t_k^c\},t_0^c}{\texttt{Minimize}} \quad t_1^{off}+t_1^c+\sum_{k=1}^{K} t_k^{dl}$$
$$\texttt{Subject to} \quad (18a)-(18d),\ (18h),\ (19a)-(19c).$$

Based on Problem (P2), we decouple the design of task assignment and wireless resource allocation in a greedy task assignment based heuristic algorithm that will be elaborated in Section V-B.

IV. Joint Task Assignment and Wireless Resource Allocation

The main thrust of the proposed scheme in this section is to first relax the binary task-assignment variables into continuous ones, then solve the relaxed convex problem in semi-closed

forms of solution, and finally attain suboptimal task assignment based on the optimal solution to the relaxed problem.

It is seen that Problem (P1) is convex, and can thus be efficiently solved by some off-the-shelf convex optimization tools such as CVX [26]. To gain more insights into the optimal rate and computation frequency allocation, in this section, we propose to solve (P1) leveraging the technique of Lagrangian dual decomposition. The (partial) Lagrangian of (P1) is expressed as

$$\mathcal{L}_1(\mathbf{\Pi}, \{t_k^{off}, t_k^{dl}, t_k^c\}, t_0^c; \eta, \beta_0, \lambda_0, \zeta_0, \boldsymbol{\lambda}^T, \boldsymbol{\beta}^T, \boldsymbol{\zeta}^T) = t_1^{off} + t_1^c + \sum_{k=1}^{K} t_k^{dl} + \eta\Big(\sum_{k=1}^{K} t_k^{off} - t_1^{off} - t_1^c\Big) + \beta_0$$
$$\Big(t_0^c - t_1^{off} - t_1^c - \sum_{k=1}^{K} t_k^{dl}\Big) + \lambda_0\left(\kappa_0 \frac{\Big(\sum_{l=1}^{L} \pi(l, K+1) C_l\Big)^3}{(t_0^c)^2} + \sum_{k=1}^{K} \frac{1}{\bar{h}_k} f\left(\frac{\sum_{l=1}^{L} \pi(l,k) T_l}{t_k^{off}}\right) t_k^{off} - E_0\right)$$
$$- \zeta_0\Big(t_0^c - \frac{\sum_{l=1}^{L} \pi(l, K+1) C_l}{f_0^{\max}}\Big) + \sum_{k=1}^{K} \lambda_k\left(\kappa_k \frac{\Big(\sum_{l=1}^{L} \pi(l,k) C_l\Big)^3}{(t_k^c)^2} + \frac{1}{\bar{g}_k} f\left(\frac{\sum_{l=1}^{L} \pi(l,k) R_l}{t_k^{dl}}\right) t_k^{dl}\right.$$
$$\left. - E_k\right) + \sum_{k=2}^{K} \beta_k\big(t_k^c - t_1^{off} - t_1^c - \sum_{j=1}^{k-1} t_j^{dl} + \sum_{j=1}^{k} t_j^{off}\big) - \sum_{k=1}^{K} \zeta_k\Big(t_k^c - \frac{\sum_{l=1}^{L} \pi(l,k) C_l}{f_k^{\max}}\Big), \quad (21)$$

where η, β_0, λ_0, and ζ_0 denote the dual variables associated with the constraints (19a), (19c), (18a), and (18c), respectively; $\boldsymbol{\lambda} = (\lambda_1, \ldots, \lambda_k)^T$ represent the dual variables associated with the total energy constraints (18b) each for one helper; $\boldsymbol{\beta} = (\beta_2, \ldots, \beta_K)^T$ are the dual variables for the constraints given by (19b); and the multipliers $\boldsymbol{\zeta} = (\zeta_1, \ldots, \zeta_K)^T$ are assigned to the constraints given by (18d). After some manipulations, (21) can be equivalently expressed as

$$\mathcal{L}_1(\mathbf{\Pi}, \{t_k^{off}, t_k^{dl}, t_k^c\}, t_0^c; \eta, \beta_0, \lambda_0, \zeta_0, \boldsymbol{\lambda}^T, \boldsymbol{\beta}^T, \boldsymbol{\zeta}^T) = \bar{\mathcal{L}}_0(\mathbf{\Pi}, t_0^c; \beta_0, \lambda_0, \zeta_0) + \zeta_0 \frac{\sum_{l=1}^{L} \pi(l, K+1) C_l}{f_0^{\max}}$$
$$+ \sum_{k=1}^{K} \Big(\bar{\mathcal{L}}_k(\mathbf{\Pi}, t_k^{off}, t_k^{dl}, t_k^c; \eta, \beta_0, \lambda_0, \boldsymbol{\lambda}^T, \boldsymbol{\beta}^T, \boldsymbol{\zeta}^T) + \zeta_k \frac{\sum_{l=1}^{L} \pi(l,k) C_l}{f_k^{\max}}\Big), \quad (22)$$

where

$$\bar{\mathcal{L}}_0(\mathbf{\Pi}, t_0^c; \beta_0, \lambda_0, \zeta_0) = (\beta_0 - \zeta_0) t_0^c + \lambda_0 \kappa_0 \frac{\Big(\sum_{l=1}^{L} \pi(l, K+1) C_l\Big)^3}{(t_0^c)^2}, \quad (23)$$

and

$$\bar{\mathcal{L}}_k(\mathbf{\Pi}, t_k^{off}, t_k^{dl}, t_k^c; \eta, \beta_0, \lambda_0, \boldsymbol{\lambda}^T, \boldsymbol{\beta}^T, \boldsymbol{\zeta}^T) = A_k t_k^{dl} + B_k t_k^c + D_k t_k^{off} + \lambda_k \kappa_k \frac{\Big(\sum_{l=1}^{L} \pi(l,k) C_l\Big)^3}{(t_k^c)^2}$$
$$+ \frac{\lambda_0}{\bar{h}_k} f\left(\frac{\sum_{l=1}^{L} \pi(l,k) T_l}{t_k^{off}}\right) t_k^{off} + \frac{\lambda_k}{\bar{g}_k} f\left(\frac{\sum_{l=1}^{L} \pi(l,k) R_l}{t_k^{dl}}\right) t_k^{dl}, \quad (24)$$

with A_k, B_k, D_k, $\forall k \in \mathcal{K}$ given by

$$A_k = \begin{cases} 1-\beta_0-\sum_{j=k+1}^{K}\beta_j & k<K \\ 1-\beta_0 & k=K \end{cases},\ B_k = \begin{cases} 1-\eta-\beta_0-\sum_{k=2}^{K}\beta_k-\zeta_1 & k=1 \\ \beta_k-\zeta_k & k>1 \end{cases}, \tag{25}$$

and

$$D_k = \begin{cases} 1-\beta_0 & k=1 \\ \eta+\sum_{j=k}^{K}\beta_j & k>1 \end{cases}, \tag{26}$$

respectively. The dual function corresponding to (22) can be expressed as

$$g(\eta,\beta_0,\lambda_0,\zeta_0,\boldsymbol{\lambda}^T,\boldsymbol{\beta}^T,\boldsymbol{\zeta}^T) = \min_{\substack{\text{(18e)}-\text{(18f)}\\ \text{(18h)},\text{(20)}}} \mathcal{L}_1(\boldsymbol{\Pi},\{t_k^{off},t_k^{dl},t_k^c\},t_0^c;\eta,\beta_0,\lambda_0,\zeta_0,\boldsymbol{\lambda}^T,\boldsymbol{\beta}^T,\boldsymbol{\zeta}^T). \tag{27}$$

As a result, the dual problem of $(\mathrm{P2})$ is formulated as

$$\begin{aligned} (\mathrm{P1\text{-}dual}): \quad & \underset{\eta,\beta_0,\lambda_0,\zeta_0,\boldsymbol{\lambda},\boldsymbol{\beta},\boldsymbol{\zeta}}{\texttt{Maximize}} \quad g(\eta,\beta_0,\lambda_0,\zeta_0,\boldsymbol{\lambda}^T,\boldsymbol{\beta}^T,\boldsymbol{\zeta}^T) \\ & \texttt{Subject to} \quad \eta\geq 0,\ \beta_0\geq 0,\ \lambda_0\geq 0,\ \zeta_0\geq 0,\ \boldsymbol{\lambda}\geq 0,\ \boldsymbol{\beta}\geq 0,\ \boldsymbol{\zeta}\geq 0. \end{aligned} \tag{28}$$

A. Dual-Optimal Solution to (P1)

In this subsection, we aim for solving problem $(\mathrm{P1\text{-}dual})$. To facilitate solving the optimum $\{t_k^{off},t_k^{dl},t_k^c\}$ and t_0^c to (27) providing that $\boldsymbol{\Pi}=\bar{\boldsymbol{\Pi}}$ and a set of dual variables are given, we decompose the above problem into $K+1$ subproblems including K for $\forall k\in\mathcal{K}$ and one for $k=K+1$ as follows.

$$\begin{aligned} (\mathrm{P1\text{-}sub1}): \quad & \underset{t_k^{off},t_k^{dl},t_k^c}{\texttt{Minimize}} \quad \bar{\mathcal{L}}_k(\bar{\boldsymbol{\Pi}},t_k^{off},t_k^{dl},t_k^c;\eta,\beta_0,\lambda_0,\boldsymbol{\lambda}^T,\boldsymbol{\beta}^T,\boldsymbol{\zeta}^T) \\ & \texttt{Subject to} \quad t_k^{off}\geq 0,\ t_k^{dl}\geq 0. \end{aligned}$$

$$(\mathrm{P1\text{-}sub2}): \quad \underset{t_0^c}{\texttt{Minimize}} \quad \bar{\mathcal{L}}_0(\bar{\boldsymbol{\Pi}},t_0^c;\beta_0,\lambda_0,\zeta_0).$$

Since these problems are independent of each other, they can be solved in parallel each for one k, $k\in\mathcal{K}\cup\{K+1\}$.

Define $\tilde{f}(x)\triangleq\frac{B}{\ln 2}(W_0(\frac{x}{e}-\frac{1}{e})+1)$ for $x\geq 0$, in which $W_0(\cdot)$ is the principal branch of Lambert W function defined as the inverse function of $xe^x=y$ [27]. Then, in accordance with the optimal solution to the above subproblems, the optimal time and power together with the optimal task assignment to (27) are shown in the following proposition.

Proposition 4.1: Given a set of dual variables, the optimal solution to (27) is given by

$$\hat{t}_k^{off} = \begin{cases} \frac{\sum_{l=1}^{L}\hat{\pi}(l,k)T_l}{\tilde{f}\left(D_k\bar{h}_k/\lambda_0\right)} & \text{if } D_k>0, \\ \inf & \text{otherwise;} \end{cases} \quad \hat{t}_k^{dl} = \begin{cases} \frac{\sum_{l=1}^{L}\hat{\pi}(l,k)R_l}{\tilde{f}(A_k\bar{g}_k/\lambda_k)} & \text{if } A_k>0, \\ \inf & \text{otherwise;} \end{cases} \tag{29}$$

$$\hat{t}_k^{c} = \begin{cases} \frac{\sum_{l=1}^{L}\hat{\pi}(l,k)C_l}{\sqrt[3]{B_k/(2\lambda_k\kappa_k)}} & \text{if } B_k>0, \\ \inf & \text{otherwise;} \end{cases} \text{ and } \hat{t}_0^{c} = \begin{cases} \frac{\sum_{l=1}^{L}\hat{\pi}(l,K+1)C_l}{\sqrt[3]{(\beta_0-\zeta_0)/(2\lambda_0\kappa_0)}} & \text{if } \beta_0-\zeta_0>0, \\ \inf & \text{otherwise.} \end{cases} \tag{30}$$

In addition, $\hat{\pi}(l,k)$'s shown in (29) and (30) denote the optimal solution to the following linear programming (LP) problem:

$$\begin{aligned} (\text{LP1}): \ \underset{\mathbf{\Pi}}{\texttt{Minimize}} \quad & \sum_{l=1}^{L}\left(\sum_{k=1}^{K}\phi_{l,k}\pi(l,k)+\phi_{l,K+1}\pi(l,K+1)\right) \\ \texttt{Subject to} \quad & \text{(18e)}-\text{(18f)}, \text{(20)}, \end{aligned}$$

where $\phi_{l,k}$, $\forall k\in\mathcal{K}$, $\forall l\in\mathcal{L}$, is given by

$$\begin{aligned} \phi_{l,k} = & \frac{A_kR_l}{\tilde{f}\left(A_k\bar{g}_k/\lambda_k\right)}+\frac{B_kC_l}{\sqrt[3]{B_k/(2\lambda_k\kappa_k)}}+\frac{D_kT_l}{\tilde{f}\left(D_k\bar{h}_k/\lambda_0\right)}+\lambda_k\kappa_kC_l\left(\frac{B_k}{2\lambda_k\kappa_k}\right)^{\frac{2}{3}} \\ & +\frac{\lambda_0}{\bar{h}_k}f\left(\tilde{f}\left(\frac{D_k\bar{h}_k}{\lambda_0}\right)\right)\frac{T_l}{\tilde{f}\left(D_k\bar{h}_k/\lambda_0\right)}+\frac{\lambda_k}{\bar{g}_k}f\left(\tilde{f}\left(\frac{A_k\bar{g}_k}{\lambda_k}\right)\right)\frac{R_l}{\tilde{f}\left(A_k\bar{g}_k/\lambda_k\right)}+\zeta_k\frac{C_l}{f_k^{\max}}, \end{aligned} \tag{31}$$

and $\phi_{l,K+1}$, $\forall l\in\mathcal{L}$, is expressed as

$$\phi_{l,K+1}=\frac{(\beta_0-\zeta_0)C_l}{\sqrt[3]{(\beta_0-\zeta_0)/(2\lambda_0\kappa_0)}}+\lambda_0\kappa_0C_l\left(\frac{\beta_0-\zeta_0}{2\lambda_0\kappa_0}\right)^{\frac{2}{3}}+\zeta_0\frac{C_l}{f_0^{\max}}. \tag{32}$$

Proof: Please refer to Appendix B. ■

Remark 4.1: Note that some useful insights can be drawn from the results in Proposition 4.1. First, with the dual variables given, $\tilde{f}(D_k\bar{h}_k/\lambda_0)$ and $\tilde{f}(A_k\bar{g}_k/\lambda_k)$ can be, respectively, interpreted (in terms of the dual function (27)) as the optimum offloading rate to helper k and the optimum results downloading rate from helper k, while $\sqrt[3]{B_k/(2\lambda_k\kappa_k)}$ and $\sqrt[3]{(\beta_0-\zeta_0)/(2\lambda_0\kappa_0)}$ represent the optimum computation frequency at the kth helper and the local user, respectively. Accordingly, when helper k enjoys good offloading (downloading) channel gain, the optimum offloading rate (downloading rate) $\tilde{f}(D_k\bar{h}_k/\lambda_0)$ ($\tilde{f}(A_k\bar{g}_k/\lambda_k)$) also gets large due to non-decreasing monotonicity of $W_0(x)$. Moreover, provided that the total energy constraint for the local user is violated incurring a larger Lagrangian multiplier λ_0 (c.f. (21)), the optimum offloading time ($\hat{t}_k^{off}$'s) and

the optimum local computation time ($\hat{t}_0^c$) under the same $\mathbf{\Pi}$ turn out to be longer, leading to reducing the total energy consumption at the local user, which complies with Lemma A.1. Accordingly, Problem (P1-dual) can be further modified as shown below.

$$\begin{aligned}(\text{P1-dual}'): \quad & \underset{\eta,\beta_0,\lambda_0,\zeta_0,\boldsymbol{\lambda},\boldsymbol{\beta},\boldsymbol{\zeta}}{\texttt{Maximize}} \quad g(\eta,\beta_0,\lambda_0,\zeta_0,\boldsymbol{\lambda}^T,\boldsymbol{\beta}^T,\boldsymbol{\zeta}^T) \\ & \texttt{Subject to} \quad (28), \\ & \qquad\qquad A_k \geq 0,\ B_k \geq 0,\ D_k \geq 0,\ \forall k \in \mathcal{K},\ \beta_0 - \zeta_0 \geq 0,. \end{aligned} \tag{33}$$

In a sum, given an initial (feasible) set of dual variables, the optimal solution to (27) is first obtained leveraging Proposition 4.1, and then the dual variables are readily updated utilizing some sub-gradient based method, e.g., ellipsoid method [28, Ellipsoid Method (notes)] until a predefined threshold controlling accuracy of the algorithm is achieved.

B. Primal-Optimal Solution to (P1)

We aim for solving (P1) in this subsection. Since there might exist multiple solutions to (LP1) in each iteration of the ellipsoid method while there is only one optimal solution to the convex problem (P1), we retrieve the primal-optimal $\mathbf{\Pi}$ to (P1) through the dual-optimal solution in the sequel. Denoting the optimum dual variables by $(\eta^*, \beta_0^*, \lambda_0^*, \zeta_0^*, \boldsymbol{\lambda}^{*T}, \boldsymbol{\beta}^{*T}, \boldsymbol{\zeta}^{*T})$, the primal-optimal solution are related to the dual-optimal one as follows (c.f. (29) and (30)).

$$\frac{\sum_{l=1}^{L}\pi(l,k)T_l}{t_k^{off}} = \tilde{f}\left(D_k^*\bar{h}_k/\lambda_0^*\right), \qquad \frac{\sum_{l=1}^{L}\pi(l,k)R_l}{t_k^{dl}} = \tilde{f}\left(A_k^*\bar{g}_k/\lambda_k^*\right), \tag{34}$$

$$\frac{\sum_{l=1}^{L}\pi(l,k)C_l}{t_k^{c}} = \sqrt[3]{B_k^*/(2\lambda_k^*\kappa_k)}, \qquad \frac{\sum_{l=1}^{L}\pi(l,K+1)C_l}{t_0^{c}} = \sqrt[3]{(\beta_0^*-\zeta_0^*)/(2\lambda_0^*\kappa_0)}, \tag{35}$$

where A_k^*'s, B_k^*'s, and D_k^*'s are obtained by plugging the optimum dual variables into (25) and (26). Then transform Problem (P1) with variables $\mathbf{\Pi}$, $\{t_k^{off}, t_k^{dl}, t_k^c\}$, and t_0^c into an LP with $\mathbf{\Pi}$ as its only variable by plugging (34) and (35) into (P1). Denoting this LP by (LP2), (LP2) is then readily solved by standard LP algorithm, e.g., simplex method.

Note that (18c) and (18d) should be satisfied with $\sqrt[3]{\frac{\beta_0^*-\zeta_0^*}{(2\lambda_0^*\kappa_0)}} \leq f_0^{\max}$ and $\sqrt[3]{\frac{B_k^*}{(2\lambda_k^*\kappa_k)}} \leq f_k^{\max}$, respectively, independent of $\mathbf{\Pi}$, and thus can be safely removed from the constraints of (LP2).

Denoting the optimal solution to (LP2) by $\mathbf{\Pi}^*$, the primal-optimal solution to (P1) are thus given by

$$t_k^{off*} = \frac{\sum_{l=1}^{L} \pi^*(l,k) T_l}{\tilde{f}\left(D_k^* \bar{h}_k / \lambda_0^*\right)}, \qquad t_k^{dl*} = \frac{\sum_{l=1}^{L} \pi^*(l,k) R_l}{\tilde{f}\left(A_k^* \bar{g}_k / \lambda_k^*\right)}, \tag{36}$$

$$t_k^{c*} = \frac{\sum_{l=1}^{L} \pi^*(l,k) C_l}{\sqrt[3]{B_k^*/(2\lambda_k^* \kappa_k)}}, \qquad t_0^{c*} = \frac{\sum_{l=1}^{L} \pi^*(l,K+1) C_l}{\sqrt[3]{(\beta_0^* - \zeta_0^*)/(2\lambda_0^* \kappa_0)}}. \tag{37}$$

C. Sub-Optimal Solution to (P0-Eqv)

In this section, we propose a suboptimal scheme to jointly optimize task assignment as well as time and power allocation for (P0-Eqv) based on the optimal solution to (P1) developed in the previous subsections.

First, we propose to round off $\pi^*(l,k)$'s as follows such that (18g) are satisfied:

$$\pi^*(l,k) = \begin{cases} 1 & \text{if } k = \hat{k}_l, \\ 0 & \text{otherwise}, \end{cases} \quad \forall l \in \mathcal{L}, \tag{38}$$

where $\hat{k}_l = \arg \max_{k \in \mathcal{K} \cup \{K+1\}} \pi^*(l,k)$ To ensure that each helper is assigned at least one task, we need to further adjust $\mathbf{\Pi}^*$ to avoid the cases where some WD is assigned with no task after rounding off as (38). The detailed procedure of constructing such $\mathbf{\Pi}$ is shown in Fig. 2.

Next, given the updated $\mathbf{\Pi}^*$, what remains to be solved is Problem (P2). Hence, it can also be solved using Lagrangian dual decomposition following similar procedures as shown in Section IV. A. The proposed joint task assignment and wireless resource allocation scheme is summarized in Algorithm 1.

Algorithm 1 The Proposed Suboptimal Algorithm for (P0-Eqv)

Input a set of dual variables satisfying (28) and (33): $\eta^{(0)}, \beta_0^{(0)}, \lambda_0^{(0)}, \zeta_0^{(0)}, \boldsymbol{\lambda}^{(0)}, \boldsymbol{\beta}^{(0)}, \boldsymbol{\zeta}^{(0)}$

1) Solve $(\text{P1-dual}')$ using ellipsoid method [28];
2) Obtain the dual-optimal solution: $\eta^*, \beta_0^*, \zeta_0^*, \boldsymbol{\lambda}^*, \boldsymbol{\beta}^*, \boldsymbol{\zeta}^*$;
3) Retrieve the primal-optimal $\mathbf{\Pi}^*$ to (P1) by solving (LP2);
4) Obtain the primal-optimal $\{t_k^{off*}, t_k^{dl*}, t_k^{c*}\}$ and t_0^{c*} to (P1) in accordance with (36) and (37);
5) Modify $\mathbf{\Pi}^*$ in accordance with the procedures shown in Fig. 2;
6) Solve (P2) given the modified $\mathbf{\Pi}^*$.

Output solution to (P0-Eqv)

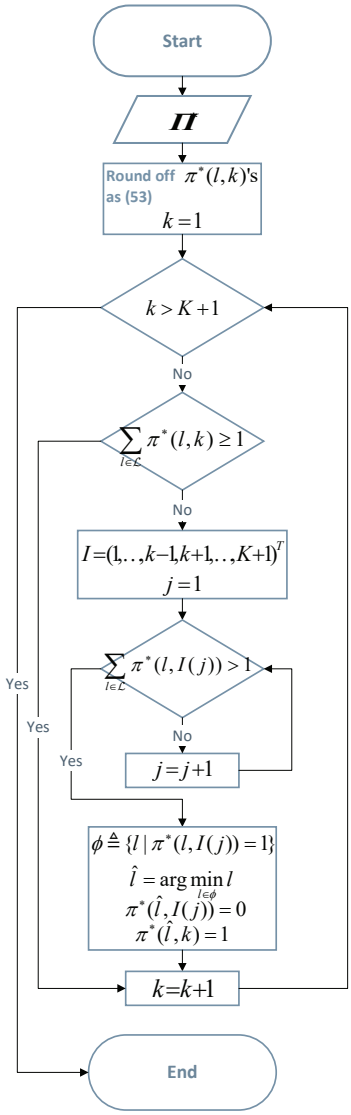


Fig. 2: The procedures to modify $\boldsymbol{\Pi}^*$ such that the constraints in (18e)-(18g) are satisfied.

D. Complexity

The complexity of Algorithm 1 mainly lies in solving (P1). Hence, we focus on discussing the complexity of solving (P1). Specifically, the complexity of solving (P1) includes solving $(\text{P1-dual}')$ by ellipsoid method via primal-dual iterations, and solving (LP1) by simplex method[4] in each iteration of the ellipsoid method. In accordance with the worst-case number of iterations

[4]The simplex method is a standard search algorithm that travels through the set of basic feasible solutions one at a time, until the optimal basic feasible solution (if it exists) is identified [29], e.g., *linprog* in Matlab.

for the ellipsoid method and the expected complexity of the simplex algorithm[5], the complexity of Algorithm 1 can be estimated by

$$\mathcal{O}\left(18(K+1)^2\log(\sqrt{\gamma}H/\epsilon)\mathcal{P}(L(K+1))\right), \tag{39}$$

where $H \triangleq \max_{\boldsymbol{h}\in\partial g(\boldsymbol{z}),\, \boldsymbol{z}\in E^{(0)}} \|\boldsymbol{h}\|$ is a Lipschitz constant for (27) over the initial ellipsoid $E^{(0)} = \{\boldsymbol{z} \mid \|\boldsymbol{z}\| \leq \sqrt{\gamma}\}$ [28, Ellipsoid Method (notes)], $\boldsymbol{h}$ is a sub-gradient of $g(\eta, \beta_0, \lambda_0, \zeta_0, \boldsymbol{\lambda}^T, \boldsymbol{\beta}^T, \boldsymbol{\zeta}^T)$ over $E^{(0)}$, and ϵ is a parameter controlling accuracy of the algorithm.

V. Low-Complexity Benchmark Schemes

In this section, we develop two low-complexity benchmark schemes, which are provided in Sections V-A and V-B, respectively.

A. Fixed-Frequency Task Assignment and Wireless Resource Allocation

In this subsection, we consider a benchmark scheme that alleviates the WDs from adjusting their computation frequency using DVFS by endowing them with the maximum computation capacities, i.e., $f_0 = f_0^{\max}$, and $f_k = f_k^{\max}$, $k \in \mathcal{K}$. Note that with the computation frequency fixed, this scheme reduces to a special case of the joint task assignment and wireless resource allocation scheme with the constraints (18c) and (18d) being active. Since we have studied a similar fixed-frequency design in [1], in the sequel, we only focus on some results that are distinguished from Section IV due to space limitation.

First, with the computation frequency fixed, the computation energy consumed by the local user and the helper turn out to be $E_0^c = \kappa_0 \sum_{l\in\mathcal{L}} \pi(l, K+1) C_l (f_0^{\max})^2$, and $E_k^c = \kappa_k \sum_{l\in\mathcal{L}} \pi(l,k) C_l (f_k^{\max})^2$, respectively. Moreover, t_0^c and t_k^c's in Problem (P0) are safely removed by replacing them with the left-hand side (LHS) of (18c) and (18d), respectively. In addition, since t_1^c cannot be adjusted

[5]Despite that the worst-case complexity of simplex algorithm is known to be exponential, it was shown in [30] that most LP can be approximated with their inputs perturbed and then solved by simplex algorithm in polynomial time.

once helper 1 is assigned with its tasks, I_1 (c.f. (14)) cannot be further simplified. To sum up, Problem (P0-Eqv) reduces to be:

$$
\begin{aligned}
(\mathrm{P0'\text{-}Eqv}): \ \underset{\mathbf{\Pi},\{t_k^{off},t_k^{dl}\}}{\texttt{Minimize}} \quad & I_1+\sum_{k=1}^{K} t_k^{dl} \\
\texttt{Subject to} \ \ & (18a)-(18b),\ (18e)-(18h), \\
& \sum_{k=1}^{K} t_k^{off} \le I_1, && \text{(40a)} \\
& t_1^{off}+\frac{\sum_{l=1}^{L}\pi(l,1)C_l}{f_1^{\max}} \le I_1, && \text{(40b)} \\
& \frac{\sum_{l=1}^{L}\pi(l,k)C_l}{f_k^{\max}} \le I_1+\sum_{j=1}^{k-1} t_j^{dl}-\sum_{j=1}^{k} t_j^{off},\ \forall k\in\mathcal{K}\setminus\{1\}, && \text{(40c)} \\
& \frac{\sum_{l=1}^{L}\pi(l,K+1)C_l}{f_0^{\max}} \le I_1+\sum_{k=1}^{K} t_k^{dl}. && \text{(40d)}
\end{aligned}
$$

Remark 5.1: Compared with Problem (P0-Eqv), Problem $(\mathrm{P0'\text{-}Eqv})$ is more sceptical to infeasibility[6], since the minimum of E_0^c (E_k^c's) cannot reach zero with t_0^c (t_k^c's) approaching to infinity (c.f. (3) ((9))). Once it is feasible, the solution to $(\mathrm{P0'\text{-}Eqv})$ can be found similarly by a simplified version of Algorithm 1. It is also worth noting that the simplified Algorithm 1 yields an approximate complexity of $\mathcal{O}(2(2K+3)^2\log(\sqrt{\gamma}H/\epsilon)\mathcal{P}(L(K+1)))$, which is lower than (39) due to decrease in the number of dual variables.

B. Greedy Task Assignment Based Wireless Resource Allocation

The joint task assignment and wireless resource allocation and its special case with fixed computation frequency both admit complexity with a polynomial factor $\mathcal{P}(L(K+1))$ induced by updating $\hat{\pi}(l,k)$'s using simplex method, to reduce computation complexity, we propose in this subsection greedy task assignment based wireless resource allocation. The main idea of this heuristic scheme is as follows. First, we obtain the objective value of problem (P2) as a cost function of the initial task assignment. Then we assign one task each time from those unallocated to a WD that yields the least amount of increase in the cost function. Next, since two different

[6]Discussion regarding the feasibility of the fixed-frequency scheme can be referred to [1].

task selection criteria are applied, we compare the results of these two sub-schemes and choose the one achieving the lower total latency.

The greedy task assignment based wireless resource allocation scheme is shown in Algorithm 2. In Algorithm 2, it is seen that for the first K tasks, each of them is in turn assigned to the helper with the best channel condition among those who have not yet been occupied. The intuition behind such assignment is that the selected helper will consume the least amount of energy in transmission (c.f. (7) and (13)), and thus any spare energy can be exploited for further latency reduction. It is also worth noting that the task with the longest (input/output) data flow is executed locally for the sake of saving data-transmission time.

The complexity of Algorithm 2 comprises that of two sub-schemes (sorting T_l's and R_l's, respectively), each of which requires solving problem (P2) for $(L-K-1)(K+1)$ times to find the right task assignment matrix. Since solving (P2) by ellipsoid method yields maximum as many as $18(K+1)^2\log(\sqrt{\gamma}H/\epsilon)$ iterations (c.f. (39)), the worst-case complexity of Algorithm 2 is given by

$$\mathcal{O}\left(36(L-K-1)(K+1)^3\log(\sqrt{\gamma}H/\epsilon)\right). \tag{41}$$

In fact, (41) suggests that even the worst-case complexity of Algorithm 2 is much less than that of Algorithm 1 (c.f. (39)) as long as $2(L-K-1)(K+1) < \mathcal{P}(L(K+1))$, which is easily verified to be true in most cases.

VI. Numerical Results

In this section, we provide numerical results to validate the effectiveness of the proposed joint task assignment and wireless resource allocation (**Joint optimization**) in Section IV, as compared against the fixed-frequency scheme (**Fixed frequency**) and the greedy task assignment based algorithm (**Greedy assignment**) presented in Section V as well as other benchmark schemes as follows.

- **Optimal** The MINLP problem (P0-Eqv) is solved by exhaustive search over feasible $\mathbf{\Pi}$ for $(K+1)^L - \sum_{i=1}^{K}(-1)^{i+1}\binom{K+1}{i}(K+1-i)^L$ times[7] with the convex problem (P2) solved each time under a given $\mathbf{\Pi}$. Note that "Optimal" is of exponential complexity and is thus too costly to implement in practice. Hence, we only provide this scheme for one numerical example in the sequel.

[7]This number take all combinations of $\mathbf{\Pi}$ satisfying (18e)-(18g) into account.

Algorithm 2 The Heuristic Algorithm for (P0-Eqv)

Input T_l, R_l, $l \in \mathcal{L}$

1) Initialize $\mathbf{\Pi}^{(0)} = \mathbf{0}_{L\times(K+1)}$;
2) Sort T_l's in ascending order such as $T_{m_1} \leq \ldots \leq T_{m_L}$;
3) $\pi^{(0)}(m_L, K+1) = 1$, $\mathcal{L}^{(0)} = \mathcal{L} \setminus \{m_L\}$, $\mathcal{K}^{(0)} = \mathcal{K} \setminus \{K+1\}$, $i = 0$;
4) **Repeat**
5) $i = i + 1$, $\mathbf{\Pi}^{i} = \mathbf{\Pi}^{(i-1)}$;
6) $\pi^{(i)}(m_i, k^*)$=1, where $k^* = \arg \max_{k\in\mathcal{K}^{(i-1)}} \bar{h}_k$;
7) $\mathcal{L}^{(i)} = \mathcal{L}^{(i-1)} \setminus \{m_i\}$, $\mathcal{K}^{(i)} = \mathcal{K}^{(i-1)} \setminus \{k^*\}$;
8) **Until** $i = K$;
9) Solve (P2) with $\mathbf{\Pi}$ given by $\mathbf{\Pi}^{(i)}$ and obtain the objective value $p^{(0)}$;
10) **Repeat**
11) $i = i + 1$, $j = 0$;
12) **Repeat**
13) $j = j + 1$, $\mathbf{\Pi}^{(j)} = \mathbf{\Pi}^{(i-1)}$;
14) $\pi^{(j)}(m_i, j) = 1$;
15) Solve (P2) with $\mathbf{\Pi}$ given by $\mathbf{\Pi}^{(j)}$ and obtain the objective value $p^{(j)}$;
16) **Until** $j = K + 1$;
17) $\pi^{(i)}(m_i, j^*) = 1$, where $j^* = \arg \min_{j\in\mathcal{K}\cup\{K+1\}} p^{(j)}$;
18) $\mathcal{L}^{(i)} = \mathcal{L}^{(i-1)} \setminus \{m_i\}$;
19) **Until** $\mathcal{L}^{(i)} = \emptyset$;
20) Solve (P2) with $\mathbf{\Pi}$ given by $\mathbf{\Pi}^{(i)}$ and obtain the objective value $p_1(\mathbf{\Pi}, \{t_k^{off}, t_k^{dl}, t_k^c\}, t_0^c)$;
21) Repeat steps 1)-20) with T_{m_l}'s, $\bar{h}_{n_k}$'s, and $p_1(\mathbf{\Pi}, \{t_k^{off}, t_k^{dl}, t_k^c\}, t_0^c)$ replaced by R_{m_l}'s, $\bar{g}_{n_k}$'s, and $p_2(\mathbf{\Pi}, \{t_k^{off}, t_k^{dl}, t_k^c\}, t_0^c)$, respectively;
22) $n^* = \arg \min_{n\in\{1,2\}} p_n(\mathbf{\Pi}, \{t_k^{off}, t_k^{dl}, t_k^c\}, t_0^c)$.

Output $\arg p_{n^*}(\mathbf{\Pi}, \{t_k^{off}, t_k^{dl}, t_k^c\}, t_0^c)$ as solution to (P0-Eqv)

- **Random assignment** In this scheme, a random matrix with its entries drawn from $i.i.d.$ uniform distribution over $[0, 1]$ is first generated, and then the procedure shown in Fig. 2 is employed to construct a feasible $\mathbf{\Pi}$ to (P0-Eqv). Next, solve problem (P2) under this given $\mathbf{\Pi}$.
- **Local execution** All of the computation tasks in $\mathcal{L}$ are executed locally with the total latency expressed as $\max\left\{\sqrt{\frac{\kappa_0(\sum_{l\in\mathcal{L}} C_l)^3}{E_0}}, \frac{\sum_{l\in\mathcal{L}} C_l}{f_0^{\max}}\right\}$ by combining (18a) and (18c).

In simulations, the K helpers are located with a distance uniformly distributed over $[0, 500]$m away from the local user. The wireless channel model consists of pathloss and Rayleigh fading. The distance-dependent pathloss model is given by $128.1 + 37.6 \log 10(d)$ in dB, where d in km is the distance between the local user and a helper. The Rayleigh fading is generated by $i.i.d.$ CSCG RVs with zero mean and unit variance. The capacitance coefficients are set all equal as $\kappa_k = \kappa_0 = 10^{-28}$, $k \in \mathcal{K}$ [13]. We also assume identical AWGN power with the transmission bandwidth of $B = 312.5$KHz and the noise power spectrum density of -169 dBm/Hz. The other parameters are set as follows unless otherwise specified. The bit-length of the input and output data are set as $T_l \sim \mathcal{U}[0, 10^4]$bits and $R_l \sim \mathcal{U}[0, 10^4]$. The amount of computation required per task is assumed to be $C_l \sim \mathcal{U}[0, 5 \times 10^6]$cycles. The energy constraints are set as $E_0 = -30$dB and $E_k = -20$dB, $\forall k \in \mathcal{K}$. The maximum frequency for local computing is $f_0^{\max} = .9$GHz, and that for remote computing is $f_k^{\max} \sim \mathcal{U}[1.5, 2]$GHz, $\forall k \in \mathcal{K}$. The total latency is obtained by averaging over channel realizations of 300 times.

A. The Effect of Wireless Resource on the Total Latency

We consider a simple scenario where the local user has $L = 5$ tasks to be executed in the present of $K = 2$ helpers. Fig. 3 shows the total latency versus the energy constraints at the helpers assuming $E_1 = E_2$. It is observed that the optimal scheme outperforms all the other ones, while our proposed "Joint optimization" achieves the second lowest total latency with little gap to the optimal solution. "Greedy assignment" sees reducing total latency with the helpers' energy constraints, and outperforms "Local execution" in most cases except for those with E_k's below around -38dB, since when the helpers also suffer from a scarcity of energy, full local execution with constant total latency of $\max\left\{\sqrt{\frac{\kappa_0(\sum_{l\in\mathcal{L}} C_l)^3}{E_0}}, \frac{\sum_{l\in\mathcal{L}} C_l}{f_0^{\max}}\right\}$ is intuitively better than computation offloading. Moreover, "Fixed frequency" starts working with E_k's above around -32dB, which is because (P0-Eqv) may be infeasible due to any of the constraints in (18b) unsatisfied.

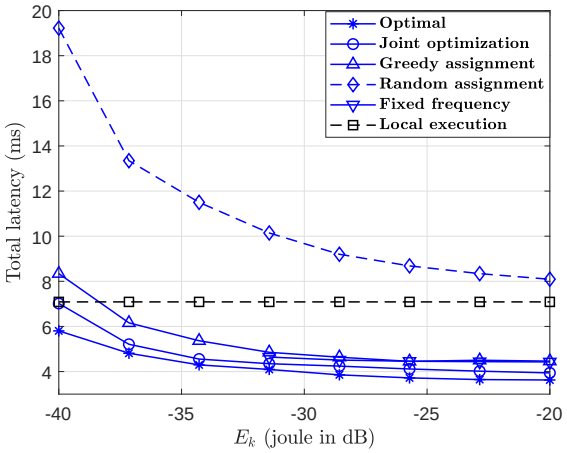


Fig. 3: The total latency versus the energy constraints with $K = 2$ and $L = 5$.

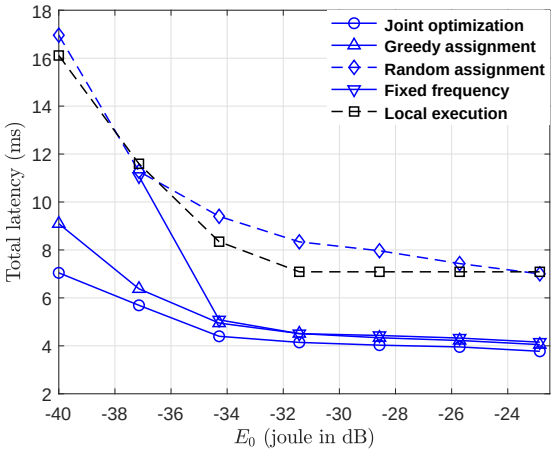


Fig. 4: The total latency versus the energy constraints at the local user with $E_1 = E_2 = -10$dB and $L = 5$.

Considering the same scenario as in Fig. 3, Fig. 4 shows the total latency versus the energy constraint at the local user with $E_1 = E_2 = -10$dB. It is seen that the average total latency decreases over E_0 for all of the schemes. Specifically, "Joint optimization", "Greedy assignment" and "Fixed frequency" admit sharp reduce in the average total latency when E_0 is less than about -34dB, and slightly go down when E_0 continues decreasing. This complies with intuition as follows. When the energy resource is scarce at the local user, a little more energy will cause significant decrease in the computation offloading time, while as for E_0 beyond -34dB, the bottleneck mainly lies in the helper' energy constraint E_k's. It is also worth noting that "Fixed

frequency" is substantially outperformed by "Joint optimization" and "Greedy assignment" when E_0 is less than about -34dB, since local computing with its full computation capacity is far from optimality under circumstances of limited energy supply. In addition, for "Local execution", $\frac{\sum_{l\in\mathcal{L}} C_l}{f_0^{\max}}$ starts taking effect when E_0 is larger than about -31.5dB.

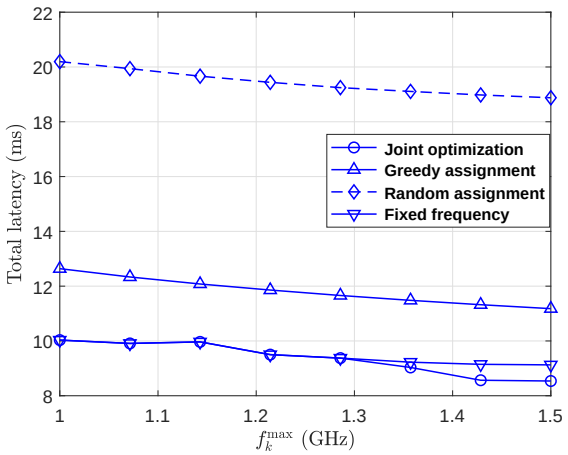
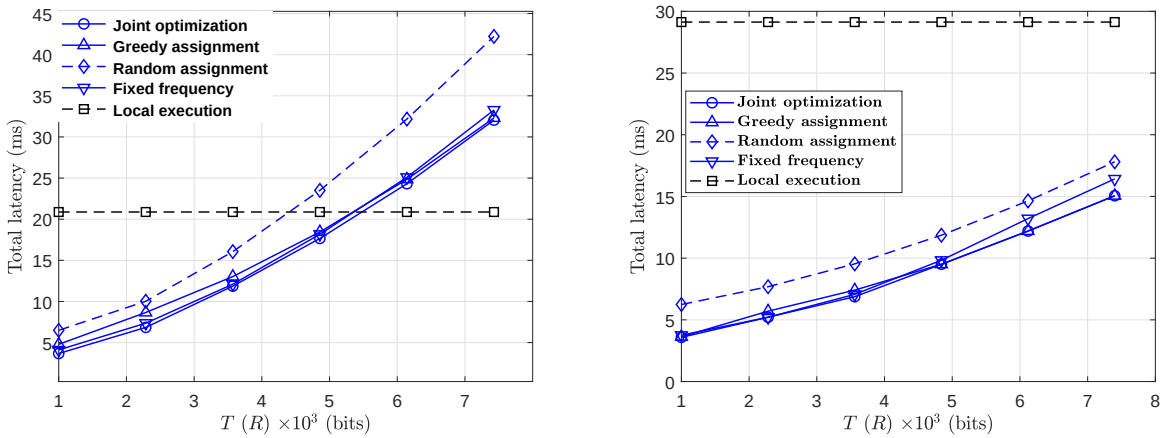


Fig. 5: The total latency versus the maximum remote computation frequency with $K = 5$ and $L = 7$.

Fig. 5 shows the total latency versus the maximum computation frequency at the helpers under the assumption of $f_1^{\max} = \ldots = f_5^{\max}$ and $L = 7$. "Fixed frequency" is observed to almost overlap with "Joint optimization" in most cases (when $f_k^{\max}$'s is below about 1.32GHz), and to outperform "Greedy assignment". This is because when $f_k^{\max}$'s is below 1.32GHz, the helpers' computing frequency obtained by "Joint optimization" tends to be their maximum capacities, i.e., $f_k^{\max}$'s. This special case turns out to be equivalent to "Fixed frequency" (c.f. Section V-A). However, when $f_k^{\max}$'s continuously increases,"Joint optimization" will suppress the helpers' computation capacities strictly below their limits such that under given energy budget of the helpers, there is still sufficient energy left for results downloading thus achieving the overall better performance. In addition, "Local execution" under this setting is equal to 39.5ms, which is too large to illustrate in the figure herein.

B. The Effect of Computation Load on the Average Total Latency

The trade-offs between the total latency and the bit-length of the task input/output data are demonstrated in Fig. 6 with different energy constraints' setup under the assumption of equal



(a) $E_0 = -30$dB, $E_1 = \ldots = E_5 = -10$dB, and $L = 8$. (b) $E_0 = -33$dB, $E_1 = \ldots = E_5 = 0$dB, and $L = 7$.

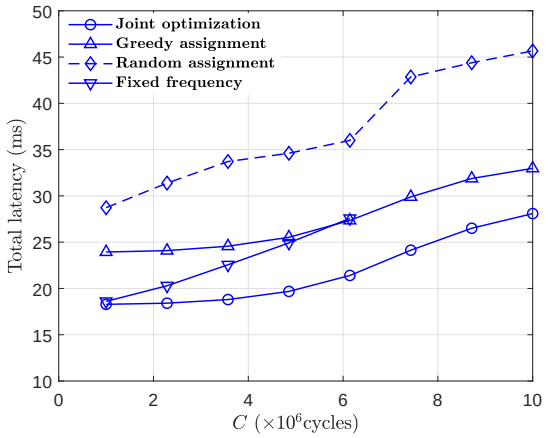
Fig. 6: The total latency versus the bit-length of the input (output) task data.

input/output length across all tasks, i.e., $T_1 = T$ and $R_1 = R$, $\forall l \in \mathcal{L}$. It is observed that the schemes of "Joint optimization", "Fixed frequency", and "Greedy assignment" almost overlap each other with little gap except that "Greedy assignment" is slightly worse than the other two in low range of data length in Fig. (6a) and "Fixed frequency" is noticeably worse than the other two in high range of data length in Fig. (6b). "Random assignment" continues being obviously outperformed by all the above three task offloading schemes especially when T (R) gets larger, which validates the importance of effective task assignment. It is also worth noting that with the energy setting higher in the local user ($E_0 = -30$dB) and lower in the helpers ($E_k = -10$dB, $\forall k \in \mathcal{K}$), when T (R) is larger than around 5.5×10^3 bits, "Local execution" becomes favourable due to increasing energy consumption in data transmission. By contrast, when the energy budget of the helpers increases to 0dB in Fig. (6b), all the task offloading schemes considerably outperform "Local execution", since with sufficient energy supply (c.f. (18b)), it costs the helpers little time to transmit even very long task-output data.

The effect of the amount of computation required per task on the total latency is shown in Fig. 7 with different energy constraints' setup assuming $C_l = C$, $\forall l \in \mathcal{L}$.[8] It is seen that "Fixed frequency" turns out to be unavailable when C is larger than about 7.4×10^6cycles and 6.0×10^6cycles in Fig. (7a) and Fig. (7b), respectively, due to infeasibility incurred for the same

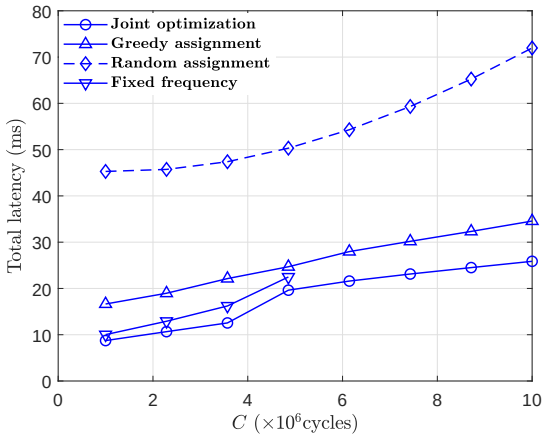
[8]Due to the relatively high regime into which the latency of "Local execution" falls in Fig. 7, we present the results in tables to improve readability.

C ($\times 10^6$cycles)	1.00	2.29	3.57	4.86	6.14	7.43	8.71	10.0
Local execution latency (ms)	7.77	20.2	39.5	62.7	89.2	119	151	185



(a) $E_0 = -30$dB, $E_1 = \ldots = E_5 = -20$dB, and $L = 7$.

C ($\times 10^6$cycles)	1.00	2.29	3.57	4.86	6.14	7.43	8.71	10.0
Local execution latency (ms)	14.1	48.9	95.5	151	215	286	364	447



(b) $E_0 = -33$dB, $E_1 = \ldots = E_6 = 0$dB, and $L = 10$.

Fig. 7: The total latency versus the amount of computation required per task.

reason as discussed for Fig. 3. It is also observed that the schemes of "Joint optimization", "Greedy assignment" and "Random assignment" increase with C very slowly when it is below about 5.0×10^6cycles (4.0×10^6cycles) in Fig. (7a) (Fig. (7b)), which is because under low to medium computation load per task, very little increase in t_1^c is already sufficient to satisfy (19b). However, when C gets larger, drastically increasing amount of computation energy yield decrease in communications energy thus prolonging the total latency. In addition, "Local execution" admits the best performance among all the schemes when the computation burden becomes below 3.1×10^6 cycles per task under the lower energy setting of the helpers in Fig. (7a), since "Local execution" is preferred for computation non-intensive tasks. However, when there is drastic difference in the energy budget between the local user and the helpers, the advantage of the task assignment schemes is prominently observed in Fig. (7b), as similarly explained for Fig. (6b).

Fig. 8 shows the total latency under different number of tasks with $K = 5$ helpers. It is seen that while the average total latency achieved by "Joint optimization", "Greedy assignment", and "Fixed frequency" steadily increase with the total number of tasks, that of "Local execution" drastically grows, which motivates cooperative MEC especially when there are a large number of tasks to be executed. Moreover, the performance of "Random assignment", almost worst among all the schemes, stresses the importance of proper task distribution schemes. "Greedy assignment" is seen to strike satisfying balance between the performance and the complexity, in

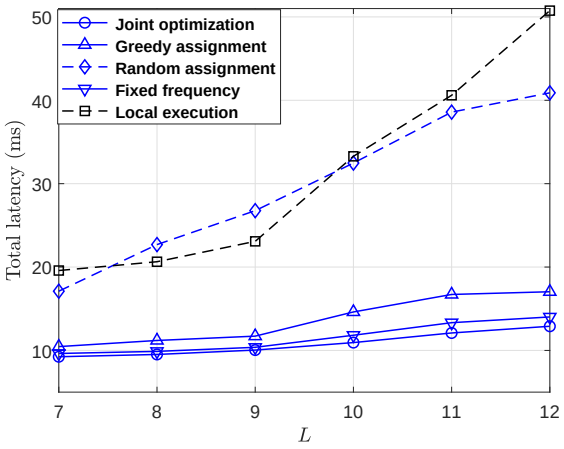


Fig. 8: The total latency versus the number of tasks with $K = 5$.

spite of increasing gap to "Joint optimization" when there are more than $L = 10$ tasks in total.

VII. Concluding Remarks

In this paper, we investigated the joint task assignment, communications rate, as well as computation frequency allocation for a D2D-enabled multi-helper MEC system assuming binary task offloading. Under a TDMA communication protocol, we aimed for minimizing the overall computation latency subject to individual energy and computation capacity constraints at both the local user and the helpers. Since the formulated problem was an MINLP that is in general difficult to solve, we proposed an efficient convex-relaxation-based algorithm to construct suboptimal task assignment based on the optimal solution to the relaxed problem. Furthermore, a benchmark scheme with fixed computation frequency and a greedy task assignment based heuristic algorithm were also developed to strike the balance between complexity and performance. Finally, numerical results verified that our proposed design is an effective solution to enhance the local user's computation latency, by exploiting D2D collaborations at the network edge.

Due to space limitation, there are several other challenging issues not addressed in this paper, which will be investigated in our future work. First, this paper assumed that all of the WDs are deployed at fixed locations with static wireless channels. In practice, these WDs may move over time. For instance, the wireless channels may fluctuate over time, and the D2D connections established between the local user and the helpers may be dropped as they move away from each other. Under these circumstances, we need to consider new design principles

(e.g., online algorithms with long-term objectives capturing dynamics of the D2D links) to combat such mobility issues [24]. Next, in this paper we considered TDMA owing to its easy implementation in practice. Other orthogonal multiple access schemes, e.g., OFDMA [13, 22], and more sophisticated NOMA schemes, e.g., sparse code multiple access (SCMA) [31], can be employed to further enhance the system performance. Under these schemes, on top of the mixed-integer task assignment, how to design the subcarrier allocation for OFDMA and the joint message decoding for NOMA are also quite challenging issues worthy of further study. In addition, our considered model assumed that the helpers have agreed to cooperate in the computation offloading, while we believe there are different types of collaboration-advocating incentive mechanisms that require sophisticated design. At last, it is also worth investigating how to extend our current single-user multi-helper MEC model to a multi-user multi-helper one by proper design of multi-user scheduling algorithms.

APPENDIX A
PROOF OF PROPOSITION 3.1

First, in order to prove Proposition 3.1, we need the following lemma.

Lemma A.1: The function $h(y,t) = f\left(\frac{y}{t}\right) t$ monotonically decreases over $t > 0$.

Proof: The monotonicity can be obtained by evaluating the first-order partial derivative of $h(y,t)$ w.r.t. t, and using the fact that $(1-x)e^x - 1 < 0$, for $x > 0$. ■

On one hand, there are two possible cases for the optimal I_k's given by (15): case 1) $\sum_{j=1}^{k} t_j^{off} + t_k^c > I_{k-1} + t_{k-1}^{dl}$; and case 2) $\sum_{j=1}^{k} t_j^{off} + t_k^c \leq I_{k-1} + t_{k-1}^{dl}$. In line with Lemma A.1, the total transmitting energy of the kth helper, i.e., E_k^{dl}'s (c.f. (13)), $\forall k \in \mathcal{K}$, monotonically decreases over t_k^{dl}'s. Hence, if the first case occurs, helper $k-1$ ($k \geq 2$) can slow down its downloading, i.e., extending t_{k-1}^{dl} until $\sum_{j=1}^{k} t_j^{off} + t_k^c = I_{k-1} + t_{k-1}^{dl}$ is satisfied (c.f. Fig. 1), such that I_k remains unchanged but the transmitting energy of helper $k-1$ gets reduced. As such, the two cases can be, w.l.o.g., merged into one as $\sum_{j=1}^{k} t_j^{off} + t_k^c \leq I_{k-1} + t_{k-1}^{dl}$, $\forall k \in \mathcal{K} \setminus \{1\}$, which suggests the following constraint on t_k^c given by

$$t_k^c \leq I_{k-1} + t_{k-1}^{dl} - \sum_{j=1}^{k} t_j^{off}. \tag{42}$$

As is seen clear that (15) reduces to

$$I_k = I_{k-1} + t_{k-1}^{dl},\ \forall k \in \mathcal{K} \setminus \{1\}, \tag{43}$$

the waiting time for helper k ($k \geq 2$) can be recursively obtained as

$$I_k = I_1 + \sum_{j=1}^{k-1} t_j^{dl}. \tag{44}$$

On the other hand, there are also two possible cases for the optimal I_1 given by (14): case 1) $t_1^{off} + t_1^c < \sum_{k=1}^K t_k^{off}$; and case 2) $t_1^{off} + t_1^c \geq \sum_{k=1}^K t_k^{off}$. Since the kth helper's computation energy E_k^c given by (9), $\forall k \in \mathcal{K}$, monotonically decreases over t_k^c's, if case 1) takes place, it is always possible for helper 1 to slow down its computation such that $t_1^{off} + t_1^c = \sum_{k=1}^K t_k^{off}$ is met without violating the other constraints. In a sum, I_1, w.l.o.g., reduces to

$$I_1 = t_1^{off} + t_1^c, \tag{45}$$

subject to (19a). Combining (44) with (45), and substituting the results for I_{k-1} in (42), (19b) follows.

Furthermore, substituting (45) for I_1 in (44) with $k = K$, it is easy to obtain that $I_K = t_1^{off} + t_1^c + \sum_{j=1}^{K-1} t_j^{dl}$, and thus the completion time T reduces to $T = t_1^{off} + t_1^c + \sum_j t_j^{dl}$. As a result, the total latency given by (17) turns out to be

$$T^{\text{total}} = \max\{t_0^c, t_1^{off} + t_1^c + \sum_{k=1}^K t_k^{dl}\}. \tag{46}$$

In addition, it can also be verified that when the optimal T^{total} given by (46) yields $t_0^c > t_1^{off} + t_1^c + \sum_{k=1}^K t_k^{dl}$, it is always possible for one of the K helpers to slow down its downloading rate such that $t_0^c = t_1^{off} + t_1^c + \sum_{k=1}^K t_k^{dl}$ without violating the other constraints. Therefore, T^{total}, w.l.o.g., can be further simplified as

$$T^{\text{total}} = t_1^{off} + t_1^c + \sum_{k=1}^K t_k^{dl}, \tag{47}$$

subject to (19c).

APPENDIX B
PROOF OF PROPOSITION 4.1

First, given a set of dual variables, we solve (P2-sub1) and (P2-sub2) leveraging some of the Karush-Kuhn-Tucker (KKT) conditions as follows.

$$D_k + \frac{\lambda_0}{\bar{h}_k}\left(f\left(\frac{\sum_{l=1}^L \bar{\pi}(l,k)T_l}{t_k^{off}}\right) - \frac{\sum_{l=1}^L \bar{\pi}(l,k)T_l}{t_k^{off}} f'\left(\frac{\sum_{l=1}^L \bar{\pi}(l,k)T_l}{t_k^{off}}\right)\right) = 0, \tag{48a}$$

$$A_k + \frac{\lambda_k}{\bar{g}_k}\left(f\left(\frac{\sum_{l=1}^{L}\bar{\pi}(l,k)R_l}{t_k^{dl}}\right) - \frac{\sum_{l=1}^{L}\bar{\pi}(l,k)R_l}{t_k^{dl}} f'\left(\frac{\sum_{l=1}^{L}\bar{\pi}(l,k)R_l}{t_k^{dl}}\right)\right) = 0, \tag{48b}$$

$$B_k - 2\lambda_k\kappa_k\left(\frac{\sum_{l=1}^{L}\bar{\pi}(l,k)C_l}{t_k^{c}}\right)^3 = 0, \tag{48c}$$

$$\beta_0 - \zeta_0 - 2\lambda_0\kappa_0\left(\frac{\sum_{l=1}^{L}\bar{\pi}(l,K+1)C_l}{t_0^{c}}\right)^3 = 0. \tag{48d}$$

Since the solution to the equation $f(x) - xf'(x) = y$ for $x > 0$ is shown to be $x = \tilde{f}(-y)$ [27], it is easy to verify that (48a) and (48b) imply (29) with $\bar{\pi}(l,k)$ replaced by $\hat{\pi}(l,k)$ therein. Similarly, (30) follows as a result (48c) and (48d). Note that to keep the objective function of problem (P2) from being infeasible, (48a)-(48d) suggest that $D_k > 0$, $A_k > 0$, $B_k > 0$, $\forall k \in \mathcal{K}$, and $\beta_0 - \zeta_0 > 0$, respectively, which also complies with the domain of the principal branch of Lambert W function.

Next, to find an optimum solution to (27) in terms of $\mathbf{\Pi}$ we substitute $\frac{\sum_{l=1}^{L}\bar{\pi}(l,K+1)C_l}{\sqrt[3]{(\beta_0-\zeta_0)/(2\lambda_0\kappa_0)}}$ for t_0^c in (23), and $\frac{\sum_{l=1}^{L}\bar{\pi}(l,k)T_l}{\tilde{f}\left(D_k\bar{h}_k/\lambda_0\right)}$, $\frac{\sum_{l=1}^{L}\bar{\pi}(l,k)R_l}{\tilde{f}(A_k\bar{g}_k/\lambda_k)}$, and $\frac{\sum_{l=1}^{L}\bar{\pi}(l,k)C_l}{\sqrt[3]{B_k/(2\lambda_k\kappa_k)}}$ for t_k^{off}, t_k^{dl}, and t_k^c, respectively, in (24), such that (22) is expressed in terms of $\bar{\pi}(l,k)$'s. Furthermore, considering $\bar{\pi}(l,k)$'s as the only (primal) variables herein, after some manipulations, the minimization of (22) subject to (18e), (18f), and (20) is formulated as problem (LP1), which can be solved using simplex method. Denoting the optimal solution to (LP1) as $\hat{\pi}(l,k)$'s, by applying the same analysis as shown in (48a)-(48d), the results given by (29) and (30) are thus obtained, which completes the proof.

REFERENCES

[1] H. Xing, L. Liu, J. Xu, and A. Nallanathan, "Joint task assignment and wireless resource allocation for cooperative mobile-edge computing," in *Proc. IEEE International Conference on Communications (ICC)*, Kansas City, MO, USA, May 2018.

[2] S. Barbarossa, S. Sardellitti, and P. D. Lorenzo, "Communicating while computing: Distributed mobile cloud computing over 5G heterogeneous networks," *IEEE Signal Process. Mag.*, vol. 31, no. 6, pp. 45–55, Nov. 2014.

[3] 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, fourth quart. 2017.

[4] ETSI, "Mobile-edge computing–introductory technical white paper," Sep. 2014.

[5] CISCO, "Fog computing and the internet of things: Extend the cloud to where the things are (White Paper)," 2015.

[6] 3GPP, "System architecture for the 5G system (release 15," Jun. 2018.

[7] Y. Mao, J. Zhang, and K. B. Letaief, "Joint task offloading scheduling and transmit power allocation for mobile-edge computing systems," in *Proc. IEEE Wireless Communications and Networking Conference (WCNC)*, San Francisco, CA, USA, Mar. 2017.

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

[9] X. Cao, F. Wang, J. Xu, R. Zhang, and S. Cui, "Joint computation and communication cooperation for energy-efficient mobile edge computing," *to appear in IEEE Internet Things J.*, 2018.

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

[11] M. Chen, M. Dong, and B. Liang, "Resource sharing of a computing access point for multi-user mobile cloud offloading with delay constraints," *IEEE Trans. Mobile Comput.*, vol. 17, no. 12, pp. 2868–2881, Dec. 2018.

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

[14] F. Wang, J. Xu, and Z. Ding, "Multiple-antenna NOMA for computation offloading in multiuser mobile edge computing systems," *to appear in IEEE Trans. Commun.*, 2018.

[15] A. Al-Shuwaili and O. Simeone, "Energy-efficient resource allocation for mobile edge computing-based augmented reality applications," *IEEE Wireless Commun. Lett.*, vol. 6, no. 3, pp. 398–401, Jun. 2017.

[16] X. He, H. Xing, Y. Chen, and A. Nallanathan, "Energy-efficient mobile-edge computation offloading for applications with shared data," *to appear in Proc. IEEE Global Communications Conference (GLOBECOM)*, 2018. [Online]. Available: https://arxiv.org/abs/1809.00966

[17] 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.

[18] J. Xu and J. Yao, "Exploiting physical-layer security for multiuser multicarrier computation offloading," *to appear in IEEE Wireless Commun. Lett.*, 2018.

[19] C. Shi, V. Lakafosis, M. H. Ammar, and E. W. Zegura, "Serendipity: Enabling remote computing among intermittently connected mobile devices," in *Proc. ACM International Symposium on Mobile Ad Hoc Networking and Computing (MobiHoc)*, Hilton Head, S.C., USA, Jun. 2012.

[20] X. Chen, L. Pu, L. Gao, W. Wu, and D. Wu, "Exploiting massive D2D collaboration for energy-efficient mobile edge computing," *IEEE Wireless Commun.*, vol. 24, no. 4, pp. 64–71, Aug. 2017.

[21] Y. Kao, B. Krishnamachari, M. Ra, and F. Bai, "Hermes: Latency optimal task assignment for resource-constrained mobile computing," *IEEE Trans. Mobile Comput.*, vol. 16, no. 11, pp. 3056–3069, Nov. 2017.

[22] T. X. Tran and D. Pompili, "Joint task offloading and resource allocation for multi-server mobile-edge computing networks," 2017. [Online]. Available: https://arxiv.org/abs/1705.00704

[23] W. Alsalih, S. Akl, and H. Hassancin, "Energy-aware task scheduling: towards enabling mobile computing over manets," in *Proc. IEEE International Parallel and Distributed Processing Symposium (PDPS)*, Denver, Colorado, USA, Apr. 2005.

[24] L. Pu, X. Chen, J. Xu, and X. Fu, "D2D fogging: An energy-efficient and incentive-aware task offloading framework via network-assisted D2D collaboration," *IEEE J. Sel. Areas Commun.*, vol. 34, no. 12, pp. 3887–3901, Dec. 2016.

[25] P. Mach and Z. Becvar, "Mobile edge computing: A survey on architecture and computation offloading," *IEEE Commun. Surveys Tuts.*, vol. 19, no. 3, pp. 1628–1656, third quart. 2017.

[26] M. Grant and S. Boyd, "CVX: Matlab software for disciplined convex programming, version 2.1," Mar. 2014. [Online]. Available: http://cvxr.com/cvx

[27] R. Corless, G. Gonnet, D. Hare, D. Jeffrey, and D. Knuth, "On the Lambert W function," *Adv. Comput. Math.*, vol. 5, no. 1, pp. 329–359, Dec. 1996.

[28] S. Boyd, "Lecture notes for EE364b: Convex Optimization II." [Online]. Available: https://stanford.edu/class/ee364b/lectures.html

[29] K. G. Murty, *Linear Programming*. John Wiley & Sons, 1983.

[30] D. A. Spielman and S.-H. Teng, "Smoothed analysis of algorithms: Why the simplex algorithm usually takes polynomial time," *J. ACM*, vol. 51, no. 3, pp. 385–463, May 2004.

[31] Z. Ding, X. Lei, G. K. Karagiannidis, R. Schober, J. Yuan, and V. K. Bhargava, "A survey on non-orthogonal multiple access for 5G networks: Research challenges and future trends," *IEEE J. Sel. Areas Commun.*, vol. 35, no. 10, pp. 2181–2195, Oct. 2017.