# Multi-user Multi-task Offloading and Resource Allocation in Mobile Cloud Systems

Meng-Hsi Chen, Ben Liang, *Fellow, IEEE*, Min Dong, *Senior Member, IEEE*

***Abstract*—We consider a general multi-user Mobile Cloud Computing (MCC) system where each mobile user has multiple independent tasks. These mobile users share the computation and communication resources while offloading tasks to the cloud. We study both the conventional MCC where tasks are offloaded to the cloud through a wireless access point, and MCC with a computing access point (CAP), where the CAP serves both as the network access gateway and a computation service provider to the mobile users. We aim to jointly optimize the offloading decisions of all users as well as the allocation of computation and communication resources, to minimize the overall cost of energy, computation, and delay for all users. The optimization problem is formulated as a non-convex quadratically constrained quadratic program, which is NP-hard in general. For the case without a CAP, an efficient approximate solution named MUMTO is proposed by using separable semidefinite relaxation (SDR), followed by recovery of the binary offloading decision and optimal allocation of the communication resource. To solve the more complicated problem with a CAP, we further propose an efficient three-step algorithm named MUMTO-C comprising of generalized MUMTO SDR with CAP, alternating optimization, and sequential tuning, which always computes a locally optimal solution. For performance benchmarking, we further present numerical lower bounds of the minimum system cost with and without the CAP. By comparison with this lower bound, our simulation results show that the proposed solutions for both scenarios give nearly optimal performance under various parameter settings, and the resultant efficient utilization of a CAP can bring substantial cost benefit.**

***Index Terms*—mobile cloud computing, computing access point, task offloading, resource allocation, energy cost, delay cost, computation cost.**

## I. Introduction

Mobile Cloud Computing (MCC) brings abundant cloud resources to extend the capabilities of resource-limited mobiles devices to improve the user experience [3] [4]. With the help of cloud resources, mobile devices can potentially reduce their energy consumption or processing delay by offloading their tasks to the cloud. However, integration between mobile devices and the cloud may affect the quality of service of those offloaded tasks and overall mobile device energy usage due to additional communication and computation delays and transceiver energy consumption.

Meng-Hsi Chen and Ben Liang are with the Department of Electrical and Computer Engineering, University of Toronto, Toronto, Canada (e-mail: {mchen, liang}@ece.utoronto.ca).

Min Dong is with the Department of Electrical, Computer and Software Engineering, University of Ontario Institute of Technology, Oshawa, Canada (e-mail: min.dong@uoit.ca).

This work has been funded in part by a Natural Sciences and Engineering Research Council (NSERC) of Canada Strategic Project Grant and in part by NSERC Discovery Grants.

Preliminary result of this work has appeared in [1] and [2].

The case of a single mobile user offloading its entire application to the cloud was studied in [5], [6]. Furthermore, offloading by multiple mobile users was considered in [7]–[9], where each user has a single application or task to be offloaded to the cloud in its entirety. Different from such whole-application offloading, the authors of [10]–[16] considered partitioning an application into multiple tasks. In all these cases, the partitioning problem results in integer programming that is NP-hard in general.

In conventional MCC systems, communication between the mobile devices and the remote cloud server often is over a long distance, which may result in large communication delay in task offloading. In contrast, with an aim to reduce the communication delay for those offloaded tasks, Mobile/Multiaccess Edge Computing (MEC), as defined by the European Telecommunications Standards Institute, refers to a distributed MCC system where computing resources are installed locally at or near the base station of a cellular network [17]–[19]. MEC shares similarities with micro cloud centers [20], cloudlets [21], fog computing [22], and cyber-foraging [23], except that the MEC computing servers are managed by a mobile service provider, which allows more direct control and resource management. Similar to the concept of MEC, one may define a general computing access point (CAP), which is a wireless access point or a cellular base station with built-in computation capability to serve the mobile users' computing tasks. These tasks may be processed locally at the mobile devices, sent to the CAP, or further forwarded to a remote cloud server. With the additional option of computation by the CAP, we can reduce the need for access to the remote cloud server, hence decreasing the communication delay and also potentially the overall energy and computation cost.

In this work, we study the joint optimization of task offloading and resource allocation in a general mobile cloud access network consisting of multiple mobile users, each having multiple independent tasks. The wireless access point may serve its conventional networking function and only forward the received tasks to the remote cloud server, or it may be a CAP that additionally has limited built-in computation capability to directly process some of the tasks by itself. We take into consideration the computation and communication energies, CAP and cloud usage costs, and communication and processing delays at local user devices, the CAP, and the remote cloud server.

The multi-user multi-task scenario adds substantial challenge to system design, since we need to jointly consider both the offloading decisions and the sharing of limited computation and communication resources among all users as they compete

with each other while offloading tasks. In particular, the delays of the offloaded tasks of a user will be affected by its assigned computation and communication resources, as well as the scheduling of those tasks in the computation and communication pipelines. Therefore, scheduling the tasks of even a single user contains a multi-machine flow-shop problem [24], which has no known optimal solution in the literature. In this work, we propose efficient heuristic solutions based on semi-definite relaxation methods, together with delay bounding techniques, iterative optimization, and further sequential performance tuning, which are numerically shown to provide nearly optimal performance. The contributions of this work are summarized below:

- *Conventional MCC with a non-computing AP*: We first consider the conventional MCC with a non-computing AP, and formulate the problem to jointly optimize the offloading decision and the communication resource allocation of all tasks, to minimize a weighted sum of the costs of energy, computation, and delay for all users. The resulting mixed integer programming problem can be reformulated as a non-convex quadratically constrained quadratic program (QCQP) [25], which is NP-hard in general. To solve this challenging problem, we first present a performance bounding framework that utilizes both the upper and lower bounds of the multi-task total communication and computation delay for each user. We then propose an efficient Multi-User Multi-Task Offloading (MUMTO) algorithm based on separable semidefinite relaxation (SDR) [26], with recovery of the binary offloading decision and subsequent optimal allocation of the communication resource.
- *MCC with a CAP*: We next consider the presence of a CAP in the MCC, aiming to jointly optimize the task offloading decisions and the allocation of computation and communication resources of all tasks. However, the availability of CAP computation further complicates mobile task offloading decisions, adding an extra dimension of variability at the CAP. To solve this challenging problem, we further propose an efficient three-step algorithm named MUMTO with CAP (MUMTO-C), which first utilizes a generalized version of the MUMTO SDR with an added CAP, and then performs additional alternating optimization and sequential tuning. We show that it always computes a locally optimal solution, which contains the binary offloading decision and subsequent optimal allocation of the computation and communication resources.
- *Lower bounds and performance*: For both two scenarios considered above, we obtain lower bounds for the minimum cost as the benchmark for performance evaluation. Simulation results show that MUMTO and MUMTO-C both give nearly optimal performance under various parameter settings. Furthermore, for the case with a CAP, we conduct simulation experiments on alternative combinations of the three components of the MUMTO-C algorithm, clarifying their roles and contributions to the overall system performance. Finally, we compare the performance of MUMTO-C against that of purely local processing, purely cloud processing, and hybrid local-cloud processing without the CAP, which demonstrates the effectiveness of the proposed algorithm in joint management of the computation and communication resources in the three-tier computing system of local devices, CAP, and remote cloud server.

*Organization*: The rest of this paper is organized as follows. In Section II, we discuss the related work. In Section III, we describe the system model. In Section IV, we provide details of the problem formulation, the proposed algorithm, and the lower bound of minimum system cost for the conventional MCC without a CAP. In Section V, the impact on the presence of the CAP is further studied. We present numerical results in Section VI and conclude in Section VII.

*Notations*: Trace and transpose of matrix $\mathbf{A}$ are denoted by $\mathrm{Tr}(\mathbf{A})$ and $\mathbf{A}^T$, respectively. A positive semi-definite matrix $\mathbf{A}$ is denoted as $\mathbf{A} \succcurlyeq 0$. Notation $\mathrm{diag}(\mathbf{a})$ denotes a diagonal matrix with diagonal elements being elements of vector $\mathbf{a}$, and $\mathbf{A}(i,j)$ denotes the $(i,j)\mathrm{th}$ entry of matrix $\mathbf{A}$.

## II. Related Work

### A. Two-Tier Offloading System

Many existing studies focus on two-tier cloud networks with mobile users and another tier of external processors.

For a single user offloading its entire application, the tradeoff between energy saving and computing performance was studied in [5], [6], [27]. Different from the above whole-application offloading, the authors of [10]–[16] considered partitioning an application into multiple tasks. Specifically, the authors of [10]–[12] focus on the implementation of offloading mechanisms from the mobile device to the cloud, while the discussion on optimizing the offloading decisions was limited. In [13], a heuristic offloading policy was proposed for a mobile user with sequential tasks. In [14]–[16], the problem of cloud offloading for a mobile user with dependent tasks was studied. All of the above studies focus on the single-user case.

The case of task offloading by multiple mobile users has been considered in [7]–[9], [28]–[31], but in all of these works, each user only has a single task to process. Without considering resource allocation, the authors of [7], [9], [28], [29] proposed different approaches to obtain the offloading decisions for each user. In [8], [30], when all tasks are always offloaded, the authors optimized the allocation of computation and communication resources. In contrast to the above studies, instead of optimizing either the offloading decision only or the resource allocation only, in this work we study the joint optimization as they are inter-dependent. The authors of [31] considered the joint allocation of offloading decision and resource allocation with a sequential optimization heuristic. The method can only be applied to the case where each user has a single task. In contrast, in this work, the system design is much more challenging since we consider the general multi-user multi-task scenario.

### B. Three-Tier Offloading System

Besides the two-tier cloud networks above, the three-tier network model, consisting of mobile users, a local computing node (e.g., cloudlet or CAP), and a remote cloud server, has

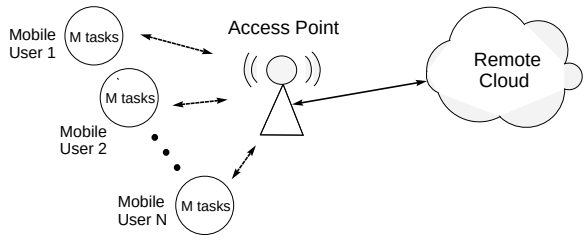



Fig. 1. Multi-user multi-task offloading system model. The AP may serve its conventional networking function and forward tasks to the remote cloud server, or be a CAP with built-in computation capability to directly process some received tasks by itself.

been studied in [32]–[38]. Compared with two-tier systems, the three-tier system adds extra flexibility for task offloading. In [32]–[35], the authors only focused on optimizing the offloading decisions without considering the allocation of computation and communication resources. However, since both computation and communication resources are limited and shared among all users, without efficiently allocating those limited resources to different users, the full benefit of task offloading cannot be realized. The joint optimization of the offloading decision and the allocation of computation and communication resources for a general three-tier multi-user multi-task offloading system has not been investigated before, and it is much more complicated to solve.

We have previously studied the scheduling of computation and communication resources in a CAP for *a single mobile user* [36] and multiple mobile users each with *a single task* only [37], [38], showing substantial system performance improvement under such simplified system models. In this work, we focus on the joint optimization problem for a general *multi-user multi-task* scenario.

## III. System Model

### A. Mobile Cloud Offloading with Multiple Users and Tasks

Consider a general cloud access network consisting of one cloud server, one AP, and $N$ mobile users, each having $M$ independent tasks, as shown in Fig. 1.[1] Examples of the AP may be a cellular base station or a WiFi access point. The AP may serve its conventional networking function and forward received tasks to the remote cloud server, or directly process some of the tasks by itself when it has built-in computation capability. In the latter case, we name it CAP. In this work, we first study a conventional mobile cloud offloading scenario without the CAP, aiming to obtain optimal offloading decisions for all mobile users' tasks as well as resource allocation. Then, we will further study a more general scenario with the presence of the CAP, showing substantial system performance improvement. Notice that we do not consider any specific queueing model for each user's tasks. We will show latter in Sections IV-C and V-C that our proposed solutions are generally applicable to any queueing discipline.

[1]We assume the same number of tasks $M$ for all users only for the notation simplicity. Our proposed solutions can be easily extended to the general scenario where each mobile user has a different number of tasks $M_i$.

TABLE I
List of Main Symbols

| Symbol | Description |
|---|---|
| $E_{ij}^l$ | local processing energy of user $i$'s task $j$ |
| $E_{ij}^t$, $E_{ij}^r$ | uplink transmitting energy and downlink receiving energy of user $i$'s task $j$ between the mobile user and the AP |
| $T_{ij}^l$, $T_{ij}^a$, $T_{ij}^c$ | local processing time, CAP processing time, and cloud processing time of user $i$'s task $j$ |
| $T_{ij}^t$, $T_{ij}^r$ | uplink transmission time and downlink transmission time of user $i$'s task $j$ between the mobile user and the AP |
| $T_{ij}^{ac}$ | transmission time of user $i$'s task $j$ between the AP and the cloud |
| $C_{\text{UL}}$, $C_{\text{DL}}$ | uplink bandwidth and downlink bandwidth for transmission between mobile users and the AP |
| $C_{\text{Total}}$ | total transmission bandwidth between mobile users and the AP |
| $c_i^u$, $c_i^d$ | uplink bandwidth and downlink bandwidth assigned to user $i$ |
| $\eta_i^u$, $\eta_i^d$ | spectral efficiency of uplink and downlink transmission between user $i$ and the AP |
| $r^{ac}$ | transmission rate between the AP and the cloud |
| $f_i^a$ | CAP processing rate assigned to user $i$'s tasks |
| $f_A$ | total CAP processing rate |
| $f^c$ | cloud processing rate for each user |
| $C_{ij}^a$ | CAP usage cost of user $i$'s task $j$ |
| $C_{ij}^c$ | cloud usage cost of user $i$'s task $j$ |

*Remark*: Our system model is a common one considered in many previous studies [7], [9], [13], [14], [16], [28]–[31], [34], where all $M$ tasks of each user are assumed to be available at the starting time. For a dynamic system where the tasks arrive at different times, we may apply our model and the proposed solution in a quasi-static manner, where the system processes the tasks in batches as they are collected [39]. Also, note that for the mobile cloud system considered, we focus on the bandwidth sharing of wireless communication links among users, while assuming that the statistics of each wireless link remain unchanged during the processing of the users' tasks. This reflects a relatively static or low-mobility scenario. The mobility issue and its effect on the offloading performance is not considered this work and will be left for future work.

The main symbols used in the system model are summarized in Table I.

### B. Cost of Local Processing

We denote by $D_{\text{in}}(ij)$, $D_{\text{out}}(ij)$, and $Y(ij)$ the input data size, output data size, and processing cycles[2] of user $i$'s task $j$, respectively.[3] For task $j$ being locally processed by user $i$, the corresponding energy consumed for processing is denoted by $E_{ij}^l$ and the processing time is denoted by $T_{ij}^l$.

### C. Cost of Remote Cloud Processing

When user $i$'s task $j$ is offloaded to the AP, we denote by $E_{ij}^t$ and $E_{ij}^r$, respectively, the energy consumed

[2]The processing cycles of user $i$'s task $j$ depends on the input data size and the application type.

[3]These quantities may be obtained by applying a program profiler [10]–[12], as similarly used in [6], [7], [9], [27], [29]–[31].

for wireless transmission and reception by the user. For the wireless connections between mobile users and the AP, we denote the uplink and downlink transmission times by $T_{ij}^t = D_{\text{in}}(ij)/(\eta_i^u c_i^u)$ and $T_{ij}^r = D_{\text{out}}(ij)/(\eta_i^d c_i^d)$, respectively, where $c_i^u$ and $c_i^d$ are uplink and downlink bandwidth allocated to user $i$, and $\eta_i^u$ and $\eta_i^d$ are the spectral efficiency of uplink and downlink transmission between user $i$ and the AP, respectively[4]. We have the following constraints on $c_i^u$ and $c_i^d$ as they are limited by the uplink bandwidth $C_{\text{UL}}$ and downlink bandwidth $C_{\text{DL}}$

$$\sum_{i=1}^{N} c_i^u \leq C_{\text{UL}}, \tag{1}$$

and

$$\sum_{i=1}^{N} c_i^d \leq C_{\text{DL}}. \tag{2}$$

We may consider also a total bandwidth constraint

$$\sum_{i=1}^{N} (c_i^u + c_i^d) \leq C_{\text{Total}}. \tag{3}$$

Since the AP has to further offload the task to the cloud, there is the additional transmission time between the AP and the cloud denoted by $T_{ij}^{ac} = (D_{\text{in}}(ij) + D_{\text{out}}(ij))/r^{ac}$, and the cloud processing time denoted by $T_{ij}^c = Y(ij)/f^c$, where $r^{ac}$ is the transmission rate between the AP and the cloud and $f^c$ is the cloud processing rate *for each user*. The rate $r^{ac}$ is assumed to be a pre-determined value regardless of the number of users, since the AP-cloud link is likely to be a high-capacity wired connection in comparison with the limited wireless links between the mobile users and the AP, so that there is no need to consider bandwidth sharing among the users. Similarly, $f^c$ is also assumed to be a pre-determined value because of the high computational capacity and dedicated service of the remote cloud server. Thus, $T_{ij}^{ac}$ and $T_{ij}^c$ only depend on user $i$'s task $j$ itself.

Finally, the cloud usage cost of processing user $i$'s task $j$ at the cloud is denoted by $C_{ij}^c$. The usage cost may depend on the data size and processing cycles of a task and the hardware and energy cost to maintain the cloud server, but such detail is outside the scope of this work. Here we simply assume that $C_{ij}^c$ is given for all $i$ and $j$.

### D. Cost of CAP Processing

When we consider the presence of a CAP, some of the offloaded tasks can be directly processed by the CAP. If user $i$'s task $j$ is processed by the CAP (i.e., instead of being further forwarded to the remote cloud), besides the communication energy (i.e., $E_{ij}^t$ and $E_{ij}^r$) and delay (i.e., $T_{ij}^t$ and $T_{ij}^r$) mentioned above, we denote the CAP processing time by $T_{ij}^a = Y(ij)/f_i^a$, where $f_i^a$ is the assigned processing rate, which is limited by the total processing rate $f_A$ at the CAP

$$\sum_{i=1}^{N} f_i^a \leq f_A. \tag{4}$$

[4]The spectral efficiency can be approximated by $\log(1+\text{SNR})$ where SNR is the link quality between user $i$ and the AP.

Similarly, denote the CAP usage cost of processing user $i$'s task $j$ at the CAP by $C_{ij}^a$. In the following, we first study the conventional MCC without considering the CAP. The impact on the presence of a CAP will be further studied in Section V.

## IV. Multi-user Multi-task Offloading without CAP

In this section, we study the conventional mobile cloud network where the AP always forwards the received tasks to the remote cloud server. In this case, we have a two-tier offloading system, and we focus on jointly optimizing the offloading decision and the communication resource allocation of all tasks, to minimize a weighted sum of the costs of energy, computation, and the delay for all users.

### A. Offloading Decision

Since there is no CAP, each mobile user can either process its tasks locally or offload some of them to the cloud for processing through the AP. Let $x_{ij}$ denote the offloading decision for task $j$ of user $i$, given by

$$x_{ij} = \begin{cases} 0, & \text{process task } j \text{ of user } i \text{ locally;} \\ 1, & \text{offload task } j \text{ of user } i \text{ to the cloud.} \end{cases}$$

### B. Problem Formulation

We aim at reducing mobile users' energy consumption and maintain the service quality of processing their tasks, measured by the delays incurred due to transmission and/or processing. For this goal, we define the total system cost as the weighted sum of total energy consumption, the costs to offload and process all tasks, and the corresponding transmission and processing delays for all users. Our objective is to minimize the total system cost by jointly optimizing the task offloading decisions $x_{ij}$ and the communication bandwidth resource allocation $\mathbf{r}_i = [c_i^u,\ c_i^d]^T$. This optimization problem is formulated as follows:

$$\min_{\{x_{ij}\},\{\mathbf{r}_i\}} \sum_{i=1}^{N} \left[ \sum_{j=1}^{M} (E_{ij}^l (1 - x_{ij}) + E_{ij}^C x_{ij}) + \rho_i \max\{T_i^L, T_i^C\} \right] \tag{5}$$

$$\text{s.t. } (1), (2), (3),$$

$$r_i^u, r_i^d, \geq 0, \forall i, \tag{6}$$

$$x_{ij} \in \{0, 1\}, \forall i, j, \tag{7}$$

where $E_{ij}^C \triangleq (E_{ij}^t + E_{ij}^r + \beta C_{ij}^c)$ is the weighted transmission energy and processing cost of offloading and processing task $j$ of user $i$ to the cloud, with $\beta$ being the relative weight; in addition, $T_i^L$ is the processing delay of tasks processed by the mobile user $i$ itself, $T_i^C$ is the overall transmission and remote-processing delay for tasks of mobile user $i$ processed at the cloud, and $\rho_i$ is the weight on the task processing delay relative to energy consumption in the total system cost.

Depending on the performance requirement, the value of $\rho_i$ can be adjusted to impose different emphasis on delay and

energy consumption.[5] The proposed optimization problem (5) can be solved by any controller in this network after collecting all required information. In practice, the controller could be the AP. That is, each user provides its information to the AP, and the AP broadcasts the obtained offloading decisions (and the corresponding resource allocations) to all users by solving problem (5).

The above mixed-integer programming problem is difficult to solve in general. Based on the offloading decision $x_{ij}$ for each task, we have the total local processing delay for each user $T_i^L = \sum_{j=1}^{M} T_{ij}^l(1-x_{ij})$, for all $i$. However, we note that the overall delay for remote processing, $T_i^C$, is challenging to calculate exactly. This is because, when there are multiple tasks offloaded by a users, the transmission times and processing times may overlap in an unpredictable manner, which depends on the offloading decision, communication resource allocation, and task scheduling order. In fact, since $T_i^C$ consists of the uplink transmission times, remote-processing time, and downlink transmissions times of all tasks, it may be viewed as the output of a multi-machine flowshop schedule, which remains an open research problem [24]. Since $T_i^C$ is not precisely tractable, we will use both upper and lower bounds of $T_i^C$ in our proposed solution and performance benchmarking. Under the MUMTO algorithm, they are shown to give total system costs that are close to each other.

### C. Multi-user Multi-task Offloading (MUMTO) Algorithm

The joint optimization problem (5) is a mixed-integer non-convex programming problem. To find an efficient solution to the original problem (5), in the following, we first propose both upper bound and lower bound formulations of $T_i^C$, then transform the optimization problem (5) into a separable QCQP, and finally propose a separable SDR approach to obtain the binary offloading decisions $\{x_{ij}\}$ and the communication resource allocation $\{\mathbf{r}_i\}$.

*1) **Bounds of Remote-Processing Delay**:* When a mobile user offloads more than one task to the cloud, there will be overlaps in the communication and processing times as mentioned above, making it difficult to exactly characterize the overall delay $T_i^C$. However, we have the following upper bound of $T_i^C$ as the *worst-case delay* formulation:

$$T_i^{C_{(U)}} = \sum_{j=1}^{M} \left( \frac{D_{\text{in}}(ij)}{\eta_i^u c_i^u} + \frac{D_{\text{out}}(ij)}{\eta_i^d c_i^d} + T_{ij}^{ac} + T_{ij}^{c} \right) x_{ij}, \ \forall i. \quad (8)$$

Since the worst-case delay sums the transmission delays and processing delays together without any overlap, it will always be greater than the real delay given the same offloading decision and resource allocation. On the other hand, we separate the offloading delays of all mobile users into several components and only consider the largest one as the lower bound of $T_i^C$:

$$T_i^{C_{(L)}} = \max\{T_i^u, T_i^d, T_i^{uac}, T_i^{dac}, T_i^{c'}\}, \ \forall i, \quad (9)$$

where $T_i^u = \sum_{j=1}^{M} D_{\text{in}}(ij)x_{ij}/(\eta_i^u c_i^u)$ and $T_i^d = \sum_{j=1}^{M} D_{\text{out}}(ij)x_{ij}/(\eta_i^d c_i^d)$ are total uplink and downlink transmission times between the user and the AP for user $i$, respectively, $T_i^{uac} = \sum_{j=1}^{M} D_{\text{in}}(ij)x_{ij}/r^{ac}$ and $T_i^{dac} = \sum_{j=1}^{M} D_{\text{out}}(ij)x_{ij}/r^{ac}$ are total uplink and downlink transmission times between the AP and the cloud for user $i$, respectively, and $T_i^{c'} = \sum_{j=1}^{M} Y(ij)x_{ij}/f^c$ is the total cloud processing time for user $i$.

In the following, we will use the worst-case delay $T_i^{C_{(U)}}$ in optimization problem (5) to obtain an approximate solution, which can provide an upper bound to the total system cost. We then use $T_i^{C_{(L)}}$ similarly, to obtain a lower bound of the total system cost, for performance benchmarking. In Section VI, by comparing both cases, we show that the MUMTO algorithm based on the worst case formulation gives nearly optimal performance.

*2) **QCQP Transformation and Semidefinite Relaxation**:* We first replace $T_i^C$ with $T_i^{C_{(U)}}$ in problem (5), and rewrite the integer constraint (7) as

$$x_{ij}(x_{ij}-1) = 0, \ \forall i, j. \quad (10)$$

We also introduce a additional auxiliary variable $t_i$ for $\max\{T_i^L, T_i^{C_{(U)}}\}$, the problem (5) is now transformed into the following equivalent problem:

$$\min_{\{x_{ij}\},\{\mathbf{r}_i,t_i\}} \sum_{i=1}^{N} \left[ \sum_{j=1}^{M} (E_{ij}^l(1-x_{ij}) + E_{ij}^C x_{ij}) + \rho_i t_i \right] \quad (11)$$

$$\text{s.t.} \ \sum_{j=1}^{M} T_{ij}^l(1-x_{ij}) \le t_i, \ \forall i, \quad (12)$$

$$\sum_{j=1}^{M} \left( \frac{D_{\text{in}}(ij)}{\eta_i^u c_i^u} + \frac{D_{\text{out}}(ij)}{\eta_i^u c_i^d} + T_{ij}^{ac} + T_{ij}^{c} \right) x_{ij} \le t_i, \forall i, \quad (13)$$

$$(1), (2), (3), (6), \text{ and } (10).$$

In order to obtain the eventual SDR formulation, we first transform the optimization problem (11) into a separable QCQP problem by the following steps.

First, we introduce two auxiliary variables $D_i^u$ and $D_i^d$, and replace constraint (13) with the following equivalent constraints:

$$D_i^u + D_i^d + \sum_{j=1}^{M} (T_{ij}^{ac} + T_{ij}^{c}) x_{ij} \le t_i, \ \forall i, \quad (14)$$

$$\sum_{j=1}^{M} \frac{D_{\text{in}}(ij)x_{ij}}{\eta_i^u c_i^u} \le D_i^u, \ \forall i, \quad (15)$$

and

$$\sum_{j=1}^{M} \frac{D_{\text{out}}(ij)x_{ij}}{\eta_i^d c_i^d} \le D_i^d, \ \forall i, \quad (16)$$

where (14) is the overall offloading delay constraint, and (15) and (16) correspond to the uplink transmission time and the downlink transmission time, respectively.

[5] To avoid mathematical redundancy, we only put the weight in front of the delay and normalize the weighted sum cost to have the unit of energy. However, it can be easily extended to an objective with some arbitrary unit (e.g., dollars).

Next, we vectorize the variables and parameters in problem (11). Define

$$\mathbf{w}_i \triangleq [x_{i1}, \ldots, x_{iM}, c_i^u, D_i^u, c_i^d, D_i^d, t_i]^T, \ \forall i, \tag{17}$$

which is the decision vector for user $i$ with all decision variables. Then, the objective in problem (11) can be rewritten as

$$\sum_{i=1}^{N} \mathbf{b}_i^T \mathbf{w}_i + \sum_{i=1}^{N}\sum_{j=1}^{M} E_{ij}^l, \tag{18}$$

where $\mathbf{b}_i \triangleq [(E_{i1}^C - E_{i1}^l), \ldots, \ (E_{iM}^C - E_{iM}^l), \ \mathbf{0}_{1\times 4}, \ \rho_i]^T$. We rewrite the local processing delay constraint (12) as

$$(\mathbf{b}_i^l)^T \mathbf{w}_i \leq -\sum_{j=1}^{M} T_{ij}^l, \ \forall i, \tag{19}$$

where $\mathbf{b}_i^l \triangleq -[T_{i1}^l, \ldots, \ T_{iM}^l, \ \mathbf{0}_{1\times 4}, \ 1]^T$. For the cloud processing delay constraint (14), it can be rewritten as

$$(\mathbf{b}_i^c)^T \mathbf{w}_i \leq 0, \ \forall i, \tag{20}$$

where $\mathbf{b}_i^c \triangleq [(T_{i1}^{ac} + T_{i1}^c), \ldots, \ (T_{iM}^{ac} + T_{iM}^c), 0, 1, 0, 1, -1]^T$. The matrix forms of constraints (15) and (16) are

$$\mathbf{w}_i^T \mathbf{A}_i^\mu \mathbf{w}_i + (\mathbf{b}_i^\mu)^T \mathbf{w}_i \leq 0, \quad \mu \in \{u, d\}, \ \forall i, \tag{21}$$

where

$$\mathbf{A}_i^{\mu'} \triangleq -\frac{1}{2}\begin{bmatrix} 0 & \eta_i^\mu \\ \eta_i^\mu & 0 \end{bmatrix}, \quad \mu \in \{u, d\},$$

$$\mathbf{A}_i^u \triangleq \begin{bmatrix} \mathbf{0}_{M\times M} & \mathbf{0}_{M\times 2} & \mathbf{0}_{M\times 3} \\ \mathbf{0}_{2\times M} & \mathbf{A}_i^{u'} & \mathbf{0}_{2\times 3} \\ \mathbf{0}_{3\times M} & \mathbf{0}_{3\times 2} & \mathbf{0}_{3\times 3} \end{bmatrix},$$

$$\mathbf{A}_i^d \triangleq \begin{bmatrix} \mathbf{0}_{(M+2)\times(M+2)} & \mathbf{0}_{(M+2)\times 2} & \mathbf{0}_{(M+2)\times 1} \\ \mathbf{0}_{2\times(M+2)} & \mathbf{A}_i^{d'} & \mathbf{0}_{2\times 1} \\ \mathbf{0}_{1\times(M+2)} & \mathbf{0}_{1\times 2} & 0 \end{bmatrix},$$

$$\mathbf{b}_i^u \triangleq [\mathbf{D}_{\text{in}}(i1), \ldots, \ \mathbf{D}_{\text{in}}(iM), \ \mathbf{0}_{1\times 5}]^T,$$

$$\mathbf{b}_i^d \triangleq [\mathbf{D}_{\text{out}}(i1), \ldots, \ \mathbf{D}_{\text{out}}(iM), \ \mathbf{0}_{1\times 5}]^T.$$

The uplink and downlink bandwidth resource constraints (1) and (2) correspond to

$$\sum_{i=1}^{N} (\mathbf{b}_i^U)^T \mathbf{w}_i = C_{\text{UL}}, \tag{22}$$

and

$$\sum_{i=1}^{N} (\mathbf{b}_i^D)^T \mathbf{w}_i = C_{\text{DL}}, \tag{23}$$

respectively, where $\mathbf{b}_i^U \triangleq [\mathbf{0}_{1\times M}, \ 1, \ \mathbf{0}_{1\times 4}]^T$ and $\mathbf{b}_i^D \triangleq [\mathbf{0}_{1\times M+2}, \ 1, \ \mathbf{0}_{1\times 2}]^T$. Similarly, the total bandwidth constraint (3) is rewritten as

$$\sum_{i=1}^{N} (\mathbf{b}_i^S)^T \mathbf{w}_i \leq C_{\text{Total}}, \tag{24}$$

where $\mathbf{b}_i^S \triangleq [\mathbf{0}_{1\times M}, 1, 0, 1, 0, 0]^T$. The constraint (6) used to ensure all variables great than or equal to 0 is replaced by

$$\mathbf{w}_i \succcurlyeq 0, \ \forall i. \tag{25}$$

Finally, we rewrite the integer constraint (10) as

$$\mathbf{w}_i^T \text{diag}(\mathbf{e}_j) \mathbf{w}_i - \mathbf{e}_j^T \mathbf{w}_i = 0, \ \forall i, j, \tag{26}$$

where $\mathbf{e}_j$ as the $(M+5)\times 1$ standard unit vector with the $j$th entry being 1.

By further defining $\mathbf{z}_i \triangleq [\mathbf{w}_i^T, 1]^T$, together with the above matrix form presentations, and dropping the constant term $\sum_{i=1}^{N}\sum_{j=1}^{M} E_{ij}^l$ from the objective function in (18), problem (11) can now be further transformed into the following homogeneous separable QCQP formulation:

$$\min_{\{\mathbf{z}_i\}} \quad \sum_{i=1}^{N} \mathbf{z}_i^T \mathbf{G}_i \mathbf{z}_i \tag{27}$$

$$\text{s.t.} \quad \mathbf{z}_i^T \mathbf{G}_i^l \mathbf{z}_i \leq -\sum_{j=1}^{M} T_{ij}^l, \ \forall i, \tag{28}$$

$$\mathbf{z}_i^T \mathbf{G}_i^c \mathbf{z}_i \leq 0, \ \forall i, \tag{29}$$

$$\mathbf{z}_i^T \mathbf{G}_i^\mu \mathbf{z}_i \leq 0, \ \mu \in \{u, d\}, \ \forall i, \tag{30}$$

$$\sum_{i=1}^{N} \mathbf{z}_i^T \mathbf{G}_i^U \mathbf{z}_i \leq C_{\text{UL}}, \ \sum_{i=1}^{N} \mathbf{z}_i^T \mathbf{G}_i^D \mathbf{z}_i \leq C_{\text{DL}}, \tag{31}$$

$$\sum_{i=1}^{N} \mathbf{z}_i^T \mathbf{G}_i^S \mathbf{z}_i \leq C_{\text{Total}}, \tag{32}$$

$$\mathbf{z}_i^T \mathbf{G}_j^I \mathbf{z}_i = 0, \ \forall i, j, \tag{33}$$

$$\mathbf{z}_i \succcurlyeq 0, \ \forall i, \tag{34}$$

where

$$\mathbf{G}_i \triangleq \begin{bmatrix} \mathbf{0} & \frac{1}{2}\mathbf{b}_i \\ \frac{1}{2}\mathbf{b}_i^T & 0 \end{bmatrix}, \quad \mathbf{G}_i^\mu \triangleq \begin{bmatrix} \mathbf{A}_i^\mu & \frac{1}{2}\mathbf{b}_i^\mu \\ \frac{1}{2}(\mathbf{b}_i^\mu)^T & 0 \end{bmatrix}, \ \mu \in \{u, d\},$$

$$\mathbf{G}_i^\pi \triangleq \begin{bmatrix} \mathbf{0} & \frac{1}{2}\mathbf{b}_i^\pi \\ \frac{1}{2}(\mathbf{b}_i^\pi)^T & 0 \end{bmatrix}, \ \pi \in \{l, c, U, D, S\},$$

$$\mathbf{G}_j^I \triangleq \begin{bmatrix} \text{diag}(\mathbf{e}_j) & -\frac{1}{2}\mathbf{e}_j \\ -\frac{1}{2}\mathbf{e}_j^T & 0 \end{bmatrix}.$$

As problems (11) and (27) are equivalent, all constraints have one-to-one correspondence.

The optimization problem (27) is a non-convex separable QCQP problem [25]. This problem is NP-hard. To show this, first, we note that problems (11) and (27) are equivalent. For problem (11), when we only consider the offloading decisions as variables (i.e., each user has already been assigned some fixed communication and computation resources), the problem is reduced to a linear integer programming problem. Then, if the $t_i$ values are further given, (e.g., $t_i = \sum_{j=1}^{M} T_{ij}$), problem (11) is reduced to the 0-1 knapsack problem, which is NP-hard.

To find an approximate solution, we apply the separable SDR approach [26], where we relax the problem into a separable semidefinite programming (SDP) problem. Specifically, define $\mathbf{Z}_i \triangleq \mathbf{z}_i \mathbf{z}_i^T$. The following equality holds:

$$\mathbf{z}_i^T \mathbf{G} \mathbf{z}_i = \text{Tr}(\mathbf{G}\mathbf{Z}_i), \tag{35}$$

with $\text{rank}(\mathbf{Z}_i) = 1$. By dropping the rank constraint $\text{rank}(\mathbf{Z}_i) = 1$, we have the following separable SDP problem:

$$\min_{\{\mathbf{Z}_i\}} \quad \sum_{i=1}^{N} \text{Tr}(\mathbf{G}_i \mathbf{Z}_i) \tag{36}$$

**Algorithm 1** MUMTO Algorithm

1: Obtain optimal solution $\mathbf{Z}_i^*$'s of the separable SDP problem (36).
2: Extract $\mathbf{Z}_i^*(M+6,k)$, for $k=1,...,M$, from $\mathbf{Z}_i^*$.
3: Record the values of $\mathbf{Z}_i^*(M+6,k)$, for $k=1,...,M$, as $\mathbf{p}_i = [p_{i1},\ldots,p_{iM}]^T$.
4: Set $x_{ij}^{\rm sdr} = \text{round}(p_{ij}), \forall i,j$.
5: Set $\mathbf{x}^{\rm sdr} = [(\mathbf{x}_1^{\rm sdr})^T,\ldots,(\mathbf{x}_N^{\rm sdr})^T]^T$, where $\mathbf{x}_i^{\rm sdr} = [x_{i1}^{\rm sdr},\ldots,x_{iM}^{\rm sdr}]^T$.
6: Solve the resource allocation problem (44) based on $\mathbf{x}^{\rm sdr}$;
7: Compare the minimum cost of (44) under $\mathbf{x}^{\rm sdr}$ with those under the local processing only and cloud processing only solutions. Select the one that yields the minimum system cost as $\mathbf{x}^{\rm sdr^*}$.
8: Output: the proposed offloading solution $\mathbf{x}^{\rm sdr^*}$ and the corresponding optimal resource allocation $\{\mathbf{r}_i^{\rm sdr^*}\}$.

$$\text{s.t.} \quad \text{Tr}(\mathbf{G}_i^l\mathbf{Z}_i) \le -\sum_{j=1}^{M} T_{ij}^l, \ \forall i, \tag{37}$$

$$\text{Tr}(\mathbf{G}_i^r\mathbf{Z}_i) \le 0, \ r\in\{c,u,d\}, \ \forall i, \tag{38}$$

$$\sum_{i=1}^{N}\text{Tr}(\mathbf{G}_i^U\mathbf{Z}_i) \le C_{\rm UL}, \ \sum_{i=1}^{N}\text{Tr}(\mathbf{G}_i^D\mathbf{Z}_i) \le C_{\rm DL}, \tag{39}$$

$$\sum_{i=1}^{N}\text{Tr}(\mathbf{G}_i^S\mathbf{Z}_i) \le C_{\rm Total}, \tag{40}$$

$$\text{Tr}(\mathbf{G}_j^I\mathbf{Z}_i) = 0, \ \forall i,j, \tag{41}$$

$$\mathbf{Z}_i(M+6,M+6) = 1, \ \forall i, \tag{42}$$

$$\mathbf{Z}_i \succcurlyeq 0, \ \forall i. \tag{43}$$

The optimal solution $\{\mathbf{Z}_i^*\}$ to the above separable SDP problem can be obtained efficiently in polynomial time using standard SDP software, such as SeDuMi [40]. However, since problem (36) is a relaxation of the problem (27), the optimal objective of the problem (36) is only a lower bound of the optimal solution of the problem (27) if $\{\mathbf{Z}_i^*\}$ does not have rank 1. Therefore, once $\{\mathbf{Z}_i^*\}$ is obtained, we still need to recover a rank-1 solution from $\{\mathbf{Z}_i^*\}$ for the original problem (5). In the following, we propose an algorithm to obtain the binary offloading decisions $\{x_{ij}\}$ and the corresponding optimal communication resource allocation $\{\mathbf{r}_i\}$ for problem (5).

*3) **Binary Offloading Decisions and Resource Allocation**:* Define the offloading solution vector as $\mathbf{x} \triangleq [\mathbf{x}_1^T,\ldots,\mathbf{x}_N^T]^T$, where $\mathbf{x}_i \triangleq [x_{i1},\ldots,x_{iM}]^T$, for all $i$. Since the rank-1 constraint has been removed from the relaxed problem (36), the obtained solution $\mathbf{Z}_i^*$ for problem (36) contains only real numbers. Our goal is to obtain appropriate offloading decisions from $\mathbf{Z}_i^*$ by mapping its elements to binary numbers. Note that only the first $M$ elements in $\mathbf{z}_i$ correspond to the offloading decision variables for user $i$ (see $\mathbf{w}_i$ in (17)). Also, we have $\mathbf{Z}_i = \mathbf{z}_i\mathbf{z}_i^T$ and $\mathbf{z}_i(M+6) = 1$, which means the last row of $\mathbf{Z}_i$ satisfies $\mathbf{Z}_i(M+6,k) = \mathbf{z}_i(k)$, for all $k$. Hence, we can use the values of $\mathbf{Z}_i^*(M+6,k)$ to recover the binary offloading decision $\mathbf{z}_i(k)$, for $k=1,...,M$. In addition, it can be shown that $\mathbf{Z}_i^*(M+6,k)\in[0,1]$, for $k=1,...,M$. Define $\mathbf{p}_i \triangleq [p_{i1},\ldots,p_{iM}]^T \triangleq [\mathbf{Z}_i^*(M+6,1),\cdots,\mathbf{Z}_i^*(M+6,M)]^T$. We have $p_{ij}\in[0,1]$, $\forall i,j$. We recover the feasible decisions $\mathbf{x}_i^{\rm sdr}$ using $\mathbf{p}_i$, where $x_{ij}^{\rm sdr} = \text{round}(p_{ij})$ is the rounding result, and obtain the overall offloading decision as $\mathbf{x}^{\rm sdr} = [(\mathbf{x}_1^{\rm sdr})^T,\ldots,(\mathbf{x}_N^{\rm sdr})^T]^T$.

Once the offloading decision $\mathbf{x}^{\rm sdr}$ is obtained, the optimization problem (5) reduces to the optimization of communication resource allocation $\{\mathbf{r}_i\}$, which is given by

$$\min_{\{\mathbf{r}_i\}} \quad \left(E + \sum_{i=1}^{N}\rho_i \max\{T_{L_i}, T_i^{C_{(U)}}\}\right) \tag{44}$$
$$\text{s.t.} \quad (1),(2),(3), \text{ and } (6),$$

where $E \triangleq \sum_{i=1}^{N}\sum_{j=1}^{M}(E_{ij}^l(1-x_{ij}) + E_{ij}^C x_{ij})$ is a constant value once $\{x_{ij}\}$ are given. This resource allocation problem (44) is convex, which can be solved optimally using standard convex optimization solvers. Note that to obtain the best offloading decision, in practice, we should compare $\mathbf{x}^{\rm sdr}$ with local processing only and cloud processing only decisions, and select the one resulting in the minimum total system cost objective of (44) as the final offloading decision $\mathbf{x}^{\rm sdr^*}$.

We summarize MUMTO in Algorithm 1. Notice that the SDP problem (36) can be solved within precision $\epsilon$ by the interior point method in $O(\sqrt{MN}\log(1/\epsilon))$ iterations, where the amount of work per iteration is $O(M^6N^4)$ [41], while there are $2^{MN}$ choices in exhaustive search to find the optimal offloading decision. In addition, once the offloading decision is made, we may schedule the multiple tasks to be offloaded in any arbitrary order. The resultant $T_i^C$ will be less than $T_i^{C_{(U)}}$. To measure the effectiveness of this solution, in the following, we introduce a lower bound of the optimal solution to the original problem (5).

### *D. Lower Bound on the Optimal Solution*

Previously, the cost function in our original optimization problem (5) considers the worst-case transmission-plus-processing delay (8) for all users. Once the offloading decision is made, we may schedule the multiple tasks to be offloaded in any arbitrary order. The resultant $T_i^C$ will be less than $T_i^{C_{(U)}}$. Therefore, the actual cost based on MUMTO will be lower than the worst-case cost.

However, we are still interested in the performance of MUMTO compared with an optimal solution. Therefore, we introduce a lower bound of the optimal solution to the original problem (5). We first introduce a new optimization problem, where $T_i^{C_{(L)}}$ are used instead of $T_i^C$ and the objective function is replaced by its lower bound, as follows:

$$\min_{\{x_{ij}\},\{\mathbf{r}_i\}} \quad \sum_{i=1}^{N}\Bigg[\sum_{j=1}^{M}(E_{ij}^l(1-x_{ij}) + E_{ij}^C x_{ij}) + \rho_i\max\{T_i^L, T_i^u, T_i^d, T_i^{uac}, T_i^{dac}, T_i^{c'}\}\Bigg] \tag{45}$$
$$\text{s.t.} \quad (1),(2),(3),(6), \text{ and } (7).$$

Notice that under the same offloading decisions and communication resource allocation, this new objective function will always give us a lower cost than the real cost.

Since the above optimization problem (45) is still non-convex, we formulate a separable SDR problem similar to (36), whose details are omitted due to page limitation. We

note that the optimal objective of this SDR problem is smaller than the optimal objective of (45). Hence, it can serve as a lower bound of the minimum total system cost defined by the original optimization problem (5). In Section VI-B, we show that MUMTO provides nearly optimal performance under a wide range of parameter settings.

## V. Multi-user Multi-task Offloading with CAP

When we consider the presence of a CAP, it may serve its conventional networking function and forward the task to the remote cloud server, or directly process the task by itself. Each task may be processed locally at the mobile device, at the CAP, or at the remote cloud server. An optimal offloading decision must take into consideration the computation and communication energies, computation costs, and communication and processing delays at all three locations. In this section, we further study the mobile cloud computing network with the presence of the CAP, aiming to jointly optimize the task offloading decisions and the communication and CAP processing resource allocation.

### A. Offloading Decision

Each mobile user can process its tasks locally or offload some of them. With the presence of a CAP, those offloaded tasks may be processed at the CAP or be further forwarded to the remote cloud. Instead of only using $x_{ij}$, we denote the offloading decisions for user $i$'s task $j$ by $x_{ij}^l, x_{ij}^a, x_{ij}^c \in \{0,1\}$, indicating whether user $i$'s task $j$ is processed locally, at the CAP, or at the cloud, respectively. The offloading decisions are constrained by

$$x_{ij}^l + x_{ij}^a + x_{ij}^c = 1. \tag{46}$$

Notice that only one of $x_{ij}^l, x_{ij}^a$, and $x_{ij}^c$ for user $i$'s task $j$ could be 1.

### B. Problem Formulation

The new total system cost is defined as the weighted sum of total energy consumption, the costs to offload and process all tasks, and the transmission and processing delays for all users. Define offloading decision vector $\mathbf{x}_{ij} \triangleq [x_{ij}^l, x_{ij}^a, x_{ij}^c]^T$. With a CAP, both communication and CAP processing resources needs to be considered, defined by $\mathbf{r}_i \triangleq [c_i^u, c_i^d, f_i^a]^T$. Similar to Section IV-B, our objective is to minimize the total system cost by jointly optimizing the task offloading decisions $\{\mathbf{x}_{ij}\}$ and the communication and CAP processing resource allocation $\{\mathbf{r}_i\}$. This optimization problem is formulated as follows:

$$\min_{\{\mathbf{x}_{ij}\},\{\mathbf{r}_i\}} \quad \sum_{i=1}^{N}\Bigg[\sum_{j=1}^{M}(E_{ij}^l x_{ij}^l + E_{ij}^A x_{ij}^a + E_{ij}^C x_{ij}^c) + \rho_i \max\{T_i^L, T_i^A, T_i^C\}\Bigg] \tag{47}$$

$$\text{s.t.} \quad (1),(2),(3),(4),(46),$$

$$c_i^u, c_i^d, f_i^a, \geq 0, \forall i, \tag{48}$$

$$x_{ij}^l, x_{ij}^a, x_{ij}^c \in \{0,1\}, \forall i, j, \tag{49}$$

where $E_{ij}^A \triangleq (E_{ij}^t + E_{ij}^r + \alpha C_{ij}^a)$ and $E_{ij}^C \triangleq (E_{ij}^t + E_{ij}^r + \beta C_{ij}^c)$ are the weighted transmission energy and processing costs of offloading and processing task $j$ of user $i$ to the CAP and cloud, with $\alpha$ and $\beta$ being their relative weights, respectively; also, $T_i^L$ is the processing delay of tasks processed by the mobile user $i$ itself, $T_i^A$ and $T_i^C$ are the overall transmission and remote-processing delays for the tasks of mobile user $i$ processed at the CAP and cloud, respectively, and $\rho_i$ is the weight on the task processing delay relative to energy consumption for user $i$. Comparing with the optimization problem (5) in the no-CAP case, the above mixed-integer programming problem (47) is even more complicated due to the additional CAP processing cost, $E_{ij}^A$, CAP processing delay, $T_i^A$, and the placement constraint (46).

For optimization problem (47), we have the overall local processing delay for each user as $T_i^L = \sum_{j=1}^{M} T_{ij}^l x_{ij}^l$, for all $i$. However, as similarly discussed in problem (5), the overall delay for CAP processing, $T_i^A$, and for cloud processing, $T_i^C$, are not precisely tractable due to multiple offloaded tasks may have overlapping transmission or processing time. Therefore, we use both upper and lower bounds of $T_i^A$ and $T_i^C$ in our proposed solution and performance benchmarking. We will show later that, with the proposed MUMTO-C algorithm, the upper and lower bounds give estimates to the total system cost that are close to each other.

### C. Multi-user Multi-task Offloading with CAP (MUMTO-C) Algorithm

To find an efficient solution to the mixed-integer non-convex programming problem (47), in the following, we first propose upper-bound and lower-bound formulations of both $T_i^A$ and $T_i^C$, then transform the optimization problem (47) into a separable QCQP and the corresponding SDR problem. Finally, we will propose a three-step MUMTO-C algorithm to obtain the binary offloading decisions $\{\mathbf{x}_{ij}\}$ and the communication and processing resource allocation $\{\mathbf{r}_i\}$.

*1) **Bounds of CAP-Processing and Cloud-Processing Delays**:* Similar to Section IV-C1, we have the following upper bounds, i.e., the *worst-case delays*:

$$T_i^{A_{(U)}} = \sum_{j=1}^{M}((T_{ij}^t + T_{ij}^r)(x_{ij}^a + x_{ij}^c) + T_{ij}^a x_{ij}^a), \tag{50}$$

$$T_i^{C_{(U)}} = \sum_{j=1}^{M}((T_{ij}^t + T_{ij}^r)(x_{ij}^a + x_{ij}^c) + (T_{ij}^{ac} + T_{ij}^c)x_{ij}^c). \tag{51}$$

In the above expressions, $T_i^{A_{(U)}}$ and $T_i^{C_{(U)}}$ represent the direct summing of the transmission delays and processing delays without any overlap. They are always greater than the actual delay given the same offloading decision and resource allocation.

For performance benchmarking, we will also need the best-case delays. By separating the offloading delays of all mobile users into several components and only considering the largest one among them, the lower bounds of $T_i^A$ and $T_i^C$ are

$$T_i^{A_{(L)}} = \max\{T_i^{u'}, T_i^{d'}, T_i^{a'}\}, \tag{52}$$

$$T_i^{C_{(L)}} = \max\{T_i^u, T_i^d, T_i^{uac}, T_i^{dac}, T_i^{c'}\}, \tag{53}$$

where $T_i^{u'} = \sum_{j=1}^{M} T_{ij}^t x_{ij}^a$ and $T_i^{d'} = \sum_{j=1}^{M} T_{ij}^r x_{ij}^a$ are the total uplink and downlink transmission times between the user and the CAP for user $i$'s tasks processed at the CAP, respectively, $T_i^u = \sum_{j=1}^{M} T_{ij}^t x_{ij}^c$ and $T_i^d = \sum_{j=1}^{M} T_{ij}^r x_{ij}^c$ are the total uplink and downlink transmission times between the user and the CAP for user $i$'s tasks processed at the cloud, respectively, $T_i^{uac} = \sum_{j=1}^{M} D_{\text{in}}(ij) x_{ij}^c / r^{ac}$ and $T_i^{dac} = \sum_{j=1}^{M} D_{\text{out}}(ij) x_{ij}^c / r^{ac}$ are the total uplink and downlink transmission times between the CAP and the cloud for user $i$, respectively, and $T_i^{a'} = \sum_{j=1}^{M} T_{ij}^a x_{ij}^a$ and $T_i^{c'} = \sum_{j=1}^{M} T_{ij}^c x_{ij}^c$ are the total CAP and cloud processing times for user $i$, respectively.

In the following subsections, we describe the details of the proposed three-step MUMTO-C algorithm, using the worst-case delays $T_i^{A_{(U)}}$ and $T_i^{C_{(U)}}$ in optimization problem (47) to obtain an approximate solution, which gives an upper bound to the actual total system cost. Furthermore, we show the local optimum property of the obtained binary offloading decisions $\{\mathbf{x}_{ij}\}$ and communication and processing resource allocation $\{\mathbf{r}_i\}$. Similarly, $T_i^{A_{(L)}}$ and $T_i^{C_{(L)}}$ are used to obtain a lower bound of the total system cost for performance benchmarking. Finally, we show in Section VI-C that MUMTO-C achieve actual system cost that is close to the lower bound of the system cost, and hence is also close to the optimal system cost.

*2) **Step 1: QCQP Transformation and Semidefinite Relaxation**:* As mentioned before, optimization problem (47) is more complicated than problem (5) due to the availability of the CAP. In order to obtain the eventual SDR formulation, we first rewrite the integer constraint (49) as

$$x_{ij}^s(x_{ij}^s - 1) = 0, \quad \forall i, j, \tag{54}$$

for $s \in \{l, a, c\}$, and replace $T_i^A$ and $T_i^C$ in (47) with $T_i^{A_{(U)}}$ and $T_i^{C_{(U)}}$, respectively. Following the similar procedure in Section IV-C2, we move the delay term from the objective to the constraints by using additional auxiliary variables $t_i$, and rewrite (47) as

$$\begin{aligned}
\min_{\{\mathbf{x}_{ij}\},\{\mathbf{r}_i,t_i\}} \quad & \sum_{i=1}^{N}\left[\sum_{j=1}^{M}(E_{ij}^l x_{ij}^l + E_{ij}^A x_{ij}^a + E_{ij}^C x_{ij}^c) + \rho_i t_i\right] \quad (55)\\
\text{s.t.} \quad & \sum_{j=1}^{M} T_{ij}^l x_{ij}^l \le t_i, \ \forall i,\\
& \sum_{j=1}^{M}\left(\frac{D_{\text{in}}(ij)}{\eta_i^u c_i^u} + \frac{D_{\text{out}}(ij)}{\eta_i^d c_i^d}\right)(x_{ij}^a + x_{ij}^c)\\
& + \sum_{j=1}^{M}\frac{Y(ij)}{f_i^a}x_{ij}^a \le t_i, \ \forall i,\\
& \sum_{j=1}^{M}\left(\frac{D_{\text{in}}(ij)}{\eta_i^u c_i^u} + \frac{D_{\text{out}}(ij)}{\eta_i^d c_i^d}\right)(x_{ij}^a + x_{ij}^c)\\
& + \sum_{j=1}^{M}(T_{ij}^{ac} + T_{ij}^c)x_{ij}^c \le t_i, \ \forall i,\\
& (1), (2), (3), (4), (46), (48), \text{ and } (54).
\end{aligned}$$

Comparing problem (55) with problem (11), we observe that they share a similar structure. Therefore, we can apply a similar procedure to transform problem (55) into a non-convex separable QCQP problem that is similar to problem (27), with the optimization vector now defined by $\mathbf{v}_i \triangleq [\tilde{\mathbf{w}}_i^T, 1]^T$, where $\tilde{\mathbf{w}}_i \triangleq [\mathbf{x}_{i1}^T, \cdots, \mathbf{x}_{iM}^T, c_i^u, D_i^u, c_i^d, D_i^d, f_i^a, D_i^a, t_i]^T$, with $D_i^u$, $D_i^d$ and $D_i^a$ being the auxiliary variables introduced corresponding to the uplink transmission time, downlink transmission time, and the CAP processing time, respectively. Auxiliary variables $D_i^u$ and $D_i^d$ are similarly defined as in (15) and (16), except that $x_{ij}$ in (15) and (16) is now replaced by $(x_{ij}^a + x_{ij}^c)$. Similar to these two constraints, the new auxiliary variable $D_i^a$ for the CAP processing time also introduces a new constraint $\sum_{j=1}^{M} Y(ij)x_{ij}^a / f_i^a \le D_i^a$. Using the separable SDR approach, we solve the relaxed separable SDP problem that is similar to problem (36), with optimization matrix defined by $\mathbf{V}_i = \mathbf{v}_i\mathbf{v}_i^T$ with size $(3M+8)\times(3M+8)$. The details are omitted to avoid redundancy.

Denote $\{\mathbf{V}_i^*\}$ as the optimal solution of the corresponding separable SDR problem for the optimization problem (55). We need to recover a rank-one solution from $\{\mathbf{V}_i^*\}$ for problem (55). However, the reconstruction of binary offloading decision $\{\mathbf{x}_{ij}\}$ in Section IV-C3, as part of the MUMTO algorithm, cannot be directly applied to find a feasible solution for problem (55) due to the additional placement constraint (46) for each user's tasks. To deal with this challenge, in the following, we propose an modified method, termed *MUMTO SDR with CAP (SDR-C)*, to obtain the binary offloading decisions $\{\mathbf{x}_{ij}\}$ and the corresponding optimal communication resource allocation $\{\mathbf{r}_i\}$ from $\{\mathbf{V}_i^*\}$.

Define $\mathbf{x} \triangleq [\mathbf{x}_1^T, \ldots, \mathbf{x}_N^T]^T$, where $\mathbf{x}_i \triangleq [\mathbf{x}_{i1}, \ldots, \mathbf{x}_{iM}]^T$, for all $i$. As similarly discussed in Section IV-C3, $\mathbf{V}_i(3M+8, k) = \mathbf{v}_i(k)$, for $k = 1, \cdots, 3M$, which correspond to offloading decision $\mathbf{x}_i$ for user $i$. It can be proven that optimal solution $\mathbf{V}_i^*(3M+8, k) \in [0, 1]$, for $k = 1, ..., 3M$. Denote $\mathbf{p}_{ij} \triangleq [p_{ij}^l, p_{ij}^a, p_{ij}^c]^T$ and $\mathbf{p}_i \triangleq [\mathbf{p}_{i1}^T, \ldots, \mathbf{p}_{iM}^T]^T \triangleq [\mathbf{V}_i^*(3M+8, 1), \cdots, \mathbf{V}_i^*(3M+8, 3M)]^T$. Then, we have each element in $\mathbf{p}_i$ having its value within $[0, 1]$, $\forall i$. We recover the feasible decisions $\mathbf{x}_i^{\text{sdr}}$ using $\mathbf{p}_i$ as follows: for $j = 1, \cdots, M$, set

$$\mathbf{x}_{ij}^{\text{sdr}} = \begin{cases} [1,0,0]^T, \text{ if } \max\limits_{s\in\{l,a,c\}} p_{ij}^s = p_{ij}^l \text{ (local processing)} \\ [0,1,0]^T, \text{ if } \max\limits_{s\in\{l,a,c\}} p_{ij}^s = p_{ij}^a \text{ (CAP processing)} \\ [0,0,1]^T, \text{ if } \max\limits_{s\in\{l,a,c\}} p_{ij}^s = p_{ij}^c \text{ (cloud processing)}, \end{cases} \tag{56}$$

The overall offloading decision is obtained as $\mathbf{x}^{\text{sdr}} = [(\mathbf{x}_1^{\text{sdr}})^T, \ldots, (\mathbf{x}_N^{\text{sdr}})^T]^T$.

After obtaining the offloading decision $\mathbf{x}^{\text{sdr}}$, optimization problem (47) is reduced to the optimization of computation and communication resource allocation $\{\mathbf{r}_i\}$, which is given by

$$\begin{aligned}
\min_{\{\mathbf{r}_i\}} \quad & \left(E + \sum_{i=1}^{N}\rho_i \max\{T_i^L, T_i^{A_{(U)}}, T_i^{C_{(U)}}\}\right) \quad (57)\\
\text{s.t.} \quad & (1), (2), (3), (4), \text{ and } (48),
\end{aligned}$$

where $E \triangleq \sum_{i=1}^{N}\sum_{j=1}^{M}(E_{ij}^{l}x_{ij}^{l} + E_{ij}^{A}x_{ij}^{a} + E_{ij}^{C}x_{ij}^{c})$ is a constant value once $\{\mathbf{x}_{ij}\}$ are given. Similar to problem (44), problem (57) is convex, so it can be solved optimally.

*3) **Step 2: Improvement to SDR-C by Alternating Optimization (AO)**:* After obtaining a feasible solution $\{\mathbf{x}^{\text{sdr}}, \{\mathbf{r}_i^{\text{sdr}^*}\}\}$ from the SDR-C step above, to further reduce the overall system cost, in the following we introduce an iterative alternating optimization method to further improve the offloading decision, by using $\{\mathbf{x}^{\text{sdr}}, \{\mathbf{r}_i^{\text{sdr}^*}\}\}$ as the starting point of iteration.

As mentioned above, given any offloading decision, the optimization problem (47) is reduced to the resource allocation problem (57), which is convex and the optimal resource allocation can be obtained. On the other hand, once the resource allocation $\{\mathbf{r}_i\}$ is given, the optimization problem (47) is reduced to the optimization of offloading decisions $\{\mathbf{x}_{ij}\}$ as follows:

$$\min_{\{\mathbf{x}_{ij}\}} \quad \sum_{i=1}^{N}\Bigg[\sum_{j=1}^{M}(E_{ij}^{l}x_{ij}^{l} + E_{ij}^{A}x_{ij}^{a} + E_{ij}^{C}x_{ij}^{c}) + \rho_i \max\{T_i^{L}, T_i^{A_{(U)}}, T_i^{C_{(U)}}\}\Bigg] \tag{58}$$

$$\text{s.t.} \quad (46) \text{ and } (54).$$

The offloading decision problem (58) is an integer programming problem. However, it can be separated into $N$ independent sub-problems, where each sub-problem only considers the offloading decision of one user. As shown in [36], this can be solved near-optimally by either using an SDR approach or relaxing the integer constraints to interval constraints. Since the optimization problem (47) can be separated into two sub-problems (57) and (58). We propose the following alternating optimization procedure to further reduce the total system cost.

Set $(\mathbf{x}^{\text{ao}^*}, \{\mathbf{r}_i^{\text{ao}^*}\}) = (\mathbf{x}^{\text{sdr}}, \{\mathbf{r}_i^{\text{sdr}^*}\})$ as the initial point. At each iteration:

i) Solve problem (58) based on $\{\mathbf{r}_i^{\text{ao}^*}\}$ to find the corresponding offloading decision $\mathbf{x}^{\text{ao}'}$.

ii) Solve problem (57) based on $\mathbf{x}^{\text{ao}'}$ to find the minimum system cost and the corresponding resource allocation $\{\mathbf{r}_i^{\text{ao}'}\}$. If this provides a lower total system cost, update $(\mathbf{x}^{\text{ao}^*}, \{\mathbf{r}_i^{\text{ao}^*}\}) = (\mathbf{x}^{\text{ao}'}, \{\mathbf{r}_i^{\text{ao}'}\})$.

Repeat steps i and ii until the the total system cost cannot be further decreased. Then output the solution of the alternating optimization procedure as $(\mathbf{x}^{\text{ao}^*}, \{\mathbf{r}_i^{\text{ao}^*}\})$.

Note that, despite the approximation in solving (58), since we only accept a better solution in each iteration, and the system cost is lower bounded, AO always converges. Furthermore, by design, the solution $(\mathbf{x}^{\text{ao}^*}, \{\mathbf{r}_i^{\text{ao}^*}\})$ is better than or at least as good as $(\mathbf{x}^{\text{sdr}}, \{\mathbf{r}_i^{\text{sdr}^*}\})$.

*4) **Step 3: Sequential Tuning (ST) to Reach Local Optimum**:* In this step, we propose an iterative procedure starting from $\{\mathbf{x}^{\text{ao}^*}, \{\mathbf{r}_i^{\text{ao}^*}\}\}$, termed sequential tuning, to further reduce the system cost and eventually achieve a local optimum for (47).

Set $(\mathbf{x}^{\text{st}^*}, \{\mathbf{r}_i^{\text{st}^*}\}) = (\mathbf{x}^{\text{ao}^*}, \{\mathbf{r}_i^{\text{ao}^*}\})$ as the initial point. At each iteration:

i) Randomly order the lists of all users and their tasks.

ii) Go through the user list one by one. For each examined user, sequentially check each of its tasks for the three possible offloading decisions, while the offloading decisions of all other tasks of all users remain unchanged. For each offloading decision, find the total system cost by solving problem (57). As soon as some user $i$ is found to admit a lower total system cost by changing the offloading decision of one of its tasks, update $(\mathbf{x}^{\text{st}^*}, \{\mathbf{r}_i^{\text{st}^*}\})$ to the new offloading decision and resource allocation that give the lower cost, and exit the iteration.

Repeat steps i and ii until $\mathbf{x}^{\text{st}^*}$ converges, i.e., no change for $\mathbf{x}^{\text{st}^*}$ can be made. Then output the solution of the sequential turning procedure as $(\mathbf{x}^{\text{st}^*}, \{\mathbf{r}_i^{\text{st}^*}\})$.

The above procedure is guaranteed to converge. This is because there is a finite number of possible values for $\mathbf{x}_i^{\text{st}}$. The iteration eventually will reach some $(\mathbf{x}^{\text{st}^*}, \{\mathbf{r}_i^{\text{st}^*}\})$, where the total system cost cannot be further reduced by modifying any user's offloading decision (and corresponding resource allocation). It is straightforward to show that $(\mathbf{x}^{\text{st}^*}, \{\mathbf{r}_i^{\text{st}^*}\})$ is a local optimum of problem (47), since it gives the lowest system cost in the joint binary-valued neighborhood of $\mathbf{x}$ and neighborhood of $\{\mathbf{r}_i\}$. This result is stated in the following proposition.

*Proposition 1:* The solution $(\mathbf{x}^{\text{st}^*}, \{\mathbf{r}_i^{\text{st}^*}\})$ obtained from the sequential tuning procedure is a locally optimal solution to the original non-convex optimization problem (47).

*5) **Overall MUMTO-C Algorithm**:* We summarize the above three-step MUMTO-C algorithm in Algorithm 2.

Even though each of the SDR-C, AO, and ST steps above can be used separately to provide a feasible solution to the original optimization problem (47), when combined together in the proposed manner, they each serves an important role in the overall algorithm design. First, SDR-C provides a suitable starting point for AO. Without it, i.e., if we start the AO iteration from some randomly chosen point in the solution space, as shown in Section VI-C, AO can converge to some highly sub-optimal solution. Second, with an appropriate starting point, AO pushes the solution to one that is closer to the optimum. This provide a suitable starting point for ST, which helps reduce the number of iterations in ST. This is an important step, since each of the ST iterations can be computationally expensive, as it potentially may require searching over a large number of tasks. Finally, ST further improves the solution, and more importantly, it guarantees that the final solution is a local optimum. Further numerical evaluation of the roles and contributions of each of these steps is given in Section VI-C.

### *D. Lower Bound on the Optimal Solution*

Similar to the case of MUMTO in Section IV-D, to study the performance of MUMTO-C compared with an optimal solution, we find a lower bound of the optimal solution by introducing a new optimization problem in which $T_i^{A_{(L)}}$ and $T_i^{C_{(L)}}$ are used instead as

$$\min_{\{\mathbf{x}_{ij}\},\{\mathbf{r}_i\}} \quad \sum_{i=1}^{N}\Bigg[\sum_{j=1}^{M}(E_{ij}^{l}x_{ij}^{l} + E_{ij}^{A}x_{ij}^{a} + E_{ij}^{C}x_{ij}^{c})$$

**Algorithm 2** MUMTO-C Algorithm

**Step 1: Initial offloading solution via SDR-C**

Transform the original problem (47) into the SDR problem and obtain the optimal solution $\{\mathbf{V}_i^*\}$.
Extract $\mathbf{V}_i^*(3M+8,k)$, for $k=1,...,3M$, from $\mathbf{V}_i^*$.
Record the values of $\mathbf{V}_i^*(3M+8,k)$, for $k=1,...,3M$, by $\mathbf{p}_i=[\mathbf{p}_{i1}^T,\ldots,\mathbf{p}_{iM}^T]^T$, where $\mathbf{p}_{ij}=[p_{ij}^l,p_{ij}^a,p_{ij}^c]^T$.
Set $\mathbf{x}^{\text{sdr}}=[(\mathbf{x}_1^{\text{sdr}})^T,\ldots,(\mathbf{x}_N^{\text{sdr}})^T]^T$, where $\mathbf{x}_i^{\text{sdr}}$ is given by (56), and solve problem (57) based on $\mathbf{x}^{\text{sdr}}$.

**Step 2: Alternating optimization (AO)**

Set $(\mathbf{x}^{\text{ao}^*},\{\mathbf{r}_i^{\text{ao}^*}\})=(\mathbf{x}^{\text{sdr}},\{\mathbf{r}_i^{\text{sdr}^*}\})$, and record the corresponding total system cost as $J^{\text{ao}^*}$; set AO = False.
**while** AO == False **do**
  Solve problem (58) based on $\{\mathbf{r}_i^{\text{ao}^*}\}$ to find the corresponding offloading decision $\mathbf{x}^{\text{ao}'}$;
  Solve problem (57) based on $\mathbf{x}^{\text{ao}'}$ to find the minimum system cost $J^{\text{ao}'}$ and $\{\mathbf{r}_i^{\text{ao}'}\}$;
  **if** $J^{\text{ao}'}<J^{\text{ao}^*}$ **then**
    Set $(\mathbf{x}^{\text{ao}^*},\{\mathbf{r}_i^{\text{ao}^*}\})=(\mathbf{x}^{\text{ao}'},\{\mathbf{r}_i^{\text{ao}'}\})$, $J^{\text{ao}^*}=J^{\text{ao}'}$;
  **else**
    Set AO = True; ▷ Exit while loop
  **end if**
**end while**

**Step 3: Sequential tuning (ST)**

Set $(\mathbf{x}^{\text{st}^*},\{\mathbf{r}_i^{\text{st}^*}\})=(\mathbf{x}^{\text{ao}^*},\{\mathbf{r}_i^{\text{ao}^*}\})$, and record the corresponding total system cost as $J^{\text{st}^*}$; set ST = False.
**while** ST == False **do**
  Randomly order the lists of all users and their tasks; set user index $n=1$; set task index $m=1$;
  **while** $n\le N$ and $m\le M$ **do**
    While keeping $\mathbf{x}_{n'm'}^{\text{st}^*}$ unchanged for all $(n',m')$ except $(n',m')=(n,m)$, inspect the three possible offloading choices of $\mathbf{x}_{nm}^{\text{st}^*}$; find their respective total system costs by solving problem (57); set $J^{\text{st}'}$ as the minimum cost among these three choices, and record the corresponding solution as $(\mathbf{x}^{\text{st}'},\{\mathbf{r}_i^{\text{st}'}\})$;
    **if** $J^{\text{st}'}<J^{\text{st}^*}$ **then**
      Set $(\mathbf{x}^{\text{st}^*},\{\mathbf{r}_i^{\text{st}^*}\})=(\mathbf{x}^{\text{st}'},\{\mathbf{r}_i^{\text{st}'}\})$, $J^{\text{st}^*}=J^{\text{st}'}$;
      $n\leftarrow N+1$;
    **else if** $n==N$ and $m==M$ **then**
      $n\leftarrow N+1$; ST = True; ▷ No change of $\mathbf{x}^{\text{st}^*}$ can be found; exit
    **else if** $n<N$ and $m==M$ **then**
      $n\leftarrow n+1$; $m\leftarrow 1$;
    **else**
      $m\leftarrow m+1$;
    **end if**
  **end while**
**end while**
Output: The offloading decision $\mathbf{x}^{\text{st}^*}$ and the corresponding resource allocation $\{\mathbf{r}_i^{\text{st}^*}\}$.

$$+\rho_i\max\{T_i^L,T_i^{A_{(L)}},T_i^{C_{(L)}}\}\Big] \tag{59}$$

$$\text{s.t.}\quad (1),(2),(3),(4),(46),(48),\text{ and }(54).$$

Under the same offloading decisions and resource allocation, the objective function in (59) is always lower than the actual cost. We apply the same approach to solve the corresponding separable SDR problem of the above non-convex problem (59). Since the optimal objective of this SDR problem is smaller than the optimal objective of (59), it can serve as a lower bound to the minimum total system cost defined by the original optimization problem (47).

TABLE II
DEFAULT PARAMETER SETTINGS.

| Description | Default Value |
|---|---|
| number of users $N$ | 5 |
| number of tasks per user $M$ | 4 |
| total CAP processing rate $f_A$ | $10\times10^9$ cycle/s |
| cloud processing rate $f^c$ | $10\times10^9$ cycle/s |
| weight on CAP usage cost $\alpha$ | $1.5\times10^{-7}$ J/bit |
| weight on cloud usage cost $\beta$ | $2.5\times10^{-7}$ J/bit |
| weight on delays $\rho_i$ | 1 J/s |

In Section VI, we show that the proposed MUMTO-C method provides not only a local optimum solution but also nearly optimal performance compared with the lower bound.

## VI. Performance Evaluation

In this section, we provide computer simulation to study the performance of both proposed MUMTO and MUMTO-C offloading solutions, respectively, under different parameter settings.

### A. Simulation Setup

The following default parameter values are used unless specified otherwise later. We adopt the mobile device characteristics from [42], which is based on a Nokia smart device. According to Tables 1 and 3 in [42], the mobile device has CPU rate $500\times10^6$ cycles/s and unit processing energy consumption $\frac{1}{730\times10^6}$ J/cycle. The local computation time per bit is $4.75\times10^{-7}$ s and local processing energy consumption per bit is $3.25\times10^{-7}$ J. We consider the x264 CBR encode application, which requires 1900 cycles/byte [42], i.e., $Y(ij)=1900D_{\text{in}}(ij)$. The input and output data sizes of each task are assumed to be uniformly distributed from 10 to 30MB and from 1 to 3MB, respectively.

The total transmission bandwidth between the mobile users and the CAP is set to 40 MHz, with no additional limit on the uplink or downlink, and the transmission and receiving energy consumptions of the mobile user are both $1.42\times10^{-7}$ J/bit as indicated in Table 2 in [42]. For simplicity, we set $\eta_i^u=\eta_i^d=3.5$ b/s/Hz for all $i$. When tasks are sent from the CAP to the cloud, the transmission rate $r^{ac}$ is 15 Mpbs. The cloud and CAP usage costs are assumed to be $C_{ij}^c=D_{\text{in}}(ij)+\lambda_1/f^c+\lambda_2/C_{\text{UL}}+\lambda_3/C_{\text{DL}}$ and $C_{ij}^a=D_{\text{in}}(ij)+\lambda_1/f_A+\lambda_2/C_{\text{UL}}+\lambda_3/C_{\text{DL}}$, respectively, where $\lambda_1=10^{18}$ bit$\times$cycle/s and $\lambda_2=\lambda_3=10^{16}$ bit $\times$ MHz, which accounts for the input data size, processing rate, and uplink and downlink capacities.

The default values for other parameters are summarized in Table II. Unless specified otherwise, these default values are used in the figures below. All simulation results are obtained by averaging over 100 realizations of the input and output data sizes of each task.

### B. Performance Evaluation for MUMTO without CAP

For comparison, we also consider the following methods: 1) the *local processing only* scheme where all tasks are processed by mobile users, 2) the *cloud processing only* scheme where all tasks are offloaded to the cloud and the cost is obtained

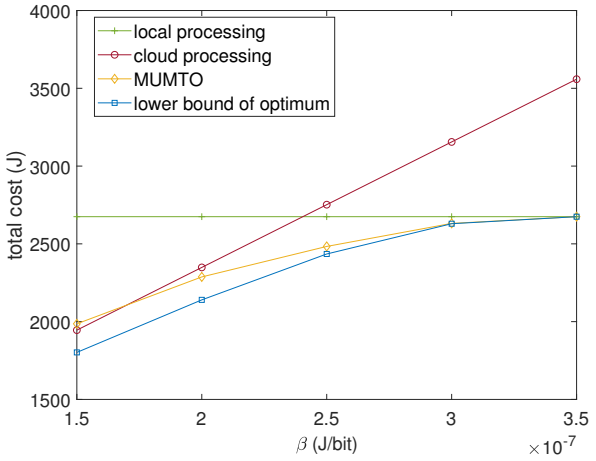



Fig. 2. Total system cost versus $\beta$ without CAP.

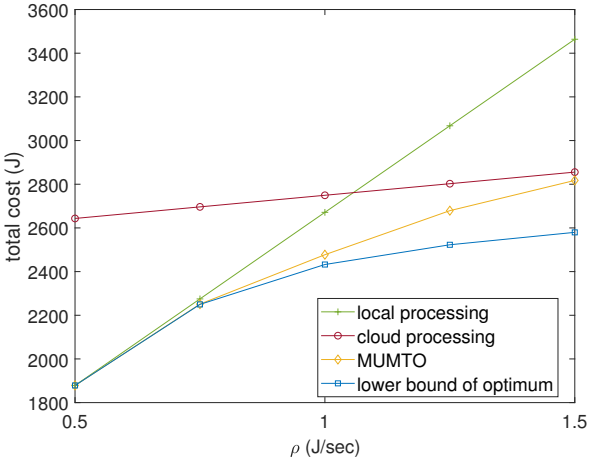



Fig. 4. Total system cost versus $\rho$ ($\rho_i$) without CAP.

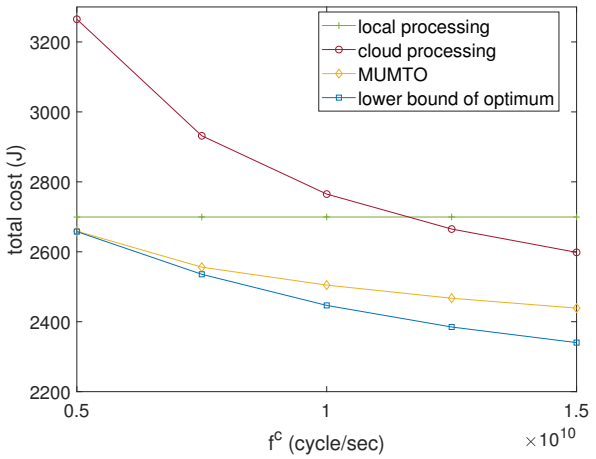



Fig. 3. Total system cost versus cloud CPU rate $f^c$ without CAP.

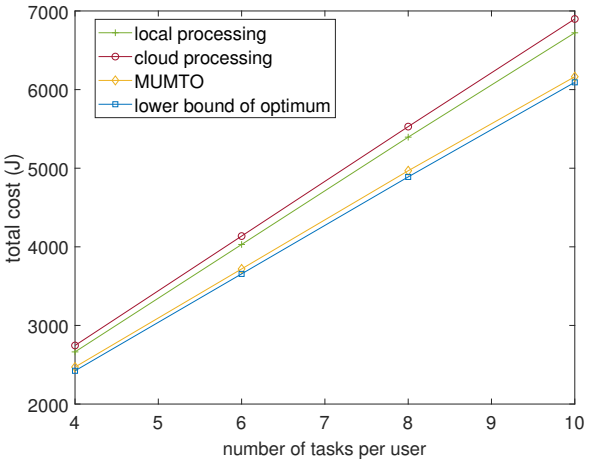



Fig. 5. Total system cost versus the number of tasks $M$ per user without CAP.

based on $T_i^{C_{(L)}}$, 3) the *lower bound of optimum*, which is obtained from the optimal objective value of the SDR of problem (45). Notice that in all figures the real cost under the same offloading decision and resource allocation will always fall between the costs of the proposed MUMTO and the lower bound of optimum.

In Fig. 2, we show the system cost vs. the weight $\beta$ on the system utility cost. When $\beta$ becomes large, all tasks are more likely to be processed by mobile users themselves. Both MUMTO and the lower bound of optimum in this case converge to the local processing only method. Though the existence of the cloud can provide additional computation capacity, the processing time at the cloud depends on the cloud CPU rate $f^c$ assigned to each user. In Fig. 3, we plot the total system cost vs. $f^c$. As expected, a more powerful per-user cloud CPU can dramatically increase system performance, and MUMTO converges to the local processing only method when the per-user cloud CPU rate is too slow to help.

In Fig. 4, we study the system cost under various values of weight $\rho_i = \rho$ on the delays. We observe that MUMTO substantially outperforms all other methods. Finally, we examine the scalability of MUMTO. Fig. 5 plot the total system cost vs. the number of tasks $M$ per user. We see that MUMTO is close to the lower bound of optimum, indicating that it is nearly optimal for all $M$ values.

### C. Performance Evaluation for MUMTO-C with CAP

*1) **Contribution of the Algorithm Components**:* To demonstrate the role and contribution of each step in the MUMTO-C algorithm, we first compare it with the following methods: 1) the *SDR-C* method where only the first step of MUMTO-C is applied, 2) the *SDR-C-ST* method where the AO step is skipped, 3) the *AO-ST* method where only the last two steps of MUMTO-C are applied by using random offloading decisions for all tasks as the starting point for the iterations of AO, 4) the *ST* method where only the last step of MUMTO-C is applied by using random offloading decisions for all tasks as the starting point for the iterations of ST, and 5) the *lower bound of optimum*, which is obtained from the optimal objective value of the SDR lower bound of problem (59).

We show the system cost and the run time ratio vs. $\alpha$ in Figs. 6 and 7, respectively. Since $\alpha$ is the weight on the CAP usage cost, more tasks compete at the CAP when $\alpha$ is smaller. We observe that MUMTO-C can reduce the system cost by up to 20% compared with purely applying SDR-C and is much

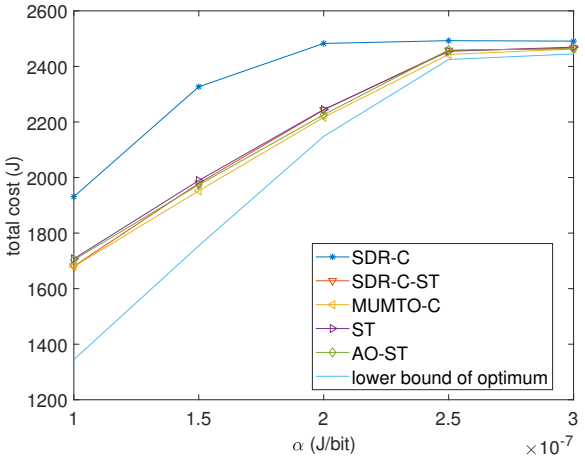



Fig. 6. Total system cost versus $\alpha$ with CAP.

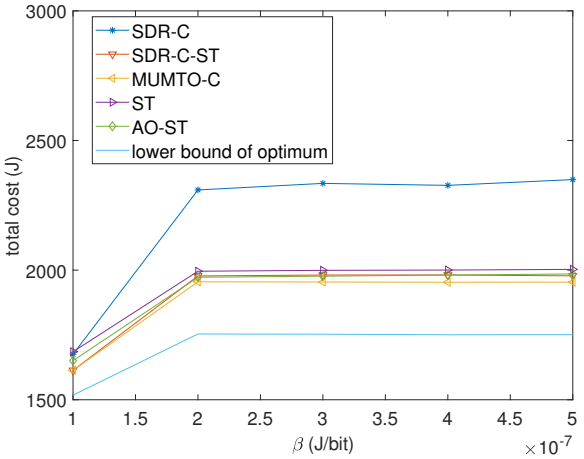



Fig. 8. Total system cost versus $\beta$ with CAP.

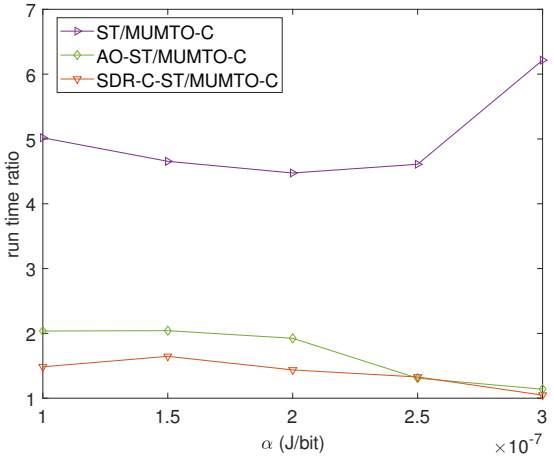



Fig. 7. Run time ratio versus $\alpha$ with CAP.

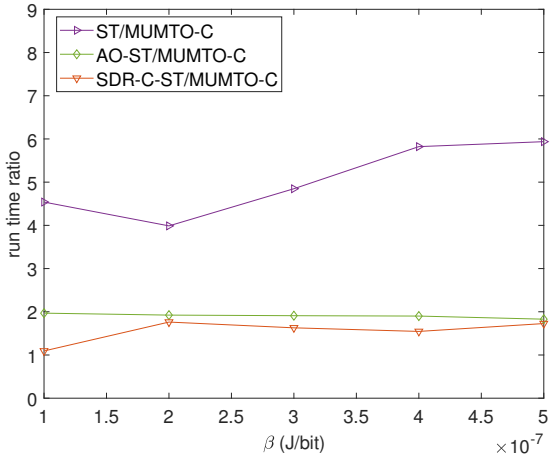



Fig. 9. Run time ratio versus $\beta$ with CAP.

closer to the lower bound of optimum with CAP. Furthermore, though SDR-C-ST, AO-ST, and ST can provide similarly low cost as MUMTO-C, which can be attributed to the sequential searching of ST, they require much longer run time to obtain their solutions. This demonstrates that we require both the SDR-C and AO steps in the proposed algorithm to provide an effective starting point for the ST step to reach a local minimum solution quickly.

Similar observations can be made in Figs. 8 and 9, where we show the system cost and the run time ratio vs. the weight $\beta$ on the cloud usage cost, and in Figs. 10 and 11, where we show the system cost and the run time ratio vs. $M$, the number of tasks per user. When $\beta$ is large, all tasks are more likely to be processed by either the mobile users or the CAP. More importantly, MUMTO-C is shown to be more scalable, since the run-time ratios are nearly linearly increasing with the number of tasks per user.

*2) **Comparison with Further Alternatives**:* For further performance evaluation, we also consider the following schemes: 1) the *local processing only* scheme, 2) the *cloud processing only* scheme, 3) the *lower bound of local-cloud*, which is the same as *lower bound of optimum* defined in Sec. VI-B, and 4) the *random mapping* scheme where each task is processed at different locations with equal probability. As shown in Figs. 12 and 13, the lower bound of optimum with CAP converges to the lower bound of local-cloud as $\alpha$ becomes large and the lower bound of local-cloud converges to the local only method as $\beta$ becomes large. In both figures, MUMTO-C is near-optimal and substantially outperforms all alternatives especially when the cost of using the CAP is small or the cost of using the cloud is large.

## VII. Conclusion

In this work, we have considered a general mobile cloud computing system consisting of multiple users and one remote cloud server, where each user has multiple independent tasks. To minimize a weighted total cost of energy, computation, and the delay of all users, we aim to find the overall optimal tasks offloading decisions and communication resource allocation. We show that the resultant optimization problem is a non-convex separable QCQP. The proposed MUMTO algorithm uses SDR and binary recovery to jointly compute the offloading decision and communication resource allocation. For the scenario with the presence of a CAP, the resultant optimization problem is even more complicated. We further propose a three-step MUMTO-C algorithm, which always compute a locally optimal solution. By comparison with a lower bound of the minimum cost for both scenarios, we show that both

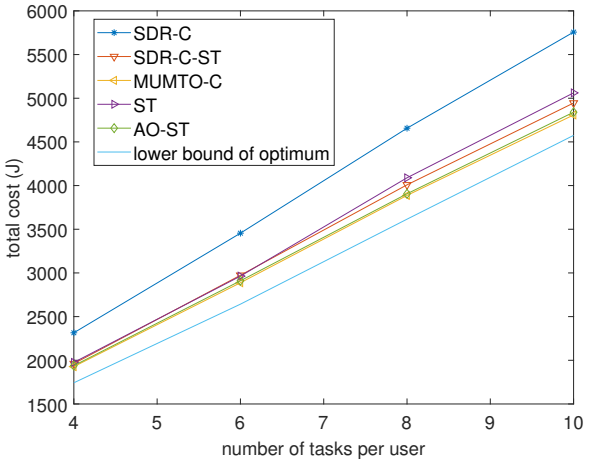



Fig. 10. Total system cost versus the number of tasks $M$ per user with CAP.

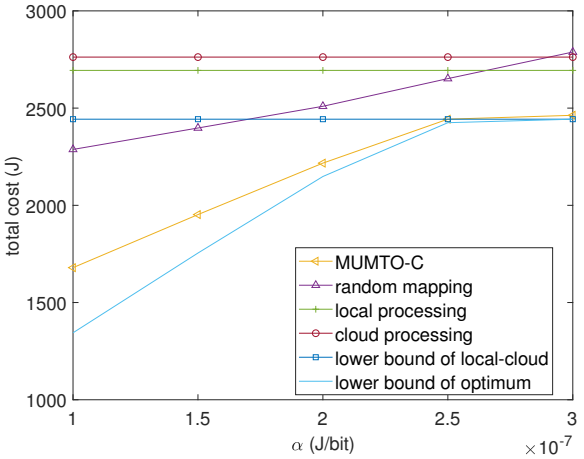



Fig. 12. Total system cost versus $\alpha$ with CAP.

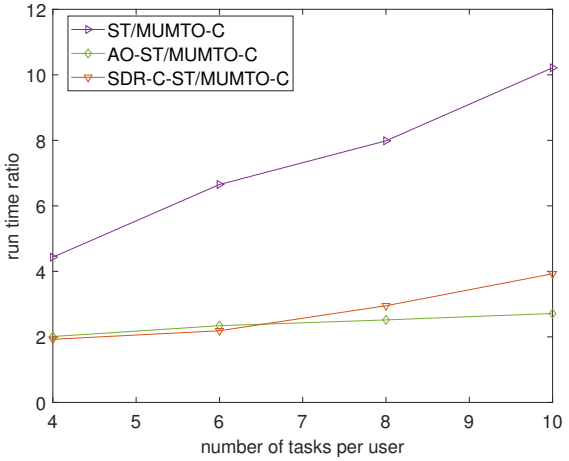



Fig. 11. Run time ratio versus the number of tasks $M$ per user with CAP.

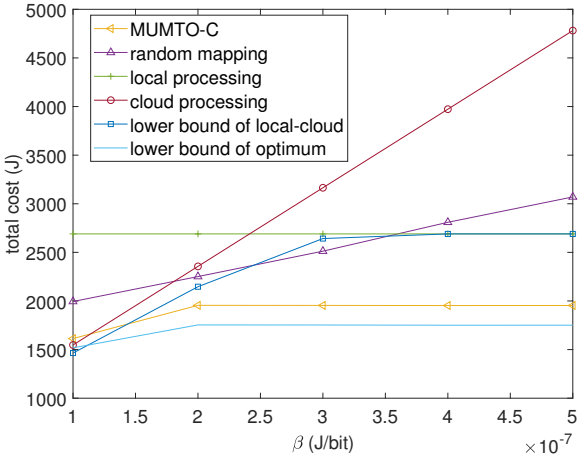



Fig. 13. Total system cost versus $\beta$ with CAP.

MUMTO and MUMTO-C give nearly optimal performance and can substantially out perform existing alternatives over a wide range of parameter settings. Finally, we remark that there are several interesting directions for future study, such as scheduling tasks with strict delay constraints, user mobility and its impact on the offloading and resource allocation, designing improved methods to better handle dynamically arriving tasks, and investigating into a more general scenario with multiple CAPs.

## REFERENCES


[1] M.-H. Chen, B. Liang, and M. Dong, "Joint offloading decision and resource allocation for multi-user multi-task mobile cloud," in *Proc. IEEE Int. Conf. Communications (ICC)*, May 2016.

[2] ——, "Joint offloading and resource allocation for computation and communication in mobile cloud with computing access point," in *Proc. IEEE Conf. on Computer Communications (INFOCOM)*, May 2017.

[3] K. Kumar, J. Liu, Y.-H. Lu, and B. Bhargava, "A survey of computation offloading for mobile systems," *Mobile Netw. Appl.*, vol. 18, no. 1, pp. 129–140, Feb. 2013.

[4] H. T. Dinh, C. Lee, D. Niyato, and P. Wang, "A survey of mobile cloud computing: architecture, applications, and approaches," *Wireless Commun. Mobile Comput.*, vol. 13, no. 18, pp. 1587–1611, 2013.

[5] K. Kumar and Y.-H. Lu, "Cloud computing for mobile users: Can offloading computation save energy?" *Computer*, vol. 43, no. 4, pp. 51–56, Apr. 2010.

[6] Y. Wen, W. Zhang, and H. Luo, "Energy-optimal mobile application execution: Taming resource-poor mobile devices with cloud clones," in *Proc. IEEE Conf. on Computer Communications (INFOCOM)*, Mar. 2012, pp. 2716–2720.

[7] X. Chen, "Decentralized computation offloading game for mobile cloud computing," *IEEE Trans. Parallel Distrib. Syst.*, vol. 26, no. 4, pp. 974–983, Apr. 2015.

[8] S. Ren and M. van der Schaar, "Efficient resource provisioning and rate selection for stream mining in a community cloud," *IEEE Trans. Multimedia*, vol. 15, no. 4, pp. 723–734, Jun. 2013.

[9] E. Meskar, T. D. Todd, D. Zhao, and G. Karakostas, "Energy aware offloading for competing users on a shared communication channel," *IEEE Trans. Mobile Comput.*, vol. 16, no. 1, pp. 87–96, Jan. 2017.

[10] E. Cuervo, A. Balasubramanian, D.-k. Cho, A. Wolman, S. Saroiu, R. Chandra, and P. Bahl, "MAUI: Making smartphones last longer with code offload," in *Proc. ACM Int. Conf. on Mobile Systems, Applications, and Services (MobiSys)*, Jan. 2010, pp. 49–62.

[11] B.-G. Chun, S. Ihm, P. Maniatis, M. Naik, and A. Patti, "Clonecloud: Elastic execution between mobile device and cloud," in *Proc. ACM Conf. on Computer Systems (EuroSys)*, Apr. 2011, pp. 301–314.

[12] S. Kosta, A. Aucinas, P. Hui, R. Mortier, and X. Zhang, "Thinkair: Dynamic resource allocation and parallel execution in the cloud for mobile code offloading," in *Proc. IEEE Conf. on Computer Communications (INFOCOM)*, Mar. 2012, pp. 945–953.

[13] W. Zhang, Y. Wen, and D. O. Wu, "Energy-efficient scheduling policy for collaborative execution in mobile cloud computing," in *Proc. IEEE Conf. on Computer Communications (INFOCOM)*, Apr. 2013, pp. 190–194.

[14] Y. H. Kao, B. Krishnamachari, M. R. Ra, and F. Bai, "Hermes: Latency optimal task assignment for resource-constrained mobile computing," in *Proc. IEEE Conf. on Computer Communications (INFOCOM)*, Apr. 2015, pp. 1894–1902.

[15] S. E. Mahmoodi, R. N. Uma, and K. P. Subbalakshmi, "Optimal joint scheduling and cloud offloading for mobile applications," *IEEE Trans. Cloud Comput.*, Apr. 2016.

[16] H. Wu, W. Knottenbelt, K. Wolter, and Y. Sun, "An optimal offloading partitioning algorithm in mobile cloud computing," in *Proc. Int. Conf. on Quantitative Evaluation of Systems*, Aug. 2016, pp. 311–328.

[17] ETSI Group Specification, "Mobile edge computing (MEC); framework and reference architecture," *ETSI GS MEC 003 V1.1.1*, 2016.

[18] B. Liang, "Mobile edge computing," in *Key Technologies for 5G Wireless Systems*, V. W. S. Wong, R. Schober, D. W. K. Ng, and L.-C. Wang, Eds., Cambridge University Press, 2017.

[19] 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, 2017.

[20] A. Greenberg, J. Hamilton, D. A. Maltz, and P. Patel, "The cost of a cloud: Research problems in data center networks," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 39, no. 1, pp. 68–73, Dec. 2008.

[21] M. Satyanarayanan, P. Bahl, R. Caceres, and N. Davies, "The case for VM-based cloudlets in mobile computing," *IEEE Pervasive Comput.*, vol. 8, no. 4, pp. 14–23, Oct. 2009.

[22] F. Bonomi, R. Milito, J. Zhu, and S. Addepalli, "Fog computing and its role in the internet of things," in *Proc. ACM SIGCOMM Workshop on Mobile Cloud Comput.*, Aug. 2012, pp. 13–16.

[23] G. Lewis and P. Lago, "Architectural tactics for cyber-foraging: Results of a systematic literature review," *J. Syst. Softw.*, vol. 107, pp. 158 – 186, 2015.

[24] M. R. Garey, D. S. Johnson, and R. Sethi, "The complexity of flowshop and jobshop scheduling," *Math. Oper. Res.*, vol. 1, no. 2, pp. 117–129, May 1976.

[25] S. Boyd and L. Vandenberghe, *Convex Optimization*. Cambridge University Press, 2004.

[26] Z.-Q. Luo, W.-K. Ma, A.-C. So, Y. Ye, and S. Zhang, "Semidefinite relaxation of quadratic optimization problems," *IEEE Signal Process. Mag.*, vol. 27, no. 3, pp. 20–34, May 2010.

[27] O. Munoz, 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.

[28] R. Kaewpuang, D. Niyato, P. Wang, and E. Hossain, "A framework for cooperative resource management in mobile cloud computing," *IEEE J. Select. Areas Commun.*, vol. 31, no. 12, pp. 2685–2700, Dec. 2013.

[29] X. Chen, L. Jiao, W. Li, and X. Fu, "Efficient multi-user computation offloading for mobile-edge cloud computing," *IEEE/ACM Trans. Netw.*, vol. 24, no. 5, pp. 2795–2808, Oct 2016.

[30] 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.

[31] X. Lyu, H. Tian, C. Sengul, and P. Zhang, "Multiuser joint task offloading and resource optimization in proximate clouds," *IEEE Trans. Veh. Technol.*, vol. 66, no. 4, pp. 3435–3447, Apr. 2017.

[32] M. R. Rahimi, N. Venkatasubramanian, S. Mehrotra, and A. V. Vasilakos, "Mapcloud: Mobile applications on an elastic and scalable 2-tier cloud architecture," in *Proc. IEEE/ACM Int. Conf. on Utility and Cloud Comput.*, Nov. 2012, pp. 83–90.

[33] M. R. Rahimi, N. Venkatasubramanian, and A. V. Vasilakos, "Music: Mobility-aware optimal service allocation in mobile cloud computing," in *Proc. IEEE Int. Conf. on Cloud Comput.*, Jun. 2013, pp. 75–82.

[34] J. Song, Y. Cui, M. Li, J. Qiu, and R. Buyya, "Energy-traffic tradeoff cooperative offloading for mobile cloud computing," in *Proc. IEEE Int. Symposium of Quality of Service (IWQoS)*, May 2014, pp. 284–289.

[35] V. Cardellini, V. De Nitto Personé, V. Di Valerio, F. Facchinei, V. Grassi, F. Lo Presti, and V. Piccialli, "A game-theoretic approach to computation offloading in mobile cloud computing," *Math. Prog.*, vol. 157, no. 2, pp. 421–449, 2016.

[36] M.-H. Chen, B. Liang, and M. Dong, "A semidefinite relaxation approach to mobile cloud offloading with computing access point," in *Proc. IEEE Int. Workshop on Signal Process. advances in Wireless Commun. (SPAWC)*, Jun. 2015, pp. 186–190.

[37] M.-H. Chen, M. Dong, and B. Liang, "Joint offloading decision and resource allocation for mobile cloud with computing access point," in *Proc. IEEE Int. Conf. Acoustics, Speech, and Signal Process. (ICASSP)*, Mar. 2016, pp. 3516–3520.

[38] ——, "Resource sharing of a computing access point for multi-user mobile cloud offloading with delay constraints," *IEEE Trans. Mobile Computing*, Mar. 2018, online early access doi: 10.1109/TMC.2018.2815533.

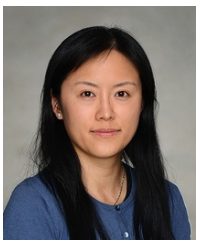
[39] D. B. Shmoys, J. Wein, and D. P. Williamson, "Scheduling parallel machines on-line," *SIAM J. Comput.*, vol. 24, no. 6, pp. 1313–1331, Dec. 1995.

[40] M. Grant, S. Boyd, and Y. Ye, "CVX: Matlab software for disciplined convex programming," 2009. [Online]. Available: http://cvxr.com/cvx/

[41] Y. Nesterov, A. Nemirovskii, and Y. Ye, *Interior-point polynomial algorithms in convex programming*. SIAM, 1994.

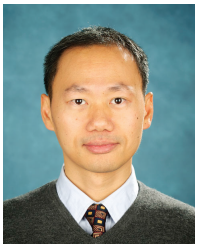
[42] A. P. Miettinen and J. K. Nurminen, "Energy efficiency of mobile clients in cloud computing," in *Proc. USENIX Conf. on Hot Topics in Cloud Comput. (HotCloud)*, Jun. 2010, pp. 4–11.


**Meng-Hsi Chen** received the B.S. degree in Electrical Engineering and M.S. degree in Communications Engineering from the National Tsing Hua University (NTHU), Hsinchu, Taiwan, in 2009 and 2011, respectively, and the Ph.D. degree in Electrical and Computer Engineering from the University of Toronto, Toronto, Canada, in 2017. His research interests are in the areas of mobile cloud systems, wireless communications, and optimization.



**Min Dong** (S'00-M'05-SM'09) received the B.Eng. degree from Tsinghua University, Beijing, China, in 1998, and the Ph.D. degree in electrical and computer engineering with minor in applied mathematics from Cornell University, Ithaca, NY, in 2004. From 2004 to 2008, she was with Corporate Research and Development, Qualcomm Inc., San Diego, CA. In 2008, she joined the Department of Electrical, Computer and Software Engineering at University of Ontario Institute of Technology, Ontario, Canada, where she is currently an Associate Professor. She also holds a status-only Associate Professor appointment with the Department of Electrical and Computer Engineering at University of Toronto. Her research interests are in the areas of statistical signal processing for communication networks, cooperative communications and networking techniques, and stochastic network optimization in dynamic networks and systems.

Dr. Dong received the Early Researcher Award from Ontario Ministry of Research and Innovation in 2012, the Best Paper Award at IEEE ICCC in 2012, and the 2004 IEEE Signal Processing Society Best Paper Award. She is a co-author of ICASSP 2016 Best Student Paper of Signal Processing for Communications and Networking at IEEE ICASSP 2016. She currently serves as an Editor for the IEEE TRANSACTIONS ON WIRELESS COMMUNICATIONS. She served as an Associate Editor for the IEEE TRANSACTIONS ON SIGNAL PROCESSING (2010-2014), and as an Associate Editor for the IEEE SIGNAL PROCESSING LETTERS (2009-2013). She was the symposium lead co-chair of the Communications and Networks to Enable the Smart Grid Symposium at the IEEE International Conference on Smart Grid Communications (SmartGridComm) in 2014. She has been an elected member of IEEE Signal Processing Society Signal Processing for Communications and Networking (SP-COM) Technical Committee since 2013.



**Ben Liang** (S'94-M'01-SM'06-F'18) received honors-simultaneous B.Sc. (valedictorian) and M.Sc. degrees in Electrical Engineering from Polytechnic University in Brooklyn, New York, in 1997 and the Ph.D. degree in Electrical Engineering with a minor in Computer Science from Cornell University in Ithaca, New York, in 2001. In the 2001 - 2002 academic year, he was a visiting lecturer and post-doctoral research associate with Cornell University. He joined the Department of Electrical and Computer Engineering at the University of Toronto in 2002, where he is now a Professor. His current research interests are in networked systems and mobile communications. He has served as an editor for the IEEE Transactions on Communications since 2014 and an associate editor for the IEEE Transactions on Mobile Computing since 2017, and he was an editor for the IEEE Transactions on Wireless Communications from 2008 to 2013 and an associate editor for Wiley Security and Communication Networks from 2007 to 2016. He regularly serves on the organizational and technical committees of a number of conferences. He is an IEEE Fellow and a member of ACM and Tau Beta Pi.