Joint Offloading and Resource Allocation for Computation and Communication in Mobile Cloud with Computing Access Point

Meng-Hsi Chen†, Ben Liang†, Min Dong‡
†Dept. of Electrical and Computer Engineering, University of Toronto, Canada
‡Dept. of Electrical, Computer and Software Engineering, University of Ontario Institute of Technology, Canada

***Abstract*—We consider a general multi-user mobile cloud computing system with a computing access point (CAP), where each mobile user has multiple independent tasks that may be processed locally, at the CAP, or at a remote cloud server. 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' tasks as well as the allocation of computation and communication resources, to minimize the overall cost of energy, computation, and delay for all users. This problem is NP-hard in general. We propose an efficient three-step algorithm comprising of semidefinite relaxation (SDR), alternating optimization (AO), and sequential tuning (ST). It is shown to always compute a locally optimal solution, and give nearly optimal performance under a wide range of parameter settings. Through evaluating the performance of different combinations of the three components of this SDR-AO-ST algorithm, we provide insights into their roles and contributions in the overall solution. We further compare the performance of SDR-AO-ST against a lower bound to the minimum cost, purely local processing, purely cloud processing, and hybrid local-cloud processing without using the CAP. Our numerical results demonstrate the effectiveness of the proposed algorithm in the joint management of computation and communication resources in mobile cloud computing systems with a CAP.**

I. Introduction

Mobile Cloud Computing (MCC) brings abundant cloud resources to extend the capabilities of resource-limited mobiles devices to improve the user experience [1]. With the help of cloud resources, mobile devices can potentially reduce their energy consumption or processing delay by offloading tasks to the cloud for data gathering, processing, and storage. 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 [1].

For a single user offloading its entire application to the cloud, the tradeoff between energy saving and computing performance was studied in [2]–[5]. The authors of [6]–[10] considered multi-user scenarios with a single application or task per user, where the entire application is offloaded to the cloud. Different from such whole-application offloading, the authors of [11]–[15] considered partitioning an application into multiple tasks. In all these cases, the partitioning problem results in integer programming that is NP-hard in general.

Instead of conventional MCC where only mobile devices and the remote cloud server can process tasks, Mobile Edge Computing (MEC), as defined by the European Telecommunications Standards Institute (ETSI), refers to an MCC system where additional computing resources are installed locally at the base station of a cellular network, in part to reduce the communication delay for those offloaded tasks [16], [17]. MEC shares similarities with micro cloud centers [18], cloudlets [19], and fog computing [20], but the MEC computing hosts are managed by the network service provider. Generalizing the concept of MEC, the computing access point (CAP) is a wireless access point or a cellular base station with built-in computation capability. The mobile users' computing tasks may be processed locally at the mobile devices, sent to the CAP, or further forwarded to a remote cloud server. In our previous works, we had studied the scheduling of computation and communication resources in a CAP for a single mobile user [21] and multiple mobile users each with a single task only [22] [23], showing substantial system performance improvement.

In this work, we consider a general cloud access network consisting of one remote cloud server, one CAP, and multiple mobile users, each having multiple independent tasks. This multi-user multi-task scenario adds substantial challenge to system design, since we need to jointly consider both the offloading decisions of all tasks for each user and the sharing of computation and communication resources among all users as they compete to offload tasks through a wireless link with limited capacity, and compete to have some of their tasks processed at a CAP with limited computation resource. Furthermore, since each user has multiple tasks, it is difficult to characterize the overall delay for each user to finish the processing of all tasks, since the transmission of some tasks and the processing of other tasks at multiple locations may overlap in time in complicated patterns.

We aim to conserve energy and reduce the cost and delay for all users. In particular, the transmission and processing delays of the offloaded tasks of a user are affected by their

This work has been funded in part by grants STPGP-447497 and RGPIN-2015-05506 from the Natural Sciences and Engineering Research Council (NSERC) of Canada and in part by the Ontario Ministry of Research and Innovation under an Early Researcher Award.

assigned computation and communication resources. Furthermore, optimal offloading decision and resource allocation must 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.

Therefore, we focus on jointly optimizing the offloading decision and the allocation of computation and communication resources of all tasks, to minimize a weighted sum of the costs of energy, cost of computation, and the delay for all users. The resultant mixed integer programming problem can be reformulated as a non-convex quadratically constrained quadratic program (QCQP) [24], which is NP-hard in general. To solve this challenging problem, we propose an efficient three-step algorithm that utilizes semidefinite relaxation (SDR) [25], alternating optimization (AO), and sequential tuning (ST). 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.

We also develop a lower bound of the minimum cost as the benchmark for performance evaluation. Simulation results show that the obtained local minimum solution gives nearly optimal performance under various parameter settings. Furthermore, we conduct simulation experiment on alternative combinations of the three components of the SDR-AO-ST algorithm, clarifying their roles and contributions to the overall system performance. Finally, we compare the performance of SDR-AO-ST 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.

The rest of this paper is organized as follows. In Section II, we describe the system model and present the problem formulation. In Section III, we provide details of three components of the proposed algorithm. For performance comparison, the lower bound of the minimum system cost is discussed in Section IV. We present numerical results in Section V and conclude in Section VI.

II. System Model and Problem Formulation

A. Mobile Cloud Offloading with Multiple Users and Tasks

Consider a general cloud access network consisting of one cloud server, one CAP, and N mobile users, each having M independent tasks, as shown in Fig. 1[1]. Each mobile user can process its tasks locally or offload some of them. Those offloaded tasks may be processed at the CAP or be further forwarded to the remote cloud. 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,

[1]We assume the same M for all users only to simplify mathematical notation. Our system model can be easily extended to the case where each mobile user has a different number of tasks.

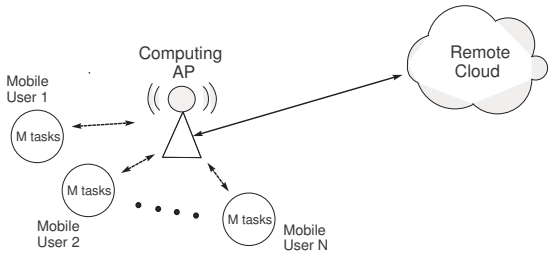

Fig. 1. System model

respectively. The offloading decisions are constrained by

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

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. Cost of Local Processing

The input data size, output data size, and processing cycles of user i's task j are denoted by $D_{\text{in}}(ij)$, $D_{\text{out}}(ij)$, and $Y(ij)$, respectively. 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 CAP Processing

Since there are multiple tasks offloaded to the CAP and some of them are processed by the CAP, we need to allocate the computation and communication resources available at the CAP. If user i's task j is offloaded to the CAP, we denote by E_{ij}^t and E_{ij}^r, respectively, the energy consumed for wireless transmission and reception by the user.

We further denote the uplink and downlink transmission times between user i and the CAP by $T_{ij}^t = D_{\text{in}}(ij)/(\eta_i^u c_i^u)$ and $T_{ij}^r = D_{\text{out}}(ij)/(\eta_i^d c_i^d)$, where c_i^u and c_i^d are the uplink and downlink bandwith allocated to user i, and η_i^u and η_i^d are the spectral efficiency of uplink and downlink transmission between user i and the CAP, respectively[2]. Furthermore, c_i^u and c_i^d are limited by the uplink bandwidth C_{UL} and downlink bandwidth C_{DL} as follows

$$\sum_{i=1}^{N} c_i^u \le C_{\text{UL}}, \tag{2}$$

and

$$\sum_{i=1}^{N} c_i^d \le C_{\text{DL}}. \tag{3}$$

We may consider also a total bandwidth constraint

$$\sum_{i=1}^{N} (c_i^u + c_i^d) \le C_{\text{Total}}. \tag{4}$$

If user i's task j is processed by the CAP (i.e., instead of further forwarded to the remote cloud), denote its processing time by $T_{ij}^a = Y(ij)/f_i^a$, where f_i^a is the assigned processing

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

rate, which is limited by the total processing rate f_A at the CAP:

$$\sum_{i=1}^{N} f_i^a \leq f_A. \tag{5}$$

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

D. Cost of Cloud Processing

If the task is further offloaded to the cloud from the CAP, 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, there is the additional transmission time between the CAP 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 CAP 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 CAP-cloud link is likely to be a high-capacity wired connection in comparison with the limited wireless links between the mobile users and the CAP, 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, which is given for all i and j.

E. Problem Formulation

We define the total system cost 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. Our objective is to minimize the total system cost by jointly optimizing the task offloading decisions $\mathbf{x}_{ij} = [x_{ij}^l, x_{ij}^a, x_{ij}^c]^T$ and the communication and CAP processing resource allocation $\mathbf{r}_i = [c_i^u, c_i^d, f_i^a]^T$. This optimization problem is formulated as follows:

$$\min_{\{\mathbf{x}_{ij}\},\{\mathbf{r}_i\}} \quad \sum_{i=1}^{N}\Bigg[\sum_{j=1}^{M}(E_{ij}^l x_{ij}^l + E_{ij}^A x_{ij}^a + E_{ij}^C x_{ij}^c) + \rho_i \max\{T_i^L, T_i^A, T_i^C\}\Bigg] \tag{6}$$

$$\text{s.t.} \quad (1),(2),(3),(4),(5),$$

$$c_i^u, c_i^d, f_i^a, \geq 0, \forall i, \tag{7}$$

$$x_{ij}^l, x_{ij}^a, x_{ij}^c \in \{0,1\}, \forall i, j, \tag{8}$$

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 the cloud, respectively, with α and β being their relative weights; in addition, $T_i^L \triangleq \sum_{j=1}^{M} T_{ij}^l x_{ij}^l$ 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 ρ_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 ρ_i can be adjusted to impose different emphasis on delay and energy consumption. For simplicity, we normalize the weighted sum cost to have the unit of energy. Problem (6) can be solved by any controller in the network after collecting all required information. For example, it may be the wireless access point or the cellular base station.

The above mixed-integer programming problem is difficult to solve in general. Moreover, we note that the overall delay for CAP processing, T_i^A, and cloud processing, T_i^C, are challenging to calculate exactly. This is because, when there are multiple tasks offloaded by a users, the transmission times and processing times of these tasks may overlap in an unpredictable manner, which depends on the offloading decision, computation and communication resource allocation, and task scheduling order. In fact, since T_i^A and T_i^C consist of the uplink transmission time, remote-processing time, and downlink transmission time of all offloaded tasks, it may be viewed as the output of a multi-machine flowshop schedule, which remains an open research problem [26]. Since both T_i^A and T_i^C are not precisely tractable, we will 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 SDR-AO-ST algorithm, the upper and lower bounds give estimates to the total system cost that are close to each other.

III. Multi-User Multi-Task Offloading Solution

The joint optimization problem (6) is a mixed-integer non-convex programming problem. To find an efficient solution to problem (6), in the following, we first present upper-bound and lower-bound formulations of both T_i^A and T_i^C, then transform the optimization problem (6) into a separable QCQP, and finally propose a three-step SDR-AO-ST algorithm containing a separable SDR approach, alternating optimization (AO), and sequential tuning (ST), to obtain the binary offloading decisions $\{\mathbf{x}_{ij}\}$ and the communication and processing resource allocation $\{\mathbf{r}_i\}$.

A. Bounds of CAP-Processing and Cloud-Processing Delays

When a mobile user offloads more than one task to the CAP or the cloud, there will be overlaps in the communication and processing times as mentioned above, making it difficult to exactly characterize the overall delays T_i^A and T_i^C. However, we have the following upper bounds, i.e., the *worst-case delays*:

$$T_i^{A_{(U)}} \triangleq \sum_{j=1}^{M}((T_{ij}^t + T_{ij}^r)(x_{ij}^a + x_{ij}^c) + T_{ij}^a x_{ij}^a), \tag{9}$$

$$T_i^{C_{(U)}} \triangleq \sum_{j=1}^{M}((T_{ij}^t + T_{ij}^r)(x_{ij}^a + x_{ij}^c) + (T_{ij}^{ac} + T_{ij}^c)x_{ij}^c). \quad (10)$$

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.

Later, for performance benchmarking, we will also need the best-case delays. We separate the offloading delays of all mobile users into several components and only consider the largest one among them as the lower bounds of T_i^A and T_i^C:

$$T_i^{A_{(L)}} \triangleq \max\{T_i^u, T_i^d, T_i^{a'}\}, \quad (11)$$

$$T_i^{C_{(L)}} \triangleq \max\{T_i^{u'}, T_i^{d'}, T_i^{uac}, T_i^{dac}, T_i^{c'}\}, \quad (12)$$

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

In the proposed SDR-AO-ST algorithm, we use the worst-case delays $T_i^{A_{(U)}}$ and $T_i^{C_{(U)}}$ in optimization problem (6) to obtain an approximate solution, which gives an upper bound to the actual total system cost. We then use $T_i^{A_{(L)}}$ and $T_i^{C_{(L)}}$ similarly, to obtain a lower bound of the total system cost, for performance benchmarking. We will show in Section V that, despite using the worst-case delays as parametric input to the proposed SDR-AO-ST algorithm, we 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.

In the next three subsections, we describe the details of the proposed three-step algorithm and show the local optimum property of the obtained binary offloading decisions $\{\mathbf{x}_{ij}\}$ and communication and processing resource allocation $\{\mathbf{r}_i\}$.

B. Step 1: QCQP Transformation and Semidefinite Relaxation

In order to obtain the eventual SDR formulation, we first transform the optimization problem (6), with $T_i^{A_{(U)}}$ and $T_i^{C_{(U)}}$ substituting for T_i^A and T_i^C, respectively, into a separable QCQP. We first rewrite the integer constraint (8) as

$$x_{ij}^s(x_{ij}^s - 1) = 0, \quad \forall i, j, \quad (13)$$

for $s \in \{l, a, c\}$. Furthermore, we introduce additional auxiliary variables $t_i = \max\{T_{L_i}, T_i^{A_{(U)}}, T_i^{C_{(U)}}\}$, $D_i^u = \sum_{j=1}^M D_{\text{in}}(ij)(x_{ij}^a + x_{ij}^c)/\eta_i^u c_i^u$, $D_i^d = \sum_{j=1}^M D_{\text{out}}(ij)(x_{ij}^a + x_{ij}^c)/\eta_i^d c_i^d$, and $D_i^a = \sum_{j=1}^M Y(ij)x_{ij}^a/f_i^a$. Let $\mathbf{d}_i = (D_i^u, D_i^d, D_i^a)$. The problem (6) is now transformed into the following form:

$$\min_{\{\mathbf{x}_{ij}\},\{\mathbf{r}_i,\mathbf{d}_i,t_i\}} \sum_{i=1}^{N}\left[\sum_{j=1}^{M}(E_{ij}^l x_{ij}^l + E_{ij}^A x_{ij}^a + E_{ij}^C x_{ij}^c) + \rho_i t_i\right] \quad (14)$$

$$\begin{aligned}
\text{s.t.} \quad & \sum_{j=1}^{M} T_{ij}^l x_{ij}^l \le t_i, \quad \forall i, \\
& D_i^u + D_i^d + D_i^a \le t_i, \quad \forall i, \\
& D_i^u + D_i^d + \sum_{j=1}^{M}(T_{ij}^{ac} + T_{ij}^c)x_{ij}^c \le t_i, \quad \forall i, \\
& \sum_{j=1}^{M} D_{\text{in}}(ij)(x_{ij}^a + x_{ij}^c) - \eta_i^u c_i^u D_i^u \le 0, \quad \forall i, \\
& \sum_{j=1}^{M} D_{\text{out}}(ij)(x_{ij}^a + x_{ij}^c) - \eta_i^d c_i^d D_i^d \le 0, \quad \forall i, \\
& \sum_{j=1}^{M} Y(ij)x_{ij}^a - f_i^a D_i^a \le 0, \quad \forall i, \\
& (1) - (5),\ (7),\ \text{and } (13).
\end{aligned}$$

Define $\mathbf{w}_i \triangleq [\mathbf{x}_{i1}^T, \ldots, \mathbf{x}_{iM}^T, c_i^u, D_i^u, c_i^d, D_i^d, f_i^a, D_i^a, t_i]^T$, for all i, and $\mathbf{e}_i$ as the $(3M+7)\times 1$ standard unit vector with the ith entry being 1. The optimization problem (14) can now be further transformed into the following equivalent QCQP formulation:

$$\begin{aligned}
\min_{\{\mathbf{w}_i\}} \quad & \sum_{i=1}^{N} \mathbf{b}_i{}^T \mathbf{w}_i \quad (15) \\
\text{s.t.} \quad & (\mathbf{b}_i^l)^T\mathbf{w}_i \le 0, \quad (\mathbf{b}_i^A)^T\mathbf{w}_i \le 0, \quad (\mathbf{b}_i^C)^T\mathbf{w}_i \le 0, \quad \forall i, \\
& \mathbf{w}_i{}^T\mathbf{A}_i^k\mathbf{w}_i + (\mathbf{b}_i^k)^T\mathbf{w}_i \le 0, \quad k \in \{u, d, a\}, \quad \forall i, \\
& (\mathbf{b}_{ij}^P)^T\mathbf{w}_i = 0, \quad \forall i, j, \\
& \sum_{i=1}^{N}(\mathbf{b}_i^U)^T\mathbf{w}_i \le C_{\text{UL}}, \quad \sum_{i=1}^{N}(\mathbf{b}_i^D)^T\mathbf{w}_i \le C_{\text{DL}}, \\
& \sum_{i=1}^{N}(\mathbf{b}_i^S)^T\mathbf{w}_i \le C_{\text{Total}}, \quad \sum_{i=1}^{N}(\mathbf{b}_i^f)^T\mathbf{w}_i \le f_A, \\
& \mathbf{w}_i{}^T\text{diag}(\mathbf{e}_p)\mathbf{w}_i - \mathbf{e}_p{}^T\mathbf{w}_i = 0,\ p \in \{1, ..., 3M\}, \quad \forall i, \\
& \mathbf{w}_i \ge \mathbf{0}, \quad \forall i,
\end{aligned}$$

where

$$\mathbf{A}_i^{u'} \triangleq -0.5\begin{bmatrix} 0 & \eta_i^u \\ \eta_i^u & 0 \end{bmatrix}, \quad \mathbf{A}_i^{d'} \triangleq -0.5\begin{bmatrix} 0 & \eta_i^d \\ \eta_i^d & 0 \end{bmatrix},$$

$$\mathbf{A}_i^u \triangleq \begin{bmatrix} \mathbf{0}_{3M\times 3M} & \mathbf{0}_{3M\times 2} & \mathbf{0}_{3M\times 5} \\ \mathbf{0}_{2\times 3M} & \mathbf{A}_i^{u'} & \mathbf{0}_{2\times 5} \\ \mathbf{0}_{5\times 3M} & \mathbf{0}_{5\times 2} & \mathbf{0}_{5\times 5} \end{bmatrix},$$

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

$$\mathbf{A}_i^{a'} \triangleq -0.5\begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix},$$

$$\mathbf{A}_i^a \triangleq \begin{bmatrix} \mathbf{0}_{(3M+4)\times(3M+4)} & \mathbf{0}_{(3M+4)\times 2} & \mathbf{0}_{(3M+4)\times 1} \\ \mathbf{0}_{2\times(3M+4)} & \mathbf{A}_i^{a'} & \mathbf{0}_{2\times 1} \\ \mathbf{0}_{1\times(3M+4)} & \mathbf{0}_{1\times 2} & 0 \end{bmatrix},$$

$$\mathbf{b}_i \triangleq [E_{i1}^l, E_{i1}^A, E_{i1}^C, \ldots, E_{iM}^l, E_{iM}^A, E_{iM}^C, \mathbf{0}_{1\times 6}, \rho_i]^T,$$

$$\mathbf{b}_i^l \triangleq [T_{i1}^l, 0, 0, \ldots, T_{iM}^l, 0, 0, \mathbf{0}_{1\times 6}, -1]^T,$$

$$\mathbf{b}_i^A \triangleq [\mathbf{0}_{1\times 3M}, 0, 1, 0, 1, 0, 1, -1]^T,$$

$$\mathbf{b}_i^C \triangleq [0, 0, (T_{i1}^{ac} + T_{i1}^c), \ldots, 0, 0, (T_{iM}^{ac} + T_{iM}^c), \mathbf{b}_i^{C'}]^T,$$

$$\mathbf{b}_i^{C'} \triangleq [0, 1, 0, 1, 0, 0, -1],$$

$$\mathbf{b}_i^u \triangleq [0, D_{\text{in}}(i1), D_{\text{in}}(i1), \ldots, 0, D_{\text{in}}(iM), D_{\text{in}}(iM), \mathbf{0}']^T,$$

$$\mathbf{b}_i^d \triangleq [0, D_{\text{out}}(i1), D_{\text{out}}(i1), \ldots, 0, D_{\text{out}}(iM), D_{\text{out}}(iM), \mathbf{0}']^T,$$

$$\mathbf{b}_i^a \triangleq [0, Y(i1), 0, \ldots, 0, Y(iM), 0, \mathbf{0}']^T, \quad \mathbf{0}' \triangleq \mathbf{0}_{1\times 7},$$

$$\mathbf{b}_{ij}^P \triangleq \mathbf{e}_{3(j-1)+1} + \mathbf{e}_{3(j-1)+2} + \mathbf{e}_{3(j-1)+3},$$

$$\mathbf{b}_i^U \triangleq [\mathbf{0}_{1\times 3M}, 1, \mathbf{0}_{1\times 6}]^T, \quad \mathbf{b}_i^D \triangleq [\mathbf{0}_{1\times(3M+2)}, 1, \mathbf{0}_{1\times 4}]^T,$$

$$\mathbf{b}_i^S \triangleq [\mathbf{0}_{1\times 3M}, 1, 0, 1, \mathbf{0}_{1\times 4}]^T, \quad \mathbf{b}_i^f \triangleq [\mathbf{0}_{1\times(3M+4)}, 1, 0, 0]^T.$$

Comparing the optimization problems (14) and (15), we note that all constraints have one-to-one correspondence. However, we omit the derivation details due to page limitation.

Note that the optimization problem (15) is a non-convex separable QCQP problem [24], which is NP-hard in general. To find an approximate solution, we apply the separable SDR approach [25], where we relax the problem into a separable semidefinite programming (SDP) problem. Specifically, define $\mathbf{Z}_i \triangleq [\mathbf{w}_i{}^T, 1]^T[\mathbf{w}_i{}^T, 1]$. By dropping the rank constraint $\text{rank}(\mathbf{Z}_i) = 1$, we have the following separable SDP problem:

$$\begin{aligned} \min_{\{\mathbf{Z}_i\}} \quad & \sum_{i=1}^N \text{Tr}(\mathbf{G}_i \mathbf{Z}_i) \qquad (16) \\ \text{s.t.} \quad & \text{Tr}(\mathbf{G}_i^r \mathbf{Z}_i) \le 0, \quad r \in \{l, A, C, u, d, a\}, \quad \forall i, \\ & \text{Tr}(\mathbf{G}_{ij}^P \mathbf{Z}_i) = 0, \quad \forall i, j, \\ & \sum_{i=1}^N \text{Tr}(\mathbf{G}_i^U \mathbf{Z}_i) \le C_{\text{UL}}, \quad \sum_{i=1}^N \text{Tr}(\mathbf{G}_i^D \mathbf{Z}_i) \le C_{\text{DL}}, \\ & \sum_{i=1}^N \text{Tr}(\mathbf{G}_i^S \mathbf{Z}_i) \le C_{\text{Total}}, \quad \sum_{i=1}^N \text{Tr}(\mathbf{G}_i^f \mathbf{Z}_i) \le f_A, \\ & \text{Tr}(\mathbf{G}_p^I \mathbf{Z}_i) = 0, \quad p \in \{1, ..., 3M\}, \quad \forall i, \\ & \mathbf{Z}_i(3M+8, 3M+8) = 1, \quad \forall i, \quad \mathbf{Z}_i \succeq \mathbf{0}, \quad \forall i, \end{aligned}$$

where

$$\mathbf{G}_i \triangleq \begin{bmatrix} \mathbf{0} & \frac{1}{2}\mathbf{b}_i \\ \frac{1}{2}\mathbf{b}_i^T & 0 \end{bmatrix},$$

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

$$\mathbf{G}_i^q \triangleq \begin{bmatrix} \mathbf{0} & \frac{1}{2}\mathbf{b}_i^q \\ \frac{1}{2}(\mathbf{b}_i^q)^T & 0 \end{bmatrix}, \quad q \in \{l, A, C, U, D, S, f\},$$

$$\mathbf{G}_{ij}^P \triangleq \begin{bmatrix} \mathbf{0} & \frac{1}{2}\mathbf{b}_{ij}^P \\ \frac{1}{2}(\mathbf{b}_{ij}^P)^T & 0 \end{bmatrix}, \quad \mathbf{G}_p^I \triangleq \begin{bmatrix} \text{diag}(\mathbf{e}_p) & -\frac{1}{2}\mathbf{e}_p \\ -\frac{1}{2}\mathbf{e}_p^T & 0 \end{bmatrix}.$$

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 [27]. However, since problem (16) is a ralaxation of the problem (14), the optimal objective of the problem (16) is only a lower bound of the optimal solution of the problem (14) 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 (14).

Define $\mathbf{x} \triangleq [\mathbf{x}_1^T, \ldots, \mathbf{x}_N^T]^T$, where $\mathbf{x}_i \triangleq [\mathbf{x}_{i1}^T, \ldots, \mathbf{x}_{iM}^T]^T$, for all i. Note that, first, only the upper-left $3M \times 3M$ submatrix of $\mathbf{Z}_i^*$, denote by $\mathbf{Z}_i'^*$, for all i, is needed to recover the offloading decision $\mathbf{x}$; second, each diagonal entry in $\mathbf{Z}_i'^*$ is always between 0 and 1. That is, take the diagonal entries of $\mathbf{Z}_i'^*$ as $\mathbf{p}_i = [\mathbf{p}_{i1}^T, \ldots, \mathbf{p}_{iM}^T]^T$, where $\mathbf{p}_{ij} = [p_{ij}^l, p_{ij}^a, p_{ij}^c]^T$; we have each element in $\mathbf{p}_{ij}$ in the interval $[0, 1]$, for all i, j. We recover the feasible decisions $\mathbf{x}_i^{\text{sdr}}$ using $\mathbf{p}_i$, where

$$\mathbf{x}_{ij}^{\text{sdr}} = \begin{cases} [1,0,0]^T, \text{ if } \max\limits_{s\in\{l,a,c\}} p_{ij}^s = p_{ij}^l \text{ (local processing)}, \\ [0,1,0]^T, \text{ if } \max\limits_{s\in\{l,a,c\}} p_{ij}^s = p_{ij}^a \text{ (CAP processing)}, \\ [0,0,1]^T, \text{ if } \max\limits_{s\in\{l,a,c\}} p_{ij}^s = p_{ij}^c \text{ (cloud processing)}, \end{cases} \quad (17)$$

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

After obtaining the offloading decision $\mathbf{x}^{\text{sdr}}$, the optimization problem (14) is reduced to the optimization of computation and communication resource allocation $\{\mathbf{r}_i\}$, which is given by

$$\begin{aligned} \min_{\{\mathbf{r}_i\}} \quad & \left(E + \sum_{i=1}^N \rho_i \max\{T_i^L, T_i^{A_{(U)}}, T_i^{C_{(U)}}\} \right) \qquad (18) \\ \text{s.t.} \quad & (2) - (5), \text{ and } (7), \end{aligned}$$

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. This resource allocation problem (18) is convex, which can be solved optimally using standard convex optimization solvers.

C. Step 2: Improvement to SDR by Alternating Optimization

After obtaining a feasible solution $(\mathbf{x}^{\text{sdr}}, \{\mathbf{r}_i^{\text{sdr}^*}\})$ from the SDR 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 (14) is reduced to the resource allocation problem (18), which is convex and optimal resource allocation can be obtained. On the other hand, if the resource allocation $\{\mathbf{r}_i\}$ is given, the optimization problem (14) is reduced to the optimization of offloading decisions $\{\mathbf{x}_{ij}\}$ as follows:

$$\begin{aligned} \min_{\{\mathbf{x}_{ij}\}} \quad & \sum_{i=1}^N \Bigg[\sum_{j=1}^M (E_{ij}^l x_{ij}^l + E_{ij}^A x_{ij}^a + E_{ij}^C x_{ij}^c) \\ & \qquad + \rho_i \max\{T_i^L, T_i^{A_{(U)}}, T_i^{C_{(U)}}\} \Bigg] \qquad (19) \\ \text{s.t.} \quad & (1) \text{ and } (13). \end{aligned}$$

The offloading decision problem (19) 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 [21], this can be solved near-optimally by either using an SDR approach or relaxing the integer constraints to interval constraints. Since the optimization problem (14) can be separated into two subproblems (18) and (19), 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 (19) based on $\{\mathbf{r}_i^{\text{ao}^*}\}$ to find the corresponding offloading decision $\mathbf{x}^{\text{ao}'}$.

ii Solve problem (18) 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 (19), 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 $(\mathbf{x}^{\text{sdr}}, \{\mathbf{r}_i^{\text{sdr}^*}\})$.

D. Step 3: Sequential Tuning 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 (6).

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 (18). 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 (6), 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 theorem.

Theorem 1: $(\mathbf{x}^{st^*}, \{\mathbf{r}_i^{st^*}\})$ *obtained from the sequential tuning procedure is a locally optimal solution to the original non-convex optimization problem* (6).

Algorithm 1 SDR-AO-ST Algorithm

Step 1: Initial offloading solution via SDR

Obtain optimal solution $\{\mathbf{Z}_i^*\}$ of problem (16). Extract the upper-left $3M \times 3M$ sub-matrices $\{\mathbf{Z'}_i^*\}$ from $\{\mathbf{Z}_i^*\}$.

Record the values of diagonal terms in $\mathbf{Z'}_i^*$ 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 (17), and solve the resource allocation problem (18) based on $\mathbf{x}^{\text{sdr}}$.

Step 2: Alternating optimization

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^{ao^*}.

Set AO = False.

while AO == False **do**

  Solve problem (19) based on $\{\mathbf{r}_i^{\text{ao}^*}\}$ to find the corresponding offloading decision $\mathbf{x}^{\text{ao}'}$;

  Solve problem (18) 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

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^{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 $n \leq N$ and $m \leq M$ **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 (18); 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 $\{\mathbf{r}_i^{\text{st}^*}\}$.

E. Overall SDR-AO-ST Algorithm

We summarize the above three-step SDR-AO-ST algorithm in Algorithm 1.

Even though each of the SDR, AO, and ST steps above can be used separately to provide a feasible solution to the original optimization problem (6), when combined together in the proposed manner, they each serves an important role in the overall algorithm design. First, SDR provides a suitable start-

ing 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 V-B, 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 V-B.

IV. Lower Bound on the Optimal Solution

Previously, the cost function in our original optimization problem (14) considers the worst-case transmission-plus-processing delays (9) and (10) 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^A and T_i^C will be less than $T_i^{A_{(U)}}$ and $T_i^{C_{(U)}}$, respectively. Therefore, the actual cost based on SDR-AO-ST will be lower than the worst-case cost.

However, we are still interested in the performance of SDR-AO-ST compared with an optimal solution. Therefore, we introduce a lower bound of the optimal solution to the original problem (6). We first introduce a new optimization problem, where $T_i^{A_{(L)}}$ and $T_i^{C_{(L)}}$ are used instead of T_i^A and T_i^C and the objective function is replaced by its lower bound, as follows:

$$\min_{\{\mathbf{x}_{ij}\},\{\mathbf{r}_i\}} \sum_{i=1}^{N} \Bigg[\sum_{j=1}^{M} (E_{ij}^l x_{ij}^l + E_{ij}^A x_{ij}^a + E_{ij}^C x_{ij}^c) + \rho_i \max\{T_i^u, T_i^d, T_i^{u'}, T_i^{d'}, T_i^{a'}, T_i^{uac}, T_i^{dac}, T_i^{c'}\} \Bigg] \tag{20}$$

$$\text{s.t.} \quad (1)-(5),\ (7),\ \text{and}\ (13).$$

Notice that under the same offloading decisions and resource allocation, the objective function in (20) is always lower than the actual cost.

Since the above optimization problem (20) is still non-convex, we formulate a separable SDR problem similar to (16), whose details are omitted due to page limitation. We note that, due to the SDP relaxation that enlarges the feasible set, the optimal objective of this SDR problem is smaller than the optimal objective of (20). Hence, it can serve as a lower bound to the minimum total system cost defined by the original optimization problem (6).

In Section V, we show that the proposed SDR-AO-ST method provides not only a local optimum solution but also nearly optimal performance compared with the lower bound.

V. Performance Evaluation

We evaluate the performance of SDR-AO-ST through Matlab simulation 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 [28], which is based on a Nokia smart device, and set the number of mobile users as $N = 5$. Each user has $M = 4$ independent tasks. According to Tables 1 and 3 in [28], the mobile device has CPU rate 500×10^6 cycles/s and unit processing energy consumption $\frac{1}{730\times10^6}$ J/cycle. The local computation time per bit is 4.75×10^{-7} s and local processing energy consumption per bit is 3.25×10^{-7} J. We consider the x264 CBR encode application, which requires 1900 cycles/byte [28], i.e., $Y(ij) = 1900 D_{\text{in}}(ij)$. 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 1.42×10^{-7} J/bit as indicated in Table 2 in [28]. For simplicity, we set $\eta_i^u = \eta_i^d = 3.5$ b/s/Hz for all i. The CPU rate assigned to each user at the remote cloud is 10×10^9 cycle/s, and the total CAP processing rate f_A is 10×10^9 cycle/s. When tasks are sent from the CAP to the cloud, the transmission rate R_{ac} is 15 Mpbs. The CAP usage cost and cloud usage cost are assumed to be $C_{ij}^a = D_{\text{in}}(ij) + \lambda_1/f_A + \lambda_2/C_{\text{UL}} + \lambda_3/C_{\text{DL}}$ and $C_{ij}^c = D_{\text{in}}(ij) + \lambda_1/f_C + \lambda_2/C_{\text{UL}} + \lambda_3/C_{\text{DL}}$, respectively, where $\lambda_1 = 10^{18}$ bits$\times$cycles/s and $\lambda_2 = \lambda_3 = 10^{16}$ bits$\times$Mbps, which accounts for the input data size, processing rate, and uplink and downlink capacities. Also, $\alpha = 1.5 \times 10^{-7}$ J/bit, $\beta = 3 \times 10^{-7}$ J/bit, and $\rho_i = 1$ J/s for all i. Finally, all simulation results are obtained by averaging over 100 realizations of the input and output data sizes of each task.

B. Contribution of the Algorithm Components

To demonstrate the role and contribution of each step in the SDR-AO-ST algorithm, we first compare it with the following methods: 1) the *SDR* method where only the first step of SDR-AO-ST is applied, 2) the *SDR-ST* method where the AO step is skipped, 3) the *AO-ST* method where only the last two steps of SDR-AO-ST 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 SDR-AO-ST 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 (20).

We show the system cost and the run time ratio vs. α in Figs. 2 and 3, respectively. Since α is the weight on the CAP usage cost, more tasks compete at the CAP when α is smaller. We observe that SDR-AO-ST can reduce the system cost by up to 20% compared with purely applying SDR and is much closer to the lower bound of optimum. Furthermore, though SDR-ST, AO-ST, and ST can provide similarly low cost as SDR-AO-ST, 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 and AO steps in

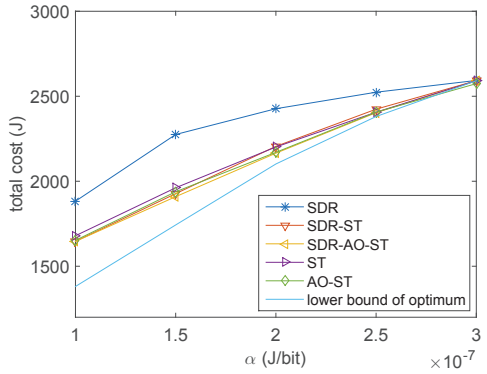

Fig. 2. Total system cost versus α.

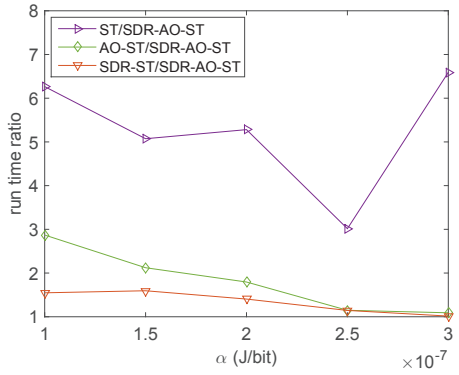

Fig. 3. Run time ratio versus α.

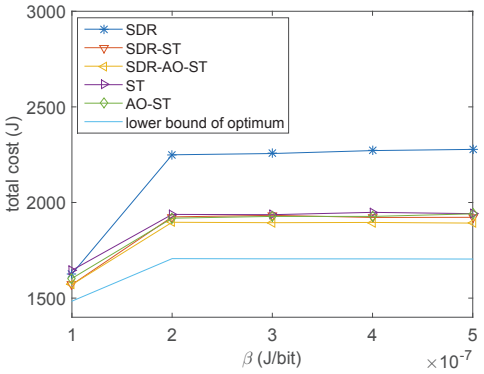

Fig. 4. Total system cost versus β.

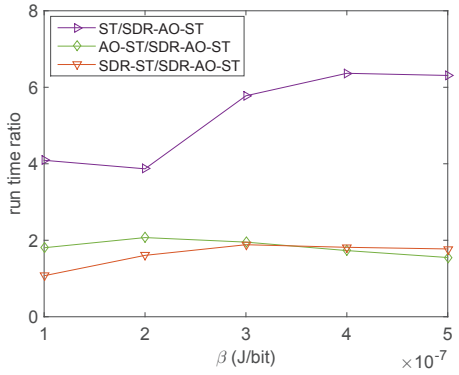

Fig. 5. Run time ratio versus β.

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. 4 and 5, where we show the system cost and the run time ratio vs. the weight β on the cloud usage cost, and in Figs. 6 and 7, where we show the system cost and the run time ratio vs. M, the number of tasks per user. When β is large, all tasks are more likely to be processed by either the mobile users or the CAP. More importantly, SDR-AO-ST is shown to be more scalable, since the run-time ratios are nearly linearly increasing with the number of tasks per user.

C. Comparison with Further Alternatives

For further performance evaluation, we also consider the following schemes: 1) the *local processing only* scheme where all tasks are processed locally by mobile users, 2) the *cloud processing only* scheme where all tasks are offloaded to the cloud and the cost is obtained based on $T_i^{C(L)}$, 3) the *lower bound of local-cloud* where the same approximation procedure as the *lower bound of optimum* is applied, except that the CAP is not used for computation, and 4) the *random mapping* scheme where each task is processed at different locations with equal probability. As shown in Figs. 8 and 9, the lower bound of optimum converges to the lower bound of local-cloud as α becomes large and the lower bound of local-cloud converges to the local only method as β becomes large. In both figures, SDR-AO-ST 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.

VI. Conclusion

We consider a general mobile cloud computing system consisting of multiple users, one CAP, 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 decision on task offloading and allocation of computation and communication resources. The proposed SDR-AO-ST algorithm uses a three-step approach to obtain a local optimum. By comparison with a lower bound of the minimum cost, we show that SDR-AO-ST substantially out performs several alternatives and often gives nearly optimal performance for a wide range of parameter settings.

References

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

[2] 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.

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

[4] W. Zhang, Y. Wen, K. Guan, D. Kilper, H. Luo, and D. Wu, "Energy-optimal mobile cloud computing under stochastic wireless channel," *IEEE Transactions on Wireless Communications*, vol. 12, no. 9, pp. 4569–4581, Sep. 2013.

[5] O. Munoz, A. Pascual-Iserte, and J. Vidal, "Optimization of radio and computational resources for energy efficiency in latency-constrained application offloading," *IEEE Transactions on Vehicular Technology*, vol. 64, no. 10, pp. 4738–4755, Oct. 2015.

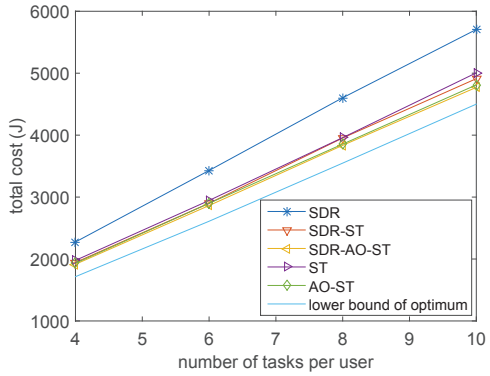

Fig. 6. Total system cost versus number of tasks per user.

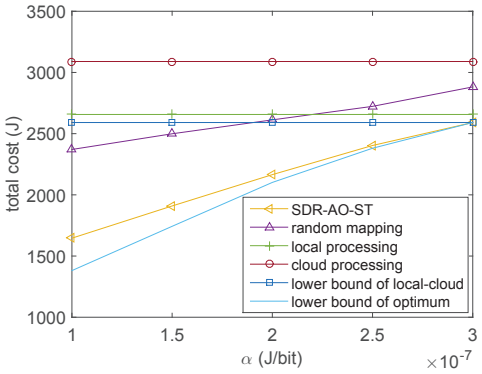

Fig. 8. Comparison of total system cost versus α.

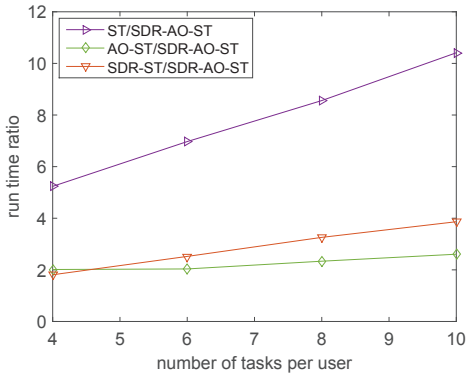

Fig. 7. Run time ratio versus number of tasks per user.

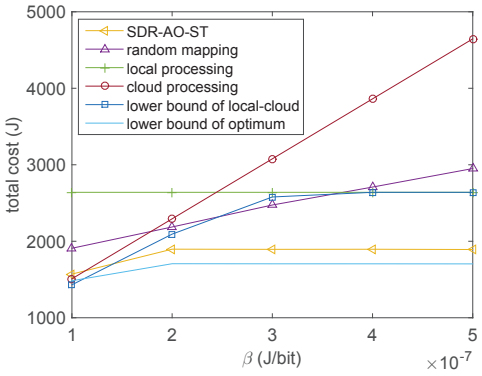

Fig. 9. Comparison of total system cost versus β.

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

[7] X. Chen, "Decentralized computation offloading game for mobile cloud computing," *IEEE Transactions on Parallel and Distributed Systems*, vol. 26, no. 4, pp. 974–983, Apr. 2015.

[8] S. Sardellitti, G. Scutari, and S. Barbarossa, "Joint optimization of radio and computational resources for multicell mobile-edge computing," *IEEE Transactions on Signal and Information Processing over Networks*, vol. 1, no. 2, pp. 89–103, Jun. 2015.

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

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

[11] 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 International Conference on Mobile Systems, Applications, and Services (MobiSys)*, Jan. 2010, pp. 49–62.

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

[13] 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 International Conference on Computer Communications (INFOCOM)*, Mar. 2012, pp. 945–953.

[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 International Conference on Computer Communications (INFOCOM)*, Apr. 2015, pp. 1894–1902.

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

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

[17] 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.

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

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

[20] 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 Computing*, Aug. 2012, pp. 13–16.

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

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

[23] M.-H. Chen, M. Dong, and B. Liang, "Multi-user mobile cloud offloading game with computing access point," in *Proc. IEEE International Conference on Cloud Networking (CLOUDNET)*, Oct. 2016, pp. 64–69.

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

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

[26] M. R. Garey, D. S. Johnson, and R. Sethi, "The complexity of flowshop and jobshop scheduling," *Mathematics of Operations Research*, vol. 1, no. 2, pp. 117–129, May 1976.

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

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