A Semidefinite Relaxation Approach to Mobile Cloud Offloading 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 mobile cloud computing scenario consisting of one user with multiple independent tasks, one computing access point (CAP), and one remote cloud server. The CAP can either process the received tasks from the mobile user or offload them to the cloud, providing additional computation capability over traditional mobile cloud computing systems. We aim to optimize the offloading decision of the user to minimize the overall cost of energy, computation, and delay. It is shown that the problem can be formulated as a non-convex quadratically constrained quadratic program, which is NP-hard in general. We propose an efficient offloading decision algorithm by semidefinite relaxation and a novel randomization mapping method. Our simulation results show that the proposed algorithm gives nearly optimal performance with only a small number of randomization iterations, and adding CAPs to the traditional dichotomy of mobile devices and remote cloud servers can drastically improve mobile cloud computing performance.**

I. Introduction

The paradigm of mobile cloud computing extends the capabilities of mobile devices to improve user experience. Although abundant cloud resources can be used to help mobile devices gather, store, and process data, the interaction between mobile devices and the cloud introduces difficult challenges. For example, while offloading tasks to the cloud, the tradeoff between energy savings and the computing performance affects the mobile user's experience. In addition, the communication delay and energy cost between mobile users and the cloud also play important roles that cannot be ignored. K. Kumar *et al.* [1] studied the general conditions that cloud offloading will be beneficial for a mobile user. M. Satyanarayanan *et al.* [2] proposed an architecture to replace the remote cloud with nearby cloudlets to reduce transmission latency. Cloud offloading of entire applications by a single user was studied in [3] [4]. The multiple user scenarios were addressed in [5] [6]. In contrast to whole application offloading, the authors of [7]–[10] considered application partitioning. In [7], E. Cuervo *et al.* proposed MAUI, a system developed to efficiently process the partitioned tasks. Clonecloud [8] and Thinkair [9] proposed further improvements. J. Niu *et al.* [10] additionally considered dynamic partitioning. In all cases, the partitioning problem results in integer programming and cannot be solved efficiently.

In this work, instead of conventional mobile cloud computing where only the mobile device and the cloud server can process tasks, we consider computing access points (CAPs), which are wireless access points or cellular base stations that also have built-in computation capability. CAPs may be provided by Internet service providers as a value-added service. Mobile devices that wish to offload a task first sends it to the CAP. The CAP may serve its conventional networking function and forward the task to the remote cloud server, or directly process the task by itself. The additional option of computation by the CAP reduces the need for access to the remote cloud server, hence decreasing the communication delay and also potentially the overall energy and computation cost.

However, the availability of CAP computation further complicates mobile task offloading decisions, adding an extra dimension of variability at the CAP. 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 paper, we focus on optimizing the offloading decision for independent tasks of a mobile user with one CAP and one remote cloud server, to minimize a weighted sum of the costs of energy, computation, and delay. We show that the problem can be formulated as a non-convex quadratically constrained quadratic program (QCQP), which is NP-hard in general. To solve this challenging problem, we propose an efficient heuristic algorithm based on semidefinite relaxation (SDR) [11] and a novel randomization mapping method. We give a numeric lower bound to the performance of the proposed algorithm. Our simulation results further demonstrate that the proposed algorithm gives nearly optimal performance with a small number of randomization iterations. Furthermore, we observe that the addition of a CAP can significantly reduce the energy and computation costs of mobile cloud computing over traditional systems where only the remote cloud server is available for task offloading.

II. System Model and Problem Formulation

Consider a cloud access scenario with a mobile user having N independent tasks, one CAP, and one remote cloud server, as shown in Fig. 1. The connection between the mobile user and the CAP is wireless, while the connection between the CAP and the cloud is usually wired. For the CAP, instead of just serving as a relay to always forward received tasks from the user to the cloud, we assume it also has the capability to process user tasks subject to its resource constraint. The mobile user needs to decide each task to be either processed

This work was funded in part by a grant from the Natural Sciences and Engineering Research Council of Canada.

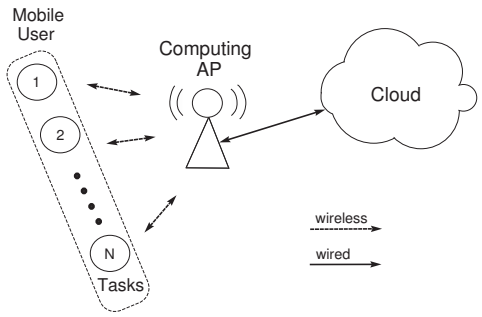

Fig. 1. System model

locally or offloaded. We further decide whether the CAP should process the offloaded task itself or further offload it to the cloud.

Denote the input and output data sizes and the application type of each task i by $D_{\text{in}}(i)$, $D_{\text{out}}(i)$, and App(i), respectively. For task i being locally processed by the mobile user, denote the corresponding energy consumption for processing by E_{l_i} and the processing time by T_{l_i}. If the task is offloaded to the CAP, denote the energy consumed for transmitting and receiving by E_{t_i} and E_{r_i} respectively. Furthermore, denote the uplink and downlink transmission times by T_{t_i} and T_{r_i}, respectively. Their values depend on the corresponding wireless link capacities, denoted by C_{UL} and C_{DL}, respectively, and other possible tasks offloaded to the CAP. If the ith task is processed by the CAP, denote its processing time by T_{a_i}. It depends on the CPU rate f_A and the number of tasks being processed at the CAP. If instead the task is further offloaded to the cloud from the CAP, we denote the required transmission time between the CAP and the cloud by T_{ac_i}, and the cloud processing time by T_{c_i}. Finally, denote by C_{a_i} and C_{c_i} the costs for letting the CAP and the cloud process the task i, respectively.

The mobile user first decides whether its task i should be processed locally or offloaded to the CAP. Denote this decision by

$$x_i = \begin{cases} 0 & \text{task } i \text{ is processed locally,} \\ 1 & \text{task } i \text{ is offloaded to the CAP.} \end{cases}$$

When task i is offloaded to the CAP, we need to determine whether the task should be processed by the CAP or further offloaded to the cloud. We denote this decision by

$$y_i = \begin{cases} 0 & \text{task } i \text{ is processed at the CAP,} \\ 1 & \text{task } i \text{ is offloaded to the cloud.} \end{cases}$$

The total cost of the mobile user is defined as the weighted sum of total energy consumption, costs to offload and process all tasks, and the corresponding worst case transmission and processing delays. We aim to minimize the worst case total cost by optimizing the processing-offloading task decisions $\{x_i\}, \{y_i\}$. The optimization problem is formulated as follows:

$$\begin{aligned} \min_{\{x_i\},\{y_i\}} & \sum_{i=1}^{N} [E_{l_i}(1-x_i) + E_{A_i} x_i (1-y_i) + E_{C_i} x_i y_i] \\ & + \rho \max\{T_L, T_A, T_C\} \qquad (1) \\ \text{s.t.} \quad & x_i, y_i \in \{0,1\}, \quad i = 1, \ldots, N, \end{aligned}$$

where $E_{A_i} \triangleq (E_{t_i} + E_{r_i} + \alpha C_{a_i})$ and $E_{C_i} \triangleq (E_{t_i} + E_{r_i} + \beta C_{c_i})$ are the weighted transmission energy and processing cost for task i being offloaded to the CAP or the cloud, respectively, with α and β being the relative weights, $T_L \triangleq \sum_{i=1}^{N} T_{l_i}(1-x_i)$ is the processing delay at the mobile user, $T_A \triangleq \sum_{i=1}^{N}((T_{t_i}+T_{r_i})x_i + T_{a_i}x_i(1-y_i))$ and $T_C \triangleq \sum_{i=1}^{N}((T_{t_i}+T_{r_i})x_i + (T_{ac_i}+T_{c_i})x_i y_i)$ correspond to the worst case sum transmission and processing delay at the CAP and the cloud, respectively. In addition, ρ is the weight on the total delay, relative to energy consumption, for the user's tasks. We can adjust ρ to place different emphasis on delay and energy consumption.

III. Mobile and CAP Offloading Solution

The optimization problem (1) can be rewritten as

$$\begin{aligned} \min_{\{x_i\},\{y_i\},t} & \sum_{i=1}^{N} [E_{l_i}(1-x_i) + E_{A_i} x_i (1-y_i) \\ & + E_{C_i} x_i y_i] + \rho t \qquad (2) \\ \text{s.t.} & \sum_{i=1}^{N} T_{l_i}(1-x_i) \le t, \qquad (3) \\ & \sum_{i=1}^{N} ((T_{t_i}+T_{r_i})x_i + T_{a_i}x_i(1-y_i)) \le t, \qquad (4) \\ & \sum_{i=1}^{N} ((T_{t_i}+T_{r_i})x_i + (T_{ac_i}+T_{c_i})x_i y_i) \le t, \qquad (5) \\ & x_i, y_i \in \{0,1\}, \quad i = 1, \ldots, N, \qquad (6) \end{aligned}$$

where (3)-(5) are the constraints on the sum transmission and processing delay when tasks are processed at the mobile user, the CAP, or the cloud, respectively.

We now examine the transmission and processing delays T_A and T_C. Note that the uplink transmission time for tasks depends on the uplink transmission capacity C_{UL} and the amount of offloaded tasks. Since the worst case offloading delay is considered, we assume the CAP starts to offload or process offloaded tasks after receiving all of them. Therefore, we have

$$\sum_{i=1}^{N} T_{t_i} x_i = \frac{\sum_{i=1}^{N} D_{\text{in}}(i) x_i}{C_{\text{UL}}}.$$

Similarly, the downlink transmission starts after all tasks are finished and waiting at the CAP. Thus, the downlink transmission time for offloaded tasks $\sum_{i=1}^{N} T_{r_i} x_i$ is given by

$$\sum_{i=1}^{N} T_{r_i} x_i = \frac{\sum_{i=1}^{N} D_{\text{out}}(i) x_i}{C_{\text{DL}}}.$$

The CAP processing time of tasks depends the CPU processing rate f_A and the amount of tasks processed at the CAP. Thus, we have

$$\sum_{i=1}^{N} T_{a_i} x_i (1-y_i) = \frac{\sum_{i=1}^{N} D_{\text{in}}(i) \text{App}(i) x_i (1-y_i)}{f_A}.$$

When tasks are offloaded to the cloud, additional transmission time $\sum_{i=1}^{N} T_{ac_i} x_i y_i = \sum_{i=1}^{N} (D_{\text{in}}(i) + D_{\text{out}}(i)) x_i y_i / R_{ac}$ will

occur, where R_{ac} is the transmission rate. At the cloud, we assume that the user will only be served by one virtual machine with the CPU processing rate f_C. The corresponding cloud processing time is

$$\sum_{i=1}^{N} T_{c_i} x_i y_i = \frac{\sum_{i=1}^{N} D_{\text{in}}(i)\text{App}(i) x_i y_i}{f_C}.$$

It follows that the delay constraints (4) and (5) at the CAP and the cloud can be respectively rewritten as

$$\frac{\sum_{i=1}^{N} D_{\text{in}}(i) x_i}{C_{\text{UL}}} + \frac{\sum_{i=1}^{N} D_{\text{in}}(i)\text{App}(i) x_i (1-y_i)}{f_A} + \frac{\sum_{i=1}^{N} D_{\text{out}}(i) x_i}{C_{\text{DL}}} \le t, \tag{7}$$

and

$$\frac{\sum_{i=1}^{N} D_{\text{in}}(i) x_i}{C_{\text{UL}}} + \frac{\sum_{i=1}^{N} (D_{\text{in}}(i) + D_{\text{out}}(i)) x_i y_i}{R_{ac}} + \frac{\sum_{i=1}^{N} D_{\text{in}}(i)\text{App}(i) x_i y_i}{f_C} + \frac{\sum_{i=1}^{N} D_{\text{out}}(i) x_i}{C_{\text{DL}}} \le t, \tag{8}$$

The optimization problem (1) is an integer programming problem. In order to find an efficient solution to this problem, in the following, we first transform it into a QCQP, and then propose an SDR approach and a novel randomization mapping method to recover the binary offloading decisions. We name it the Local-Access-Cloud (LAC) offloading solution.

A. Semidefinite Relaxation

To convert the optimization problem (1) into a QCQP problem, we first replace the integer constraint (6) by

$$x_i(x_i - 1) = 0,\ y_i(y_i - 1) = 0,\ i = 1, \ldots, N. \tag{9}$$

Define $\mathbf{w} = [x_1 \ldots x_N, y_1 \ldots y_N, t]^T$, $\mathbf{E}_j = [E_{j_1} \ldots E_{j_N}]^T$, for $j = l, t, r$, $\mathbf{C}_k = [C_{k_1} \ldots C_{k_N}]^T$, for $k = a$ or c, $\mathbf{D}_s = [D_s(1) \ldots D_s(N)]^T$, for $s =$ in or out, and $\mathbf{App} = [\text{App}(1) \ldots \text{App}(N)]^T$. Define $\mathbf{e}_i$ and $\mathbf{e}'_i$ as the $N \times 1$ and $(2N+1) \times 1$ standard unit vectors with the ith entry being 1, respectively. In addition, define $\mathbf{0}$ and $\mathbf{0}'$ as the $N \times N$ zero matrix and $N \times 1$ zero vector, respectively. The optimization problem (1) can now be transformed into the following QCQP formulation:

$$\min_{\mathbf{w}}\ \mathbf{w}^T \mathbf{A}_0 \mathbf{w} + \mathbf{b}_0^T \mathbf{w} + \mathbf{E}_l^T \mathbf{1}_{N\times 1} \tag{10}$$

$$\text{s.t. } \mathbf{b}_l^T \mathbf{w} \le -\mathbf{T}_l^T \mathbf{1}_{N\times 1}, \tag{11}$$

$$\mathbf{w}^T \mathbf{A}_a \mathbf{w} + \mathbf{b}_a^T \mathbf{w} \le 0, \tag{12}$$

$$\mathbf{w}^T \mathbf{A}_c \mathbf{w} + \mathbf{b}_c^T \mathbf{w} \le 0, \tag{13}$$

$$\mathbf{w}^T \text{diag}(\mathbf{e}'_p) \mathbf{w} - \mathbf{e}'^T_p \mathbf{w} = 0, \quad p = 1, \ldots, 2N, \tag{14}$$

where

$$\mathbf{A}_0 = \begin{bmatrix} \mathbf{0} & \mathbf{A}'_0 & \mathbf{0}' \\ \mathbf{A}'^T_0 & \mathbf{0} & \mathbf{0}' \\ \mathbf{0}'^T & \mathbf{0}'^T & 0 \end{bmatrix},\ \mathbf{A}'_0 = \frac{1}{2}\text{diag}(-\alpha \mathbf{C}_a + \beta \mathbf{C}_c),$$

$$\mathbf{A}_a = \begin{bmatrix} \mathbf{0} & \mathbf{A}'_a & \mathbf{0}' \\ \mathbf{A}'^T_a & \mathbf{0} & \mathbf{0}' \\ \mathbf{0}'^T & \mathbf{0}'^T & 0 \end{bmatrix},\ \mathbf{A}'_a = \frac{-1}{2f_A}\text{diag}(\mathbf{D}_{\text{in}})\text{diag}(\mathbf{App}),$$

$$\mathbf{A}_c = \begin{bmatrix} \mathbf{0} & \mathbf{A}'_c & \mathbf{0}' \\ \mathbf{A}'^T_c & \mathbf{0} & \mathbf{0}' \\ \mathbf{0}'^T & \mathbf{0}'^T & 0 \end{bmatrix},$$

$$\mathbf{A}'_c = \frac{1}{2}\left[\frac{1}{R_{ac}}\text{diag}(\mathbf{D}_{\text{in}} + \mathbf{D}_{\text{out}}) + \frac{1}{f_C}\text{diag}(\mathbf{D}_{\text{in}})\text{diag}(\mathbf{App})\right],$$

$$\mathbf{b}_0 = [(-\mathbf{E}_l + \mathbf{E}_t + \mathbf{E}_r + \alpha \mathbf{C}_a)^T\ \mathbf{0}'^T\ \rho]^T,$$

$$\mathbf{b}_l = -[\mathbf{T}_l^T\ \mathbf{0}'^T\ 1]^T,$$

$$\mathbf{b}_a = [\mathbf{b}'^T_a\ \mathbf{0}'^T\ -1]^T,$$

$$\mathbf{b}'_a = \frac{1}{C_{\text{UL}}}\mathbf{D}_{\text{in}} + \frac{1}{C_{\text{DL}}}\mathbf{D}_{\text{out}} + \frac{1}{f_A}\text{diag}(\mathbf{D}_{\text{in}})\mathbf{App},$$

$$\mathbf{b}_c = \left[(\frac{1}{C_{\text{UL}}}\mathbf{D}_{\text{in}} + \frac{1}{C_{\text{DL}}}\mathbf{D}_{\text{out}})^T\ \mathbf{0}'^T\ -1\right]^T.$$

Comparing the optimization problems (10) and (2), constraints (11)-(13) correspond to the processing delay constraints (3)-(5), and constraint (14) represents the integer constraint (9).

We further define $\mathbf{z}^T = [\mathbf{w}^T 1]^T$. After dropping the constant term $\mathbf{E}_l^T \mathbf{1}_{N\times 1}$ in the objective of (10), we can homogenize the optimization problem (10) as

$$\begin{aligned} \min_{\mathbf{z}}\ & \mathbf{z}^T \mathbf{G}_0 \mathbf{z} \\ \text{s.t. } & \mathbf{z}^T \mathbf{G}_l \mathbf{z} \le -\mathbf{T}_l^T \mathbf{1}_{N\times 1}, \\ & \mathbf{z}^T \mathbf{G}_a \mathbf{z} \le 0, \\ & \mathbf{z}^T \mathbf{G}_c \mathbf{z} \le 0, \\ & \mathbf{z}^T \mathbf{G}_p \mathbf{z} = 0, \quad p = 1, \ldots, 2N, \end{aligned} \tag{15}$$

where

$$\mathbf{G}_0 = \begin{bmatrix} \mathbf{A}_0 & \frac{1}{2}\mathbf{b}_0 \\ \frac{1}{2}\mathbf{b}_0^T & 0 \end{bmatrix},$$

$$\mathbf{G}_l = \begin{bmatrix} \mathbf{0}_{(2N+1)\times(2N+1)} & \frac{1}{2}\mathbf{b}_l \\ \frac{1}{2}\mathbf{b}_l^T & 0 \end{bmatrix},$$

$$\mathbf{G}_a = \begin{bmatrix} \mathbf{A}_a & \frac{1}{2}\mathbf{b}_a \\ \frac{1}{2}\mathbf{b}_a^T & 0 \end{bmatrix},$$

$$\mathbf{G}_c = \begin{bmatrix} \mathbf{A}_c & \frac{1}{2}\mathbf{b}_c \\ \frac{1}{2}\mathbf{b}_c^T & 0 \end{bmatrix},$$

$$\mathbf{G}_p = \begin{bmatrix} \text{diag}(\mathbf{e}'_p) & -\frac{1}{2}\mathbf{e}'_p \\ -\frac{1}{2}\mathbf{e}'^T_p & 0 \end{bmatrix},\quad p = 1, \ldots, 2N.$$

Note that the optimization problem (15) is a non-convex QCQP problem, which is NP-hard in general. To solve it, we apply the SDR approach to relax it into a semidefinite programming (SDP) problem. Define $\mathbf{X} = \mathbf{z}\mathbf{z}^T$. By dropping the rank constraint $\text{rank}(\mathbf{X}) = 1$, we have the following SDP problem:

$$\begin{aligned} \min_{\mathbf{X}}\ & \text{Tr}(\mathbf{G}_0 \mathbf{X}) \\ \text{s.t. } & \text{Tr}(\mathbf{G}_l \mathbf{X}) \le -\mathbf{T}_l^T \mathbf{1}_{N\times 1}, \\ & \text{Tr}(\mathbf{G}_a \mathbf{X}) \le 0, \\ & \text{Tr}(\mathbf{G}_c \mathbf{X}) \le 0, \\ & \text{Tr}(\mathbf{G}_p \mathbf{X}) = 0, \quad p = 1, \ldots, 2N, \\ & \mathbf{X}(2N+2, 2N+2) = 1,\ \ \mathbf{X} \succeq \mathbf{0}. \end{aligned} \tag{16}$$

The above SDP problem can be solved efficiently in polynomial time using a standard SDP software, such as SeDuMi [12].

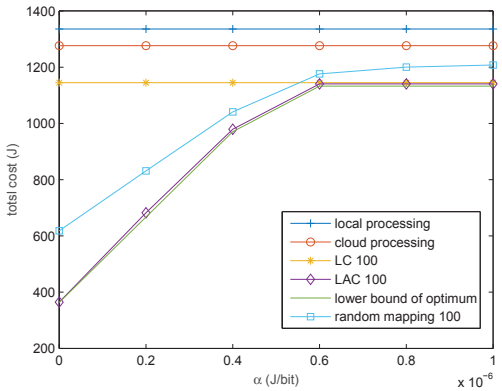

Fig. 2. The total system cost vs. α (J/bit).

Denote $\mathbf{X}^*$ as the optimal solution of the SDP problem (16). We need to recover a rank-1 solution of the original problem (1) from $\mathbf{X}^*$. In the following, we propose a randomization method to obtain our binary offloading decisions.

B. Binary Offloading Decisions via Randomization

A common approach [11] to obtain a feasible solution of the original problem (1) is to randomly generate vectors from the Gaussian distribution with zero mean and covariance $\mathbf{X}^*$ for L times, and then map them to the feasible set $\{0,1\}^{2N}$ by using the sign of each element in these vectors. Among the generated vectors, the one that yields the best objective value of the original problem (1) will be chosen as the desired solution. However, the above randomization procedure does not fully utilize the information provided in $\mathbf{X}^*$, because only the sign of each element in the randomly generated vectors is used to obtain the feasible solution. Instead, we propose the following improved method.

Define $\mathbf{v} = [x_1, \ldots, x_N, y_1, \ldots, y_N]^T$. Notice that, first, only the upper-left $2N \times 2N$ sub-matrix of $\mathbf{X}^*$, denote by $\mathbf{X}'^*$, is needed to recover the solution $\mathbf{v}$. Second, each diagonal term in $\mathbf{X}'^*$ is always between 0 and 1, corresponding to the probability that each element in $\mathbf{v}$ is 1. Denote $\mathbf{u} = [u_1, \ldots, u_{2N}]^T$, where $u_j = X'^*_{jj}$ is the jth diagonal term of $\mathbf{X}'^*$, and $\mathbf{\Sigma} = \mathbf{X}'^* - \mathbf{u}\mathbf{u}^T$. It can be shown that $\mathbf{\Sigma}$ is always positive semidefinite by using a property of the Schur complement. We omit the details due to page limitation. We generate L i.i.d. random vectors $\tilde{\mathbf{v}}^{(l)} \sim \mathcal{N}(\mathbf{u}, \mathbf{\Sigma})$, $l = 1, \ldots, L$, and map each element $\tilde{v}_j^{(l)}$ of $\tilde{\mathbf{v}}^{(l)}$ to 1 with probability u_j by using the inverse Q-function, to arrive at a feasible solution $\mathbf{v}^{(l)}$. We then choose the one among these feasible solutions that gives the minimum objective value of the optimization problem (1) as $\mathbf{v}^o$.

The details of the above-mentioned LAC offloading algorithm are described in Algorithm 1, in which $Q^{-1}(u_j)$ is the inverse Q-function with the parameter u_j. We observe from simulation results that a small number of randomization trials (e.g., $L = 100$) is enough to give system performance very close to the optimal one.

IV. Simulation Results

In this section, we provide computer simulations to show the near-optimal performance of our proposed LAC offloading solution under different parameter settings. In the following, we describe the default parameter values, unless otherwise indicated later. We adopt the mobile device characteristics from [13], which is based on Nokia N900, and set the number of tasks as $N = 10$. According to Table 1 and 3 in [13], the mobile user has CPU rate 500×10^6 cycles/s and processing energy consumption $\frac{1}{730 \times 10^6}$ J/cycle, and the local computation time 4.75×10^{-7} s/bit and local processing energy consumption 3.25×10^{-7} J/bit are calculated when the x264 CBR encode application (1900 cycles/byte) is considered as App(i) in our simulations. We set both uplink and downlink transmission capacities at 72.2 Mbps (e.g., IEEE 802.11n) between the mobile user and the CAP, 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 [13].

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 CPU rates of the CAP and the Cloud are 5×10^9 cycle/s and 10×10^9 cycle/s, respectively. When tasks are offloaded to the cloud, the transmission rate R_{ac} is 15 Mpbs. Also, we set the values of cost C_{a_i} and C_{c_i} to be the

Algorithm 1 LAC Offloading Algorithm

1: Obtain optimal solution $\mathbf{X}^*$ of the SDP problem (16). Extract the upper-left $2N \times 2N$ sub-matrix $\mathbf{X}'^*$ from $\mathbf{X}^*$. Set the number of randomization trials as L.
2: Compute $\mathbf{u} = [u_1, \ldots, u_{2N}]^T$, where $u_j = X'^*_{jj}$, and $\mathbf{\Sigma} = \mathbf{X}'^* - \mathbf{u}\mathbf{u}^T$.
3: **for** $l = 1, ..., L$ **do**
4: Generate $\tilde{\mathbf{v}}^{(l)} \sim \mathcal{N}(\mathbf{u}, \mathbf{\Sigma})$;
5: **for** $j = 1, ..., 2N$ **do**
6: $\gamma_j = Q^{-1}(u_j)\sqrt{\Sigma_{jj}} + u_j$;
7: $v_j^{(l)} = (\text{sgn}(\tilde{v}_j^{(l)} - \gamma_j) + 1)/2$;
8: **end for**
9: Form the lth feasible solution $\mathbf{v}^{(l)} = [v_1^{(l)}, \ldots, v_{2N}^{(l)}]^T$ of the original problem (1).
10: **end for**
11: Choose among $\mathbf{v}^{(1)}, \ldots, \mathbf{v}^{(L)}$ the one that yields the minimum objective value of the original problem (1); Set it as $\mathbf{v}^o$.
12: Compare $\mathbf{v}^o$ with the solutions from local processing only and cloud processing only. Set the best one among them as the solution $\mathbf{v}^s$.
13: Output: the proposed offloading solution $\mathbf{v}^s = [x_1^s, \ldots, x_N^s, y_1^s, \ldots, y_N^s]^T$.

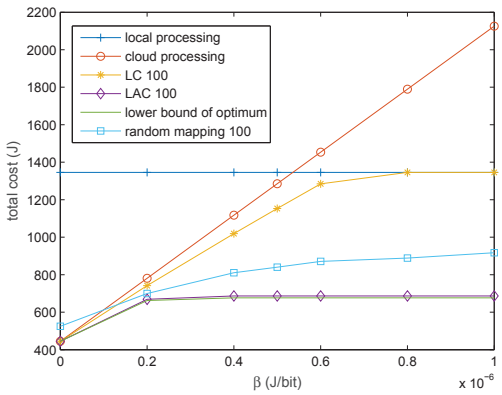

Fig. 3. The total system cost vs. β (J/bit).

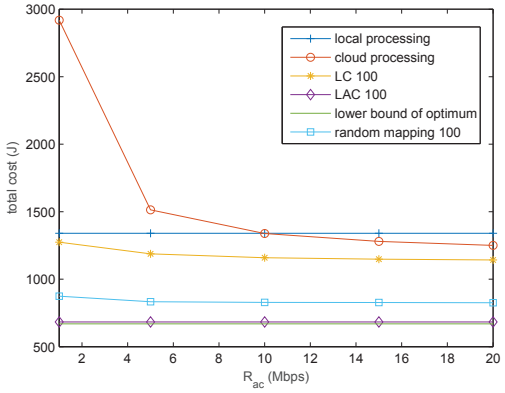

Fig. 4. The total system cost versus transmission rate R_{ac} (Mbps).

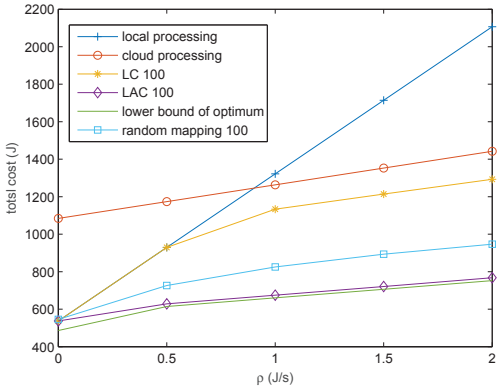

Fig. 5. The total cost under different policies versus ρ (J/s).

same as that of the input data size $D_{\text{in}}(i)$, and $\alpha = 2 \times 10^{-7}$ J/bit and $\beta = 5 \times 10^{-7}$ J/bit. The weight of the maximum delay to process all tasks ρ is 1 J/s.

For comparison, we also consider the following methods: 1) the *local processing only* method where all tasks are processed by the mobile user, 2) the *cloud processing only* method where all tasks are offloaded to the cloud, 3) the *Local-Cloud (LC) offloading* method where the same approximation procedure as the LAC method is applied except that there is no CAP, and 4) the *random mapping* method where all x_i and y_i are chosen i.i.d. with equal probability. In addition, we also plot a lower bound of the minimum cost. It is obtained from the optimal objective value of the SDR problem (16). In the following figures, we use "LAC 100," "LC 100," and "random mapping 100" to indicate that $L = 100$ for the randomization trials in these methods, respectively. Our simulation shows that $L = 100$ is sufficient for LAC to provide near-optimal performance, despite the 3^N decision space of the optimization problem. Finally, all simulation results are obtained by averaging over 100 realizations of the input and output data sizes of each task.

In Figs. 2 and 3, we show the system cost vs. the weights α and β on the CAP processing cost and cloud processing cost, respectively. Since α is the weight on the CAP processing cost, the costs of the LAC method and optimal solution increase as α increases. When α becomes large, no tasks will be processed at the CAP, and the performance converges to that of the LC method. When β becomes large, all tasks are more likely to be processed by either the mobile user or the CAP. The LC method in this case converges to the local processing method.

When a task is offloaded to the cloud, additional transmission time T_{ac_i} will occur, which is inversely proportional to the transmission rate R_{ac} between the CAP and the cloud. In Fig. 4, we plots the total system cost vs. R_{ac}. As expected, R_{ac} mainly affects the cloud processing method and the LC method. In Fig. 5, we study the system cost under various values of the weight ρ on the processing delay. We observe that with the help of the CAP, our LAC method outperforms all other methods. In addition, in all figures, we observe that it provides near-optimal performance when compared with the lower bound of the minimum cost.

V. Conclusion

A mobile cloud computing system consisting of one user with multiple tasks, one CAP, and one remote cloud server has been considered. We aim to minimize a weighted total cost of energy, computation, and delay through optimal tasks offloading by the mobile user. The optimization problem resulted in a non-convex QCQP. We have thus proposed an efficient heuristic algorithm to solve this problem using SDR and a new randomization mapping method. Simulation results suggest that the proposed algorithm gives nearly optimal performance, thus enabling the often substantial benefit of using a CAP between the mobile users and the remote cloud server.

References

[1] K. Kumar and Y.-H. Lu, "Cloud computing for mobile users: Can offloading computation save energy?" *Computer*, vol. 43, pp. 51–56, 2010.

[2] M. Satyanarayanan, P. Bahl, R. Caceres, and N. Davies, "The case for vm-based cloudlets in mobile computing," *IEEE Pervasive Computing*, vol. 8, pp. 14–23, 2009.

[3] S. Barbarossa, S. Sardellitti, and P. Di Lorenzo, "Computation offloading for mobile cloud computing based on wide cross-layer optimization," in *Proc. Future Network and Mobile Summit (FutureNetworkSummit)*, 2013, pp. 1–10.

[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, pp. 4569–4581, 2013.

[5] S. Ren and M. van der Schaar, "Efficient resource provisioning and rate selection for stream mining in a community cloud," *IEEE Transactions on Multimedia*, vol. 15, pp. 723–734, 2013.

[6] O. Munoz, A. Pascual Iserte, J. Vidal, and M. Molina, "Energy-latency trade-off for multiuser wireless computation offloading," in *Proc. IEEE Wireless Communications and Networking Conference (WCNC) Workshops*, 2014, pp. 29–33.

[7] 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)*, 2010, pp. 49–62.

[8] 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)*, 2011, pp. 301–314.

[9] 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)*, 2012, pp. 945–953.

[10] J. Niu, W. Song, L. Shu, and M. Atiquzzaman, "Bandwidth-adaptive application partitioning for execution time and energy optimization," in *Proc. IEEE International Conference on Communications (ICC)*, 2013, pp. 3660–3665.

[11] 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, pp. 20–34, 2010.

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

[13] A. P. Miettinen and J. K. Nurminen, "Energy efficiency of mobile clients in cloud computing," in *Proc. the 2nd USENIX Conference on Hot Topics in Cloud Computing (HotCloud)*, 2010.