Multi-user Mobile Cloud Offloading Game with Computing Access Point

Meng-Hsi Chen and Ben Liang
Dept. of Electrical and Computer Engineering
University of Toronto
Toronto, Canada
Email: {mchen, liang}@ece.utoronto.ca

Min Dong
Dept. of Electrical, Computer and Software Engineering
University of Ontario Institute of Technology
Oshawa, Canada
Email: min.dong@uoit.ca

***Abstract*—We consider a multi-user mobile cloud computing system with a computing access point (CAP) where each mobile user has multiple dependent tasks to be processed using a round-by-round schedule. The CAP can either process the received tasks from mobile users or offload them to the cloud. In each round, we aim to jointly optimize the offloading decisions of all users and the CAP, together with communication and processing resource allocation, to minimize the overall cost of energy, computation, and the maximum delay among all users. Since the centralized optimization problem is non-convex and mobile users may not follow the obtained solution, we further formulate a mobile cloud offloading game. We show the existence of a Nash equilibrium (NE) of this game and propose an algorithm to attain the NE. Simulation results show that our proposed algorithm gives nearly optimal performance in terms of the total system cost under various parameter settings.**

***Keywords*- mobile cloud; computing access point; game theory; offloading decision; resource allocation**

I. Introduction

By offloading tasks to the cloud for data gathering, storage, and processing, mobile users can potentially reduce its own device energy consumption or processing delay of each task [1] [2]. However, the integration between mobile devices and the cloud introduces additional communication delay and transceiver energy consumption, which may affect the quality of service (QoS) of those offloaded tasks and overall mobile device energy usage [3].

Prior works on mobild cloud computing include scenarios where a single user offloads a single application [3] [4], multiple users each offload a single application [5]–[8], a single user offloads multiple tasks [9]–[12], and multiple users each offload multiple tasks [13]. Furthermore, instead of conventional mobile cloud computing where only mobile devices and the cloud server can process tasks, a novel Computing Access Point (CAP) was proposed in our previous work [14], which can be a wireless access point or a cellular base station with built-in computation capability. Mobile cloud computing through CAPs is similar to the concept of Mobile Edge Computing [15], micro cloud centers [16], cloudlets [17], and fog computing [18], but the user computing tasks may be processed locally at the mobile devices, at the CAP, or further forwarded by the CAP to a remote cloud server. By solving the centralized offloading optimization problem, we previously showed substantial system performance improvement by considering the CAP for both single-user [14] and multi-user [19] scenarios.

In this work, we study the interaction between *selfish* mobile users and the CAP. Each mobile user has multiple sequentially ordered tasks with a round-by-round schedule where the head-of-queue task from each user is processed in each round. We consider jointly the offloading decision and the allocation of communication and computation resources among all users, with an aim to conserve energy and maintain QoS of all users. Different from [19], in which a heuristic method was proposed to solve the centralized non-convex mixed integer programming problem sub-optimally, we use a game theoretic approach by letting mobile users distributively compute their own offloading decisions based on their respective cost function, while the CAP decides the allocation of communication and computation resources. We show that a Nash equilibrium (NE) exists in our formulated game, and we propose an algorithm which leads to an NE in finite steps. We further show that the users will truthfully report their task information to the CAP, so that the CAP can compute the NE and no user will have incentive to deviate from it. Simulation shows that our proposed game-based solution can achieve near-optimal performance in terms of the overall system cost under various parameter settings.

The rest of this paper is organized as follows. In Section II, we describe the system model and present the centralized problem formulation. In Section III, we provide details of the game formulation, the existence of the NE, and the proposed algorithm to find a NE. Simulation is provided in Section IV, and conclusion is given in Section V.

II. System Model and Centralized Optimization

Consider a cloud access network consisting of one remote cloud server, one CAP, and N mobile users, as shown in Fig. 1. Each mobile user has M sequentially ordered tasks, and we consider a round-by-round schedule where one task from each user is processed (for a total of N tasks) in each round. After finishing the current round, a new round will start until all tasks are processed. Note that this system model can be easily extended to the case where each mobile user

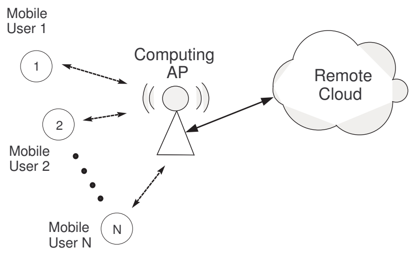

Figure 1. System model

has a different number of tasks, by considering only those users with tasks to process in the current round. In such a round-by-round system, it suffices to optimize the offloading decisions and resources allocation for mobile users in a single round. This will be the focus of the rest of this paper.

A. Per-Round Offloading Decision

In each round, every mobile user has one task to be either processed locally or offloaded. Furthermore, an offloaded task may be processed at the CAP or be further forwarded to the remote cloud. Denote the offloading decisions by

$$x_{l_i} + x_{a_i} + x_{c_i} = 1, \quad i = 1, \dots, N, \tag{1}$$

where $x_{l_i}, x_{a_i}, x_{c_i} \in \{0, 1\}$ indicate whether user i's task is processed locally, at the CAP, or at the cloud, respectively. Notice that only one of x_{l_i}, x_{a_i}, and x_{c_i} for user i could be 1.

B. Cost of Local Processing

The input data size, output data size, and processing cycles of user i's task are denoted by $D_{\text{in}}(i)$, $D_{\text{out}}(i)$, and $Y(i)$, respectively. When this task is processed locally, the processing energy is denoted by E_{l_i} and the processing time is denoted by T_{l_i}.

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 further allocate the communication and computation resources available at the CAP. For user i's task being offloaded to the CAP, we denote by E_{t_i} and E_{r_i}, respectively, the energy consumed for wireless transmission and reception by the user. We further denote the uplink and downlink transmission times by $T_{t_i} = D_{\text{in}}(i)/r_{u_i}$ and $T_{r_i} = D_{\text{out}}(i)/r_{d_i}$, respectively, where r_{u_i} and r_{d_i} are uplink and downlink data rates allocated to user i. Furthermore, r_{u_i} and r_{d_i} and limited by the uplink and downlink capacities C_{UL} and C_{DL} as follows:

$$\sum_{i=1}^{N} r_{u_i} \leq C_{UL}, \tag{2}$$

and

$$\sum_{i=1}^{N} r_{d_i} \leq C_{DL}. \tag{3}$$

Table I
NOTATION AND CORRESPONDING DESCRIPTION.

Notation	Description
E_{l_i}	local processing energy of user i's task
E_{t_i}, E_{r_i}	uplink transmitting energy and downlink receiving energy of user i's task to and from CAP, respectively
T_{l_i}, T_{c_i}	local processing time and cloud processing time of user i's task
T_{t_i}, T_{r_i}	uplink transmission time and downlink transmission time of user i's task between mobile user and CAP
T_{ac_i}	transmission time of user i's task between CAP and cloud
C_{UL}, C_{DL}	uplink transmission capacity and downlink transmission capacity between mobile users and CAP
r_{u_i}, r_{d_i}	uplink transmission rate and downlink transmission rate assigned to user i
C_{c_i}	system utility cost of user i's task
r_{ac}	transmission rate for each user between CAP and cloud
f_C	cloud processing rate for each user
f_{a_i}	CAP processing rate assigned to user i

If this task is processed by the CAP, denote its processing time by $T_{a_i} = Y(i)/f_{a_i}$, where f_{a_i} is the assigned processing rate, which is limited by the total processing rate f_A:

$$\sum_{i=1}^{N} f_{a_i} \leq f_A. \tag{4}$$

D. Cost of Cloud Processing

If the task is further offloaded to the cloud from the CAP, besides all the costs mentioned above except for T_{a_i}, there is additional transmission time between the CAP and the cloud denoted by $T_{ac_i} = (D_{\text{in}}(i) + D_{\text{out}}(i))/r_{ac}$, and cloud processing time denoted by $T_{c_i} = Y(i)/f_C$, where we assume the wired transmission rate r_{ac} between the AP and the cloud and the cloud processing rate f_C *for each user* are pre-determined values. Thus, T_{ac_i} and T_{c_i} only depend on task i itself. Finally, the cloud utility cost of processing user i's task at the cloud is denoted by C_{c_i}. The above notations are summarized in Table I.

E. Centralized Optimization Problem Formulation

Our goal is to reduce the mobile users' energy consumption and maintain the QoS to their tasks. Therefore, 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 maximum transmission and processing delays among all users. We aim to minimize the total system cost by jointly optimizing the task offloading decisions $\mathbf{x}_i = (x_{l_i}, x_{a_i}, x_{c_i})$ as well as the communication and CAP processing resource allocation $\mathbf{r}_i = (r_{u_i},\ r_{d_i},\ f_{a_i})$. The optimization problem is formulated as follows:

$$\min_{\{\mathbf{x}_i\},\{\mathbf{r}_i\}} \quad \left[\sum_{i=1}^{N} \alpha_i (E_{l_i} x_{l_i} + E_{A_i} x_{a_i} + E_{C_i} x_{c_i}) + \max_i \{T_{L_i} + T_{A_i} + T_{C_i}\} \right] \tag{5}$$

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

$$r_{u_i}, r_{d_i}, f_{a_i} \geq 0, \quad i = 1, \dots, N, \tag{6}$$

$$x_{l_i}, x_{a_i}, x_{c_i} \in \{0,1\}, \quad i = 1,\dots,N, \tag{7}$$

where $E_{A_i} \triangleq (E_{t_i} + E_{r_i})$, $E_{C_i} \triangleq (E_{t_i} + E_{r_i} + \beta C_{c_i})$ is the weighted transmission energy and processing cost for task i being offloaded to the cloud, with β being the relative weight, $T_{L_i} \triangleq T_{l_i} x_{l_i}$ is the processing delay at the mobile user, $T_{A_i} \triangleq (D_{\text{in}}(i)/r_{u_i} + D_{\text{out}}(i)/r_{d_i} + Y(i)/f_{a_i})x_{a_i}$ and $T_{C_i} \triangleq (D_{\text{in}}(i)/r_{u_i} + D_{\text{out}}(i)/r_{d_i} + T_{ac_i} + T_{c_i})x_{c_i}$ correspond to the transmission and processing delay at the CAP and the cloud, respectively, and α_i is the weight on energy consumption relative to the delay. We can adjust α_i to place different emphasis on energy consumption and delay.

The optimization problem (5) is a non-convex mixed-integer programming problem. Furthermore, even when an optimal solution to (5) can be obtained, there is no reason to expect that selfish users will necessarily follow the prescribed solution. Next, through a game theoretic approach, we consider a method to allow the mobile users distributedly compute their own offloading decisions, while the CAP decides the allocation of communication and computation resources. In simulation we will show that the proposed game theoretic solution can achieve near-optimal performance in terms of the objective of (5).

III. Multi-user Mobile Cloud Offloading Game

A. Game Formulation

In this section, we model the interaction between mobile users and the CAP as a mobile cloud offloading game, aiming to find an efficient solution to problem (5).

Let us consider a strategic form game

$$G_{\text{MCO}} = (\mathcal{I}, (\mathcal{A}_i)_{i\in\mathcal{I}}, (u_i)_{i\in\mathcal{I}}), \tag{8}$$

where $\mathcal{I} = \{1, ..., N\}$ is the player set containing all mobile users, $\mathcal{A}_i$ is the strategy set containing all possible strategies for user i, and u_i is the corresponding cost function that user i aims to minimize. Here, u_i is a function of the *strategy profile* $\mathbf{a} = (a_i, a_{-i})$, where $a_i \in \mathcal{A}_i$, $a_{-i} = (a_1, ..., a_{i-1}, ..., a_{i+1}, ..., a_N) \in \mathcal{A}_{-i} = \prod_{j\neq i} \mathcal{A}_j, \forall i$. More specifically, we define the strategy set and the cost function for user i as

$$A_i = \left\{ a_i = (x_{li}, x_{ai}, x_{ci}) \middle| x_{l_i} + x_{a_i} + x_{c_i} = 1; \right. \\ \left. x_{l_i}, x_{a_i}, x_{c_i} \in \{0,1\} \right\}, \tag{9}$$

and

$$u_i(\mathbf{a}) = \alpha_i(E_{l_i}x_{l_i} + E_{A_i}x_{a_i} + E_{C_i}x_{c_i}) \\ + \max_j\{T_{L_j} + T_{A_j} + T_{C_j}\}, \tag{10}$$

respectively. By choosing a_i, user i can decide where to process its task to minimize its cost function u_i, which is defined as the weighted sum of user i's energy consumption and the delay of this round. Without loss of generality, we assume that all mobile users are willing to participate in the game. That is, for each user, the expected cost to participate in the game is smaller than the cost of processing its task locally without joining the game. Since there are multiple tasks for each mobile user, the mobile user aims to reduce the delay of each round so that the next round can start earlier. Therefore, the cost function $u_i(\mathbf{a})$ in (10) expresses the user's objective to balance its energy usage and the overall delay that it experiences.

After receiving some offloading decisions $\mathbf{a}$ from all mobile users, the CAP will assign communication and computation resources to each user to minimize the overall system cost by solving the resource allocation problem as follows:

$$\min_{\{\mathbf{r}_i\}} \quad \left(\mathbf{E} + \max_i\{T_{L_i} + T_{A_i} + T_{C_i}\}\right) \tag{11}$$
$$\text{s.t.} \quad (2), (3), (4), (6),$$

where $\mathbf{E} \triangleq \sum_{i=1}^N \alpha_i(E_{l_i}x_{l_i} + E_{A_i}x_{a_i} + E_{C_i}x_{c_i})$ is a constant depending on $\mathbf{a}$. This resource allocation problem (11) is convex, which can be solved optimally using standard convex optimization solvers, such as SeDuMi [20].

Notice that problem (11) requires task information from all users. In the following, we show that there is no incentive for any user to provide false task information. First, we note that if a user is found by the CAP to provide false information, it will be prohibited from participating in the system, so no user will both provide false information and offload its task to the CAP in the same round, when its deceit will be noticed by the CAP. Next, we claim that a mobile user cannot further decrease its own cost by providing false information to the CAP without being noticed. The detailed proof is omitted due to page limitation. The intuition is the following: if user i participates in the game but provides false information to the CAP, it needs to guarantee the corresponding Nash equilibrium defined in Definition 1 below contains $a_i^* = (1, 0, 0)$ (i.e, local processing). Otherwise, the CAP will find that user i does not truthfully report its information. However, by providing false information and processing its tasks locally, user i will lengthen the delay of this round and incur a higher cost compared with the cost of directly processing its task locally without participating in the game. Therefore, all mobile users who are willing to participate in the game will always truthfully provide their task information to the CAP.

B. Game Structure Properties

Definition 1 ([21]). *The strategy profile $\mathbf{a}^*$ is a Nash equilibrium if $u_i(a_i^*, a_{-i}^*) \leq u_i(a_i, a_{-i}^*)$, for any $a_i \in \mathcal{A}_i$, $\forall i$.*

Definition 1 implies that, by employing strategies corresponding to the Nash equilibrium (NE), no player can decrease its cost by unilaterally changing its own strategy. However, the NE may not always exists for a strategic form game, especially when the game is not carefully formulated. In Proposition 1, we will prove that the proposed game G_{MCO} has at least one NE. Before that, we need the following definitions and lemma.

Definition 2 ([22]). *A strategic form game G is an ordinal potential game (OPG) if there exists an ordinal potential function $\phi : \prod_i \mathcal{A}_i \rightarrow R$ such that*

$$\begin{aligned}\operatorname{sgn}(u_i(a_i, a_{-i}) - u_i(a_i', a_{-i})) \\ = \operatorname{sgn}(\phi(a_i, a_{-i}) - \phi(a_i', a_{-i})), \forall i, \quad (12)\end{aligned}$$

where $a_i, a_i' \in \mathcal{A}_i, a_{-i} \in \mathcal{A}_{-i}$.

Definition 3 (Finite Improvement Property [22]). *A path in G is a sequence $(\mathbf{a}[0], \mathbf{a}[1], ...)$ where for every $k \geq 1$ there exists a unique player i such that $a_i[k] \neq a_i[k-1] \in \mathcal{A}_i$ while $a_{-i}[k] = a_{-i}[k-1]$. $(\mathbf{a}[0], \mathbf{a}[1], ...)$ is an improvement path if, for all $k \geq 1$, $u_i(\mathbf{a}[k]) < u_i(\mathbf{a}[k-1])$, where player i is the unique deviator at step k. G has the finite improvement property (FIP) if every improvement path in G is finite.*

Lemma 1 ([22]). *Every OPG with finite strategy sets possesses at least one pure-strategy NE and has the FIP.*

To show our mobile cloud offloading game G_{MCO} always has a NE, we will prove that G_{MCO} is indeed a potential game as stated in Proposition 1.

Proposition 1. *The proposed mobile cloud offloading game G_{MCO} is an OPG with the potential function* (13)*, and, therefore, it always has an NE and the FIP.*

Proof: We first construct a function

$$\begin{aligned}\phi(\mathbf{a}) = \Bigg[\sum_{i=1}^{N} \alpha_i (E_{l_i} x_{l_i} + E_{A_i} x_{a_i} + E_{C_i} x_{c_i}) \\ + \max_i \{T_{L_i} + T_{A_i} + T_{C_i}\} \Bigg]. \quad (13)\end{aligned}$$

Define $\mathbf{E}_i \triangleq \alpha_i(E_{l_i} x_{l_i} + E_{A_i} x_{a_i} + E_{C_i} x_{c_i})$ and $\mathbf{T}_i \triangleq T_{L_i} + T_{A_i} + T_{C_i}$. Given two different strategy profiles $\mathbf{a} = (a_i, a_{-i})$ and $\mathbf{a}' = (a_i', a_{-i})$, where only user i chooses different strategies a_i and a_i', respectively, the difference between $\phi(\mathbf{a})$ and $\phi(\mathbf{a}')$ is

$$\begin{aligned}\phi(\mathbf{a}) - \phi(\mathbf{a}') = \; & \mathbf{E}_i + \sum_{j \neq i} \mathbf{E}_j + \max\{\mathbf{T}_i, \max_{j \neq i}\{\mathbf{T}_j\}\} \\ & - \mathbf{E}_i' - \sum_{j \neq i} \mathbf{E}_j - \max\{\mathbf{T}_i', \max_{j \neq i}\{\mathbf{T}_j\}\} \\ = \; & \mathbf{E}_i + \max\{\mathbf{T}_i, \max_{j \neq i}\{\mathbf{T}_j\}\} \\ & - \mathbf{E}_i' - \max\{\mathbf{T}_i', \max_{j \neq i}\{\mathbf{T}_j\}\} \\ = \; & u_i(\mathbf{a}) - u_i(\mathbf{a}').\end{aligned}$$

Since $\phi(\mathbf{a})$ in (13) satisfies the condition of the potential function of an OPG defined in (12), it is indeed a potential function of G_{MCO}. Therefore, G_{MCO} is an OPG which always has an NE and the FIP. ■

Algorithm 1 Mobile Cloud Offloading Algorithm

```
1:  Take any initial strategy profile $\mathbf{a}[0]$ and obtain the corresponding optimal resource allocation $\{\mathbf{r}_i^*[0]\}$ by solving (11).
2:  Set NE = False and $k = 0$.
3:  while NE == False do flag = 0 and $i = 1$;
4:      while $flag == 0$ and $i \leq N$ do
5:          Calculate $\{\mathbf{r}_i^{'*}\}$ for $(a_i', a_{-i}[k])$, for all $a_i' \in \mathcal{A}_i$;
6:          if $u_i(\mathbf{a}[k]) > u_i(a_i', a_{-i}[k]), a_i' \in \mathcal{A}_i$ then
7:              Set $a_i[k+1] = a_i'$, $a_{-i}[k+1] = a_{-i}[k]$;
8:              Set $\mathbf{a}[k+1] = (a_i[k+1], a_{-i}[k+1])$, flag = 1;
9:              $k = k + 1$;
10:         else if $i == N$ then
11:             Set flag = 1, NE = True;
12:         else
13:             $i = i + 1$;
14:         end if
15:     end while
16: end while
17: Output: the NE $\mathbf{a}^*$ of $G_{\mathrm{MCO}}$ and the corresponding resource allocation $\{\mathbf{r}_i^*\}$.
```

C. Mobile Cloud Offloading Algorithm

In this section we propose a mobile cloud offloading algorithm based on FIP to find an NE of G_{MCO}. Since the CAP has all information from the mobile users and need to assign the communication and computation resources to all offloaded tasks based on the strategy profile, it can compute the NE $\mathbf{a}^*$ of G_{MCO} and send $\mathbf{a}^*$ to all users. Due to the property of the NE, the users will follow $\mathbf{a}^*$ and have no incentive to deviate from it.

The CAP first initiates a starting strategy profile $\mathbf{a}[0]$ containing a set of hypothetical offloading decisions for all users. Based on $\mathbf{a}[0]$, it obtains the optimal communication and computation resources allocation for all tasks by solving (11). Then, the CAP takes an arbitrarily ordered list of all users and one-by-one examines each user's strategy set. Once it finds a user i so that it can decrease the value of the potential function (13) by changing this particular user i's strategy from $a_i[0]$ to another $a_i' \in \mathcal{A}_i$ while $a_{-i}[0]$ remains the same, it updates the strategy profile from $\mathbf{a}[0]$ to $\mathbf{a}[1]$ where $a_i[1] = a_i'$ and $a_{-i}[1] = a_{-i}[0]$. The CAP repeats the same procedure to find an improvement path $(\mathbf{a}[0], \mathbf{a}[1], \mathbf{a}[2], ...)$ of G_{MCO}.

The details of the mobile cloud offloading algorithm are given in Algorithm 1. Since G_{MCO} is an OPG and every improvement path is finite due to the FIP, the CAP will finally reach an NE $\mathbf{a}^*$ where no mobile user can further decrease its cost by unilaterally changing its strategy.

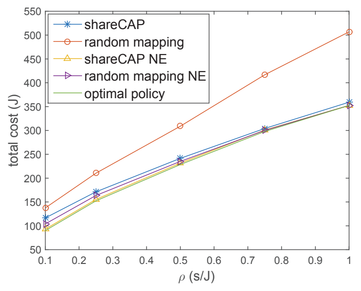

Figure 2. The total cost vs. α ($\alpha_i = \alpha$).

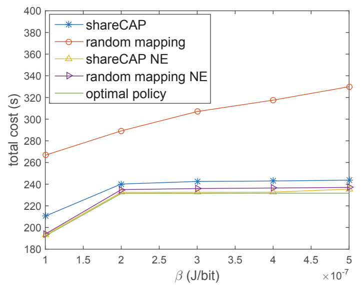

Figure 3. The total cost vs. β.

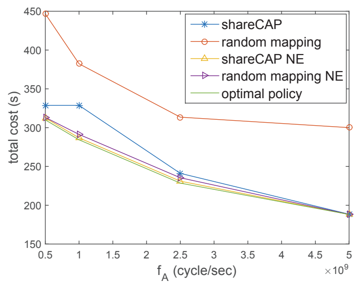

Figure 4. The total cost vs. f_A.

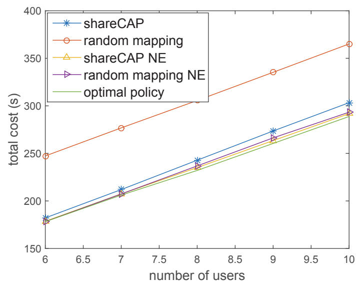

Figure 5. The total cost vs. number of users N.

Note that the general potential game approach has also been used in [7], [8] to find offloading decisions for users without a CAP. In this work, we study the impact of the CAP and additionally aim to optimize both communication and computation resource allocation for each user, leading to substantially more complex problem formulation and solution.

IV. Simulation Results

In this section, we provide computer simulation result to study the performance of our proposed algorithm under different parameter settings. Unless otherwise indicated, we use the following default parameter values. We adopt the mobile device characteristics from [23], which is based on Nokia N900, and set the number of users as $N = 8$. According to Tables 1 and 3 in [23], the mobile device has CPU rate 500×10^6 cycles/s and unit processing energy consumption $\frac{1}{730 \times 10^6}$ J/cycle. We consider the x264 CBR encoding application, which requires 1900 cycles/byte [23], leading to local computation time 4.75×10^{-7} s/bit and local processing energy consumption 3.25×10^{-7} J/bit. The input and output data sizes of each task are assumed to be uniformly distributed from 10MB to 30MB and from 1MB to 3MB, respectively. We assume $C_{\text{UL}} = C_{\text{DL}} = 72.2$Mbps according to IEEE 802.11n. The transmission and receiving energy per bit at each mobile device are both 1.42×10^{-7} J/bit as indicated in Table 2 in [23]. The CPU rates of the CAP and each sever at the remote cloud are 2.5×10^9 cycle/s and 5×10^9 cycle/s, respectively. For offloading a task to the cloud, the transmission rate is $R_{ac} = 15$Mpbs. Also, we set the cloud utility cost C_{c_i} to be the same as that of the input data size $D_{\text{in}}(i)$, and $\beta = 3 \times 10^{-7}$ J/bit. We further set $\alpha_i = \alpha = 0.5$ s/J. We assume each user has $M = 100$ tasks, with the input and output data size of each task being independently and identically generated. We plot the averaged total system cost over 100 random realizations for each data point.

Since our goal is to minimize the overall system cost in each round, we use the total system cost in (5) as the performance metric for our proposed algorithm and other alternative methods. As shown in Algorithm 1, we need a starting strategy profile $\mathbf{a}[0]$ and corresponding resource allocation $\{\mathbf{r}_i^*[0]\}$. In all figures shown, we provide the performance of two NEs obtained by two different starting profiles: 1) the *random mapping NE*, where in $\mathbf{a}[0]$ each task is processed at different locations with equal probability; and 2) the *shareCAP NE*, where $\mathbf{a}[0]$ is obtained by the shareCAP method using both semidefinite relaxation (SDR) and randomization to solve the non-convex mixed integer programming sub-optimally [19].

For comparison, we also consider the following methods: 1) the *random mapping* method, where the offloading decisions are exactly the same as $\mathbf{a}[0]$ for the *random mapping NE*, 2) the *shareCAP* method, where the offloading decisions are exactly the same as $\mathbf{a}[0]$ for the *shareCAP NE* , and 3) the *optimal policy*, where the optimal value is obtained by an exhaustive search.

In Fig. 2, we study the system cost versus weight $\alpha_i = \alpha$ on the energy consumption. We observe that *shareCAP* is much better than *random mapping*, but both *random mapping NE* and *shareCAP NE* obtained by our proposed algorithm give near-optimal performance. This indicates that even if we use a random starting strategy profile for our algorithm, the resulting NE is still near-optimal in terms of the total system cost. Next, we show the system cost vs. weights β on the cloud processing cost in Fig. 3. As expected, we see that when β becomes large, all tasks are more likely to be processed by either the mobile user or the CAP. In

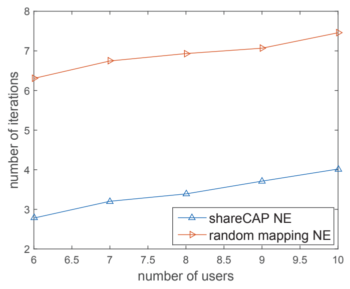

Figure 6. Number of iterations vs. number of users N.

Fig. 4, we plots the total system cost vs. f_A. Although the CAP can provide additional computation capacity, all tasks processed at the CAP need to share the CAP CPU rate f_A by optimally allocating the processing rate f_{ai} to each user's task. As expected, the more powerful CAP can dramatically increase system performance. In Fig. 5, we plot the total system cost vs. the number of users N. We see that both NE solutions are close to the optimal policy, indicating that our proposed algorithm is nearly optimal for various N values.

In Figs. 2-5, our proposed algorithm provides the nearly the same performance as the optimal policy under different parameter settings, despite the decision space of the centralized optimal problem (5) being very large as 3^N. This demonstrates that our proposed game can solve the original centralized non-convex problem near-optimally.

Finally, Fig. 6 shows the number of iterations required in our algorithm to find an NE versus N. We see that only a small number of iterations (i.e., the length of the improvement path in G_{MCO}) is needed as N increases. Although *random mapping NE* needs slightly more iterations than *shareCAP NE*, the former avoids the the computational complexity incurred by SDR to find the starting strategy profile.

V. Conclusion

A multi-user mobile cloud computing system with a CAP has been considered, in which each mobile user has multiple dependent tasks to be processed round-by-round. Since the centralized optimization problem is non-convex and mobile users may not follow the obtained solution, a mobile cloud offloading game is proposed instead. We show the existence of the NE of the proposed game and propose a algorithm for the CAP to find it. Through simulation, we show that the proposed algorithm gives nearly optimal performance.

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] N. Fernando, S. W. Loke, and W. Rahayu, "Mobile cloud computing: A survey," *Future Generation Computer Systems*, vol. 29, no. 1, pp. 84 – 106, Jan. 2013.

[3] 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.

[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] S. Sardellitti, G. Scutari, and S. Barbarossa, "Joint optimization of radio and computational resources for multicell mobile cloud computing," in *Proc. IEEE Workshop on Signal Processing Advances in Wireless Communications (SPAWC)*, Jun. 2014, pp. 354–358.

[6] E. Meskar, T. Todd, D. Zhao, and G. Karakostas, "Energy efficient offloading for competing users on a shared communication channel," in *Proc. IEEE International Conference on Communications (ICC)*, Jun. 2015, pp. 3192–3197.

[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] X. Chen, L. Jiao, W. Li, and X. Fu, "Efficient multi-user computation offloading for mobile-edge cloud computing," *to appear in the IEEE/ACM Transactions on Networking*.

[9] 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.

[10] 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.

[11] 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.

[12] S. Mahmoodi, K. Subbalakshmi, and V. Sagar, "Cloud offloading for multi-radio enabled mobile devices," in *Proc. IEEE International Conference on Communications (ICC)*, Jun. 2015, pp. 5473–5478.

[13] 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.

[14] 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.

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

[16] 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.

[17] 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.

[18] 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.

[19] 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.

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

[21] M. J. Osborne and A. Rubinstein, *A Course in Game Theory*. The MIT press, 1994.

[22] D. Monderer and L. S. Shapley, "Potential games," *Games and Economic Behavior*, vol. 14, no. 1, pp. 124–143, Jun. 1996.

[23] 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.