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 mobile 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
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

\[ x_i, x_{ai}, x_{ei} \in \{0, 1\} \]

where \( x_i \), \( x_{ai} \), and \( x_{ei} \) indicate whether user \( i \)'s task is processed locally, at the CAP, or at the cloud, respectively. Notice that only one of \( x_i, x_{ai}, \) and \( x_{ei} \) 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 \( D_{in}(i) \), \( D_{out}(i) \), and \( Y(i) \), respectively. When this task is processed locally, the processing energy is denoted by \( E_{li} \) and the processing time is denoted by \( T_{li} \).

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 \( E_{ci} \) and \( E_{cf} \), respectively, the energy consumed for wireless transmission and reception by the user. We further denote the uplink and downlink transmission times by \( T_{ci} = D_{in}(i)/r_{ui} \) and \( T_{cf} = D_{out}(i)/r_{di} \), respectively, where \( r_{ui} \) and \( r_{di} \) are uplink and downlink data rates allocated to user \( i \). Furthermore, \( r_{ui} \) and \( r_{di} \) are limited by the uplink and downlink capacities \( C_{UL} \) and \( C_{DL} \) as follows:

\[ \sum_{i=1}^{N} r_{ui} \leq C_{UL}, \quad (2) \]

and

\[ \sum_{i=1}^{N} r_{di} \leq C_{DL}. \quad (3) \]

D. Cost of Cloud Processing

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

\[ \sum_{i=1}^{N} f_{ai} \leq f_A. \quad (4) \]

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 \( x_i = (x_i, x_{ai}, x_{ei}) \) as well as the communication and CAP processing resource allocation \( r_i = (r_{ui}, r_{di}, f_{ai}) \). The optimization problem is formulated as follows:

\[ \min_{\{x_i\}, \{r_i\}} \left[ \sum_{i=1}^{N} \alpha_i (E_i x_i + E_{Ai} x_{ai} + E_{Ci} x_{ei}) + \max_i \{T_{Li} + T_{Ai} + T_{Ci}\} \right] \quad (5) \]

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

\[ r_{ui}, r_{di}, f_{ai} \geq 0, \quad i = 1, \ldots, N. \quad (6) \]
\[ x_l, x_a, x_c_i \in \{0, 1\}, \quad i = 1, \ldots, N, \quad (7) \]

where \( E_{A_i} \triangleq (E_t + E_r_i), \quad E_{C_i} \triangleq (E_t + E_r_i + \beta C_{c_i}) \) is the weighted transmission energy and processing cost for task \( i \) being offloaded to the cloud, with \( \beta \) being the relative weight, \( T_{A_i} \triangleq T_i x_l \) is the processing delay at the mobile user, \( T_{A_i} \triangleq (D_a(i)/r_{u_i} + D_{ua}(i)/r_{d_i} + Y(i)/f_{a_i}) x_{a_i} \), and \( T_{C_i} \triangleq (D_a(i)/r_{u_i} + D_{ua}(i)/r_{d_i} + T_{ac} + T_{c_i}) x_{c_i} \) correspond to the transmission and processing delay at the CAP and the cloud, respectively, and \( \alpha_i \) is the weight on energy consumption relative to the delay. We can adjust \( \alpha_i \) 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 to distribute 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

\[ G_{MCO} = (\mathcal{I}, (A_i)_{i \in \mathcal{I}}, (u_i)_{i \in \mathcal{I}}), \quad (8) \]

where \( \mathcal{I} = \{1, \ldots, N\} \) is the player set containing all mobile users, \( A_i \) 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 \( a_i \in A_i, \quad a_{-i} = (a_1, \ldots, a_{i-1}, a_{i+1}, \ldots, a_N) \in A_{-i} = \prod_{j \neq i} A_j, \forall i. \)

More specifically, we define the strategy set and the cost function for user \( i \) as

\[ A_i = \left\{ a_i = (x_{l_i}, x_{a_i}, x_{c_i}) \mid x_{l_i} + x_{a_i} + x_{c_i} = 1; \right. \]

\[ x_{l_i}, x_{a_i}, x_{c_i} \in \{0, 1\}, \quad (9) \]

and

\[ u_i(\mathbf{a}) = \alpha_i(E_{L_i} x_{l_i} + E_{A_i} x_{a_i} + E_{C_i} x_{c_i}) \]

\[ + \max_j \{T_{L_j} + T_{A_j} + T_{C_j}\}, \quad (10) \]

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 \( a \) 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:

\[ \min_{\{\alpha_i\}} \left( E + \max_i \{T_{L_i} + T_{A_i} + T_{C_i}\} \right) \quad (11) \]

s.t. (2), (3), (4), (6),

where \( E \triangleq \sum_{i=1}^{N} \alpha_i(E_{L_i} x_{l_i} + E_{A_i} x_{a_i} + E_{C_i} x_{c_i}) \) is a constant depending on \( a \). 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 \( a_i \in A_i, \forall i. \)

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 pure-strategy NE and has the FIP. Since it is indeed a potential function of an OPG defined in (12), it is therefore a potential function of $G_{MCO}$. Therefore, $G_{MCO}$ is an OPG which always has an NE and the FIP.

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 an OPG 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

$$
\phi(a) = \sum_{i=1}^{N} \alpha_i(E_i x_i + E_A a_i + E_C c_i) + \max_i \{T_i + T_A + T_C\}.
$$

(13)

Define $E_i \Delta \alpha_i(E_i x_i + E_A a_i + E_C c_i)$ and $T_i \Delta T_i + T_A + T_C$. Given two different strategy profiles $a = (a_i, a_{-i})$ and $a' = (a'_i, a'_{-i})$, where only user $i$ chooses different strategies $a_i$ and $a'_i$, respectively, the difference between $\phi(a)$ and $\phi(a')$ is

$$
\phi(a) - \phi(a') = E_i + \sum_{j \neq i} E_j + \max_{j \neq i} \{T_i, \max_{j \neq i} \{T_j\}\} - E_i' - \sum_{j \neq i} E_j - \max_{j \neq i} \{T_i', \max_{j \neq i} \{T_j\}\}
$$

$$
= E_i + \max_i \{T_i, \max_{j \neq i} \{T_j\}\} - E_i' - \max_{j \neq i} \{T_i', \max_{j \neq i} \{T_j\}\}
$$

$$
= u_i(a) - u_i(a').
$$

Since $\phi(a)$ 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.

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 $a^*$ of $G_{MCO}$ and send $a^*$ to all users. Due to the property of the NE, the users will follow $a^*$ and have no incentive to deviate from it.

The CAP first initiates a starting strategy profile $a[0]$ containing a set of hypothetical offloading decisions for all users. Based on $a[0]$, 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 $a'_i \in A_i$, while $a_{-i}[0]$ remains the same, it updates the strategy profile from $a[0]$ to $a[1]$ where $a_i[1] = a'_i$ and $a_{-i}[1] = a_{-i}[0]$. The CAP repeats the same procedure to find an improvement path $(a[0], a[1], a[2], \ldots)$ 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 $a^*$ where no mobile user can further decrease its cost by unilaterally changing its strategy.
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 results 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 $500 \times 10^6$ cycles/s and unit processing energy consumption $\frac{1.23 \times 10^7}{1}$ J/cycle. We consider the x264 CBR encoding application, which requires 1900 cycles/byte [23], leading to local computation time $4.75 \times 10^{-7}$ s/bit and local processing energy consumption $3.25 \times 10^{-7}$ 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 $C_{UL} = C_{DL} = 72.2$Mbps according to IEEE 802.11n. The transmission and receiving energy per bit at each mobile device are both $1.42 \times 10^{-7}$ J/bit as indicated in Table 2 in [23]. The CPU rates of the CAP and each sever at the remote cloud are $2.5 \times 10^9$ cycle/s and $5 \times 10^9$ cycle/s, respectively. For offloading a task to the cloud, the transmission rate is $R_{ac} = 15$Mbps. Also, we set the cloud utility cost $C_{ac}$ to be the same as that of the input data size $D_{in}(i)$, and $\beta = 3 \times 10^{-7}$ J/bit. We further set $\alpha_0 = \alpha = 0.5$ 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 $\alpha[0]$ and corresponding resource allocation $\{r^*_i[0]\}$. In all figures shown, we provide the performance of two NEs obtained by two different starting profiles: 1) the random mapping NE, where in $\alpha[0]$ each task is processed at different locations with equal probability; and 2) the shareCAP NE, where $\alpha[0]$ 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 $\alpha[0]$ for the random mapping NE, 2) the shareCAP method, where the offloading decisions are exactly the same as $\alpha[0]$ 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 $\alpha_3 = \alpha$ 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 $\beta$ on the cloud processing cost in Fig. 3. As expected, we see that when $\beta$ becomes large, all tasks are more likely to be processed by either the mobile user or the CAP. In
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 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


