Multi-User Mobile Offloading Game with Multiple Computing Access Points

Deng Pan and Eric Jiang
Dept. of Electrical and Computer Engineering, University of Toronto, Canada

***Abstract*—Mobile Cloud Computing (MCC) can improve the performance of mobile applications by enabling tasks to be offloaded to powerful remote servers. Further performance enhancements to MCC may be possible by providing wireless and computational resources to mobile users near the mobile edge. This system is known as Mobile Edge Computing (MEC), and one example of such a system is adding computational servers to the wireless base stations. One substantial area of research here is the offloading and resource allocation problem - the determination of the optimal distribution of tasks and allocation of communication and computational resources between the various units in the system (i.e. the mobile device itself, computational servers at the edge, and cloud servers).**

In this work, we consider a multi-user mobile edge computing system with multiple computing access points (CAPs) where each mobile user has multiple dependent tasks to be processed. The user has three options: process locally at the device, offload to one of the CAPs, which can in turn offload to the remote cloud server. Using a round-by-round schedule, it suffices to for us to jointly optimized the offloading decisions of users and CAPs, along with the associated computational resource allocation. This non-convex centralized optimization can be solved with a game-theoretic approach, by which we prove that selfish mobile users is willing to participate in and truthfully follow the decisions of the FIP algorithm. In simulations, we compare the communication and computational costs evaluated at a Nash Equilibrium and the mean random placement of offloading strategies for each user at various parameter settings.

I. Introduction

Advances in mobile computing have resulted in increasingly sophisticated programs being run on mobile and embedded applications, such as GPS navigation, autonomous robots, and environmental sensing [1]. As a result, modern computing has branched off from traditional platforms of desktops and mainframe computers. Limited system resources and a lack of network bandwidth typically exemplify these emergent and novel platforms [2]. Thus, there is an increasing demand to have complex programs run smoothly and efficiently on such platforms.

Mobile cloud computing (MCC) seeks to improve the quality of service (QoS) for mobile applications, allowing the mobile user to perform more intensive tasks on their mobile devices. The process of offloading involves migrating programs outside of the users immediate computing environment. This form of surrogate computing should occur if the expected application QoS will increase after the device connected to a cloud network offloads the given task [2]. Benefits of offloading includes improvements to the application's energy consumption and processing delay, which can be quantified as a metric for QoS. Mobile edge computing (MEC) aims to improve upon MCC by adding radio/computational resources near the mobile device and at the 'edge' or base station, thereby reducing the latency caused by communications between the user and the servers and moving towards ultra-reliable low latency communications (URLLC) envisaged for 5G networks. This is especially important for tasks such as virtual reality or Internet-of-things. Thus, offloading is a solution to augment mobile systems capabilities by migrating computation to more resourceful servers (i.e. the cloud) [1]. A substantial literature has been developed in recent years on research problems in these areas. One of these problems is that of resource allocation. An MCC/MEC system will likely have multiple users and tasks simultaneously in operation, each with their own communication/computational requirements and desired QoS. In particular, mobile bandwidth at the base station and computational power at the cloud or edge server will be in contention — thus, optimal task placement and scheduling will be important to the proper functioning of this system.

In the work of Chen et al. [1], a multi-user, multi-task system was considered, with a single computing access point (CAP) and one remote cloud available for offloading tasks, and bandwidth on one wireless channel and processing rate at the CAP as resources to be allocated to each user. There, the authors formulated the offloading and resource allocation problem as a non-convex mixed integer programming problem, where the objective is a weighted sum of total energy consumption and processing time. To solve the problem, the authors used game theory to iterate through a set of offloading strategy, then solved the convex resource allocation problem for each offloading strategy until a Nash Equilibrium was reached. This was guaranteed to be found as the system was shown to be an ordinal potential game — thus a finite improvement path algorithm can be used to find the Nash equilibrium. In this paper, we extend their work to consider the case of multiple CAPs, each with their own wireless channel and CPU. We formulate an analogous mixed-integer programming problem to the original work, then use the game-theoretic approach to arrive at a Nash equilibrium. In this approach, a single user iterates through all the possible offloading decisions, based on the decision of the other users. For each offloading decision, a convex resource allocation problem is solved, which is then used to compute the user's individual objective. The user changes their decision if an improvement to their own objective is found. This process cycles through the set of users until no user changes their strategy, at which point a Nash Equilibrium (NE) is reached. Subsequent simulations

show that our solution achieves substantial improvements over a randomized mapping of strategies under a series of different parameter settings.

II. Related Work

Various cost/benefit studies conducted focus on whether to offload computation to the cloud and attempts to make offloading attractive [2]. Prior works on mobile cloud computing models the offloading in a pure client-server scenario [3][4][5]. This led to general energy and delay conditions established by Kumar et al. [2] and transformed into a mobile offloading decision algorithm. Additionally, custom architectures have been implemented to replace remote cloud servers with nearby cloudlets to reduce transmission latency as proposed by Satyanarayanan et al. [6]. The challenges of cloud offloading from single to multiple user scenarios have been studied in [4][5][7]. Moreover, E. Cuervo et al. developed a task partitioning system, known as MAUI, to efficiently process the mobile users applications in [8]. However, in all above cases, the resultant integer programming cannot be solved easily [9].

Chen et al. managed to tackle this challenge by redesigning the conventional mobile cloud architecture through the introduction of the device known as a computing access point [1]. CAPs can be conceptualized as devices with built-in computation capability, such as wireless access points or cellular base stations [1][9]. Introduction of a CAP in between the dichotomy of mobile users and remote cloud servers have led to significant system performance improvements in both single and multi-user scenarios [9][10]. By considering multiple independent tasks, the joint decision and resource allocation problem can be formulated as a non-convex quadratically constrained quadratic program (QCQP), which is NP-hard in general. A heuristic algorithm based on semidefinite relaxation (SDR) proposed in [9] successfully reduced the energy and computation costs with a CAP-based system.

III. Problem Formulation

Consider a mobile network consisting of one cloud server, M CAPs, and N mobile users, as shown in Figure 1. Each user has K sequentially ordered tasks, which are processed in a round-by-round schedule, where each user processes one task each round, and a new round begins only once all the tasks in the current round are finished. In this system, only one round needs to be considered for making offloading and resource allocation decisions, as decisions between rounds are independent. Thus in the following, only one round will be considered.

A. Offloading Decision

In each round, a user's task may be processed by the user locally, or offloaded to one of M CAPs. If a task is offloaded to a CAP, it may be processed there or further offloaded to the remote cloud server. Denote the offloading decision for each user by:

$$\mathbf{x_i} = [x_{i0}, x_{i1}, \ldots, x_{iM}] \in \{0,1\}^{M+1} \quad (1)$$

$$\theta_i \in \{0,1\} \quad (2)$$

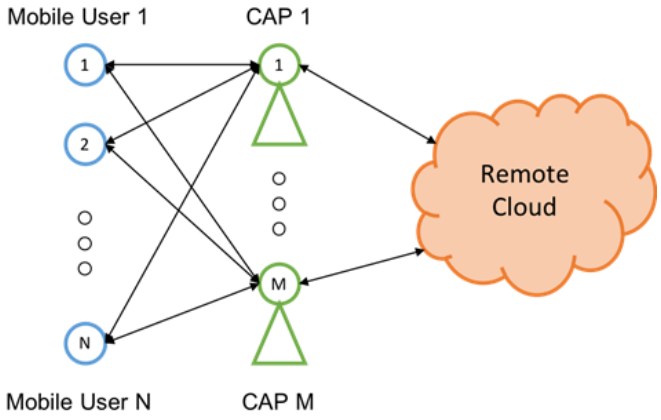

Fig. 1. Mobile Cloud System Model

where $x_{ij} = 1$ if user i's task is offloaded to CAP j ($x_{i0} = 1$ when the task is locally processed), and $\theta_i = 1$ if a task that is offloaded to a CAP is processed at the cloud. Because the tasks are atomic and cannot be placed in multiple locations, only one of $x_{i0}, x_{i1}, \ldots, x_{iM}$ can be one — this can be expressed through the following constraint:

$$\sum_{j=0}^{M} x_{ij} = 1 \quad (3)$$

B. Local Processing

Each task consists of the following parameters: an input data size $D_{in}(i)$, output data size $D_{out}(i)$, and processing size (in terms of number of processing cycles) $Y(i)$. For local computation, the processing time required is $T_{l_i} = Y(i)/f_{L_i}$, where f_{L_i} is the processing rate of the mobile device for user i. The energy consumption for the user is $E_{l_i} = \kappa_i Y(i) f_{L_i}^2$, where κ is a constant that depends on CPU architecture.

C. CAP Processing

For tasks processed at the CAPs, the bandwidth and processing rate allocated to each user must be determined in order to compute the energy and time required to complete a task. Each CAP has its own wireless channel where users can offload their tasks to. The uplink transmission time is then given as $T_{t_{ij}} = D_{in}(i)/r_{ij}^u$, where r_{ij}^u is the uplink data rate to CAP j assigned to user i. The downlink transmission time is then given as $T_{r_{ij}} = D_{out}(i)/r_{ij}^d$, where r_{ij}^d is the downlink data rate to CAP j assigned to user i. The rates r_{ij}^u and r_{ij}^d are further limited by link capacities as follows:

$$\sum_{i=1}^{N} r_{ij}^u \leq C_j^{UL} \quad (4)$$

$$\sum_{i=1}^{N} r_{ij}^d \leq C_j^{DL} \quad (5)$$

Furthermore, the task is processed at the CAP with processing time $T_{a_{ij}} = Y(i)/f_{ij}^a$, where f_{ij}^a is the processing

rate at CAP j assigned to user i, and constrained by the total processing rate of the CAP as follows:

$$\sum_{i=1}^{N} f_{ij}^{a} \leq f_{j}^{A} \tag{6}$$

The total time required for processing task i on CAP j can then be expressed as follows:

$$T_{ij}^{A} = T_{t_{ij}} + T_{r_{ij}} + T_{a_{ij}} \tag{7}$$

The energy consumption required for CAP processing can be expressed as:

$$E_{a_i} = E_{t_i} + E_{r_i} \tag{8}$$

where E_{t_i} and E_{r_i} represent the uplink and downlink transmission energy costs required by the user, respectively.

D. Cloud Processing

For cloud processing, the uplink and downlink transmission energies and times are identical to the CAP processing case. However, there is now a further transmission time between the CAP and the cloud, expressed by $T_{ac_i} = (D_{in}(i) + D_{out}(i))/r_{ac}^{j}$, where r_{ac} is a predetermined wired transmission rate between each CAP and the cloud. The computation time in this case is $T_{c_i} = Y(i)/f_C$, where f_C is a predetermined processing rate assigned to each user at the cloud. Thus, the total cloud processing time can be expressed as:

$$T_{ij}^{C} = T_{t_{ij}} + T_{r_{ij}} + T_{ac_j} + T_{c_i} \tag{9}$$

The energy consumption required can be expressed as:

$$E_{c_i} = E_{t_i} + E_{r_i} + \beta C_{C_i} \tag{10}$$

where C_{C_i} is the cloud utility cost, and β the relative weight.

IV. Centralized Optimization Problem

To maintain an adequate quality of service for each user, we seek to minimize both energy consumption and processing time for the tasks among the users. Thus, our centralized optimization problem considers a weighted sum of the total energy consumption among all the users and the round time, which is the maximum processing time among all the tasks. Our decision variables are the offloading decisions $\mathbf{x_i}$ and θ_i, as well as the resource allocation decisions $\mathbf{r_i^u} = [r_{i1}^{u}, \ldots, r_{iM}^{u}]$, $\mathbf{r_i^d} = [r_{i1}^{d}, \ldots, r_{iM}^{d}]$, and $\mathbf{f_i^a} = [f_{i1}^{a}, \ldots, f_{iM}^{a}]$. Thus, the optimization problem is as follows:

$$\min_{\{\mathbf{x_i}\},\{\mathbf{r_i^u}\},\{\mathbf{r_i^d}\},\{\mathbf{f_i^a}\}} \left\{ \sum_{i=1}^{N} \alpha_i (E_{L_i} + E_{A_i} + E_{C_i}) + \max_{i \in \{1,\ldots,N\}} \{T_{L_i} + T_{A_i} + T_{C_i}\} \right\} \tag{11}$$

subject to:

$$E_{L_i} = E_{l_i} x_{i0}, \qquad T_{L_i} = T_{l_i} x_{i0} \tag{12}$$

$$E_{A_i} = \sum_{j=1}^{M} E_{a_i} x_{ij} (1 - \theta_i), \quad T_{A_i} = \sum_{j=1}^{M} T_{ij}^{A} x_{ij} (1 - \theta_i) \tag{13}$$

$$E_{C_i} = \sum_{j=1}^{M} E_{c_i} x_{ij} \theta_i, \qquad T_{C_i} = \sum_{j=1}^{M} T_{ij}^{C} x_{ij} \theta_i \tag{14}$$

$$(1-10),$$

$$r_{ij}^{u}, r_{ij}^{d}, f_{ij}^{a} \geq 0 \tag{15}$$

where α_i is a relative weight between the energy cost and time requirement.

V. Multi-User Offloading Game

Because this is an NP-hard mixed integer programming problem, we use a game theoretic approach to arrive at a solution.

The parameters of the game are as follows:

$$\mathcal{I} = \{1, \ldots, N\} \tag{16}$$

$$\mathcal{A}_i = \{a_i = (\mathbf{x_i}, \theta_i) | \text{ (1-3)}\} \tag{17}$$

$$u_i(a_1, \ldots, a_N) = \alpha_i (E_{L_i} + E_{A_i} + E_{C_i}) + \max_{j \in \mathcal{I}} \{T_{L_j} + T_{A_j} + T_{C_j}\} \tag{18}$$

where $\mathcal{I}$ is the set of players (which in this case are the mobile users), $\mathcal{A}_i$ is the set of all possible strategies for player i, and u_i is the individual cost function associated with each player.

For each set of offloading decisions $\mathbf{A} = \{a_i\}_{i \in \mathcal{I}}$ made by the users, the communication and computational resources are assigned to the users by the CAPs through the following optimization problem:

$$\min_{\{\mathbf{r_i^u}\},\{\mathbf{r_i^d}\},\{\mathbf{f_i^a}\}} \left\{ \sum_{i=1}^{N} \alpha_i (E_{L_i} + E_{A_i} + E_{C_i}) + \max_{i \in \mathcal{I}} \{T_{L_i} + T_{A_i} + T_{C_i}\} \right\} \tag{19}$$

subject to: (4-10), (12-15).

Note that the resource allocation problem is convex. The energy consumption is a constant term (as the offloading decisions are fixed), while the time cost is a maximum of sums of positive reciprocal functions, which are themselves convex. All of the constraints are linear — thus the feasible set is convex. Because of this, the problem can be solved optimally using a convex optimization solver.

Our objective is to find the Nash equilibrium of the game, which is defined as follows:

Definition 1 [1]: *The strategy profile $a^* = (a_1^*, ..., a_N^*)$ is a **Nash Equilibrium** if and only if no user i can reduce their cost by altering their strategy*[1] *- i.e.,*

$$u_i(a_i^*, a_{-i}^*) \geq u_i(a_i, a_{-i}^*), \forall a_i \neq a_i^*, \forall i \in \{1, \ldots, N\} \tag{20}$$

[1]In this paper, we consider only pure strategy Nash equilibria.

where $a_{-i} = \{a_1, \ldots, a_N\} \setminus a_i$.

To show that at least one Nash equilibrium exists in this game, we prove that the game is an ordinal potential game, which is defined as follows:

Definition 2 [1] *An **ordinal potential game** G is a game where there exists a potential function $\Phi(\boldsymbol{a})$ such that:*

$$u_i(a_i, a_{-i}) - u_i(a_i', a_{-i}) > 0 \iff \Phi(a_i, a_{-i}) - \Phi(a_i', a_{-i}) > 0, \forall i \in U$$

Due to space limitations, we omit the proof that this game exhibits an ordinal potential, and simply state outright that the global objective defined in (11) is in fact a potential function for this game:

$$\Phi(a_i, a_{-i}) = \sum_{i=1}^{N} \alpha_i (E_{L_i} + E_{A_i} + E_{C_i}) + \max_{i \in \{1,\ldots,N\}} \{T_{L_i} + T_{A_i} + T_{C_i}\} \quad (21)$$

All ordinal potential games have at least one Nash equilibrium. Furthermore, a Nash equilibrium can be reached using the finite improvement path algorithm, which is defined below:

FIP-ALGORITHM

```
Initialize each user's strategy to a random value, (a_1, ..., a_n)
NE = False
while NE = False
    Improvement = False
    for all users i ∈ I
        for all possible strategies a_i' ≠ a_i ∈ A_i
            if u(a_i', a_-i) < u(a_i, a_-i)
                a_i = a_i'
                Improvement = True
                break
    if Improvement = False
        NE = True
```

Thus, a Nash equilibrium for our system can be found using the above algorithm. We begin with a random strategy for each user. For each user, the optimal resource allocation is calculated for each strategy by solving the the optimization problem in (19), given the strategies of the other users. The costs of each are computed until no user can improve their cost by changing their strategy, at which point a Nash equilibrium is reached. By the properties of an ordinal potential game, this algorithm is guaranteed to converge in a finite number of steps.

VI. SIMULATION RESULTS

For our system, we use the mobile device characteristics from [1], which was based on the Nokia N900. Unless otherwise stated, we set the number of users as $N = 10$, number of CAPS $M = 3$, and number of tasks per user $K = 20$ (which are independent and identically generated). The local computation time is $4.75(10^{-7})$ s/bit, and local energy consumption

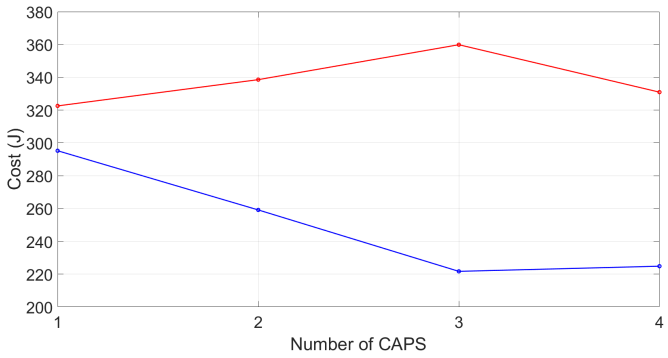

Fig. 2. Total Cost vs number of CAPS M

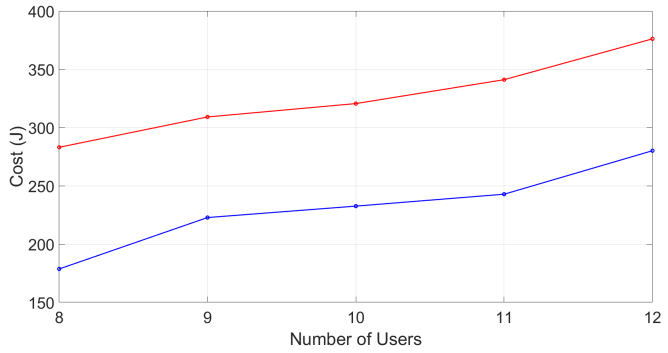

Fig. 3. Total Cost vs number of users N

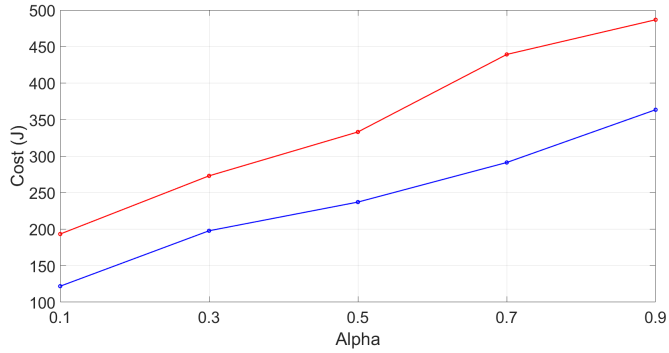

Fig. 4. Total Cost vs α

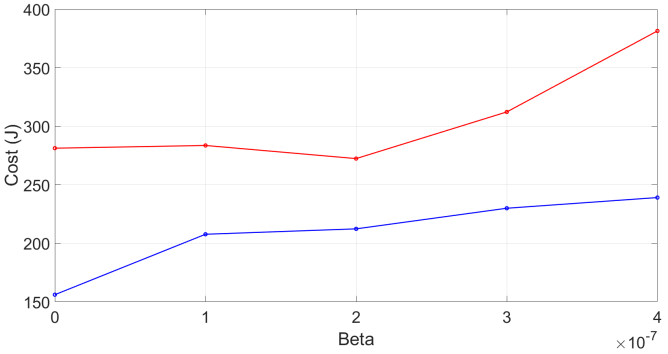

Fig. 5. Total Cost vs β

is $3.25(10^{-7})$ J/bit. The input and output data sizes for each task is uniformly distributed between 10MB and 30MB, and we assume 1 byte of input data corresponds with one CPU cycle. From IEEE 802.11n, we have $C_{UL} = C_{DL} = 72.2$ Mbps. The energy per bit for transmission and receiving at the mobile device is $1.42(10^{-7})$ J/bit. Each CAP has a CPU rate of $2.5(10^9)$ cycles/s, and the cloud server has a CPU rate of $5(10^9)$ cycles/s. The transmission rate from the CAP to the cloud is 15 Mbps. The cloud utility cost C_{C_i} is set to the input data size, and we set the relative weights $\alpha = 0.5s/J$ and $\beta = 3(10^{-7})$ J/bit.

In the plots shown in Figures 2-5, we compare the global objective cost of the Nash equilibrium compared with a random placement of tasks, averaged over the K tasks. In each plot, the red line denotes the result from random placement, while the blue line denotes the result from the Nash equilibrium. Each of our figures show a substantial gain from the random mapping of offloading decisions that is achieved by

Figure 2 shows the total cost plotted against the number of CAPS M in the system. It is expected that the cost would decrease as the number of CAPS increases, as that entails a greater number of offloading decisions available to the users along with less contention in the bandwidth. However, our plot showed that in our current configuration, there is no gain to be had from increasing the number of CAPS from 3 to 4. This suggests that there exists a point in the system after which adding additional CAPs no longer improves the system performance.

Figure 3 shows the cost plotted against the number of users N in the system. As expected, we see that an increase in the number of users results in an increase in the total cost.

Figures 4 and 5 plot the total cost against the relative weights α and β, respectively. With lower α, the weight given to the energy consumption is reduced, allowing the system to offload tasks more to the CAP and cloud to reduce the processing time (since the energy consumption is almost always greater for local processing compared to offloading). With lower β, the utility cost of using the cloud decreases, allowing more users to offload their tasks there and thus reduce their processing time. However, the plot shows that after a certain point, an increase in β has little effect on the total cost, as there are too few tasks being processed at the cloud for a further increase in β to have an impact.

VII. Conclusion

A multi-user mobile cloud computing system with multiple CAPs has been simulated. Formulation as a mobile cloud offloading problem gives rise to a difficult non-convex problem. By formulating the problem as a game, we demonstrate the existence of a NE and the finite improvement path property. This led to the development of a FIP algorithm that solves for the NE solution and the corresponding resource allocation vectors. The simulation demonstrated the effects of various parameters on the global utility (cost) function being minimized with default parameter setting coming from a model of Nokia N900. This global utility can be proven as the potential function that the offloading game formulated uses to satisfy the OPG definition. Future work that may result from this includes considering dependent tasks, multiple CPUs at the CAP, or allowing tasks to be split between the multiple CAPs.

References

[1] M. Chen, M. Dong and B. Liang, "Multi-user mobile cloud offloading game with computing access point," in 2016, Available: https://ieeexplore.ieee.org/document/7776577. DOI: 10.1109/CloudNet.2016.52.

[2] K. Kumar and Y.-H. Lu, Cloud computing for mobile users: Can offloading computation save energy? Computer, vol. 43, no. 4, pp. 5156, Apr. 2010.

[3] 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. 129140, Feb. 2013.

[4] 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. 54735478.

[5] 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. 411.

[6] M. Satyanarayanan, P. Bahl, R. Caceres, and N. Davies, The case for vm-based cloudlets in mobile computing, IEEE Pervasive Computing, vol. 8, pp. 1423, 2009.

[7] 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. 723734, 2013.

[8] 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. 4962.

[9] 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. 186190.

[10] 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.