Gaming and Learning Approaches for Multi-user Computation Offloading

(Invited Paper)

Sowndarya Sundar and Ben Liang
Department of Electrical and Computer Engineering
University of Toronto, Ontario, Canada
{ssundar, liang}@ece.utoronto.ca

***Abstract*—We consider both offline and online computational offloading of tasks from multiple users to a cloud or nearby cloud at the edge. We model the offline problem as an N-player finite game where each user has access to information from other users, and we use an optimization approach to find a mixed-strategy Nash equilibrium solution. We also consider a practical online version wherein tasks arrive over time and a user does not require information from other users. We suggest a solution to this online problem by adopting a payoff-based reinforcement learning algorithm, which converges to a pure-strategy solution. Through simulation, we observe that the trends of the Nash equilibrium obtained from the offline technique and the pure-strategy point obtained from the online solution are similar. While the offline algorithm obtains a better solution on average, the online algorithm is much faster, particularly for larger systems.**

I. Introduction

Mobile cloud computing enables the computational offloading, or migration, of tasks from an application on a mobile device to a remote cloud or a nearby cloudlet. A mobile application can be partitioned into a number of tasks, and offloading decisions can be made to reduce the overall completion time [1]–[3]. However, most of the existing works only address the single-user offloading scenario or assume a centralized decision-making mechanism that schedules the tasks for all users [4]–[10]. Such centralized schemes require the communication of information from all devices to the decision maker and may not be in the best interests of all users.

In this work, we consider multiple mobile users, each of whom requires multiple tasks to be executed. These users may offload their tasks to a nearby shared cloud-at-the-edge or execute the tasks locally on their own device. The processors at the cloud have varying speeds. Each user aims to identify task scheduling decisions with an objective of minimizing their own completion time, which is a combination of processing time and communication time. This lends itself to a game theoretic model.

We first consider the offline version of this problem wherein each user has knowledge of its set of tasks in advance. Furthermore, it is aware of the task information of other users. We can then model this problem as an N-player finite game, and we propose to identify a mixed-strategy Nash equilibrium for this game by utilizing an optimization problem formulation modified from [11].

However, in a practical setting, each user might not have access to the task and decision information of the other users. Furthermore, tasks may arrive at a user dynamically over time. Thus each user is required to make decisions by observing its own completion time and learning over time to identify the best strategy. We model this online model as a repeated game, and, we propose a payoff-based reinforcement learning (RL) algorithm based on the Erev-Roth model. This algorithm converges to a pure-strategy equilibrium point but does not guarantee a Nash equilibrium. Through simulation, we observe the convergence of the online RL algorithm. We also compare and contrast the online and offline solutions against each other in order to study their relative advantages and disadvantages.

The rest of the paper is organized as follows. In Section II, we present the related work. Section III describes the system model and the problem formulation. In Section IV, we present the proposed solution approaches. Section V presents the simulation results, and we conclude the paper in Section VI.

II. Related Work

There are few existing works that look to tackle the multi-user computational offloading problem using game theory. In [12], the problem of multi-user offloading with one wireless access point and one cloud server

is modeled as a decentralized computational offloading game, and it is proved that the game always admits a Nash equilibrium solution. But the authors only consider the case where each user has a single task to execute.

In [13], a three tier model consisting of mobile devices, one resource-constrained cloudlet and one remote cloud is considered. The authors formulate a generalized Nash equilibrium problem, and a distributed algorithm is proposed to compute an equilibrium strategy. However, the cloudlet resource is assumed to be a single entity where any number of tasks can be executed simultaneously with the processing times of all tasks involved being affected by the load. While this simplifies the model to a congestion game, it penalizes a user whose task is already being executed at the cloudlet when other tasks arrive. A similar model with a computing access point at the mobile edge and a remote cloud server is considered in [14], where an ordinal potential game is used to find a Nash equilibrium solution for the offline problem.

Our model allows for processors at the cloud to have varying speeds. Additionally, the processors at the cloud are finite in nature and can run just one task at a time, with any task arriving later to be added to a queue. The mobile devices are considered to be finite-capacity devices. More importantly, we also explore an online version of this problem where users have access to only their own task information. To the best of our knowledge, the computational offloading problem for such a game-theoretic model has not been studied in the literature.

III. System Model and Problem Formulation

A. Tasks and Cloud Model

We consider a system with N mobile device users who are represented by players in our game theoretic model. Each mobile user $i \in \{1, \dots N\}$ wishes to complete a set of tasks (potentially belonging to a mobile application). The processing time for each task $j \in \mathcal{J}_i$, where $\mathcal{J}_i$ is the set of tasks corresponding to user i, is t_j on its local processor. Each task is released or becomes available for processing at time s_j. Furthermore, each task has a communication time c_j associated with it if this task were to be processed at the cloud. This communication time can be calculated as $c_j = \frac{d_j}{\tau}$ where d_j is the input data required for task execution and τ is the transmission rate of the channel.

The system also includes a finite-capacity cloudlet or edge-cloud, which may be located on a base station near to the users. We shall use the term "cloud" to refer to this computing service provider from here on for simplicity. Each processor on the cloud is assumed to have unary capacity, i.e., it can execute only one task at a time. It can be noted that devices that have non-unary capacity, or multi-core, may be represented by combining multiple unary capacity processors. However, the processors at the cloud can run at different speeds. Each user can execute its tasks either locally on its own mobile device processor or remotely at one of the cloud processors. The "speed-up" factor for each cloud processor r is α_{ir}, which indicates the amount of speed-up user i's tasks can achieve while utilizing the cloud processor r instead of running on the processor locally at the mobile device. Thus, the speed-up factors for the local processors can be considered to be one. Let $\mathcal{R}_i$ denote the set of processors to which user i can offload its tasks, $\mathcal{C}$ be set of cloud processors, and $\mathcal{R}$ be the set of all processors (local and cloud).

B. Offline Model: N-Player Finite Game

In the offline version, each user has knowledge of its set of tasks in advance, including their processing times and release times. Furthermore, it is aware of the task information of other users in addition to their set of strategies, i.e. possible offloading decisions. Given the strategies of the other players, each player i wishes to identify its best response strategy that minimizes its completion time, T_i. This can be represented by an optimization problem where the decision variables x_{jrp} are defined as follows:

$$x_{jrp} := \begin{cases} 1 & \text{if task } j \text{ is on processor } r \text{ at position } p, \\ 0 & \text{if otherwise,} \end{cases}$$

$\forall\ j \in \{1, \dots P\}$, $r \in \mathcal{R}$, and $p \in \{1, \dots P\}$, where $P = \sum_{i=1}^{N} |\mathcal{J}_i|$.

The position index p indicates the position in the queue of each task on the processor where it is executed. The position of each task is required because it determines the completion time of the task and consequently affects the maximum completion time for its associated user. Let auxiliary variables C_j represent the completion time of offloaded task j. The optimization formulation is as follows:

$$\min_{\{x_{jrp}\}} T_i \tag{1}$$

$$\text{s.t.} \sum_{p} \sum_{r \in R_i} x_{jrp} = 1, \quad \forall i \in \{1 \dots N\},\ j \in \mathcal{J}_i, \tag{2}$$

$$\sum_{i=1}^{N} \sum_{j \in J_i} x_{jrp} \leq 1, \quad \forall r \in \mathcal{R},\ p \in \mathcal{P}, \tag{3}$$

$$C_j - C_k + (2 - x_{jrp} - x_{kr(p-1)})I \geq \alpha_{ir} t_j, \quad \forall i \in \{1 \dots N\},\ j \in \mathcal{J}_i,\ k \in P,\ p \in \mathcal{P},\ r \in \mathcal{R}, \tag{4}$$

$$C_j \geq t_j \sum_p \sum_{r \in \mathcal{R}_i} \alpha_{ir} x_{jrp} + s_j + c_j \sum_p \sum_{r \in \mathcal{C}} x_{jrp}, \quad \forall i \in \{1 \dots N\},\ j \in \mathcal{J}_i, \tag{5}$$

$$T_i \geq C_j, \quad \forall j \in \mathcal{J}_i, \tag{6}$$

$$x_{jrp} \in \{0, 1\},\ \forall j \in \mathcal{P},\ r \in \mathcal{R},\ p \in \mathcal{P}. \tag{7}$$

The objective function (1) deals with the minimization of the completion time for user i. Constraint (2) ensures each task is only processed once at a particular position on a particular processor, and (3) is the resource constraint on each processor. Constraint (4) uses a large integer I to create a precedence constraint depending on the value of x_{jrp}. Constraint (5) sets the completion times for each task to be at least the sum of its processing time, release time, and communication time. Constraint (6) ensures that the completion time for each user is equal to the completion time of its last executed task. Constraint (7) ensures that the decision variables are binary.

Thus, this problem can be modelled as an N-player finite game where the mobile users are the players of the game, and all of them have access to a shared finite-capacity cloudlet resource. The completion time for a task from a particular user depends on the other users' strategies, since this affects the load at the processors at the cloud. This game can be denoted by $\mathcal{G}(\mathcal{N}, \Delta_i, T_i)$, where the more generic mixed-strategy representation of Δ_i is considered since the N-player finite game is only guaranteed to give us a mixed-strategy equilibrium by the Nash Equilibrium Theorem [15].

C. Online Model: Repeated Game

In the offline model above, each user requires information from the other users in order to solve its optimization problem. A more practical assumption would be an online model where each user does not require task information from the other users. Furthermore, in many applications, the tasks arrive one at a time, and a user needs to identify a strategy for the task that has arrived.

A repeated game is one consisting of a number of repetitions of some base game. If the tasks arriving at a particular user can be assumed to have identical processing times or have some sense of stationarity, then such an online model can be viewed as a repeated game, where during each round a single task arrives at each user, i.e., the base game corresponds to scheduling of a single task by each user. In application to practical scenarios, we may say that the user learns each time the same type of task is required to be executed. Thus, a user makes decisions by observing and learning from its own historical performance.

IV. Solution Approaches

In the following section, we propose methods to identify solutions for both the offline and online versions of the problem.

A. Optimization-Based Offline Solution

The offline problem is modelled as an N-player finite game, and each player i has $m_i = |\mathcal{R}_i|^{|\mathcal{J}_i|}$ possible strategies to choose from. Across all the players we have a total of $\prod_i^N |\mathcal{R}_i|^{|\mathcal{J}_i|}$ possible pure-strategy combinations. We define $\Sigma = (\sigma_{\mathbf{1}}, \sigma_{\mathbf{2}} \dots \sigma_{\mathbf{N}})$, where $\sigma_{\mathbf{i}} = (\sigma_{i0}, \sigma_{i1} \dots \sigma_{i(m_i-1)})$ refers to mixed strategy of user i and σ_{ij} is the mixed-strategy probability assigned to pure strategy s_{ij}. Let the cost (completion time) for user i for mixed strategy profile Σ be $a_i(\Sigma)$. Let O_i be the expected completion time for player i. Modifying from the approach suggested in [11], we can find a mixed-strategy Nash equilibrium using the following optimization. The details are omitted due to page limitation.

$$\min_{\Sigma} \sum_{i=1}^{N} (a_i(\Sigma) - O_i) \tag{8}$$

$$\text{s. t.} \sum_{j=1}^{m_i} \sigma_{ij} = 1 \quad \forall i \in 1 \dots N \tag{9}$$

$$O_i - a_i(s_{ij}, \sigma_{-\mathbf{i}}) \leq 0 \quad \forall j \in 1 \dots m_i, \ \forall i \in 1 \dots N \tag{10}$$

$$\sigma_{ij} \geq 0 \quad \forall j \in 1 \dots m_i, i \in 1 \dots N \tag{11}$$

The optimal value of σ ensures that the objective is minimized and Nash equilibrium is found by obeying constraint (10). In [11], a similar non-linear optimization problem is solved using a sequential quadratic programming based quasi Newton method, and we adopt this method to identify a solution to problem (8).

B. Reinforcement-Learning Based Online Solution

The online solution is found through an iterative algorithm based on the Erev-Roth model. For each user i, a score is assigned to each of its strategies in every iteration. We define $\mathbf{z}_i^k = (z_{i0}^k, z_{i1}^k \dots z_{i(|R_i|-1)}^k)$ as the strategy scores and $\mathbf{e}_i^k = (e_{i0}^k, e_{i1}^k \dots e_{i(|R_i|-1)}^k)$ as the pure strategy unit vector for user i at iteration k, where the 0^{th} strategy corresponds to local execution. Our algorithm has the following major steps:

1) Set initial values $z_{ij}^k = \frac{t_i}{\alpha_{ij}}$ for every strategy j.
2) If $t_i \geq C_i$:
   Assign $\mathbf{z}_i^{(k+1)} = \mathbf{z}_i^k + (t_i - C_i)\mathbf{e}_i^k$.
3) If $t_i < C_i$:
   - Reset $\mathbf{e}_i^k = 0$.
   - Set $e_{i0}^k = 1$.
   - Assign $\mathbf{z}_i^{(k+1)} = \mathbf{z}_i^k + (C_i - t_i)\mathbf{e}_i^k$.
4) Mixed strategy $\sigma_\mathbf{i}{}^{(k+1)} = \frac{\mathbf{z}_i^{(k+1)}}{Z_i^{(k+1)}}$, where $Z_i^{(k+1)} = \sum_{j=1}^{m_i} z_{ij}^{(k+1)}$.

The payoff in our model is the amount of time a user saves by offloading a task rather than running it on the local device. The strategy that maximises this value will also minimize the completion time of the task. Hence, if $t_i \geq C_i$, we add this obtained payoff to the score of the strategy $z_i^{(k+1)}$, and if not, we add the loss to the local processor to indicate that the processing at the local device is a better strategy. The corresponding mixed strategy probability $\sigma_i^{(k+1)}$ increases if the score $z_{ij}^{(k+1)}$ for strategy j increases. Thus, this strategy is more likely to be picked in the future and consequently, after a number of iterations, we can expect the probability of a particular strategy to converge to 1 for identical tasks arriving at a user. Additionally, we also introduce a cut-off parameter μ such that when σ_{ij}^k for some strategy j is less than μ, we set σ_{ij}^k to zero. This helps reduce the number of iterations and result in faster convergence. The algorithm stops when the users have attained a pure-strategy equilibrium value. While this algorithm cannot assure a Nash equilibrium, its ability to converge to a solution by using minimal information provides potentially beneficial trade-off to the offline scheme.

V. Simulation Results

We run MATLAB simulation to assess the performance of the two proposed algorithms. We consider a three-processor cloud with speed-up factors $\alpha_{i1} = 0.5$, $\alpha_{i2} = 0.2$, and $\alpha_{i3} = 0.8$ for every user i. We consider the arrival of one task at a time at each user, and the processing times of the tasks are generated randomly from a uniform distribution in (10, 100) ms, the release times are uniform in (0, 20) ms and the input data are uniform in (10, 100) KB. The data rate is set to 5 MBps.

Figure 1 depicts the variation in task completion time over multiple iterations in a network consisting of 3 users. We can see that the online RL algorithm solution converges to a particular pure strategy within a few hundred iterations. Figure 2 depicts the corresponding change in probability for choosing the equilibrium pure strategy as the number of iterations increases. We see that the probability of the equilibrium strategy increases and converges to 1. The probabilities of other strategies converge towards zero.

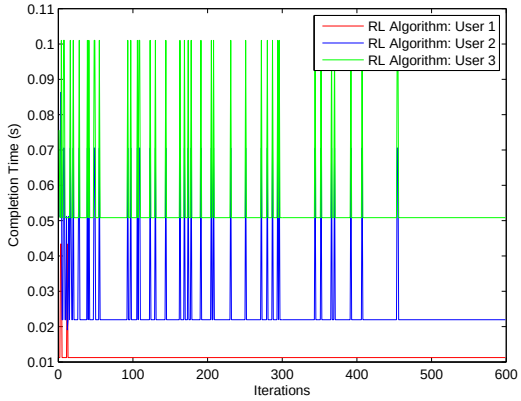

Fig. 1: Convergence of the task completion time for the online algorithm.

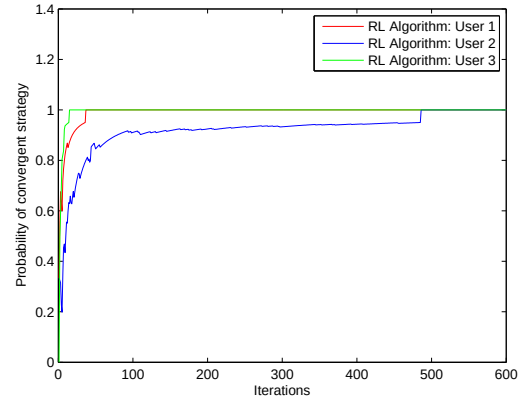

Fig. 2: Convergence of the online algorithm.

Figure 3 compares the task completion time value obtained from the Nash equilibrium offline solution and that obtained from the converged RL solution for the online problem. We consider the same simulation setup as that for Figures 1 and 2, but we vary the number of users in the system from 2 to 6. We run multiple randomized realizations and take the average among them to plot each point in the graph. We compare the task completion times for two specific users and on average across all users. We can see that the trajectory followed by both the online and offline completion times are similar, while the offline completion time is lower on average. This is understandable as each user has access to all of the information about the other users in the offline system, whereas in the online system, each user only learns by observing its own completion time.

Table I depicts the run-times of the offline and on-

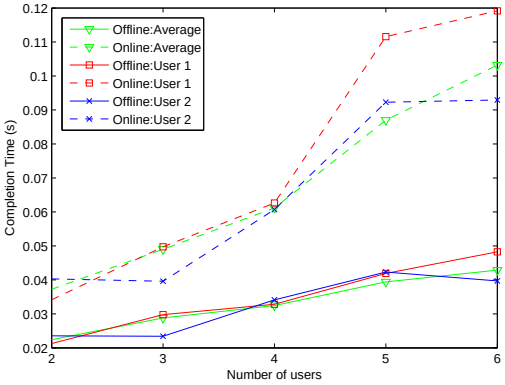

Fig. 3: Converged task completion time vs. the number of users, for User 1, User 2, and the average over all users.

TABLE I: Run-time (in seconds)

N	Offline	Online
2	0.0561	1.4664
3	0.1663	2.1657
4	2.6503	1.1439
5	21.5912	1.1463
6	333.2687	4.7291

line algorithms for the same simulation scenario. The online algorithm scales well with the increase in the number of users, but the offline optimization can be time-consuming, particularly for larger systems.

VI. Conclusion

We have considered both offline and online versions of the multi-user computational offloading problem. We have modelled the offline problem as an N-player finite game and suggested using an optimization problem to find a mixed-strategy Nash equilibrium. We have modelled the online problem as a payoff-based RL algorithm in order to consider a practical scenario where a user does not have access to information from the other users and tasks arrive dynamically over time.

While the online algorithm requires no external information and converges to a pure strategy solution, it does not guarantee a Nash equilibrium solution. On the other hand, the offline solution guarantees a mixed-strategy Nash equilibrium and can account for non-identical tasks, but each user requires information about all its own tasks in advance in addition to task information from all other users. Through simulation, we observe that the trends of the mixed-strategy Nash equilibrium obtained from the offline solution and the pure-strategy equilibrium point obtained from the online solution are similar. The offline algorithm gives smaller task completion times on average, but the online algorithm is much faster. One may choose between the two proposed solution techniques based on application and requirements.

References

[1] M. Jia, J. Cao, and L. Yang, "Heuristic offloading of concurrent tasks for computation-intensive applications in mobile cloud computing," in *Proc. IEEE INFOCOM Workshop on Computer Communications*, pp. 352–357, 2014.

[2] Y. Zhang, H. Liu, L. Jiao, and X. Fu, "To offload or not to offload: an efficient code partition algorithm for mobile cloud computing," in *Proc. IEEE International Conference on Cloud Networking (CloudNet)*, pp. 80–86, 2012.

[3] M.-A. Hassan Abdel-Jabbar, I. Kacem, and S. Martin, "Unrelated parallel machines with precedence constraints: application to cloud computing," in *Proc. IEEE International Conference on Cloud Networking (CloudNet)*, pp. 438–442, 2014.

[4] 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)*, pp. 49–62, 2010.

[5] A. Mtibaa, K. A. Harras, and A. Fahim, "Towards computational offloading in mobile device clouds," in *Proc. IEEE Conference on Cloud Computing Technology and Science (CloudCom)*, pp. 331–338, 2013.

[6] A. Mtibaa, K. A. Harras, K. Habak, M. Ammar, and E. W. Zegura, "Towards mobile opportunistic computing," in *Proc. IEEE International Conference on Cloud Computing*, pp. 1111–1114, 2015.

[7] R. K. Balan, D. Gergle, M. Satyanarayanan, and J. Herbsleb, "Simplifying cyber foraging for mobile devices," in *Proc. ACM International Conference on Mobile Systems, Applications, and Services (MobiSys)*, pp. 272–285, 2007.

[8] K. Habak, M. Ammar, K. A. Harras, and E. Zegura, "Femto clouds: Leveraging mobile devices to provide cloud service at the edge," in *Proc. IEEE International Conference on Cloud Computing*, pp. 9–16, 2015.

[9] W. Zhang, Y. Wen, and D. O. Wu, "Energy-efficient scheduling policy for collaborative execution in mobile cloud computing," in *Proc. IEEE INFOCOM*, pp. 190–194, 2013.

[10] B. Y.-H. Kao and B. Krishnamachari, "Optimizing mobile computational offloading with delay constraints," in *Proc. IEEE Global Communication Conference (Globecom)*, pp. 8–12, 2014.

[11] B. Chatterjee, "An optimization formulation to compute nash equilibrium in finite games," in *Proc. IEEE International Conference on Methods and Models in Computer Science*, pp. 1–5, 2009.

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

[13] V. Cardellini, V. D. N. Personé, V. Di Valerio, F. Facchinei, V. Grassi, F. L. Presti, and V. Piccialli, "A game-theoretic approach to computation offloading in mobile cloud computing," *Mathematical Programming*, vol. 157, no. 2, pp. 421–449, 2016.

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

[15] J. F. Nash *et al.*, "Equilibrium points in n-person games," in *Proc. Nat. Acad. Sci. USA*, vol. 36, no. 1, pp. 48–49, 1950.