Joint Offloading Decision and Resource Allocation with Uncertain Task Computing Requirement

Nima Eshraghi and Ben Liang
Department of Electrical and Computer Engineering, University of Toronto, Canada
Email: {neshraghi, liang}@ece.utoronto.ca

***Abstract*— We study the problem of joint offloading decision and resource allocation for mobile cloud networks with a computing access point (CAP) and a remote cloud center. We consider the case where the task computing requirement is not fully known before their execution. We aim to jointly optimize the offloading decisions as well as the allocation of computation and communication resources, to minimize a weighted sum of the average cost and cost variation. The problem is formulated as a mixed-integer program. We propose an efficient algorithm, termed Task Offloading and Resource Allocation with Uncertain Computing (TORAUC), and show that it always converges to a Karush-Kuhn-Tucker (KKT) point of an alternate form of the original problem, which has its binary constraints removed but guarantees an offloading decision solution that is arbitrarily close to binary. We extend TORAUC to TORAUC-MP for the case of a multi-processor CAP. Through trace-based simulation, we study the performance of TORAUC and TORAUC-MP. We observe that TORAUC is nearly optimal, and both algorithms substantially outperform several alternatives.**

I. Introduction

Recent advances in smart mobile devices have laid the foundation to support a broad range of interactive services such as augmented reality, online gaming, and social networking. As mobile applications grow in complexity, so does the demand on computing resources. The insufficient computational resources and limited energy supply in mobile devices, hence, impede the accommodation of high computation demand in emerging resource-hungry applications. With the help of cloud computing, mobile devices can potentially reduce their energy consumption by offloading computation-intensive tasks to the resource-rich cloud environment. Nevertheless, offloading to remote cloud servers can incur significant delays, particularly if a large amount of data needs to be communicated over already congested backhaul links.

Different from conventional mobile cloud networks, in Mobile Edge Computing (MEC) additional computing resources are deployed in close proximity to end users [1] [2]. Therefore, MEC serves as an attractive alternative to those offloaded tasks that require low latency. The concept of MEC may be abstracted by a computing access point (CAP), which is a wireless access point, such as a cellular base station, with built-in computation capability [3]. The computation tasks can be processed locally at the mobile device, offloaded to the CAP, or further forwarded by the CAP to a remote cloud center. There is a cost, which may include delay, energy consumption, and usage charges, associated with the task execution, and it depends on both the task and the processor where the task is served.

This work has been funded in part by the Natural Sciences and Engineering Research Council (NSERC) of Canada.

While many prior studies consider task offloading when the processing times of all tasks are available [3] - [7], the computing cycles that a task requires is generally uncertain until it is processed to completion [4]. Thus, in this work, we consider task offloading when the computing cycle requirement is not fully known before task execution. Such a system needs robustness to limit performance degradation despite the processing time uncertainty.

In particular, we study the problem of joint offloading decision and resource allocation under uncertain computing requirement, for multiple competing mobile users. The presence of uncertainty adds substantial challenge to the system design, complicating task scheduling and user competition in the sharing of communication and computation resources at the CAP. In fact, limited available shared resources can drastically affect task offloading decision under processing time uncertainty. Under-allocation of resources results in excessive user delay. Over-allocation, on the other hand, would adversely impact other users in the shared system. Consequently, satisfactory and sustained computation performance can only be achieved by a careful robust design.

A resource allocation policy is robust if its performance is not affected immensely across a wide range of operational conditions [5]. Here, we account for robustness in terms of maintaining a desired level of cost efficiency despite the uncertainty involved in the task computation requirements. The contributions of this work are as follows:

- We study joint offloading decision and allocation of communication and computation resources in a computing network with one CAP and a remote cloud server, where the computation requirement of the tasks is uncertain. We consider a system cost that accounts for energy consumption, processing delay, and communication delay. We formulate a robust optimization problem toward the objective of minimizing a weighted sum of the average cost and cost variation. It is a mixed-integer program and is non-convex even after binary relaxation.
- For the case where the CAP has a single processor, we propose an efficient algorithm, termed Task Offloading and Resource Allocation with Uncertain Computing (TORAUC). It is based on the construction of an iteratively updated sequence of locally tight approximate

geometric programming (GP) problems. We show that it converges to a KKT point of an alternate form of the original problem that has its binary constraints removed but guarantees an offloading decision solution that is arbitrarily close to binary.
- We further extend our solution to the case where the CAP has multiple processors. We propose the TORAUC-Multi-Processor (TORAUC-MP) algorithm. It is shown to have the same convergence property as TORAUC.
- Through trace-based simulations with Google cluster data [6], we study the impact of system settings on the performance of TORAUC and TORAUC-MP. We observe that TORAUC is nearly optimal over a wide range of parameter settings. Both algorithms outperform several alternatives.

The rest of the paper is organized as follows. In Section II we present the related works. Section III describes system model and problem formulation. We present TORAUC, prove its optimality, and derive its complexity in Section IV. Extension to multi-processor CAP is studied in Section V. We evaluate the performance of TORAUC via trace-driven simulations in Section VI, followed by concluding remarks in Section VII.

II. Related Works

The problem of task offloading and resource allocation has been studied in the literature under various settings with different objectives. In this section, we summarize existing works that investigate this problem for a three-tier cloud computing network. We further present related works on job assignment under processing time uncertainty.

A. Offloading and Resource Optimization in MEC with Known Processing Times

The joint optimization of offloading decision and allocation of communication and computation resources for a three-tier network, consisting of mobile users, a local computing node (e.g, cloudlet or CAP), and a remote cloud server, has been studied in [3] - [7]. In [3], scheduling of computation and communication resources is studied for a CAP with multiple users, each with a single task. The multi-user multi-task scenario is further studied in [8] [9], where upper and lower bounds are used for the overall processing delay, to address the complicated overlapping delays in offloading and processing multiple tasks in a shared medium. Task offloading with dynamic voltage frequency scaling is discussed in [10]. Fairness-aware cost minimization in a fog computing network for a multi-user single task scenario is studied in [7]. These works assume that the processing times of the tasks are exactly known a priori. This assumption is impractical, since processing requirement of a task is generally unknown until the task is executed to completion [4]. In this work, we account for the uncertainty of the task processing time.

B. Unknown Processing Times

Job assignment under processing time uncertainty is well studied in theoretical computer science. In [11], an energy-efficient scheduling policy is proposed, where tasks are scheduled based on an online estimation of their required computation cycle. For such estimation, the scheduler requires the probability distribution of the computing cycle demand. The authors of [12] consider the scheduling of dependent tasks, with the goal of both maximizing robustness and minimizing the makespan. The robustness metric is based on the slack of a task, which represents the window within which the task can be delayed without affecting the makespan. In [13], a task scheduling problem is considered to minimize the worst-case makespan under discrete scenario of uncertain processing times. In [14] and [15], two different models of partial information about task processing times are considered, and semi-online task scheduling algorithms are proposed with guaranteed competitive ratios on the makespan. In [16], dynamic job shop scheduling problem is studied under uncertain processing times. To minimize the total flow time, a clustering approach is proposed to classify the bottleneck machines. The focus of all of the above works is on the design of efficient task scheduling algorithms. They do not concern the allocation of computation or communication resources. In this work, we aim for robust minimization of a weighted sum of energy consumption, processing delay, and communication delay, through joint consideration of the offloading decision and allocation of computation and communication resources. To the best of our knowledge, this problem has not been studied in the literature.

III. System Model and Problem Formulation

A. Mobile Cloud Offloading System Model

Consider a cloud access network consisting of N mobile users, one CAP, and a remote cloud center. The CAP is a wireless access point (e.g., a cellular base station) equipped with computation resources. Instead of just serving only as a relay to forward the received tasks from users to cloud center, the CAP also has the capability to execute users' tasks subject to its computation resource constraints.

1) Users and Tasks: We denote the set of all users by $\mathcal{N} = \{1, \ldots, N\}$. Each mobile user has one computation task to be either executed locally at the mobile device, offloaded and served at the CAP, or sent further and processed at the remote cloud servers. We assume that user tasks are indivisible, i.e., computation tasks cannot be further divided into sub-tasks, and thus, the entire task must be executed in a single processor. Denote the offloading decisions for user i by $x_i^l, x_i^a, x_i^c \in \{0, 1\}$, indicating whether user i's task is processed locally, at the CAP, or at the cloud center, respectively. Then, the offloading decisions are constrained by

$$x_i^l + x_i^a + x_i^c = 1, \quad i \in \mathcal{N}. \tag{1}$$

Note that only one of x_i^l, x_i^a, and x_i^c for user i can be non-zero.

Similar to existing studies [3] - [7], we assume that all tasks are available at time zero. If the tasks arrive dynamically in

time, one can apply our model and proposed solution in a quasi-static manner, where the system processes the tasks in batches that are collected over time intervals [4].

The computation task of user i is represented by (L_i, ω_i), where L_i denotes the size of the task in bits including programming instructions and input data, and ω_i is the number of computing cycles required to process the task. We assume that L_i are known but precise values of ω_i are not available. This represents a realistic scenario where the size of the task can be measured but its processing time is generally uncertain until it is processed to completion [4]. Nevertheless, given the application type, some statistical information about the tasks (such as expected value of the computing cycles) can be reasonably inferred through measurements and experimental studies [17]. The revealed statistical information can be leveraged to limit performance degradation in the presence of uncertainty. Hence, in this work, we assume that although ω_i is random, its expected value $\overline{\omega_i}$ and some measure of its uncertainty, such as standard deviation or upper and lower bounds, are available. In particular, we consider a certain range of ω_i, denoted by Δ_i^{ω}, which is generally defined and may represent, e.g., the full range of possible ω_i values or some multiples of standard deviation around the mean value.

2) Local Execution Model: Denote the processing rate of user i's local mobile device by f_i^l, which can be adjusted depending on the application requirements and is limited by the device computing capability $F_i^{l^{\max}}$, i.e.,

$$f_i^l \leq F_i^{l^{\max}}. \tag{2}$$

The time delay for locally processing user i's task is denoted by $T_i^l = \omega_i / f_i^l$. The energy consumption model for local processing, denoted by $E_i^l(\omega_i, f_i^l)$, is linear in the task computing cycle requirement ω_i, and is generally a polynomial function of f_i^l [18]. Different mobile devices may have different coefficients and exponents that can be captured by the general energy model $E_i^l(\omega_i, f_i^l)$.

3) CAP Processing Model: To offload the computation tasks, the data bits need to be transmitted to the CAP over the wireless channel. For spectrum sharing among users, as an illustrative example, we consider orthogonal frequency devision, but this work can be extended to other sharing methods. The data rate of user i is given by $b_i \log(1 + \frac{h_i p_i}{\sigma})$, where p_i is the transmit power, h_i is the channel power gain, σ is the noise power at the CAP receiver, and b_i is the allocated bandwidth to user i and is constrained by the total available bandwidth as follows:

$$\sum_{i=1}^{N} b_i \leq B. \tag{3}$$

Energy consumed by user i for wireless transmission to the CAP is denoted by $E_i^a = p_i L_i / (b_i \log(1 + \frac{h_i p_i}{\sigma}))$. For notational convenience we define $\eta_i = \log(1 + \frac{h_i p_i}{\sigma})$.

Since there are possibly multiple tasks offloaded to the CAP, we need to further allocate computation resources available at the CAP as well. Denote f_i^a as the assigned CAP processing rate to user i, which is constrained by the total processing rate F^a at the CAP:

$$\sum_{i=1}^{N} f_i^a \leq F^a. \tag{4}$$

Initially, we consider a CAP model where F^a is concentrated in a single fast processor. In Section V, we will extend our consideration to the case where the CAP has multiple processors.

If the task is served by the CAP, the time delay is mainly contributed by wireless communication latency and processing time and is denoted by $T_i^a = \frac{L_i}{b_i \eta_i} + \frac{\omega_i}{f_i^a}$. Hence, considering shared and limited resources, careful joint allocation of communication and computation resources is necessary to improve user latency.

4) Cloud Processing Model: We further assume that the remote cloud center provides an essentially infinite number of processors, each with processor rate of f^c, possibly through leasing of virtual machines. If task i is offloaded to the remote cloud, besides the wireless communication delay, there is an additional transmission time between the CAP and cloud center, and the time required to complete the process at the cloud server. Therefore, the overall delay can be expressed as $T_i^c = \frac{L_i}{b_i \eta_i} + \frac{L_i}{r_i^{ac}} + \frac{\omega_i}{f^c}$, where r_i^{ac} is the rate allocated to user i to transmit its task over the limited-capacity backhaul link between the CAP and cloud. Let R^{ac} be the capacity of backhaul. We have

$$\sum_{i=1}^{N} r_i^{ac} \leq R^{ac}. \tag{5}$$

The consumed energy by user i in this case is due to wireless transmission and is denoted by $E_i^c = p_i L_i / b_i \eta_i$.

Note that although a vast number of servers are available at the cloud center, and each can be fully devoted to a user, the overall delay does not only depend on the task itself as multiple users compete to reach the cloud through a limited backhaul link.

B. Problem Formulation

The goal is to reduce the total system cost and maintain a desired level of performance for every potential realization of uncertainty. The cost incurred by each user is defined as the weighted sum of the user energy consumption and the task processing delay as follows:

$$C_i = E_i + \rho_i T_i, \tag{6}$$

where $E_i = E_i^l x_i^l + E_i^a x_i^a + E_i^c x_i^c$ and $T_i = T_i^l x_i^l + T_i^a x_i^a + T_i^c x_i^c$, and ρ_i is the relative weight of time delay compared with energy usage.

The overall system cost involves some uncertainty brought by unknown computation cycles required for task execution. Despite this uncertainty, statistical information provided by experimental studies [17] can be exploited to limit the risk of cost fluctuation. Reducing cost fluctuation is necessary to the maintenance of a certain desired system performance for

different realizations of uncertainty. To model cost fluctuation, we consider the effect of Δ_i^ω, as defined in Section III-A.1, on user i's range of cost variation, denoted by Δ_i^C. Since C_i is a linear function of ω_i, the relation between Δ_i^C and Δ_i^ω is also given by (6) as follows:

$$\Delta_i^C = \rho_i \left[x_i^l \frac{\Delta_i^\omega}{f_i^l} + x_i^a \frac{\Delta_i^\omega}{f_i^a} + x_i^c \frac{\Delta_i^\omega}{f^c} \right] + x_i^l E_i^l(\Delta_i^\omega, f_i^l). \quad (7)$$

As a special example, if we use $\Delta_i^\omega = \omega_i^{ub} - \omega_i^{lb}$, where ω_i^{ub} and ω_i^{lb} are the upper and lower bounds of ω_i, then Δ_i^C represents the maximum range of cost fluctuation.

Our objective is to minimize the expected cost as well as the cost fluctuation by jointly optimizing the task offloading decisions $\mathbf{x}_i = [x_i^l, x_i^a, x_i^c]$, and the resource allocation vector $\mathbf{r}_i = [f_i^l, f_i^a, b_i, r_i^{ac}]$. Hence, we consider the following optimization problem

$$\begin{aligned} \underset{\{\mathbf{x}_i, \mathbf{r}_i\}}{\text{minimize}} \quad & \mathbb{E}\left[\sum_{i=1}^N C_i\right] + \gamma \sum_{i=1}^N \Delta_i^C, && (8) \\ \text{subject to} \quad & (1), (2), (3), (4), (5), \\ & f_i^l, f_i^a, b_i, r_i^{ac} \geq 0, \ \forall i, && (9) \\ & x_i^l, x_i^a, x_i^c \in \{0, 1\}, \ \forall i, && (10) \end{aligned}$$

where $\mathbb{E}[.]$ is the expectation with respect to the required computing cycles $\{\omega_i\}$, and γ is the weight on stabilizing the system cost compared with the expected cost.

Optimization problem (8) is a mixed-integer program that is hard to solve in general. Moreover, even if we relax the binary constraint (1) so that the task offloading decision variables can vary within the range $[0, 1]$, problem (8) is non-convex due to its non-convex objective and constraints. Next, we propose the TORAUC algorithm, discuss its optimality, and further study its effectiveness in solving this problem.

IV. Task Offloading and Resource Allocation with Uncertain Computing Requirement (TORAUC)

The TORAUC algorithm belongs to the general framework of successive convex approximation (SCA). In particular, locally tight approximate monomials are introduced to upperbound the offloading decision constraints. The key is to carefully choose the approximation functions to guarantee the convergence of the algorithm to a KKT point with offloading decisions arbitrarily close to binary. We now present the details of TORAUC, concluding with a discussion on its convergence properties and computational complexity.

A. TORAUC Algorithm

TORAUC is comprised of three major steps: 1) Replace binary constraint with relaxed constraints that provide offloading decisions arbitrarily close to binary; 2) Find locally tight upperbound approximate monomial functions; 3) Iteratively form and solve a sequence of geometric programming (GP) problems optimally and update the approximate functions based on the previous solutions.

1) Alternate Problem Formulation: In order to reformulate problem (8) in the GP form, equality constraints must be monomials as in the standard GP format. Nevertheless, the offloading decision constraint in (1) cannot be directly written as a monomial. Moreover, the feasible set of problem (8) is not continuous as the offloading decisions can only take binary values. Therefore, we relax the offloading decisions and introduce the equivalent constraints (15)-(17) below, to ensure the satisfaction of offloading decision constraints (1) and (10) in the original problem. Furthermore, we move the energy and delay terms involved in the cost function (6), and cost variation Δ_i^C from the objective (8) to the constraints by introducing auxiliary variables $\{E_i, T_i, \delta_i\}$ to obtain the following problem:

$$\begin{aligned} \underset{\{\mathbf{x}_i, \mathbf{r}_i, E_i, T_i, \delta_i\}}{\text{minimize}} \quad & \sum_{i=1}^N [E_i + \rho_i T_i + \gamma \delta_i] && (11) \\ \text{subject to} \quad & (2) - (5), (9) \\ & x_i^l E_i^l(\overline{\omega}_i, f_i^l) + x_i^a \frac{p_i L_i}{b_i \eta_i} + x_i^c \frac{p_i L_i}{b_i \eta_i} \leq E_i, \forall i, && (12) \\ & x_i^l \frac{\overline{\omega}_i}{f_i^l} + x_i^a \left[\frac{L_i}{b_i \eta_i} + \frac{\overline{\omega}_i}{f_i^a} \right] \\ & \qquad + x_i^c \left[\frac{L_i}{b_i \eta_i} + \frac{L_i}{r_i^{ac}} + \frac{\overline{\omega}_i}{f^c} \right] \leq T_i, \forall i, && (13) \\ & \rho_i \left[x_i^l \frac{\Delta_i^\omega}{f_i^l} + x_i^a \frac{\Delta_i^\omega}{f_i^a} + x_i^c \frac{\Delta_i^\omega}{f^c} \right] \\ & \qquad + x_i^l E_i^l(\Delta_i^\omega, f_i^l) \leq \delta_i, \forall i, && (14) \\ & 0 \leq x_i^s \leq 1, \ \text{for } s = l, a, c, \ \forall i, && (15) \\ & M x_i^s x_i^t \leq 1, \ \text{for } s, t = l, a, c, \ s \neq t, \ \forall i, && (16) \\ & x_i^l + x_i^a + x_i^c \geq 1, \ \forall i, && (17) \end{aligned}$$

where M in (16) is a parameter chosen to be sufficiently large to have the multiplication of the decision variables arbitrarily close to zero. Constraint (16) ensures that for every user i, the decision tuple (x_i^l, x_i^a, x_i^c) contains at least two zero elements, and since each element is less than 1 by constraint (15) and they sum to at least 1 by constraint (17), the tuple also includes an offloading decision element with the value of 1. As $M \to \infty$, the set of constraints (15)-(17) are equivalent to constraints (1) and (10), and so the optimization problem (11) is equivalent to problem (8).

2) Monomial Approximate Functions: Toward the GP formulation, the constraint (17) needs to be further approximated as there is a posynomial on the right-hand side. Proper choice of approximation functions is of high importance as it directly affects the convergence property. The following lemma provides monomial approximation to a posynomial.

Lemma 1. Let h_j be arbitrary positive values, and β_j be positive constants that sum to one. We have

$$\sum_j h_j \geq \prod_j \left(\frac{h_j}{\beta_j} \right)^{\beta_j}.$$

The equality holds if $\beta_j = h_j / (\sum_j h_j)$, $\forall j$.

Proof. Let $v_j = h_j/\beta_j$. By Jensen's inequality we have

$$\log(\sum_i \beta_j v_j) \geq \sum_j \beta_j \log(v_j)$$
$$\Rightarrow \log(\sum_j h_j) \geq \sum_j \beta_j \log(\frac{h_j}{\beta_j})$$
$$\Rightarrow \sum_i h_j \geq \prod_j \left(\frac{h_j}{\beta_j}\right)^{\beta_j}. \quad (18)$$

The condition of equality is obvious. □

Using Lemma 1, we have the following upperbound monomial approximation for all i:

$$\frac{1}{x_i^l + x_i^a + x_i^c} \leq \frac{1}{\left(\frac{x_i^l}{\beta_i^l}\right)^{\beta_i^l}\left(\frac{x_i^a}{\beta_i^a}\right)^{\beta_i^a}\left(\frac{x_i^c}{\beta_i^c}\right)^{\beta_i^c}}, \quad (19)$$

where β_i^l, β_i^a, and β_i^c are arbitrary positive constants that sum to one. Thus, the following monomial inequalities provide a stronger condition than constraint (17).

$$\frac{1}{\left(\frac{x_i^l}{\beta_i^l}\right)^{\beta_i^l}\left(\frac{x_i^a}{\beta_i^a}\right)^{\beta_i^a}\left(\frac{x_i^c}{\beta_i^c}\right)^{\beta_i^c}} \leq 1, \quad \forall i. \quad (20)$$

3) GP Formulation and Update Rule: We replace the constraint (17) by the monomial constraints in (20). This leads to a GP formulation as follows:

$$\underset{\{\mathbf{x}_i,\mathbf{r}_i,\mathbf{E}_i,\mathbf{T}_i,\delta_i\}}{\text{minimize}} \quad \sum_{i=1}^{N}[E_i + \rho_i T_i + \gamma\delta_i] \quad (21)$$
$$\text{subject to} \quad (2)-(5),(9),(12)-(16) \text{ and } (20).$$

The optimization problem (21) is a standard GP that can be transformed into a convex program and solved efficiently and optimally [19].

In TORAUC, we iteratively update the approximate functions and solve a sequence of GP problems of the above form. In particular, by solving each GP, TORAUC tries to improve the accuracy of the approximations to a distinct minimum in the original feasible set of (8). We update parameters β_i^l, β_i^a, and β_i^c as follows:

$$\beta_i^s = \frac{x_i^{s*}}{x_i^{l*} + x_i^{a*} + x_i^{c*}}, \ s = l, a, c, \ \forall i, \quad (22)$$

where $\mathbf{x}_i^* = \{x_i^{l*}, x_i^{a*}, x_i^{c*}\}$ is the optimal decisions of the approximated problem (21) in the previous iteration. An exact description of TORAUC is given in Algorithm 1. We next show that TORAUC converges to a KKT point of problem (11).

B. TORAUC Convergence Properties

In this section, we prove that the proposed TORAUC algorithm converges to a KKT point of problem (11), with offloading decisions arbitrarily close to binary.

Algorithm 1 TORAUC algorithm

Input: An initial feasible offloading decision $\{\mathbf{x}_i\}$; error tolerance ϵ.
Output: Locally optimal offloading decision $\{\mathbf{x}_i\}$ and resource allocation $\{\mathbf{r}_i\}$.
1: Compute for each user i, β_i^l, β_i^a, and β_i^c using (22).
2: Solve the resulting GP problem (21) to find an optimal solution P^t.
3: Repeat from Step 1, and terminate if $||P^t - P^{t-1}|| \leq \epsilon$.

In particular, problem (11) belongs to the following family of non-convex programs

$$\underset{\mathbf{x}}{\text{minimize}} \quad f_0(\mathbf{x})$$
$$\text{subject to} \quad f_j(\mathbf{x}) \leq 1, j = 1, 2, \ldots, m,$$

where $f_0(\mathbf{x})$ is a convex function, and $f_j(\mathbf{x})$, $\forall j$, are non-convex. Denote convex approximate functions by $\tilde{f}_j(\mathbf{x}) \approx f_j(\mathbf{x})$. It has been shown that any SCA method guarantees to converge to a KKT point of the original if the following properties hold in each iteration [20]:
(P1) $f_j(\mathbf{x}) \leq \tilde{f}_j(\mathbf{x}), \ \forall \mathbf{x}$,
(P2) $f_j(\mathbf{x}^*) = \tilde{f}_j(\mathbf{x}^*)$, where $\mathbf{x}^*$ is the optimal solution of the approximated problem in the previous iteration,
(P3) $\nabla f_j(\mathbf{x}^*) = \nabla \tilde{f}_j(\mathbf{x}^*)$.
We next show that TORAUC satisfies all three properties.

Proposition 1. The TORAUC algorithm converges to a KKT point of optimization problem (11).

Proof. To ensure that TORAUC converges to a KKT point, it is sufficient to prove that the approximation functions introduced in (19) satisfy the three properties P1-P3.

(P1) In each iteration, since β_i^s, for $s = l, a, c$, are all positive and $\sum_s \beta_i^s = 1$ according to (22), we always have (19) by Lemma 1.

(P2) We further show that the equality in (19) holds at the optimal solution of the approximate problem. For $s = l, a, c$, according to (22), we have

$$\left(\frac{x_i^{s*}}{\beta_i^{s*}}\right)^{\beta_i^{s*}} = \left(x_i^{l*} + x_i^{a*} + x_i^{c*}\right)^{\frac{x_i^{s*}}{x_i^{l*}+x_i^{a*}+x_i^{c*}}}.$$

Combining the above for different cases of s yields

$$x_i^{l*} + x_i^{a*} + x_i^{c*} = \left(\frac{x_i^{l*}}{\beta_i^{l*}}\right)^{\beta_i^{l*}}\left(\frac{x_i^{a*}}{\beta_i^{a*}}\right)^{\beta_i^{a*}}\left(\frac{x_i^{c*}}{\beta_i^{c*}}\right)^{\beta_i^{c*}}. \quad (23)$$

(P3) First we note that the gradient of $x_i^l + x_i^a + x_i^c$ is an all-one vector. Let us define $F(\mathbf{x}_i) = \left(\frac{x_i^l}{\beta_i^l}\right)^{\beta_i^l}\left(\frac{x_i^a}{\beta_i^a}\right)^{\beta_i^a}\left(\frac{x_i^c}{\beta_i^c}\right)^{\beta_i^c}$. Then, for $s = l, a, c$,

$$\left.\frac{\partial F(\mathbf{x}_i)}{\partial x_i^s}\right|_{\mathbf{x}_i^*} = \left(\frac{x_i^{s*}}{\beta_i^{s*}}\right)^{-1}\left[\left(\frac{x_i^{l*}}{\beta_i^{l*}}\right)^{\beta_i^{l*}}\left(\frac{x_i^{a*}}{\beta_i^{a*}}\right)^{\beta_i^{a*}}\left(\frac{x_i^{c*}}{\beta_i^{c*}}\right)^{\beta_i^{c*}}\right].$$

Replacing β_i^s by the updates given in (22) and using (23) we have

$$\left.\frac{\partial F(\mathbf{x}_i)}{\partial x_i^s}\right|_{\mathbf{x}_i^*} = \left[x_i^{l*} + x_i^{a*} + x_i^{c*}\right]^{-1}\left[x_i^{l*} + x_i^{a*} + x_i^{c*}\right] = 1.$$

Hence, one can conclude that (P3) also holds. □

C. Computational Complexity Analysis

The GP problem (21) can be solved within ϵ precision by the interior point method in $O(\log(V/\epsilon)/\log(\mu))$ time, where V is the total number of constraints in problem (21), and μ is the updating accuracy parameter [19]. It is easy to show that $V = 11N+3$. Moreover, computations required for monomial approximations in (19) is $O(N)$. Therefore, the amount of computations in each iteration is $O(N\log(N/\epsilon)/\log(\mu))$. The number of iterations in the algorithm required to arrive at solution depends on the curvature of the objective function, which is difficult to quantify. In Section VI, we will present numerical results for the run time of TORAUC.

V. Extension to Multi-Processor CAP Scenario

In this section, we further study the scenario where instead of a single powerful processor, the CAP contains multiple processors, each can be shared among all users.

A. Multi-Processor CAP System Model

Having multiple processors at the CAP complicates the offloading decision. In fact, due to the task indivisibility, users need to further decide among the available processors at the CAP. Assume that the CAP contains K identical processors, denoted by $\mathcal{K}$, each with maximum speed $F_a^{\max}$. Denote by y_i^k the fraction of the CAP processor k computing rate that is assigned to user i, and let $\mathbf{y}_i = [y_i^1, \ldots, y_i^K]$. Task i can only be executed by CAP processors if the decision is to offload to the CAP. Thus,

$$\sum_{k=1}^{K} y_i^k \leq x_i^a, \quad \forall i. \tag{24}$$

Furthermore, the sum of the processing rate fractions assigned to users is less than 1

$$\sum_{i=1}^{N} y_i^k \leq 1, \quad \forall k. \tag{25}$$

Since the computation tasks are indivisible, each user i can only receive computation resources from one of the available processors at the CAP. Similar to (16), this is formulated as

$$M y_i^k y_i^q \leq 1, \ k, q \in \mathcal{K}, \ k \neq q, \ \forall i. \tag{26}$$

Note that (26) implies that for every user and for a sufficiently large M, only one of the processing rate fractions can be non-zero.

Furthermore, the CAP processing rate assigned to a user is limited to the total portion of computing resources allocated to the user, and the maximum computing capability of a CAP processor. Thus, we introduce an auxiliary variable z_i, representing the overall portion of a CAP processor assigned to user i, and we have

$$f_i^a \leq z_i F_a^{\max}, \quad \forall i, \tag{27}$$

$$z_i \leq \sum_{k=1}^{K} y_i^k, \quad \forall i. \tag{28}$$

Thus, we formulate the problem of joint offloading decision and resource allocation for a multi-processor CAP with unknown task computing cycle as

$$\begin{aligned} \underset{\{\mathbf{x}_i, \mathbf{r}_i, \mathbf{y}_i, \mathbf{z}_i\}}{\text{minimize}} \quad & \mathbb{E}\left[\sum_{i=1}^{N} C_i\right] + \gamma \sum_{i=1}^{N} \Delta_i^C, \\ \text{subject to} \quad & (1)-(5), (10), (24)-(28), \\ & f_i^l, f_i^a, b_i, r_i^{ac}, z_i, y_i^k \geq 0, \ \forall i, \forall k. \end{aligned} \tag{29}$$

Due to the additional CAP processors, problem (29) is more complicated than the original problem (8). If the tasks were divisible, a subset of the CAP processors could jointly serve the users' tasks, and problem (29) would be reduced to problem (8), where we only deal with the overall CAP processing rate assignment. Nevertheless, since computation tasks are indivisible, each user task can only be processed by one of the processors at the CAP. In fact, in problem (29) the processing rate of each user is constrained by the maximum speed of one CAP processor. Hence, the feasible set of (29) is smaller compared with that of (8), and the solution of (8) can serve as a lower bound to (29). To tackle this problem, in the following, we show how TORAUC can be extended for a multi-processor CAP.

B. TORAUC-MP Algorithm

Comparing optimization problem (29) with problem (8), we observe that they share a similar structure except that additional multi-processor CAP processing constraints (24)-(28) are imposed on the system. Following the procedure in Section IV, we first replace constraints (1) and (10) by constraints (15)-(17). We move the energy and delay and cost variation terms from the objective of (29) to the constraints by introducing auxiliary variables $\{E_i, T_i, \delta_i\}$ to form the following alternate problem:

$$\begin{aligned} \underset{\{\mathbf{x}_i, \mathbf{r}_i, \mathbf{y}_i, \mathbf{z}_i, \mathbf{E}_i, \mathbf{T}_i, \delta_i\}}{\text{minimize}} \quad & \sum_{i=1}^{N} \left[E_i + \rho_i T_i + \gamma \delta_i\right] \\ \text{subject to} \quad & (2), (3), (5), (12)-(17), \ (24)-(28). \end{aligned} \tag{30}$$

In order to cast the problem in the standard GP form, we need to further approximate constraints (17) and (28). Therefore, Lemma 1 is used again to provide a monomial upperbound approximation for (28) as

$$\frac{z_i}{\sum\limits_{k=1}^{K} y_i^k} \leq \frac{z_i}{\prod\limits_{k=1}^{K} \left(\frac{y_i^k}{\alpha_i^k}\right)^{\alpha_i^k}}, \tag{31}$$

where α_i^k are arbitrary positive constants such that $\sum_k \alpha_i^k = 1, \forall i$.

We replace constraint (28) by its upperbound monomial approximation given in (31). This leads to a GP formulation

Algorithm 2 TORAUC-MP algorithm

Input: An initial feasible offloading decision $\{\mathbf{x}_i\}$; An initial feasible CAP processing rate $\{\mathbf{y}_i\}$; vector error tolerance ϵ.

Output: Locally optimal offloading decision $\{\mathbf{x}_i\}$ and resource allocation $\{\mathbf{y}_i, \mathbf{r}_i\}$.

1: Compute for each user i, β_i^l, β_i^a, and β_i^c using (22).

2: Compute for each user i, α_i^k, $\forall k \in \mathcal{K}$, using (34).

3: Solve the resulting GP problem (21) to find an optimal solution P^t.

4: Repeat from Step 1, and terminate if $||P^t - P^{t-1}|| \leq \epsilon$.

as follows:

$$\underset{\{\mathbf{x}_i,\mathbf{r}_i,\mathbf{y}_i,\mathbf{z}_i,\mathbf{E}_i,\mathbf{T}_i,\delta_i\}}{\text{minimize}} \quad \sum_{i=1}^{N} \left[E_i + \rho_i T_i + \gamma \delta_i\right] \tag{32}$$

$$\text{subject to} \quad \frac{z_i}{\prod_{k=1}^{K} \left(\frac{y_i^k}{\alpha_i^k}\right)^{\alpha_i^k}} \leq 1, \forall i, \tag{33}$$

$$(2), (3), (5), (12) - (16), (20), (24) - (27),$$

where constraint (33) comes from (28) and (31). TORAUC-MP iteratively solves a series of GPs given by problem (32) and the solution at each step is leveraged to better approximate the constraints (20) and (33) for the next round. In particular, similar to $\{\beta_i^s\}$ updates in (22), α_i^k updates are given by

$$\alpha_i^k = \frac{y_i^{k^*}}{\sum_q y_i^{q^*}}, \ \forall i, \tag{34}$$

where $\{y_i^{k^*}\}$ is the optimal solution at the previous iteration. The details of TORAUC-MP are given in Algorithm 2. Similar to TORAUC, TORAUC-MP runs in $O(N \log(V/\epsilon)/\log(\mu))$, where V is the number of constraints. Note that in this case $V = \frac{NK(K-1)}{2} + 14N + K + 2$.

Proposition 2. The TORAUC-MP algorithm converges to a KKT point of the optimization problem (30).

Proof. It is sufficient to show that the approximations in (20) and (33) meet properties P1-P3. It is shown in the proof of Proposotion 1 that (20) satisfies the three properties. The structure of the upperbound monomial approximate functions in (33) is similar to (20). Thus, it can be shown similarly (33) satisfies these properties P1-P3. □

VI. Trace-Driven Simulation Results

We investigate the performance of TORAUC via extensive simulations over a generic cloud access network, using Google cluster traces [6].

A. Simulation Setup

We use Google cluster usage traces to extract the users' required computing cycles. To account for the relationship between the task computing cycle and data size, we follow the method in [21] and assume $\omega_i f_c = \Gamma L_i$, where Γ is a random variable with Gamma distribution with parameters $\alpha = 4$ and $\beta = 200$. We consider the full range of possible ω_i values, i.e., $\Delta_i^\omega = \omega_i^{ub} - \omega_i^{lb}$.

The mobile device's processor speed is set to 1.2×10^9 cycles/s. The device energy consumption is modeled by $E_i^l(\omega_i, f_i^l) = \alpha_l \omega_i f_i^{l^{\kappa_i}}$ with parameter values $\alpha_l = 10^{-27}$ and $\kappa_i = 2.3$, as given in [18]. In addition, the total wireless bandwidth between mobile users and the CAP is set to 40 MHz. Processing rate assigned to each user at the remote cloud is 10×10^9 cycles/s, and the overall CAP processing speed of 10×10^9 cycles/s is considered. In the multi-processor scenario, the overall computing rate is equally divided among the CAP processors.

We further compare the performance of TORAUC with the following alternatives: 1) *Random mapping*, where each task is served on a processor chosen with equal probability; and 2) *Pessimistic scheme*, which conservatively minimizes the maximum cost using w_i^{ub} as the required computing cycle; 3) *Optimistic scheme*, which optimistically minimizes the minimum cost using ω_i^{lb} as the required computing cycle; and 4) *Optimal policy*, where the optimal binary decisions are obtained by exhaustive search.

B. Single CAP Processor

In Fig. 1-3, we study the effect of various parameters on the performance of TORAUC in terms of average cost and cost variation. We set by default $\gamma = 0.1$, $F^a = 10^{10}$ (cycles/s), and $N = 6$, but vary each of them in different figures. The optimal policy is obtained by exhaustive search over the decision space, so it has an exponential computational complexity, i.e., $O(3^N)$. Therefore, we only compute the optimal policy for up to 8 users as the required run time becomes excessively high for cases beyond 8 users.

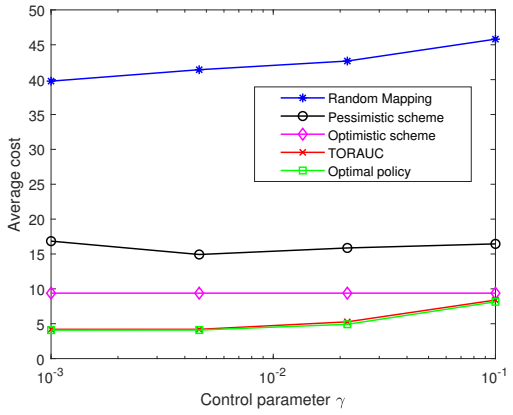
We observe that TORAUC significantly outperforms all alternatives, and is nearly optimal over a wide range of parameter values. From Fig 1b, we see that naive algorithms such as the optimistic scheme and random mapping perform poorly. A more conservative pessimistic scheme performs better than the other naive algorithms. However, since it only targets the worst-case cost, it is ineffective in reducing the average cost or cost variation. In comparison, within the range of parameter values under consideration, TORAUC incurs up to 15% lower cost variation and three times lower average cost.

C. Multiple CAP Processors

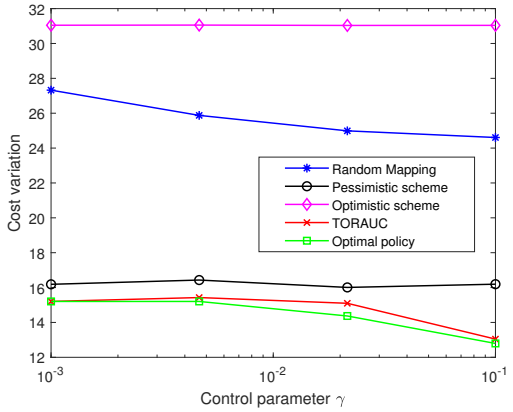
In Fig. 4 we further study the performance of TORAUC-MP, where we assume that there are $K = 4$ processors at the CAP. In this case, we are unable to compute the optimal policy because of the drastically increased decision space. We see that TORAUC-MP remains favorable against the other alternatives. We also observe the tradeoff between the average cost and the cost variation as we vary γ.

D. Computation Time Complexity

In Table I, we show the run time of TORAUC for $K = 1$ and TORAUC-MP for $K = 4$, in comparison with exhaustive

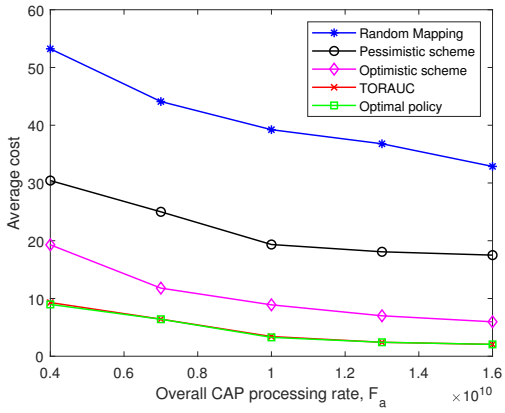


(a) Effect of control parameter γ.

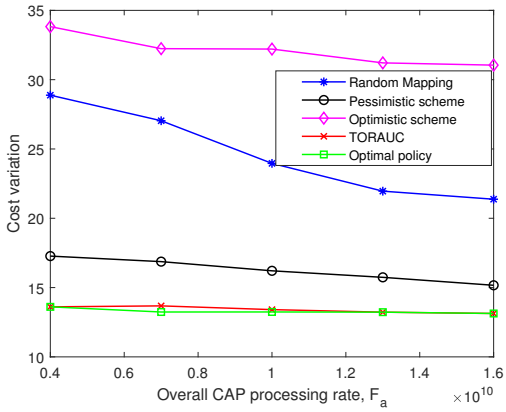


(b) Effect of control parameter γ.

Fig. 1: TORAUC performance vs. control parameter γ



(a) Effect of CAP computing rate F^a (cycles/s).



(b) Effect of CAP computing rate F^a (cycles/s).

Fig. 2: TORAUC performance versus the CAP rate F^a.

TABLE I: Average run time (seconds) comparison

No. of users	TORAUC	TORAUC-MP	Optimal Policy (K=1)
4	10.32	16.59	71.32
6	14.98	23.56	392.77
8	21.85	32.26	2139.62
12	29.79	49.36	N/A
20	53.09	95.17	N/A
30	93.56	171.83	N/A

searching for an optimal policy for $K = 1$. They are obtained on a computer with Intel core i7-85500 4.0 GHz processor and 16 GB RAM. This table suggests that both proposed algorithms have nearly linear run time in the number of users, despite the exponential search space to find an optimal policy. Furthermore, the substantial run time improvement of TORAUC is achieved with negligible cost penalty. Combining this with the complexity analysis in Section IV-C and Section V-B, we see that both algorithms converge quickly and scale well with respect to the number of users.

VII. Conclusion

We have studied joint offloading and allocation of computation and communication resources with unknown task computing requirement in a three-tier computing system. The objective is to minimize a weighted sum of expected cost and cost variation. We propose efficient algorithms TORAUC, for when there is a single processor at the CAP, and TORAUC-MP, for when there are multiple processors at the CAP. They both converge to a KKT point of an approximate problem that is arbitrarily close to the original problem. Through trace-driven simulation, we show that TORAUC gives nearly optimal performance, and both algorithms outperform existing alternatives.

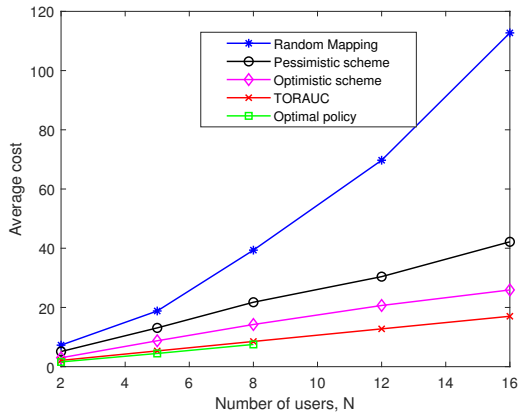
References

[1] Y. Mao, C. You, J. Zhang, K. Huang, and K. B. Letaief, "A survey on mobile edge computing: The communication perspective," *IEEE Communications Surveys & Tutorials*, vol. 19, no. 4, pp. 2322–2358, August 2017.

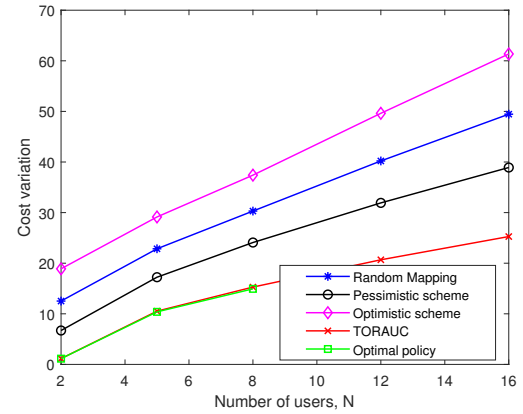
[2] B. Liang, "Mobile Edge Computing," in *Key Technologies for 5G Wireless Systems*, V.W.S Wong, R. Schober, D.W.K Ng, and L.-C. Wang, Cambridge University Press, 2017.

[3] M.-H. Chen, M. Dong, and B. Liang, "Resource sharing of a computing access point for multi-user mobile cloud offloading with delay constraints," *IEEE Transactions on Mobile Computing*, vol. 17, no. 12, pp. 2868–2881, December 2018.

[4] D. B. Shmoys, J. Wein, and D. P. Williamson, "Scheduling parallel machines on-line," *SIAM Journal on Computing*, vol. 24, no. 6, pp. 1313–1331, December 1995.

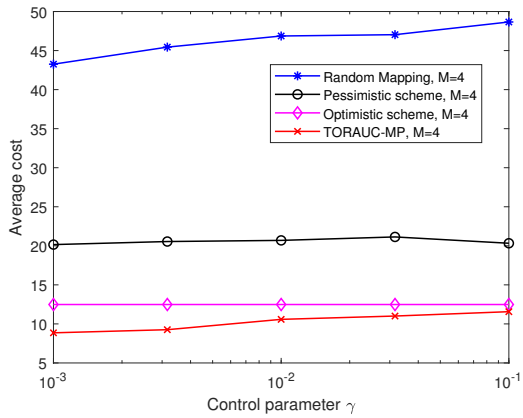


(a) Effect of number of users N.

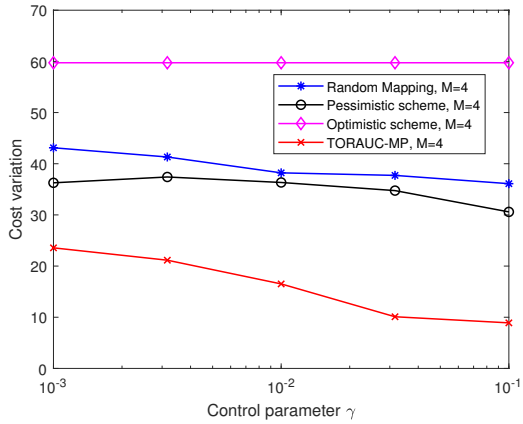


(b) Effect of number of users N.

Fig. 3: TORAUC performance vs. the number of users N.



(a) Effect of control parameter γ.



(b) Effect of control parameter γ.

Fig. 4: TORAUC-MP performance vs. the control parameter γ.

[5] S. Ali, A. A. Maciejewski, H. J. Siegel, and J.-K. Kim, "Measuring the robustness of a resource allocation," *IEEE Transactions on Parallel and Distributed Systems*, vol. 15, no. 7, pp. 630–641, June 2004.

[6] J. Wilkes, "More Google cluster data," Google research blog, November 2011, posted at http://googleresearch.blogspot.com/2011/11/more-googlecluster-data.html.

[7] J. Du, L. Zhao, J. Feng, and X. Chu, "Computation offloading and resource allocation in mixed fog/cloud computing systems with min-max fairness guarantee," *IEEE Transactions on Communications*, vol. 66, pp. 1594–1608, April 2018.

[8] M.-H. Chen, B. Liang, and M. Dong, "Joint offloading and resource allocation for computation and communication in mobile cloud with computing access point," in *Proc. IEEE Conference on Computer Communications (INFOCOM)*, May 2017.

[9] M. Chen, B. Liang, and M. Dong, "Multi-user multi-task offloading and resource allocation in mobile cloud systems," *IEEE Transactions on Wireless Communications*, vol. 17, no. 10, pp. 6790–6805, October 2018.

[10] T. Q. Dinh, J. Tang, Q. D. La, and T. Q. Quek, "Offloading in mobile edge computing: Task allocation and computational frequency scaling," *IEEE Transactions on Communications*, vol. 65, no. 8, pp. 3571–3584, April 2017.

[11] W. Yuan and K. Nahrstedt, "Energy-efficient soft real-time CPU scheduling for mobile multimedia systems," *ACM Operating Systems Review (SIGOPS)*, vol. 37, no. 5, pp. 149–163, October 2003.

[12] Z. Shi, E. Jeannot, and J. J. Dongarra, "Robust task scheduling in non-deterministic heterogeneous computing systems," in *Proc. IEEE International Conference on Cluster Computing*, September 2006.

[13] W. Naji, V.-D. Cung, and M.-L. Espinouse, "Robust preemptive scheduling on unrelated parallel machines under uncertain processing times," in *Proc. IEEE International Conference on Control, Decision and Information Technologies (CoDIT)*, April 2017.

[14] J. P. Champati and B. Liang, "Single restart with time stamps for computational offloading in a semi-online setting," in *Proc. of IEEE International Conference on Computer Communications (INFOCOM)*, May 2017.

[15] J. P. Champati and B. Liang, "Semi-online algorithms for computational task offloading with communication delay," *IEEE Transactions on Parallel and Distributed Systems*, vol. 28, no. 4, pp. 1189–1201, April 2017.

[16] D. Karunakaran, Y. Mei, G. Chen, and M. Zhang, "Toward evolving dispatching rules for dynamic job shop scheduling under uncertainty," in *Proc. Genetic and Evolutionary Computation Conference*, July 2017.

[17] A. P. Miettinen and J. K. Nurminen, "Energy efficiency of mobile clients in cloud computing," in *Proc. of USENIX Conference on Hot Topics in Cloud Computing*, June 2010.

[18] J. Kwak, Y. Kim, J. Lee, and S. Chong, "Dream: Dynamic resource and task allocation for energy minimization in mobile cloud systems," *IEEE Journal on Selected Areas in Communications*, vol. 33, no. 12, pp. 2510–2523, September 2015.

[19] S. Boyd and L. Vandenberghe, *Convex Optimization*. Cambridge Univ. Press, 2004.

[20] M. Chiang, "Geometric programming for communication systems," *Foundations and Trends in Communications and Information Theory*, vol. 2, no. 1–2, pp. 1–154, 2005.

[21] W. Zhang, Y. Wen, K. Guan, D. Kilper, H. Luo, and D. O. Wu, "Energy-optimal mobile cloud computing under stochastic wireless channel," *IEEE Transactions on Wireless Communications*, vol. 12, no. 9, pp. 4569–4581, August 2013.