I/O Interference Aware Multiuser Computation Offloading for Virtualized Edge Computing

Zezu Liang*, Yuan Liu†, Kaibin Huang‡, and Tat-Ming Lok*

*Department of Information Engineering, The Chinese University of Hong Kong, Shatin, Hong Kong
†School of Electronic and Information Engineering, South China University of Technology, Guangzhou 510641, China
‡Department of Electrical and Electronic Engineering, The University of Hong Kong, Hong Kong
Emails: {lz017, tmlok}@ie.cuhk.edu.hk, eeyliu@scut.edu.cn, huangkb@eee.hku.hk

***Abstract*—Mobile-edge computing (MEC) is an emerging technology for enhancing the computational capabilities of mobile devices and reducing their energy consumption via offloading complex computation tasks to the nearby servers. Multiuser MEC at servers is widely realized via parallel computing based on virtualization. Due to finite shared I/O resources, interference between virtual machines (VMs), called I/O interference, arises that degrades the computation performance. In this paper, we study the problem of joint radio-and-computation resource allocation (RCRA) in multiuser MEC systems in the presence of I/O interference. Specifically, we formulate a sum offloading rate maximization problem by joint offloading-user scheduling, the offloaded size control, and time allocation for communication (offloading and downloading) and computation. The problem is a non-convex mixed-integer programming problem. An optimal algorithm with low complexity is designed based on a decomposition approach and Dinkelbach method. The simulation results demonstrate considering of I/O interference can endow on an offloading controller robustness against the performance-degradation factor.**

I. Introduction

Recently, mobile-edge computing (MEC), which provides users computing services at the network edge, is envisioned as a promising way to enable computation-intensive and latency-sensitive mobile applications. Compared with cloud computing, users in MEC systems offload tasks to the proximate edge servers [e.g., base stations (BSs) and access points (APs)] for execution, which avoids data delivery over the backhaul networks and thereby dramatically reduces latency [1].

An essential technology for implementing MEC is virtualization, referring to sharing of a physical machine (server) by multiple computing processes via the execution of virtual machines (VMs). Specifically, each VM is a virtual computer configured with a certain amount of the server's hardware resource (such as CPU, memory and I/O bus). According to technical standards for the MEC server architecture [2], the virtualization functionality is supported by a virtualization layer and a virtualization manager. The virtualization layer virtualizes the MEC hosting infrastructure by abstracting the detailed hardware implementation, while the virtualization manager provides the virtualized computer infrastructure as a service (IaaS). Applications run on top of an IaaS and are deployed within the packaged-operating systems (i.e.,VMs) that are well-isolated with the others. To this end, the MEC server can isolate co-hosted applications and provide multi-service support. Nevertheless, it has been shown in the literature [3]–[5] that sharing the same physical platform can incur the so-called I/O interference among VMs, resulting in a certain degree of computation-speed reduction for each VM. As far as we know, prior research of this issue focuses on the interference modeling [4]–[6] and computation resource provisioning [7]. No previous works related to the computation offloading coupled with joint radio-and-computational resource allocation (RCRA) have been studied before.

In this paper, we investigate the multiuser offloading problem in an MEC system in the presence of I/O interference. Although the joint optimization of RCRA has been studied in various scenarios, e.g., [8]–[13], only the works in [13], [14] are closely related to this paper, as they both address parallel computation at a MEC server for joint RCRA. However, simultaneous computation processes at the same server are assumed in [13], [14] to be independent and conditioned on partitioned computation resources. The effect of I/O interference is neglected despite its being an importance issue in virtualization.

Omitting I/O interference in multiuser MEC based on virtualization leads to the unrealistic assumption that the total computation resource at a server remains fixed regardless of the number of VMs. In reality, the resource reduces as the number grows due to I/O interference. Thus, the number of VMs per server is usually constrained in practice, so as to maintain the efficiency in resource utilization. Despite its importance, I/O interference has received little attention in the literature. It motivates the current work on accounting for the factor in resource allocation for MEC systems.

In this paper, we revisit the RCRA problem in multiuser offloading and address the following two practical issues that have received scant attention in the literature.

1) (I/O interference) The I/O interference in practical parallel computing has been largely neglected in the existing "cake-slicing" model of computing-resource allocation (see e.g., [13], [14]). Considering I/O interference introduces a *dilemma*: scheduling more offloading users

This work was supported in part by the Natural Science Foundation of China under Grant U1701265 and the Pearl River Science and Technology Nova Program of Guangzhou under Grant 201710010111.

978-1-5386-8088-9/19/$31.00 ©2019 IEEE

increases the multiplexing gain in parallel computing but degrades the speeds of individual VMs due to their interference.

2) (Result downloading) The communication overhead for computation-result downloading is commonly assumed in the literature to be negligible compared with that for offloading. The assumption does not always hold in applications such as augmented reality and image processing. Considering downloading complicates scheduling as the policy needs account for not only users' uplink channel states but also downlink states as well as the output-input-size ratio for each task.

In this paper, we consider a multiuser MEC system where parallel computing at the server is based on virtualization. The I/O interference is modelled using a practical model developed based on measurement data [6]. While the literature focuses on offloading latency, we consider offloading, parallel computing and downloading as factors contributing to latency. Based on the assumptions, scheduling policies are designed by solving a RCRA problem for *sum offloading rate maximization* under the latency constraint. The main contributions are summarized as follows: We jointly optimize the offloading-user scheduling, offloaded-data sizes, and communication-and-computation time division. By analyzing its properties, we present a solution approach of decomposing the problem into master and slave sub-problems. The former optimizes the number of offloading users and given the number, the later optimizes offloading-user set, offloaded-data sizes, and time division (offloading, computing, downloading). By adopting Dinkelbach method, an efficient iterative algorithm is designed to solve the slave problem that is a *combinatorial-optimization* problem. With the algorithm, the master problem can be then solved by a simple search over a finite integer set of possible numbers of offloading users.

The rest of this paper is organized as follows. In Section II, we present our system model and problem formulation. In Section III, we propose an optimal algorithm to solve the problem of sum offloading rate maximization. Finally, simulation results and conclusions are provided in Section IV and Section V, respectively.

II. System Model and Problem Formulation

A. System Model

Consider an MEC system shown in Fig. 1, consisting of one AP integrated with an MEC server and K users. Partial offloading is assumed in this paper so that each user can partition its computation task into two independent parts for local computing and offloading to the server. The two operations are simultaneous assuming that the communication modulars and user CPUs are separated. All of the users have to complete their tasks within a fixed duration T (in second) so as to meet a real-time requirement. The system operation is divided into three *sequential phases*: 1) TDMA-based task offloading by users, 2) parallel computing at the server, and 3) TDMA-based computation-result downloading from the server

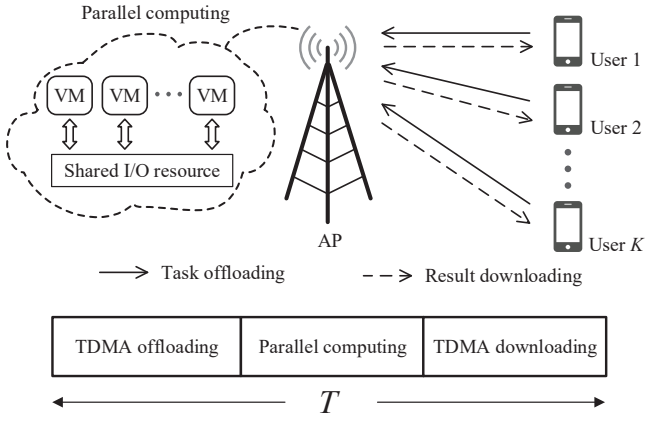

Figure 1: A multiuser MEC system comprising a single AP and K users.

to users. Corresponding models and assumptions are described as follows.

1) Offloading and Downloading Phases: Let ℓ_i denote the input data bits offloaded by user i to the server. It is assumed that each input bit generated γ_i bits of computation result. Then for an offloaded data ℓ_i, the computed result contains $\gamma_i \ell_i$ bits. The transmission delay for user i for offloading and downloading can be written separately as

$$t_i^u = a_i \ell_i, \tag{1}$$

$$t_i^d = b_i \gamma_i \ell_i, \tag{2}$$

where a_i and b_i are the required time for transmitting a single bit in uplink and downlink, respectively, which are the inverse of the corresponding uplink and downlink rates.

2) Parallel-Computing Phase with Virtualization: After receiving all the offloaded tasks, the server executes them in parallel by creating multiple VMs. We consider the important factor of I/O interference in parallel computing [15] and adopt a model developed in the literature based on measurement data [1], [6], which is described as follows. Group the user indices into the set $\mathcal{K}$. The subset $\mathcal{S} \subseteq \mathcal{K}$ identifies the set of scheduled offloading users, t_e the time allocated to the parallel-computing phase, and r_i the expected computation-service rate (bits/sec) of a VM given task i when running in isolation. Following [1], [6], a performance degradation factor $d > 0$ is defined to specify the percentage reduction in the computation-service rate of a VM when multiplexed with another VM. [1] Suppose that one VM is created and assigned to a task, the degraded computing rate for each task is modeled as $r_i(1+d)^{1-|\mathcal{S}|}$ [6], where $|\mathcal{S}|$ denotes the number of tasks (or offloading users) for parallel computing. Therefore, for given t_e, the numbers of offloadable bits are constrained by

$$0 \le \ell_i \le t_e r_i (1+d)^{1-|\mathcal{S}|}, \quad \forall i \in \mathcal{S}. \tag{3}$$

[1]The parameter d depends on the specific VM multiplexing and placement strategy [16], [17]. Its value can be estimated by theoretical studies or statistical observations.

The constraints in (3) show that the maximum number of offloadable bits per user decreases with the number of offloaded tasks due to the I/O interference in parallel computing. Moreover, relaxing the duration for parallel computing (t_e) can accommodate more offloaded bits ($\{\ell_i\}$), however, at the cost of less time for the offloading and downloading phases. This introduces a tradeoff between the three phases under the following total-latency constraint:

$$\sum_{i\in\mathcal{S}} t_i^u + t_e + \sum_{i\in\mathcal{S}} t_i^d = \sum_{i\in\mathcal{S}} \ell_i \left(a_i + b_i\gamma_i\right) + t_e \le T. \quad (4)$$

B. Problem Formulation

In this paper, we aim at maximizing the weighted sum of the users' offloading rates under the latency constraint, by joint scheduling, offloaded-bits control, and three-phase time allocation. Here, the sum offloading rate is defined as the sum offloadable bits over the time duration T. Let ω_i denote a positive weight assigned to user i based on the users' priority. Mathematically, the optimization problem can be formulated as

$$(\text{P1}): \max_{\mathcal{S}\subseteq\mathcal{K},\{\ell_i\},t_e} R = \frac{1}{T}\sum_{i\in\mathcal{S}} \omega_i \ell_i \quad (5a)$$

$$\text{s.t.} \sum_{i\in\mathcal{S}} \ell_i \left(a_i + b_i\gamma_i\right) + t_e \le T, \quad (5b)$$

$$0 \le \ell_i \le t_e r_i \left(1+d\right)^{1-|\mathcal{S}|}, \quad \forall i \in \mathcal{S}. \quad (5c)$$

Problem (P1) is a mixed-integer programming problem comprising both a combinatorial variable $\mathcal{S}$ and continuous variables $(\{\ell_i\}, t_e)$ and non-convex constraints (5c). Therefore, Problem (P1) is *non-convex*. Though such a problem is usually difficult to solve exactly, an algorithm is designed in sequel to find the optimal solution.

III. Sum Offloading Rate Maximization

In this section, we develop an optimal algorithm for solving Problem (P1). First, an important property of the optimal offloading-user set $\mathcal{S}^*$ will be obtained, which allows tractable analysis of the optimal offloading scheme and thereby simplifies the problem. Subsequently, an iterative algorithm based on the Dinkelbach method is proposed to exactly solve the simplified problem.

A. Sum Offloading Rate Maximization for a Given Offloading-User Set

We made a key observation that Problem (P1) becomes a linear programming (LP) problem if the offloading-user set is given. The conditional optimal offloading strategy, specified by the offloaded-data sizes $\{\ell_i^*\}$, satisfies the following property.

Lemma 1. *Given an arbitrary offloading-user set $\mathcal{S}$, the optimal offloading strategy $\{\ell_i^* \mid i \in \mathcal{S}\}$ must be the maximum or minimum value in the constraint in* (5c)*.*

The proof is presented in the longer version of this paper [18]. Lemma 1 indicates that the optimal offloading strategy of each scheduled user follows a binary policy, i.e., offloading with the maximum data size or nothing. Accordingly, we can divide the elements in $\{\ell_i \mid i \in \mathcal{S}\}$ into two groups, with one group $\widetilde{\mathcal{S}}$ for the users offloading maximum bits and the other performing no offloading, i.e.,

$$\ell_i = \begin{cases} t_e r_i \left(1+d\right)^{1-|\mathcal{S}|}, & i \in \widetilde{\mathcal{S}}, \\ 0, & i \in \mathcal{S}\backslash\widetilde{\mathcal{S}}. \end{cases} \quad (6)$$

Note that $\widetilde{\mathcal{S}}$ is needed to be determined and we first use $\widetilde{\mathcal{S}}$ to express $\{\ell_i\}$ and t_e. It is intuitive that the equality must hold in constraint (5b) for the optimal solution of Problem (P1). Then, by substituting (6) into (5b), we obtain the conditional optimal parallel-computing time t_e as

$$t_e = \frac{T(1+d)^{|\mathcal{S}|-1}}{(1+d)^{|\mathcal{S}|-1} + \sum_{i\in\widetilde{\mathcal{S}}}(a_i + b_i\gamma_i)r_i}. \quad (7)$$

Combining (6) and (7), Problem (P1) for a given $\mathcal{S}$ can be formulated as one for determining the subset $\widetilde{\mathcal{S}}$ in $\mathcal{S}$:

$$\max_{\widetilde{\mathcal{S}}\subseteq\mathcal{S}} \quad R = \frac{\sum_{i\in\widetilde{\mathcal{S}}}\omega_i r_i}{(1+d)^{|\mathcal{S}|-1} + \sum_{i\in\widetilde{\mathcal{S}}}(a_i + b_i\gamma_i)r_i}. \quad (8)$$

It is observed that, for any given $\mathcal{S}$, if $\widetilde{\mathcal{S}}^* \neq \mathcal{S}$, R in (8) can be further improved via replacing $\mathcal{S}$ with the smaller subset $\widetilde{\mathcal{S}}^*$. In other words, there exists the users $i \in \mathcal{S}\backslash\widetilde{\mathcal{S}}^*$ who offload zero bits but are scheduled to unnecessarily create a VM at the server, resulting in waste of resources. Thereby, removing them from the offloading-user set and only allocating VMs to the full offloading users can further increase the sum offloading rate. By the above argument, the necessary condition for $\mathcal{S}$ being optimal of Problem (P1) is $\widetilde{\mathcal{S}}^* = \mathcal{S}$ in Problem (8). That is, all the scheduled users offload their maximum bits, or otherwise the given $\mathcal{S}$ is not optimal.

Therefore, we re-define R as the sum offloading rate achieved by $\widetilde{\mathcal{S}} = \mathcal{S}$ in Problem (8), i.e.,

$$R = \frac{\sum_{i\in\mathcal{S}}\omega_i r_i}{(1+d)^{|\mathcal{S}|-1} + \sum_{i\in\mathcal{S}}(a_i + b_i\gamma_i)r_i}. \quad (9)$$

We have the following proposition to identify whether $\mathcal{S}$ meets the necessarily optimal condition.

Proposition 1. *$\widetilde{\mathcal{S}} = \mathcal{S}$ is the optimal solution of Problem* (8) *if and only if the given offloading-user set $\mathcal{S}$ satisfies*

$$R \le \min_{i\in\mathcal{S}} \left\{ \frac{\omega_i}{a_i + b_i\gamma_i} \right\}. \quad (10)$$

Due to limited space, we provide the detailed derivation in [18]. To better understand (10), we multiply the term $t_e(1+d)^{1-|\mathcal{S}|}$ in both the numerator and denominator of R and using the result that $\ell_i = t_e r_i (1+d)^{1-|\mathcal{S}|}$, $\forall i \in \mathcal{S}$, then R in (9) can be rewritten as

$$R = \frac{\sum_{i\in\mathcal{S}}\omega_i \ell_i}{t_e + \sum_{i\in\mathcal{S}}(a_i + b_i\gamma_i)\ell_i}. \quad (11)$$

The numerator in (11) denotes the weighted sum offloaded bits and the denominator denotes the total time and equals T. Then, R in (11) can be physically interpreted as the weighted sum-offloading rate of the system with users of set $\mathcal{S}$. On the

other hand, $\frac{\omega_i}{a_i+b_i\gamma_i}$ can be rewritten as $\frac{\omega_i\ell_i}{(a_i+b_i\gamma_i)\ell_i}$, where the numerator denotes the weighted amount of offloaded bits of user i while the denominator denotes the transmission duration that includes both offloading and downloading time. Therefore, $\frac{\omega_i}{a_i+b_i\gamma_i}$ can be regarded as the weighted transmission rate of user i. Proposition 1 implies that the system offloading rate should be less than or equal to the minimum transmission rate among users in $\mathcal{S}$ if it solves Problem (P1).

B. Offloading User Scheduling

Building on the results from the last subsection, we present in this subsection an efficient scheduling algorithm for computing the optimal offloading-user set. To this end, the variables $\{\ell_i\}$ and t_e can be expressed in term of $\mathcal{S}$ when $\mathcal{S}$ meets the necessarily optimal condition (10). This simplifies Problem (P1) as a scheduling problem that finds the optimal offloading-user set under constraint (10):

$$\begin{aligned} \max_{\mathcal{S}\subseteq\mathcal{K}} \quad & R = \frac{\sum_{i\in\mathcal{S}}\omega_i r_i}{(1+d)^{|\mathcal{S}|-1}+\sum_{i\in\mathcal{S}}(a_i+b_i\gamma_i)r_i} \\ \text{s.t.} \quad & (10). \end{aligned} \tag{12}$$

The problem can be further reduced to an unconstrained optimization problem using the following useful result.

Proposition 2. *Constraint* (10) *can be removed from Problem* (12) *without loss of optimality.*

The proof is presented in [18] due to the space limitation. Using Proposition 2, Problem (12) can be safely relaxed into the following non-constrained optimization problem:

$$\max_{\mathcal{S}\subseteq\mathcal{K}} \quad R = \frac{\sum_{i\in\mathcal{S}}\omega_i r_i}{(1+d)^{|\mathcal{S}|-1}+\sum_{i\in\mathcal{S}}(a_i+b_i\gamma_i)r_i}. \tag{13}$$

However, with the non-convex term $(1+d)^{|\mathcal{S}|-1}$ in the denominator of R, Problem (13) is still challenging to solve. To tackle this difficulty, we fix $|\mathcal{S}|=m$, with $m=1,\cdots,K$. For a given m, since term $(1+d)^{|\mathcal{S}|-1}$ becomes a constant, Problem (13) is reduced to a *mixed-integer linear fractional programming* problem. We solve Problem (13) by decomposing it into master-and-slave problems without loss of the optimality. The slave problem is determining the optimal offloading-user set using the Dinkelbach method [19] for a given number of scheduled users m. Then the master problem is obtaining the optimal value of m, denoted as m^*, by a simple search. The detailed solutions of the decomposed problems are presented in the sequel, which yield Algorithm 1 for computing the optimal scheduled-user set $\mathcal{S}^*$.

1) Optimal scheduling for a given number of scheduled users: In this section, we solve Problem (P1) conditioned on a given number of offloading users m, i.e., $|\mathcal{S}|=m$. To this end, we introduce a set of binary variables $\mathbf{x}=[x_1,\cdots,x_K]$, where $x_i=1$ means that user i is scheduled (i.e., $i\in\mathcal{S}$), and $x_i=0$ otherwise. Then, using the binary variables and conditioned on $|\mathcal{S}|=m$, Problem (P1) can be transformed into a combinatorial optimization problem as

(Slave Problem)

$$\begin{aligned} \max_{\mathbf{x}} \quad & R_m = \frac{\sum_{i=1}^K x_i\omega_i r_i}{(1+d)^{m-1}+\sum_{i=1}^K x_i(a_i+b_i\gamma_i)r_i} = \frac{N(\mathbf{x})}{D(\mathbf{x})} \\ \text{s.t.} \quad & \sum_{i=1}^K x_i = m, \quad x_i\in\{0,1\}, \quad i=1,\cdots,K, \end{aligned} \tag{14}$$

where $N(\mathbf{x}) \triangleq \sum_{i=1}^K x_i\omega_i r_i$ and $D(\mathbf{x}) \triangleq (1+d)^{m-1}+\sum_{i=1}^K x_i(a_i+b_i\gamma_i)r_i$. Let R_m^* denotes the maximum conditional sum offloading rate from solving the slave problem. For ease of notation, we define the feasible set for Problem (14) as $\mathcal{F}_m \triangleq \{\mathbf{x}|\sum_{i=1}^K x_i = m \text{ and } x_i\in\{0,1\}, i=1,\cdots,K\}$. Since the objective function has a fractional form, the problem can be solved by *non-linear fractional programming*. To this end, define a function $g(\cdot)$ of the conditional rate R_m by an optimization problem in a substrative form:

$$g(R_m) = \max_{\mathbf{x}\in\mathcal{F}_m}\left[N(\mathbf{x})-D(\mathbf{x})R_m\right]. \tag{15}$$

Let $\mathbf{x}^*$ be an optimal solution of Problem (14). We have the following property.

Lemma 2. *The maximum conditional sum offloading rate R_m^* that solves Problem* (14) *can be achieved if and only if*

$$\begin{aligned} g(R_m^*) &= \max_{\mathbf{x}\in\mathcal{F}_m}\left[N(\mathbf{x})-D(\mathbf{x})R_m^*\right] \\ &= N(\mathbf{x}^*)-D(\mathbf{x}^*)R_m^* = 0. \end{aligned} \tag{16}$$

The proof is presented in [18] due to the space limitation. Lemma 2 reveals the fact that the targeted fractional-form problem in (14) shares the solution $\mathbf{x}^*$ as the subtractive-form problem in (15) when $R_m = R_m^*$. This provides an indirect method for solving the former using an iterative algorithm derived in the sequel, in which the derived condition $g(R_m)=0$ is applied to checking the optimal convergence.

Based on Dinkelbach method [19], we propose an iterative algorithm to obtain R_m^* in (16), thereby solving the slave problem in (14). Specifically, we concern the optimal solution to the subtractive-form Problem (15) for a given R_m:

$$\begin{aligned} g(R_m) = \max_{\mathbf{x}\in\mathcal{F}_m}\Bigg\{ & \sum_{i=1}^K x_i r_i\left[\omega_i - R_m(a_i+b_i\gamma_i)\right] \\ & - R_m(1+d)^{m-1}\Bigg\}. \end{aligned} \tag{17}$$

To facilitate exposition, we can rewrite the expression of $g(R_m)$ as

$$g(R_m) = \max_{\mathbf{x}\in\mathcal{F}_m}\left\{\frac{\sum_{i=1}^K x_i\psi_i(R_m)}{t_e(1+d)^{1-m}} - R_m(1+d)^{m-1}\right\}, \tag{18}$$

with

$$\begin{aligned} \psi_i(R_m) &= x_i r_i\left[\omega_i - R_m(a_i+b_i\gamma_i)\right]t_e(1+d)^{1-m} \\ &= x_i\left[\omega_i\ell_i - R_m(a_i+b_i\gamma_i)\ell_i\right] \\ &= \omega_i\ell_i - R_m(t_i^u+t_i^d), \end{aligned} \tag{19}$$

where the second equality is derived by the result that $\ell_i = t_e r_i (1+d)^{1-m}$ and the last equality is obtained by substituting (1) and (2).

- **Step 1**: The objective of the optimization in (17) can be interpreted as one for maximizing the total system revenue. It follows that the optimal solution, denoted as $\mathbf{x}^*$, is to select m users having the largest per-user revenue:

$$x_i^* = \begin{cases} 1, & \text{if } \psi_i(R_m) \text{ is one of the } m \text{ largest,} \\ 0, & \text{otherwise,} \end{cases} \quad (20)$$

  with $i = 1, \cdots, K$, where $\psi_i(R_m)$ is defined in (19).
- **Step 2**: Given $\mathbf{x}^*$ computed in Step 1, the sum offloading rate R_m can be updated as

$$R_m = \frac{N(\mathbf{x}^*)}{D(\mathbf{x}^*)}, \quad (21)$$

  where $N(\cdot)$ and $D(\cdot)$ are given in (14). Then the per-user revenues $\{\psi_i(R_m)\}$ are updated using the new value of R_m.

Based on the Dinkelbach method, the above two steps are iterated till $g(R_m) = 0$. Since this is the optimality condition according Lemma 2, the convergence of the iteration yields the maximum R_m^* and the corresponding m scheduled users $\mathcal{S}^*(m) = \{i \mid x_i^* = 1\}$. It can be proved that the convergence rate is *superlinear* (see e.g., [19]).

2) Finding the optimal number of scheduled users: With the slave problem in (14) solved in the preceding sub-section, the master problem is to optimize m:

(Master Problem)

$$\max_{1 \le m \le K} R_m^* = \frac{\sum_{i \in \mathcal{S}^*(m)} \omega_i r_i}{(1+d)^{m-1} + \sum_{i \in \mathcal{S}^*(m)} (a_i + b_i \gamma_i) r_i}. \quad (22)$$

To solve the problem, an intelligent search for m^* over $\{1, 2, \cdots, K\}$ seems difficult for the reason that $\{R_m^*\}$ is not a monotone sequence. Due to the lack of monotonicity, we resort to enumerating all possible values of m from 1 to K to find m^*. The complexity of the exhaustive search is reasonable as it scales only linearly with the total number of users K.

3) Overall Algorithm and Its Complexity: The overall algorithm for solving the scheduling problem in (13) is shown in Algorithm 1 which combines the iterative algorithm for solving the slave problem and the exhaustive search for solving the master problem. The complexity of the overall algorithm is discussed as follows. The iterative algorithm for solving the slave problem using Dinkelbach method has complexity upper bounded by $O(\log K)$ [20]. Solving the master problem repeats at most K runs of the iterative algorithms. Therefore, the worst-case complexity of the overall algorithm is $O(K \log K)$.

IV. Simulation Results

In this section, we provide simulation results to evaluate the proposed algorithm. The parameters are set as follows: we let $T = 35$ ms and $\omega_i = 1$, $\forall i \in \mathcal{K}$. The uplink and downlink transmission rates are uniformly distributed in $[100, 150]$ Mbps and $[150, 200]$ Mbps, respectively. The computation-service rate r_i follows uniform distribution over $[1 \times 10^7, 2 \times 10^7]$ bits/sec. The ratio of output/input data is $\gamma_i = 10^{-x}$, where x is uniformly distributed in $[0.5, 1.5]$.

Algorithm 1 Iterative User Scheduling Algorithm Based on Dinkelbach Method

1: **for** $m = 1, \cdots, K$ **do**
2: **initialize** $R_m = 0$.
3: **repeat**
4: For a given R_m, compute $\mathbf{x}^*$ according to (20);
5: Update $R_m = \frac{N(\mathbf{x}^*)}{D(\mathbf{x}^*)}$;
6: **until** $g(R_m) = 0$.
7: Return $R_m^* = R_m$, $\mathbf{x}_m^* = \mathbf{x}$.
8: **end for**
9: Return $m^* = \arg\max_{1 \le m \le K} \{R_m^*\}$, $R^* = R_{m^*}^*$ and $\mathcal{S}^* = \{i \mid x_i^* = 1, i \in \mathcal{K}\}$.

Output: R^* and $\mathcal{S}^*$.

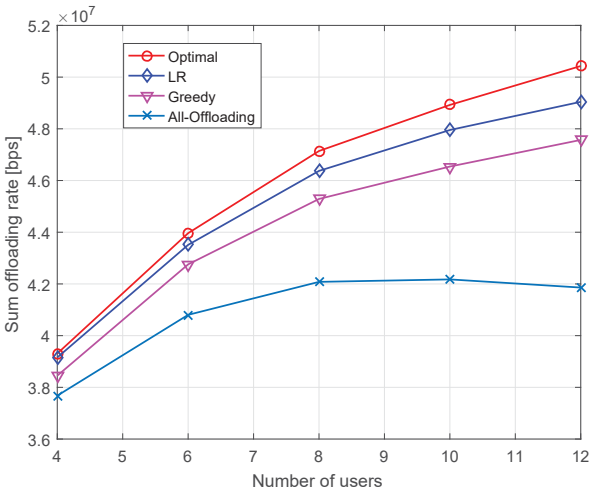

Figure 2: Sum offloading rate vs. the number of users K.

For performance comparison, we introduce three benchmark algorithms in the following.

1) *All-Offloading:* All the users are scheduled to offload, i.e., $\mathcal{S} = \mathcal{K}$.
2) *Greedy:* $\mathcal{S}$ is obtained through a greedy method, i.e., selecting users in the descending order of the transmission rate (i.e., $\frac{\omega_i}{a_i + b_i \gamma_i}$) until condition (10) is invalid.
3) *Linear Programming Relaxation (LR):* $\mathcal{S}$ is obtained by solving K slave problems in (14) using linear programming relaxation [21].

Note that the three benchmarks are used to find the offloading-user set, then the rest problem is reduced to a LP that can be solved efficiently.

In Fig. 2, we compare the sum offloading rate performance of different algorithms when the number of users K varies from 4 to 12, where d is set as 0.1. First, we can see that the sum offloading rate is increasing with K in the cases of

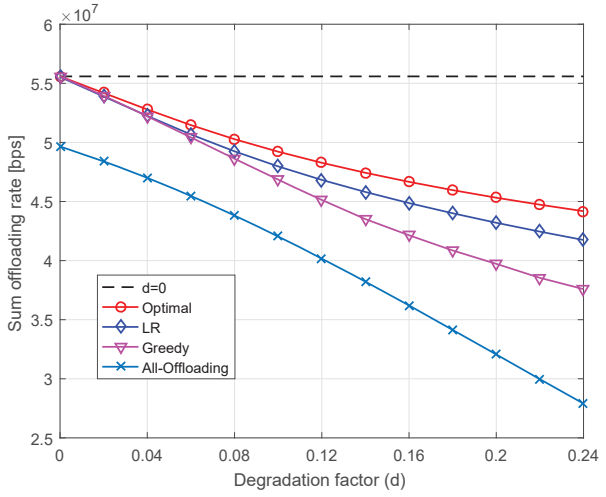

Figure 3: Sum offloading rate vs. the degradation factor d.

the optimal, LR and greedy algorithms, while for the scheme that all users offload, it grows slowly when $K \leq 10$ and begins to decrease afterwards. This is because the former three algorithms have more flexible user-scheduling schemes to balance the degradation impact caused by I/O interference and thus have superior system performance. In contrast, the last algorithm with no control on the number of offloading users, will suffer more severe performance degradation as K increases. Besides, it can be observed that the optimal algorithm outperforms the benchmark algorithms especially when K is large. For instance, when $K = 12$, the optimal algorithm obtains about 3%, 6%, and 20% performance improvements over the three benchmarks respectively.

In Fig. 3, we illustrate the relationship between the degradation factor d and the sum offloading rate performance, where $K = 10$. As expected, the sum offloading rate is decreasing with d in all considered algorithms while the descending rate of the optimal algorithm is the slowest. This indicates that our proposed algorithm has the best performance resistance against the I/O-interference effect. Another interesting observation is that, the performance of LR and greedy algorithms is close-to-optimal when d is small. On the other hand, the line of $d = 0$ can been seen as the sum offloading rate of the conventional case without considering the I/O interference issue. Its performance gap with the optimal algorithm can be interpreted as the overestimation of the system performance builded on the optimistic assumption of no I/O interference.

V. Conclusions

In this paper, we studied joint radio-and-computation resource allocation in a multiuser MEC system, where the computation interference issue has been considered. We formulated the problem of sum offloading rate maximization by joint offloading scheduling, offloaded-data sizes, and communication-and-computation time division. We proposed an optimal solution with low-complexity to solve the non-convex problem. Simulation results demonstrated that our proposed algorithm achieves superior performance gain compared with the benchmark schemes.

References

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

[2] European Telecommunications Standards Institute (ETSI), "Mobile-edge Computing-Introductory technical white paper," September 2014.

[3] X. Pu, L. Liu, Y. Mei, S. Sivathanu, Y. Koh, and C. Pu, "Understanding performance interference of I/O workload in virtualized cloud environments," in *IEEE Cloud*, 2010, pp. 51–58.

[4] S. Ibrahim, B. He, and H. Jin, "Towards pay-as-you-consume cloud computing," in *IEEE SCC*, 2011, pp. 370–377.

[5] S.-g. Kim, H. Eom, and H. Y. Yeom, "Virtual machine consolidation based on interference modeling," *J. Supercomput.*, vol. 66, no. 3, pp. 1489–1506, December 2013.

[6] D. Bruneo, "A stochastic model to investigate data center performance and QoS in IaaS cloud computing systems," *IEEE Trans. Parallel Distrib. Syst.*, vol. 25, no. 3, pp. 560–569, March 2014.

[7] W. B. Slama, Z. Brahmi, and M. M. gammoudi, "Interference-aware virtual machine placement in cloud computing system approach based on fuzzy formal concepts analysis," in *IEEE WETICE*, 2018, pp. 48–53.

[8] C. You, K. Huang, H. Chae, and B. H. Kim, "Energy-efficient resource allocation for mobile-edge computation offloading," *IEEE Trans. Wireless Commun.*, vol. 16, no. 3, pp. 1397–1411, March 2017.

[9] X. Chen, "Decentralized computation offloading game for mobile cloud computing," *IEEE Trans. Parallel Distrib. Syst.*, vol. 26, no. 4, pp. 974–983, April 2015.

[10] M. Liu and Y. Liu, "Price-based distributed offloading for mobile-edge computing with computation capacity constraints," *IEEE Commun. Lett.*, vol. 7, no. 3, pp. 420–423, June 2018.

[11] S. Sardellitti, G. Scutari, and S. Barbarossa, "Joint optimization of radio and computational resources for multicell mobile-edge computing," *IEEE Trans. Signal Inf. Process. Netw.*, vol. 1, no. 2, pp. 89–103, June 2015.

[12] Y. Liu, "Exploiting NOMA for cooperative edge computing," *IEEE Wireless Commun. Mag.*, 2019.

[13] X. Lyu, H. Tian, C. Sengul, and P. Zhang, "Multiuser joint task offloading and resource optimization in proximate clouds," *IEEE Trans. Veh. Technol.*, vol. 66, no. 4, pp. 3435–3447, April 2017.

[14] Y. Mao, J. Zhang, S. H. Song, and K. B. Letaief, "Stochastic joint radio and computational resource management for multi-user mobile-edge computing systems," *IEEE Trans. Wireless Commun.*, vol. 16, no. 9, pp. 5994–6009, September 2017.

[15] X. Pu, L. Liu, Y. Mei, S. Sivathanu, Y. Koh, C. Pu, and Y. Cao, "Who is your neighbor: Net I/O performance interference in virtualized clouds," vol. 6, no. 3, pp. 314–329, July 2013.

[16] J. N. Matthews, W. Hu, M. Hapuarachchi, T. Deshane, D. Dimatos, G. Hamilton, M. McCabe, and J. Owens, "Quantifying the performance isolation properties of virtualization systems," in *ExpCS'07*, 2007.

[17] M. Mishra and A. Sahoo, "On theory of VM placement: Anomalies in existing methodologies and their mitigation using a novel vector based approach," in *IEEE Cloud*, 2011, pp. 275–282.

[18] Z. Liang, Y. Liu, T. Lok, and K. Huang, "Multiuser computation offloading and downloading for edge computing with virtualization," *CoRR*, vol. abs/1811.07517, 2018. [Online]. Available: http://arxiv.org/abs/1811.07517

[19] W. Dinkelbach, "On nonlinear fractional programming," *Management Science*, vol. 13, no. 7, pp. 492–498, 1967. [Online]. Available: https://doi.org/10.1287/mnsc.13.7.492

[20] T. Matsui, Y. Saruwatari, and M. Shigeno, "An analysis of dinkelbach's algorithm for 0-1 fractional programming problems," 1992.

[21] T. Q. Dinh, J. Tang, Q. D. La, and T. Q. S. Quek, "Offloading in mobile edge computing: Task allocation and computational frequency scaling," *IEEE Trans. Commun.*, vol. 65, no. 8, pp. 3571–3584, August 2017.