Group Delay-Aware Scalable Mobile Edge Computing Using Service Replication

Shimaa A. Mohamed, Sameh Sorour, *Senior Member, IEEE*, and Hossam S. Hassanein, *Fellow, IEEE*

***Abstract*—The number of individuals and groups of users offloading independent and inter-related computational tasks to mobile edge computing (MEC) servers is rapidly increasing, thus overloading them and raising the risk of service interruptions. Hence, reactive service replication has been suggested to enable individuals and groups of users to access services from remote edge servers, thus guaranteeing system scalability. This paper proposes a task offloading and service replication scheme on local and remote MEC servers. The scheme minimizes the response time of all users while satisfying the delay requirements of user groups in traffic-heavy and multimedia-intense applications (e.g., online gaming, multimedia conferencing, augmenting reality). We formulate an integer linear problem that minimizes the average response time of all users while satisfying the time and time difference constraints of the user groups running the same applications. We then use linear relaxation programming using Lagrangian analysis and solve the problem using a numerical solver. In addition, we compare the optimal solution to distance-based and resource-based greedy approaches. The results demonstrate the merits of our proposed optimized decision scheme compared to these two greedy approaches.**

***Index Terms*—Mobile edge computing, computation offloading, service replication, lagrangian analysis.**

I. Introduction

MOBILE devices currently need complex computations because of their resource limitations (processing power, battery lifetime, and storage capacity). One conventional solution to this problem is computation offloading, where part or all such computations are offloaded to remote resourceful cloud servers, thus saving processing power and energy. However, offloading such computations to the cloud may cause long communication latency between cloud servers and mobile devices. Mobile Edge Computing (MEC) has emerged as an alternative that enables mobile devices to access cloud-like computing services at edge servers located within the mobile subscribers' radio access network. MEC's primary objective is reducing latency by offloading the computations and data storage to these edge servers [2]. Despite their significant potential to improve the performance of delay-sensitive applications, edge servers have relatively limited computation resources compared to cloud servers. This fact may cause edge servers to get overloaded with computations to the extent of causing service interruption for a subset of its subscribed users [2]. This scenario is real during high-density user demand, such as viewing live sports events and concerts.

A. Related Work

Partial and full task offloading and resource allocation strategies have been extensively investigated in single server edge computing environments to minimize the overloading and prevent service interruption. Liu et al. [3] use a Markov Decision Process to solve a proposed stochastic optimization problem, which minimized the latency for one user with multiple tasks connected to one MEC server. In [4], Mao et al. study task offloading, scheduling, and transmit power for MEC systems with independent tasks by proposing a joint optimization problem that minimized the weighted sum of the latency and device energy consumption. Other works considered a multi-server MEC system. In [5], Liu et al. investigate the power-delay trade-off in a multi-user MEC system by using Lyapunov stochastic optimization for optimizing the transmission power and allocated user tasks between local and on-server computations. In [6], Ranadheera et al. propose an energy-efficient mechanism for computation offloading to MEC servers by activating a subset of servers and satisfying users' latency requirements by using the theory of minority games. In [7], Dinh et al. propose an optimization problem on offloading from a single mobile device to multiple access points, besides local computation on the mobile devices. Their goal is to minimize the mobile device's total latency and energy consumption while jointly optimizing the mobile device's task allocation decisions and frequency scaling. Two cases for the mobile device were considered, fixed computing capacity and elastic computing capacity. They studied how the computing capacity range can influence the task allocation decision. In [8], Du et al. propose a joint task offloading and resource allocation scheme for multi-server edge computing environments. This scheme intended to maximize service capacity (defined by the number of served mobile devices) and minimize the service cost (measuring the service latency and power consumption experienced by users). However, this work assumed that some users could be left unserved due to a lack of resources. Gu et al. [9] investigate the task offloading problem

Manuscript received 27 August 2021; revised 14 March 2022 and 30 May 2022; accepted 4 July 2022. Date of publication 26 July 2022; date of current version 14 November 2022. This paper extends our earlier work [1] published at IEEE Global Communications Conference, December 2020. This work was supported by the Natural Sciences and Engineering Research Council of Canada (NSERC) under Grant ALLRP 549919-20. The review of this article was coordinated by Prof. Bin Lin. *(Corresponding author: Shimaa A. Mohamed.)*

Shimaa A. Mohamed is with the School of Computing, Queen's University, Kingston, ON K7L 3N6, Canada, and also with the City of Scientific Research and Technological Applications, SRTA-city, Alexandria 5220211, Egypt (e-mail: 16saa4@queensu.ca).

Sameh Sorour and Hossam S. Hassanein are with the School of Computing, Queen's University, Kingston, ON K7L 3N6, Canada (e-mail: sameh.sorour@queensu.ca; hossam@cs.queensu.ca).

Digital Object Identifier 10.1109/TVT.2022.3193887

in the MEC-enabled Ultra Dense Network (UDN) architecture. They propose an offloading algorithm to minimize the task response time while satisfying the energy budget constraints.

On the other hand, another direction aimed to optimize task allocation and resource scheduling to reduce MEC's cost while assuming that the capacity of the edge servers will always satisfy the demands of the users. In [10], Zhang et al. investigated the joint task scheduling and dynamic resource management problem to reduce the cost of an edge computing system consisting of multiple collocated edge servers. They modeled this problem as an optimization problem, whose objective was to minimize the system cost given delay requirements. In [11], Li et al. proposed a mixed-integer non-convex optimization problem to maximize the number of offloaded tasks and minimize the energy consumption of all the user devices and edge servers by selecting the best edge servers with capacity for offloading. In [12], Long et al. investigated the computation offloading and communication and computation resource allocation scheme for the MEC system. They designed a multi-objective computation offloading resource allocation algorithm optimizing the uplink and downlink spectrum resources, computation resources, and offloading strategy.

In most of these works, the notion of scalability in the MEC system was ignored under the assumption of guaranteed service availability regardless of the number of user devices and capacities of the edge servers. With the increasing number of devices and complexity of the applications that run on them, these devices impose a massive load at their local edge servers resulting in network bottlenecks and service interruptions [2]. In [13], Maheshwari et al. analyzed the scalability and capacity performance of a hybrid edge-cloud system designed to support latency-sensitive applications, namely an augmented reality application with service constraints. Their results presented a guide for selecting the right balance between edge and cloud resources in such scenarios. On the other side of the spectrum, the distribution of offloaded tasks between edge servers and peer user devices was proposed [14]. The purpose of this work is to determine the best load partitioning to minimize the excess delay and energy consumption compared to the no offloading alternative.

Many researchers have suggested that load balancing mechanisms and server clustering ensure service scalability within the layer of the edge server. In [15], Liu et al. proposed a first-come-first-serve queuing and service model for scenarios with multiple MEC servers to improve the blocking probability and waiting time experienced by users through clustering edge servers together for load sharing. When the buffer is full, the system administrator handles user requests differently according to three proposed load-sharing schemes: No Sharing, Random Sharing, and Least Loaded Sharing. In [16], Beraldi et al. proposed a cooperative load balancing scheme to minimize the blocking states at each edge server. Each edge server has a buffer to store service requests for future execution. The upcoming demands migrate to a neighboring edge server in a full buffer case, which accepts the demand if its queue length is less than a specific threshold. In [17], Ndikumana et al. proposed collaborative data caching and computation offloading among collaborating MEC servers. The allocation of caches and computation resources into multiple service requesters is done based on their demands and payments thus, maximizing utilization from the MEC server. In [18], Zhang et al. introduced the fiber-wireless technology to promote the load balancing of the edge servers' computation resources in vehicular edge computing networks. They proposed the software-defined networking-based load-balancing task offloading scheme between vehicles and edge servers to minimize the processing delay of the vehicles' computation tasks. In [19], Li et al. investigated the capability of partial processing the tasks of mobile devices between edge computing and cloud computing. They proposed a joint communication and computation resource allocation problem to minimize the weight-sum latency of all mobile devices. In [20], Zhao et al. investigated the collaborative computation between cloud computing and MEC to jointly optimizing computation offloading decision and computation resource allocation by proposing a collaborative computation offloading and resource allocation optimization scheme. In [21], Liu et al. propose an online offloading framework for multiple mobile applications, which consists of dependent tasks and propose a heuristic algorithm to minimize the average makespan of multiple mobile applications to the MEC-Cloud architecture. The logical sequence of these dependent tasks can be represented by a direct acyclic graph (DAG) [22], [23], where they use release time and deadline to ensure the sequence of executing dependent tasks.

The above works did not consider the concept of users grouping, where users are split into groups due to participating in the same activity under *time* and *time different* constraints. In addition, our proposed work seeks to improve the scalability of real-time applications that cannot tolerate queuing at the MEC servers to be executed.

B. Our Contributions

In this paper, we consider multiple sets of users forming service groups due to each group's participation in the same synchronized activity, such as online gaming or augmented reality. In addition, our proposed scheme improves the scalability of real-time offered services by replicating them on-demand at underloaded remote edge servers in the MEC system to serve all users concurrently when the local servers cannot sustain the load of all user groups.

The main contributions of this paper are the following:

- We manage task offloading for groups of users requesting to perform delay-correlated computations on multiple MEC servers.
- We formulate an integer linear problem that minimizes the average response time of all users while satisfying the time and time difference constraints of the user groups running the same applications.
- We deal with the linear programming (LP) relaxation of our problem, which is used to derive the optimal decisions using Lagrangian analysis.
- We solve the proposed LP problem numerically and apply a proposed rounding algorithm to reach the integer values of the decision variables.
- We compare the proposed solution with the ILP solution and show that the LP relaxation solution is performing almost the same while saving big on the time complexity.

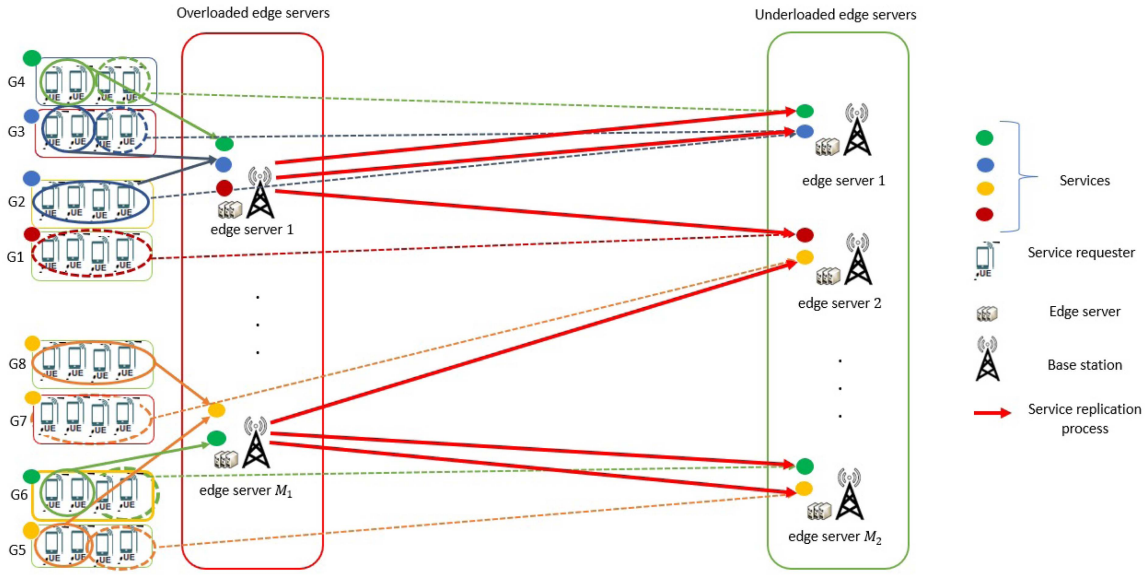


Fig. 1. System Model.

- We compare the performance of our proposed LP relaxation solution with distance-based and resource-based greedy approaches.

The rest of the paper is organized as follows; Section II details our system model and parameters, and Section III discusses the problem setup. Section IV describes the solution using linear relaxation. Section V presents the performance evaluation and the comparison with the two greedy approaches. We conclude the paper and highlight future research directions in Section VI.

II. System Model

The system model is depicted in Fig. 1, in which the MEC system consists of two entities, namely a service provider and a set of users requesting service from this service provider. The service provider manages a cluster of edge servers, namely, local edge servers and remote edge servers. We assume that users can all connect to their local edge servers, which are typically collocated with their nearest base station (BS), usually identified according to their received signal strength. Whenever needed, each user can also wirelessly connect to one of a subset of remote servers with different levels of connection qualities, which are impacted by its distance and fading conditions. The local edge server of every User U_u always achieves the least response time. The MEC servers are divided into two subsets, namely the overloaded and underloaded sets. The overloaded set typically consists of local servers whose local users have demands that exceed their capacities. In this case, the un-served users of overloaded edge servers are allowed to access the remote underloaded servers after the latter replicates their needed service(s) on them. Let $\mathcal{S}^o = \left\{S_1^o, S_2^o, \ldots, S_{M_1}^o\right\}$ be the set of overloaded edge servers in the system, $M_1 = |\mathcal{S}^o|$ denotes the number of overloaded local edge servers, $\mathcal{S}^u = \left\{S_1^u, S_2^u, \ldots, S_{M_2}^u\right\}$ be the set of underloaded edge servers in the system, $M_2 = |\mathcal{S}^u|$ denotes the number of underloaded remote edge servers, and $\mathcal{S} = S_j^o \bigcup \mathcal{S}^u$ be the set of the possible edge servers that a user can offload its task to one of them.

All edge servers connect via backhaul links, which are helpful for service replications whenever needed. Besides, their computational resources are divisible among multiple users. Let $F_{S_j^o}^o$ denote the available computing capacity of each overloaded edge server $S_j^o, S_j^o \in \mathcal{S}^o$ and $F_{S_i^u}^u$ denote the available computing capacity of each underloaded edge server $S_i^u, S_i^u \in \mathcal{S}^u$ in the MEC system, their units are gigahertz (Gcycles/sec).

On the user side, $\mathcal{U}^j = \left\{U_1^j, U_2^j, \ldots, U_{N_j}^j\right\}$ denotes the set of user devices attached to their local overloaded edge server $S_j^o, S_j^o \in \mathcal{S}^o$. Moreover, let $\mathcal{U} = \bigcup_{j=1}^{M_1} \mathcal{U}^j$ represent the set of all such users $\{U_1, U_2, \ldots, U_N\}$. Subsets of these users may be participating in the same common traffic-heavy, real-time, and multimedia-intense application/activity, such as online gaming, multimedia conferencing, and augmented reality. We will refer to each of these subsets of users $\mathcal{U}^{G_g} = \left\{U_1^{G_g}, U_2^{G_g}, \ldots, U_{N_g}^{G_g}\right\}$[1], and all such groups form a set $\mathcal{G} = \{G_1, G_2, \ldots, G_{NG}\}$ of groups. Being involved in the same application, the members of each group are typically constrained with application-dependent QoS requirements $T_{G_g}^{max}$, $F_{G_g}^{min}$, and d_{G_g}, denoting the maximum delay all members within the group can tolerate for having their tasks completed, minimum CPU speed these tasks can be processed with, and maximum delay difference between any two users belonging to the same group, respectively. By delay difference, we mean the difference between the delay experienced by any pair of users in the group. This parameter is of extreme importance in many of the considered applications, where users must get responses for their offloaded computations within a very small difference in time, or else the application fails. An example of such a case is online gaming, in which players must see the results of each others' actions almost at the same time or within a bounded (typically very small) time difference. Otherwise, different players will be

[1]This model captures a special case scenario where any one or multiple users are solely involved in applications. In this case, each of these users forms a separate group of size 1 (i.e., $N_g = 1$)

seeing other users' actions with different delays, which may cause some of them to make wrong decisions in their next actions/moves, thus totally ruining the game.

At execution time, each user sends its task to either its local edge server or one of its accessible remote edge servers. A user's task $\mathcal{T}_u$ is defined by the following 3-tuple:

$$\mathcal{T}_u = (C_u, B_u, G_g), \tag{1}$$

where u refers to User $U_u, U_u \in \mathcal{U}$, C_u (measured in megacycles) denotes the total number of the CPU cycles required to complete user U_u's task, B_u (measured in megabytes) describes the size of the information that User U_u needs to send to the edge server for it to execute this task, and $G_g, G_g \in \mathcal{G}$ denotes the ID of the group user U_u belongs to. Given this description of the system, we can formulate in the next section our group-delay aware task offloading and service replication problem. The aim of this problem is to minimize the average response time of all users served by both local and remote edge servers while satisfying all the QoS requirements of users and groups. By response time, we mean the total time between a user submitting a task and getting the response of it. This includes the transmission time of the task to the edge server, the computation time at the server, and (if needed), the replication time of the application on a remote edge server when a user is forced to send its task to a remote edge server.

III. Problem Setup

A. Definition of Objective Variables

Let $x_{ui}, \forall u, i$, be the task offloading indicators, where u and i refer to User $U_u, U_u \in \mathcal{U}$ and Server $S_i, S_i \in \mathcal{S}$, respectively. In our formulation, x_{ui} is set to 1 when User U_u offloads its task to Server S_i, and is set to 0 otherwise.

As in typical MEC settings, users offload their tasks and access services at their assigned edge server through a direct wireless channel between them. The rate at which the instructions and data of user U_u's computational task (i.e., $\mathcal{T}_u$) is sent to Server $S_i, S_i \in \mathcal{S}$ is expressed as [11]:

$$R_{ui} = B \log_2 \left(1 + \frac{P_u H_{ui}}{\sigma_i B}\right). \tag{2}$$

where B is channel bandwidth, P_u is the transmission power of User U_u, σ_i denotes the noise power spectral density at Server S_i, and H_{ui} represents the channel power gain between User U_u to Server S_i defined as:

$$H_{ui} = d_{ui}^{-\alpha}. \tag{3}$$

where d_{ui} is the distance between User U_u and Server S_i, and α is the path loss exponent, which is typically set to 4 in mobile edge environments [24].

Consequently, the transmission time of Task $\mathcal{T}_u$ of User U_u to Server S_i can be expressed as:

$$T_{ui}^{T} = \frac{B_u}{R_{ui}}. \tag{4}$$

The transmission time includes any re-transmission due to packet loss. In addition, the computation time of task $\mathcal{T}_u$ at Server S_i can be defined as:

$$T_{ui}^{C} = \frac{C_u}{f_u}. \tag{5}$$

where $f_u \geq F_{G_g}^{min}$ is the CPU frequency assigned to the task of User $\mathcal{U}_u$ that belongs to Group $G_g, G_g \in \mathcal{G}$. Finally, if the task of User U_u is offloaded to remote Server $S_i^u, S_i^u \in \mathcal{S}^u$, an additional time T_{ui}^{R} will be required to replicate and deploy this service on that remote edge server. Its value depends on the required service of Group G_g that User U_u belong to. Thus, the response time of the user U_u's task $\mathcal{T}_u$ at Server S_i is given by:

$$T_{ui}^{resp} = T_{ui}^{T} + T_{ui}^{C} + T_{ui}^{R}. \tag{6}$$

B. Problem Formulation

Given the preceding definition of the objective variables, the task offloading problem achieving our target can be expressed as follows:

$$\min_{x_{ui}} \frac{1}{N} \sum_{i \in \mathcal{S}} \sum_{u=1}^{N} x_{ui} T_{ui}^{resp} \tag{7a}$$

$$S.T. \quad \sum_{i \in \mathcal{S}} x_{ui} = 1, \qquad \forall U_u \in \mathcal{U} \tag{7b}$$

$$\sum_{u \in \mathcal{U}^j} x_{ui} f_u \leq F_{S_j^o}^{o}, \qquad \forall S_j^o \in \mathcal{S}^o \tag{7c}$$

$$\sum_{u=1}^{N} x_{ui} f_u \leq F_{S_i^u}^{u}, \qquad \forall S_i^u \in \mathcal{S}^u \tag{7d}$$

$$\sum_{i \in \mathcal{S}} x_{ui} T_{ui}^{resp} \leq T_{G_g}^{max}, \quad \forall G_g \in \mathcal{G}, \forall U_u^{G_g} \in \mathcal{U}^{G_g} \tag{7e}$$

$$\sum_{i \in \mathcal{S}} x_{ui} T_{ui}^{resp} - \sum_{i \in \mathcal{S}} x_{ji} T_{ji}^{resp} \leq d_{G_g}, \quad \forall G_g \in \mathcal{G}, \forall U_u^{G_g}, U_j^{G_g} \in \mathcal{U}^{G_g}, j \neq u \tag{7f}$$

$$x_{ui} \in \{0, 1\}, \qquad \forall U_u \in \mathcal{U}, \forall S_i \in \mathcal{S} \tag{7g}$$

The objective function in (7a) minimizes the average response time of all users' tasks by determining the best edge server each User $\mathcal{U}_u$ should offload their tasks to. Constraint (7b) ensures that each task is offloaded to only one server, whereas Constraint (7c) ensures that the sum of assigned CPU computing capacities to the subset of users offloading their tasks to their local edge server does not exceed its total available CPU computing capacity. Meanwhile, Constraint (7d) ensures that the sum of assigned CPU computing capacities to all users offloading their tasks to each remote edge server does not exceed its total available CPU computing capacity. Constraint (7e) ensures that the total response time of a user's task belonging to any Group G_g does not exceed this group's maximum delay requirement. Constraint (7f) ensures that the delay difference between two users belonging to the same Group G_g does not exceed its application's upper limit. Finally, Constraint (7g) imposes a binary decision value for the variable x_{ui} of all users and servers.

It is cleared that optimization problem (7) is an integer linear problem (ILP). To solve this problem, we relax the ILP optimization problem into a linear programming (LP) by relaxing the integer decision variables, deriving analytical and solver-based solutions for them, and then restoring the integer decision variables using a rounding greedy approach. We then compare this solution to both a solver-based solution to the ILP problem, and two greedy approaches that can satisfy the scalability and group-delay awareness settings considered in this paper.

IV. Solution Using Linear Relaxation

In this section, we relax the previous optimization problem in (7) into a linear programming (LP) formulation. The objective function and constraints are similar to the integer linear problem (7) except constraint (7g), which is replaced by $0 \leq x_{ui} \leq 1$. In [26], the output of a linear problem is an integer if each square submatrix of constraint coefficient matrices is unimodular. This can be determined by calculating the determinant of each coefficient matrix. We investigated whether this property applies to our problem, but it, unfortunately, failed for some sub-matrices.

A. Analytical Solutions

The linear problem version of (7) is a convex optimization problem. The Karush-Kuhn-Tucker (KKT) conditions thus provide necessary and sufficient conditions for an optimal solution [27]. Therefore, applying the KKT conditions to the constraints of the problem and the Lagrangian function's gradient allows us to find the analytical solution of the real offloading decision variables x_{ui}. The Lagrangian function associated with the linear optimization problem is given by (8) shown at the bottom of this page, where X is the vector of offloading decisions (i.e., $X = [x_{ui}]$), u refers to user $U_u, \forall U_u \in \mathcal{U}$ and i refers to the edge server $S_i, \forall S_i \in \mathcal{S}$, and:

- $\lambda_{local} = [\lambda_{local_i}], \lambda_{remote} = [\lambda_{remote_i}], \alpha = [\alpha_u], \sigma = [\sigma_{gul}]$, are the associated Lagrange multiplier to inequality constraints (7c), (7d), (7e), (7f), respectively.
- $\beta = [\beta_{ui}], \gamma = [\gamma_{ui}]$ are the associated Lagrange multiplier to the lower bound and upper bound inequality constraint of x_{ui}, respectively.
- $\nu = [\nu_u]$ is the associated Lagrange multiplier to the equality constraint (7b).

Appendix A shows the derived equations by applying the KKT conditions. After applying the KKT conditions on the equality and inequality constraints, the following theorem illustrates the optimal solution for the linear problem.

Theorem 1: The optimal offloading decisions of the linear optimization problem can be expressed as in (9), shown at the bottom of this page.

Proof: Appendix B shows the proof of Theorem 1.

B. Numerical Results

From (9), we found that the analytical solutions did not yield closed-form expressions for the relaxed problem. Therefore, we solve the linear problem using a numerical solver. We then employ the rounding greedy approach, described in Algorithm 1, to restore the binary values of the decision variables x_{ui}. This algorithm is based on the one defined in [28]. The worst-case time complexity of the rounding algorithm is $O(N^2 M_2^2)$. The time complexity of the linear-relaxation-based solution is the summation of the complexities of solving the linear problem and the rounding algorithm. This time complexity was shown to be less complex than that of solving the original ILP [29]–[31].

$$
\begin{aligned}
&L(X, \lambda_{local}, \lambda_{remote}, \alpha, \sigma, \beta, \gamma, \nu) \\
&= \sum_{i\in\mathcal{S}}\sum_{u=1}^{N} \frac{T_{ui}^{resp}}{N} x_{ui} + \sum_{i\in\mathcal{S}^o} \lambda_{local_i} \left(\sum_{u\in\mathcal{U}^i} f_u x_{ui} - F_{S_i^o}^o \right) + \sum_{i\in\mathcal{S}^u} \lambda_{remote_i} \left(\sum_{u=1}^{N} f_u x_{ui} - F_{S_i^u}^u \right) \\
&\quad + \sum_{g=1}^{NG} \sum_{u\in\mathcal{U}^{G_g}} \alpha_u \left(\sum_{i\in\mathcal{S}} T_{ui}^{resp} x_{ui} - T_{G_g}^{max} \right) + \sum_{g=1}^{NG} \sum_{u\in\mathcal{U}^{G_g}} \sum_{j\in\mathcal{U}^{G_g}, j\neq u} \sigma_{guj} \left(\sum_{i\in\mathcal{S}} x_{ui} T_{ui}^{resp} - \sum_{i\in\mathcal{S}} x_{ji} T_{ji}^{resp} - d_{G_g} \right) \\
&\quad - \sum_{i\in\mathcal{S}}\sum_{u=1}^{N} \beta_{ui} x_{ui} + \sum_{i\in\mathcal{S}}\sum_{u=1}^{N} \gamma_{ui} (x_{ui} - 1) + \sum_{u=1}^{N} \nu_u \left(\sum_{i\in\mathcal{S}} x_{ui} - 1 \right) \\
&\qquad\qquad \forall S_j^o \in \mathcal{S}^o, \forall S_i^u \in \mathcal{S}^u, \forall G_g \in \mathcal{G}, \forall U_u^{G_g}, \in \mathcal{U}^{G_g}
\end{aligned} \tag{8}
$$

When $\forall S_j^o \in \mathcal{S}^o, \forall U_u, \in \mathcal{U}^j$,

$$
x_{ui}^* = \begin{cases} 0, & if \gamma_{ui}^* = 0, \beta_{ui}^* > 0, \frac{T_{ui}^{resp}}{N} + \lambda_{local_i}^* f_u + \alpha_u^* T_{ui}^{resp} + \nu_u^* \neq 0 \\ 1, & if \gamma_{ui}^* > 0, \beta_{ui}^* = 0, \frac{T_{ui}^{resp}}{N} + \lambda_{local_i}^* f_u + \alpha_u^* T_{ui}^{resp} + \nu_u^* = -\gamma_{ui}^* \end{cases}
$$

Otherwise, when $\forall S_i^u \in \mathcal{S}^u, \forall U_u, \in \mathcal{U}$,

$$
x_{ui}^* = \begin{cases} 0, & if \gamma_{ui}^* = 0, \beta_{ui}^* > 0, \frac{T_{ui}^{resp}}{N} + \lambda_{remote_i}^* f_u + \alpha_u^* T_{ui}^{resp} + \nu_u^* \neq 0 \\ 1, & if \gamma_{ui}^* > 0, \beta_{ui}^* = 0, \frac{T_{ui}^{resp}}{N} + \lambda_{remote_i}^* f_u + \alpha_u^* T_{ui}^{resp} + \nu_u^* = -\gamma_{ui}^* \end{cases} \tag{9}
$$

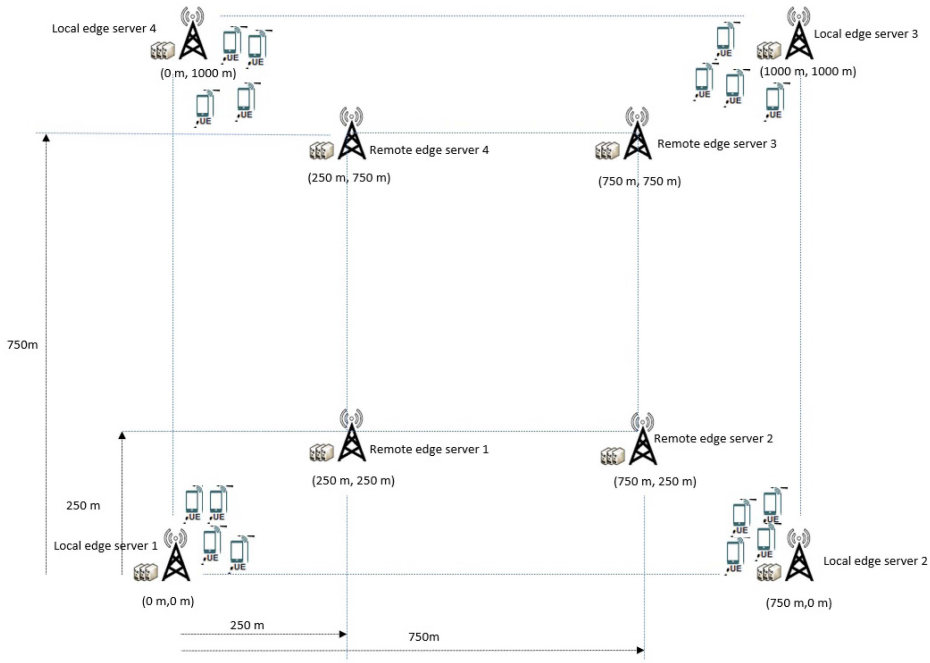


Fig. 2. Simulation setup.

Algorithm 1: Rounding Algorithm.

Inputs:
The linear relaxed decision variables.

Steps:
1- **Sort** the set of decision variables x^*_{ui}, $\forall S_i \in \mathcal{S}, \forall U_u \in \mathcal{U}$ in a descending order.
2- $round(x^*_{ui}) \in \{0, 1\}$:
3- **Check** $round(x^*_{ui})$
4- **If** $round(x^*_{ui}) = 1$:
5- **If** constraints (7b) to (7f) are achieved.
6- $x_{ui} = 1$
7- **Else**, $x_{ui} = 0$
8- **End If**
9- **Else**,
10- $x_{ui} = 0$
11- **End If**

Output:
The output of linear optimized solution.

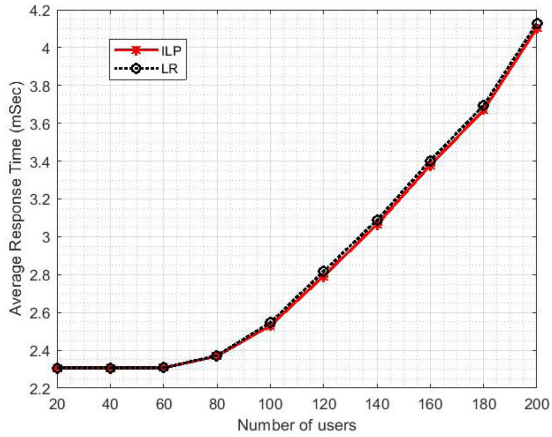
V. Performance Evaluation

Here, we present the simulation results showing the performance of the ILP and linear-relaxation-based solution and the merits of the latter.

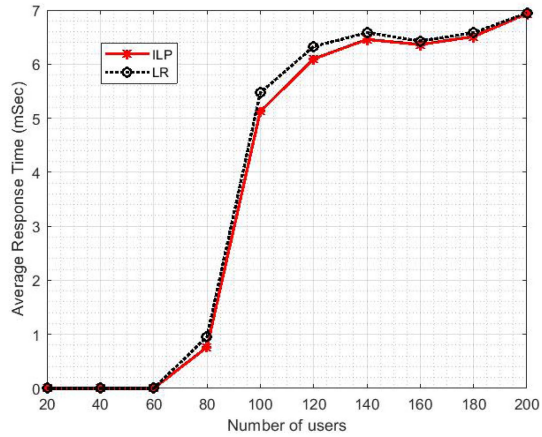
A. Simulation Setup

In our simulations, there are eight edge servers deployed in an area of dimensions 1000X1000 m. Fig. 2 shows the distribution of users and servers in the area of interest. For wireless access, the bandwidth for uplink and downlink channels is equal and is set to B = 1 MHz. The user devices' transmission power is set to $P_u = 20$ dBm, and the noise power spectral density is set to -174 dBm/HZ. The computation frequency of each MEC server follows a uniform distribution in the range of [400–800] GHz. The data size of the tasks follows a uniform distribution[2] in the range of [0.01–0.05] Megabits. The computing cycle of any one task follows a uniform distribution in the range of [100–700] cycles/byte. The edge servers are deployed at fixed locations in the simulation area. All user devices are then deployed near to (yet at random distances from) the number of specific edge servers designated as overloaded edge servers $\mathcal{S}^o$. There are four overloaded local edge servers and four underloaded remote edge servers. For each overloaded local edge server, there are up to three groups of users; each playing a multiplayer game using a virtual reality application with different QoS requirements, the maximum delay difference between any two users belonging to the same group follows a uniform distribution in the range of [4–11] mSec and the maximum delay of the group is selected randomly between 25 and 40 mSec. The numerical solver employed to derive the optimized ILP and LP solutions are the corresponding solvers in the MATLAB Optimization Toolbox [25]. The merits of the ILP and linear-relaxation-based solutions are compared with two greedy approaches: (1) the distance-based approach in which users close to their overloaded local server offload their tasks to it and the remaining users offload their tasks to their closest available underloaded remote servers (i.e., the closest ones that can still satisfy both of their group's QoS requirements). (2) The resource-based approach in which users offload tasks to the overloaded local server then to underloaded remote servers in descending order of their CPU computing cycles while satisfying the group's QoS requirements.

[2]We experimented with other distributions for data size but found the results to be very similar. Hence these results are not explicitly depicted in the paper.



(a)



(b)

Fig. 3. Average response time of integer linear programming (ILP) vs. linear-relaxation-based (LR). (a) All users. (b) Users whose tasks are assigned to remote servers.

B. Simulation Results

In Fig. 3, we plot the average response time of the integer linear programming (ILP) and the linear-relaxation-based (LR) solutions as the number of users increased. In Fig. 3(a), the ILP and the LR solutions have almost the same average response time of all users. While the worst case of the average response time of the LR solution can achieve up to 5% more than the ILP solution of users whose tasks are assigned to the remote edge servers when the number of users equals 100 as shown in Fig. 3(b). Therefore, the following simulation results will only compare the linear-relaxation-based solution compared with distance-based and resource-based greedy approaches. Fig. 4 shows the simulation time of the ILP and the LR solutions as the number of users increased. The ILP solution is more complex than the LR solution for larger number of users, namely 100 users and above. The LR solution saves up to 89% of the simulation time compared to the ILP when the number of users equals 140.

Fig. 5 shows that the LR solution outperforms the two greedy approaches in the average response time for larger number of users, namely 100 users and above. As for smaller number of users, the figures shows that the LR scheme does not provide performance gains compared to the two schemes. This can be interpreted by the fact that most users in this scenario will execute their tasks at their local servers, and only a very small number of users will be assigned to execute their tasks at remote servers. This makes the impact of such assignment using the

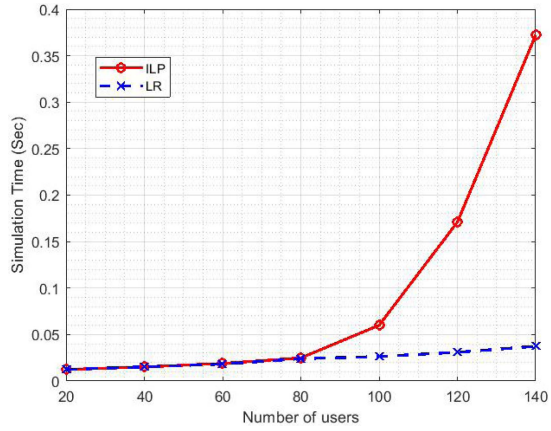
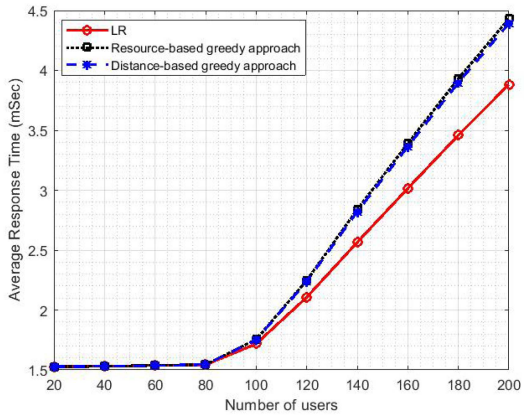
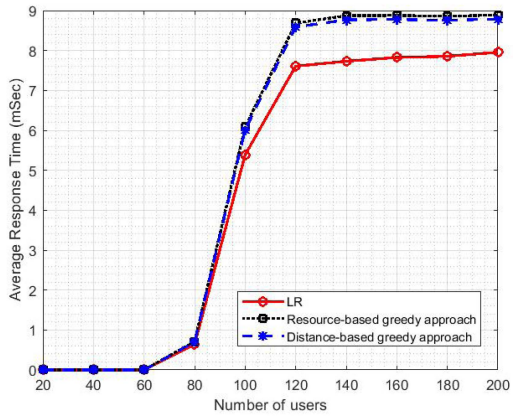


Fig. 4. Simulation time of integer linear programming (ILP) vs. linear-relaxation-based (LR).



(a)



(b)

Fig. 5. Average response time vs. the number of users. (a) All users. (b) Users whose tasks are assigned to remote servers.

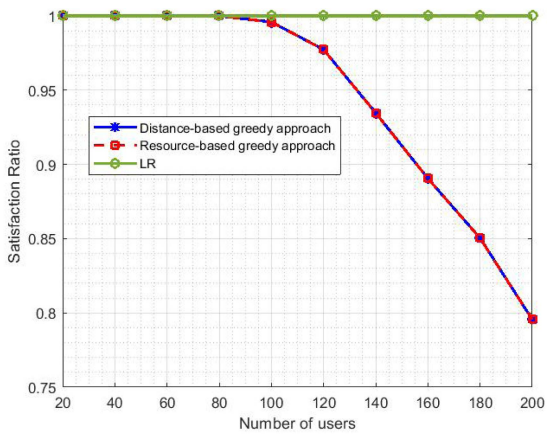


Fig. 6. Satisfaction ratio vs. the number of users.

three schemes quite indifferent, unlike the case of larger number of users, where the LR solution minimizes the response time of all users while satisfying the delay requirements of user groups in traffic-heavy and multimedia-intense applications compared with distance-based and resource-based greedy schemes. In Fig. 5(a), the LR solution can achieve 11% and 12% response time reduction over the solutions achieved by the distance-based and resource-based greedy schemes, respectively for all users when the number of users equals 200. While the LR solution can achieve up to 9% and 10% response time reduction over the solutions achieved by distance-based and resource-based greedy schemes, respectively for users whose tasks are assigned to remote edge servers when the number of users equals 200 as shown in Fig. 5(b).

Fig. 6 shows the satisfaction ratio, which is the ratio of the number of users who satisfy the time difference constraint to all number of users in the system for our proposed LR solution compared with the distance-based and resource-based greedy baseline solutions when the number of users varies from 20 to 200. The distance-based and resource-based greedy approaches do not meet all group's QoS requirements in terms of the delay difference (i.e., the delay difference between two users belonging to the same group does not satisfy the value d_{G_g} for a fraction of the users). This degradation becomes more evident for the larger number of users, in which case it is harder for these algorithms to satisfy such requirements. On the other hand, our proposed LR solution achieves this group's QoS delay difference requirements for all users

VI. Conclusion

In this paper, we investigated the problem of task offloading with service replications in scalable MEC environments for groups of users involved in similar multimedia-intense applications. By scalable MEC, we mean MEC systems that allow local overloaded servers, which cannot immediately admit the tasks of all their local users, to replicate the services of the un-served users on remote underloaded servers so that these users get their tasks executed on them. Our study aimed to minimize the average response time of all users' offloaded tasks while respecting the application-imposed delays and delay difference requirements for each group of users involved in the same application. The problem was formulated as an integer linear program and was solved numerically using the MATLAB Optimization Toolbox. To reduce complexity, we proposed a linear-relaxation-based solution, derived its optimal solution of its relaxed linear problem using Lagrangian analysis and KKT conditions, solved numerically using the MATLAB Optimization Toolbox, and presented a greedy rounding algorithm to restore binary values for the decision variables. Simulation results show that this linear-relaxation-based solution both achieves almost the same performance of the ILP solution and outperforms the distance-based and resource-based greedy schemes that respect our model's scalability and group requirement settings. More gains were achievable by our scheme compared to the two greedy schemes as the number of users increases. Up to 11% and 12% were observed by our proposed scheme compared to both the distance-based and resource-based greedy approaches, respectively. Compared to the ILP solution, our proposed solution achieved up to 89% reduction in run time.

In our future work, we plan to consider scenarios from the service provider's perspective. Due to service replication, the overloaded local service providers typically assume additional costs to pay the remote servers for helping serve the former's users. We thus aim to extend this study to minimize the overloaded local service provider cost in addition to minimizing the offloading average response time and while still considering the users' and groups' QoS requirements.

Appendix A
KKT Conditions

In the following equations, we apply the KKT conditions on the equality and inequality constraints of the relaxed linear relaxation problem.

- Primal Feasibility conditions:

$$\sum_{u\in\mathcal{U}^i} f_u x^*_{ui} - F^o_{S^o_i} \le 0 \quad \forall S^o_i \in \mathcal{S}_o \tag{10a}$$

$$\sum_{u=1}^{N} f_u x^*_{ui} - F^u_{S^u_i} \le 0 \quad \forall S^u_i \in \mathcal{S}_u \tag{10b}$$

$$\sum_{i\in\mathcal{S}} T^{resp}_{ui} x^*_{ui} - T^{max}_{G_g} \le 0$$

$$\forall G_g \in \mathcal{G}, \forall U^{G_g}_u \in \mathcal{U}^{G_g} \tag{10c}$$

$$\sum_{i\in\mathcal{S}} x^*_{ui} T^{resp}_{ui} - \sum_{i\in\mathcal{S}} x^*_{ji} T^{resp}_{ji} d_{G_g} \le 0,$$

$$\forall G_g \in \mathcal{G}, \forall U^{G_g}_j, U^{G_g}_u \in \mathcal{U}^{G_g} \tag{10d}$$

$$\sum_{i\in\mathcal{S}} x^*_{ui} = 1 \quad \forall U_u \in \mathcal{U} \tag{10e}$$

$$-x^*_{ui} \le 0 \quad \forall S_i \in \mathcal{S}, \forall U_u \in \mathcal{U} \tag{10f}$$

$$x^*_{ui} - 1 \le 0 \quad \forall S_i \in \mathcal{S}, \forall U_u \in \mathcal{U} \tag{10g}$$

- Dual Feasibility conditions:

$$\lambda^*_{local_i}, \lambda^*_{remote_i}, \alpha^*_u, \sigma^*_{guj}, \beta^*_{ui}, and \gamma^*_{ui} \geq 0, \quad \forall S_i \in \mathcal{S}, \forall U_u \in \mathcal{U} \tag{11a}$$

$$\frac{\partial L(X^*, \lambda^*_{local}, \lambda^*_{remote}, \alpha^*, \sigma^*, \beta^*, \gamma^*, \nu^*)}{\partial x^*_{ui}} = 0$$

$$\frac{T^{resp}_{ui}}{N} + \lambda^*_{local_i} f_u + \alpha^*_u T^{resp}_{ui} + \sigma^*_{guj} T^{resp}_{ui} - \beta^*_{ui} + \gamma^*_{ui} + \nu^*_u = 0$$

$$\forall S^o_i \in \mathcal{S}^o, \forall G_g \in \mathcal{G}, \forall U^{G_g}_u, U^{G_g}_j \in \mathcal{U}^{G_g}, j \neq u \tag{11b}$$

$$\frac{\partial L(X^*, \lambda^*_{local}, \lambda^*_{remote}, \alpha^*, \sigma^*, \beta^*, \gamma^*, \nu^*)}{\partial x^*_{ui}} = 0$$

$$\frac{T^{resp}_{ui}}{N} + \lambda^*_{remote_i} f_u + \alpha^*_u T^{resp}_{ui} + \sigma^*_{guj} T^{resp}_{ui} - \beta^*_{ui} + \gamma^*_{ui} + \nu^*_u = 0$$

$$\forall S^u_i \in \mathcal{S}^u, \forall G_g \in \mathcal{G}, \forall U^{G_g}_u, U^{G_g}_j \in \mathcal{U}^{G_g}, j \neq u \tag{11c}$$

$$\frac{\partial L(X^*, \lambda^*_{local}, \lambda^*_{remote}, \alpha^*, \sigma^*, \beta^*, \gamma^*, \nu^*)}{\partial x^*_{ji}} = 0$$

$$\sigma^*_{guj} T^{resp}_{ji} = 0 \quad \forall S_i \in \mathcal{S}, \forall G_g \in \mathcal{G}, \forall U^{G_g}_u, U^{G_g}_j \in \mathcal{U}^{G_g}, j \neq u \tag{11d}$$

- Complementary slackness conditions:

$$\lambda^*_{local_i} \left(\sum_{u \in \mathcal{U}^i} f_u x^*_{ui} - F^o_{S^o_i} \right) = 0, \quad \forall S^o_i \in \mathcal{S}^o \tag{12a}$$

$$\lambda^*_{remote_i} \left(\sum_{u=1}^{N} f_u x^*_{ui} - F^u_{S^u_i} \right) = 0, \quad \forall S^u_i \in \mathcal{S}^u \tag{12b}$$

$$\alpha^*_u \left(\sum_{i \in \mathcal{S}} T^{resp}_{ui} x^*_{ui} - T^{max}_{G_g} \right) = 0, \quad \forall G_g \in \mathcal{G}, U^{G_g}_u \in \mathcal{U}^{G_g} \tag{12c}$$

$$\sigma_{guj} \left(\sum_{i \in \mathcal{S}} x_{ui} T^{resp}_{ui} - \sum_{i \in \mathcal{S}} x_{ji} T^{resp}_{ji} - d_{G_g} \right) = 0, \quad \forall G_g \in \mathcal{G}, \forall U^{G_g}_j, U^{G_g}_u \in \mathcal{U}^{G_g} \tag{12d}$$

$$\beta^*_{ui} x^*_{ui} = 0, \quad \forall S_i \in \mathcal{S}, \forall U_u \in \mathcal{U} \tag{12e}$$

$$\gamma^*_{ui} \left(x^*_{ui} - 1 \right) = 0, \quad \forall S_i \in \mathcal{S}, \forall U_u \in \mathcal{U} \tag{12f}$$

APPENDIX B
PROOF OF THEOREM 1

Using the equations we derived using KKT conditions in Appendix A, we can see from (11d) that $T^{resp}_{ji} \neq 0$, and $\sigma^*_{guj} \geq 0$, and thus $\sigma^*_{guj} = 0$. By substituting in (11b) and (11c), we get:

$$\frac{T^{resp}_{ui}}{N} + \lambda^*_{local_i} f_u + \alpha^*_u T^{resp}_{ui} - \beta^*_{ui} + \gamma^*_{ui} + \nu^*_u = 0 \quad \forall S^o_i \in \mathcal{S}^o, \forall U_u \in \mathcal{U} \tag{13}$$

$$\frac{T^{resp}_{ui}}{N} + \lambda^*_{remote_i} f_u + \alpha^*_u T^{resp}_{ui} - \beta^*_{ui} + \gamma^*_{ui} + \nu^*_u = 0 \quad \forall S^u_i \in \mathcal{S}^u, \forall U_u, U_j \in \mathcal{U}, j \neq u \tag{14}$$

By Multiplying (13) by x^*_{ui} and using (12e), (12f), substitute in the result of multiplication, we get:

$$x^*_{ui} = \frac{-\gamma^*_{ui}}{\frac{T^{resp}_{ui}}{N} + \lambda^*_{local_i} f_u + \alpha^*_u T^{resp}_{ui} + \nu^*_u} = \frac{-\gamma^*_{ui}}{\beta^*_{ui} - \gamma^*_{ui}} \quad \forall S^o_i \in \mathcal{S}^o, \forall U_u, \in \mathcal{U}^j \tag{15}$$

Similarly, by multiplying (14) by x^*_{ui}, and using (12e), (12f), substitute in the result of multiplication, we get:

$$x^*_{ui} = \frac{-\gamma^*_{ui}}{\frac{T^{resp}_{ui}}{N} + \lambda^*_{remote_i} f_u + \alpha^*_u T^{resp}_{ui} + \nu^*_u} = \frac{-\gamma^*_{ui}}{\beta^*_{ui} - \gamma^*_{ui}} \quad \forall S^u_i \in \mathcal{S}^u, \forall U_u, \in \mathcal{U} \tag{16}$$

From (15), (12e), and (12f), we only have two viable cases:

- *Case 1:* $x^*_{ui} = 0$ when $\gamma^*_{ui} = 0$ and $\beta^*_{ui} > 0$, which means that $\frac{T^{resp}_{ui}}{N} + \lambda^*_{local_i} f_u + \alpha^*_u T^{resp}_{ui} + \nu^*_u > 0$.
- *Case 2:* $x^*_{ui} = 1$ when $\gamma^*_{ui} > 0$ and $\beta^*_{ui} = 0$, which means that $\frac{T^{resp}_{ui}}{N} + \lambda^*_{local_i} f_u + \alpha^*_u T^{resp}_{ui} + \nu^*_u = -\gamma^*_{ui}$. Consequently, ν^*_u should be negative.

Otherwise, when $\gamma^*_{ui} = \beta^*_{ui} = 0$, $\gamma^*_{ui} = \beta^*_{ui} > 0$, or $\gamma^*_{ui} > 0$ and $\beta^*_{ui} > 0$, x^*_{ui} has no valid value.

From (16), (12e), and (12f), we only have two viable cases:

- *Case 1:* $x^*_{ui} = 0$ when $\gamma^*_{ui} = 0$ and $\beta^*_{ui} > 0$, which means that $\frac{T^{resp}_{ui}}{N} + \lambda^*_{remote_i} f_u + \alpha^*_u T^{resp}_{ui} + \nu^*_u > 0$.
- *Case 2:* $x^*_{ui} = 1$ when $\gamma^*_{ui} > 0$ and $\beta^*_{ui} = 0$, which means that $\frac{T^{resp}_{ui}}{N} + \lambda^*_{remote_i} f_u + \alpha^*_u T^{resp}_{ui} + \nu^*_u = -\gamma^*_{ui}$. Consequently, ν^*_u should be negative.

Otherwise, when $\gamma^*_{ui} = \beta^*_{ui} = 0$ (because of the limitation of x^*_{ui} when γ^*_{ui} and β^*_{ui} tend to zero), $\gamma^*_{ui} = \beta^*_{ui} > 0$, or $\gamma^*_{ui} > 0$ and $\beta^*_{ui} > 0$, x^*_{ui} has no valid value.

REFERENCES

[1] S. A. Mohamed, S. Sorour, and H. S. Hassanein, "Group-delay aware task offloading with service replication for scalable mobile edge computing," in *Proc. IEEE Glob. Commun. Conf.*, 2020, pp. 1–6.

[2] A. Ahmed and E. Ahmed, "A survey on mobile edge computing," in *Proc. IEEE 10th Int. Conf. Intell. Syst. Control*, 2016, pp. 1–8.

[3] J. Liu, Y. Mao, J. Zhang, and K. B. Letaief, "Delay-optimal computation task scheduling for mobile-edge computing systems," in *Proc. IEEE Int. Symp. Inf. Theory*, 2016, pp. 1451–1455.

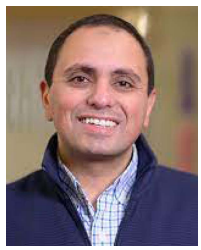
[4] Y. Mao, J. Zhang, and K. B. Letaief, "Joint task offloading scheduling and transmit power allocation for mobile-edge computing systems," in *Proc. IEEE Wireless Commun. Netw. Conf.*, 2017, pp. 1–6.

[5] C.-F. Liu, M. Bennis, and H. V. Poor, "Latency and reliability-aware task offloading and resource allocation for mobile edge computing," in *Proc. IEEE Globecom Workshops*, 2017, pp. 1–7.

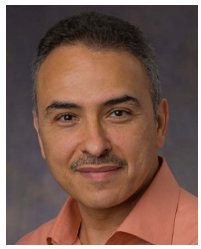
[6] S. Ranadheera, S. Maghsudi, and E. Hossain, "Computation offloading and activation of mobile edge computing servers: A minority game," *IEEE Wireless Commun. Lett.*, vol. 7, no. 5, pp. 688–691, Oct. 2018.

[7] 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, Aug. 2017.
[8] W. Du et al., "Service capacity enhanced task offloading and resource allocation in multi-server edge computing environment," Mar. 2019, *arXiv:1903.04709*.
[9] B. Gu, M. Alazab, Z. Lin, X. Zhang, and J. Huang, "AI-Enabled task offloading for improving quality of computational experience in ultra dense networks," *ACM Trans. Internet Technol.*, vol. 22, no. 3, pp. 1–17, Aug. 2022.
[10] Y. Zhang, X. Chen, Y. Chen, Z. Li, and J. Huang, "Cost efficient scheduling for delay-sensitive tasks in edge computing system," in *Proc. IEEE Int. Conf. Serv. Comput.*, 2018, pp. 73–80.
[11] P. Li, Y. Luo, K. Wang, and K. Yang, "Energy minimization and offloading number maximization in wireless mobile edge computing," in *Proc. IEEE Glob. Commun. Conf.*, 2018, pp. 1–6.
[12] L. Long, Z. Liu, Y. Zhou, L. Liu, J. Shi, and Q. Sun, "Delay optimized computation offloading and resource allocation for mobile edge computing," in *Proc. IEEE 90th Veh. Technol. Conf.*, 2019, pp. 1–5.
[13] S. Maheshwari, D. Raychaudhuri, I. Seskar, and F. Bronzino, "Scalability and performance evaluation of edge cloud systems for latency constrained applications," in *Proc. IEEE/ACM Symp. Edge Comput.*, 2018, pp. 286–299.
[14] U. Yaqub and S. Sorour, "Multi-objective resource optimization for hierarchical mobile edge computing," in *Proc. IEEE Global Commun. Conf.*, 2018, pp. 1–6.
[15] L. Liu, S. Chan, G. Han, M. Guizani, and M. Bandai, "Performance modeling of representative load sharing schemes for clustered servers in multiaccess edge computing," *IEEE Internet Things J.*, vol. 6, no. 3, pp. 4880–4888, Jun. 2019.
[16] R. Beraldi, A. Mtibaa, and H. Alnuweiri, "Cooperative load balancing scheme for edge computing resources," in *Proc. IEEE 2nd Int. Conf. Fog Mobile Edge Comput.*, 2017, pp. 94–100.
[17] A. Ndikumana, S. Ullah, T. LeAnh, N. H. Tran, and C. S. Hong, "Collaborative cache allocation and computation offloading in mobile edge computing," in *Proc. IEEE 19th Asia-Pacific Netw. Operations Manage. Symp.*, 2017, pp. 366–369.
[18] J. Zhang, H. Guo, J. Liu, and Y. Zhang, "Task offloading in vehicular edge computing networks: A load-balancing solution," *IEEE Trans. Veh. Technol.*, vol. 69 no. 2, pp. 2092–2104, Feb. 2020.
[19] J. Ren, G. Yu, Y. He, and G. Y. Li, "Collaborative cloud and edge computing for latency minimization," *IEEE Trans. Veh. Technol.*, vol. 68, no. 5, pp. 5031–5044, May 2019.
[20] J. Zhao, Q. Li, Y. Gong, and K. Zhang, "Computation offloading and resource allocation for cloud assisted mobile edge computing in vehicular networks," *IEEE Trans. Veh. Technol.*, vol. 68, no. 8, pp. 7944–7956, Aug. 2019.
[21] J. Liu, J. Ren, Y. Zhang, X. Peng, Y. Zhang, and Y. Yang, "Efficient dependent task offloading for multiple applications in MEC-Cloud system," *IEEE Trans. Mobile Comput.*, to be published, doi: 10.1109/TMC.2021.3119200.
[22] S. Sundar and B. Liang, "Offloading dependent tasks with communication delay and deadline constraint," in *Proc. IEEE Conf. Comput. Commun.*, 2018, pp. 37–45.
[23] Y. Zhang, "Resource scheduling and delay analysis for workflow in wireless small cloud," *IEEE Trans. Mobile Comput.*, vol. 17, no. 3, pp. 675–687, Mar. 2018.
[24] L. Yuchong, W. Jigang, W. Yalan, and C. Long, "Task scheduling in mobile edge computing with stochastic requests and M/M/1 servers," in *Proc. IEEE 21st Int. Conf. High Perform. Comput. Commun.; IEEE 17th Int. Conf. Smart City; IEEE 5th Int. Conf. Data Sci. Syst.*, 2019, pp. 2379–2382.
[25] [Online]. Available: https://www.mathworks.com/products/optimization.html
[26] B. Doerr, "Linear discrepancy of totally unimodular matrices," *Combinatorica*, vol. 24, no. 1, pp. 117–125, Jan. 2004.
[27] S. Boyd and L. Vandenberghe, *Convex Optimization*. Cambridge, U.K.: Cambridge Univ. Press, Mar. 2004.
[28] R. Cont and M. Heidari, "Optimal rounding under integer constraints," Dec. 2014, *arXiv:1501.00014*.
[29] C. H. Papadimitriou, "On the complexity of integer programming," *J. ACM*, vol. 28, no. 4, pp. 765–768, 1981.
[30] N. Megiddo, " On the complexity of linear programming," *Advances in Economic Theory: Fifth World Congress*, Cambridge, U.K.: Cambridge Univ. Press, Jan. 1987, pp. 225–268.
[31] C. C. Gonzaga, "On the complexity of linear programming," *Resenhas do Instituto de Matemática e Estatística da Universidade de São Paulo*, vol. 2, no. 2, pp. 197–207, 1995.

Shimaa A. Mohamed received the B.Sc. and the M.Sc. degrees in electrical and communication engineering from Alexandria University, Alexandria, Egypt, in 2010 and 2015, respectively, and the Ph.D. degree from Queen's University, Kingston, ON, Canada, in 2021. She is a Researcher with the Informatics Research Institute, City of Scientific Research and Technological Applications. Her research interests include edge computing, cloud computing, Internet of things, and software-defined networking.



Sameh Sorour (Senior Member, IEEE) was an Assistant Professor with the School of Computing, Queen's University, Kingston, ON, Canada, where he led several projects on autonomous and connected vehicles, edge intelligence and wireless networks and services. He was a Editor of the IEEE COMMUNICATIONS LETTERS. His research interests include the broad areas of advanced computing, learning, and networking technologies for cyber-physical, and autonomous systems. Topics of particular interest include cloud/edge/IoT networking, computing, learning, and their applications in multimodal/coordinated autonomous driving, autonomous/electric mobility on demand systems, and cyber-physical systems.



Hossam S. Hassanein (Fellow, IEEE) is currently a Leading Authority in the areas of broadband, wireless and mobile networks architecture, protocols, control and performance evaluation. His record spans more than 600 publications in journals, conferences and book chapters, in addition to numerous keynotes and plenary talks in flagship venues. He is the founder and Director of the Telecommunications Research Lab with Queen's University, Kingston, ON, Canada, School of Computing, with extensive international academic and industrial collaborations. He was the recipient of the several recognition and Best Paper awards at top international conferences, the 2016 IEEE Communications Society Communications Software Technical Achievement Award for outstanding contributions to routing and deployment planning algorithms in wireless sensor networks, and the 2020 IEEE IoT, Ad Hoc and Sensor Networks Technical Achievement and Recognition Award for significant contributions to technological advancement of the Internet of Things, ad hoc networks and sensing systems. Dr. Hassanein is a former Chair of the IEEE Communication Society Technical Committee on Ad hoc and Sensor Networks (TC AHSN). He is an IEEE Communications Society Distinguished Speaker and Distinguished Lecturer (during 2008–2010).