Delay and QoS Aware Low Complex Optimal Service Provisioning for Edge Computing

Taewoon Kim, *Member, IEEE,* Jenn-Wei Lin*, and Chi-Ting Hsieh

***Abstract*—Edge computing can be used as a distributed counterpart of the cloud computing, and task offloading with much shorter delays can be achieved. In general, edge servers are resource-limited, but the services are becoming further diversified to meet the ever-increasing user demand. When an edge server cannot process the received service request of a particular service type for not locally having the corresponding service execution image, the request needs to be forwarded to another on which the particular image is installed. In this paper, an optimal scheduling method that forwards such requests is proposed by using the variant multiple-knapsack framework. Then, to find the optimal solution, we propose to reformulate it as an integer linear program. As the problem size increases, however, the problem easily becomes intractable, so we propose a distributed solution by using both Lagrangian relaxation and decomposition theory. To evaluate the effectiveness of the proposed approaches, we carry out simulations with different network layouts and with different request arrival scenarios. The evaluation results show that the centralized Lagrange dual solution can yield near-optimal solutions. In addition, the decomposed distributed solution can significantly reduce the computational and operational complexity at the expense of the total reward.**

***Index Terms*—Multi-Access Edge Computing, MEC Missing Problem, Service Request Forwarding, Variant Multiple-Knapsack Problem, Lagrangian Relaxation.**

I. Introduction

EDGE computing (EC) is a promising computing paradigm where compute and storage resources are placed at the network edge, in close proximity to end users. Compared to the centralized cloud computing (CC), edge computing has additional benefits such as reduced response time, battery saving at the user devices, bandwidth saving, etc. [1]. Nowadays, edge computing has been incorporated into the mobile cellular network (e.g. 5G), called multi-access edge computing (MEC), by having an edge server (ES) collocated with one or more base stations (BSs) or with radio access networks (RANs) [2]. With such deployment, a substantial amount of data can be stored, and computationally heavy tasks can be processed near the mobile user. In addition, European Telecommunications Standards Institute (ETSI) has initiated the standardization for MEC by launching the MEC Industry Specification Group (ISG) [3].

When providing a mobile data service or offloading service with MEC, each ES (i.e., MEC server) has limited computation, communication, and storage capacities in general [4]. Therefore, it is impossible and inefficient to deploy all the service execution images corresponding to different service types at each ES, and thus, the service image deployment strategies for MEC has been extensively studied [4]–[7]. In addition, user mobility is another challenging issue. Regardless of the mobile user's location, an ES co-located with the BS which the user is associated with is responsible for handling the service requests from the mobile user. For not having installed all the service execute images at ES, the *MEC service missing* (MSM) problem may occur when the responsible ES cannot provide the desired service to the received request from a mobile user. In such a situation, the responsible ES becomes a *missing ES* and the desired service request is a *missed request*.

There are two possible solutions to the MSM problem in general. One is *service request forwarding*, where the missed request is forwarded from the missing ES to another *serving ES* on which the service execution image for the missed request is installed. The other is *service image migration*, where the service execution image is migrated to the missing ES [8]. In general, the migration approaches may cause a service delay and non-trivial amount of data transmissions over a wide area network (WAN); for example, the service image migration for face recognition service takes 247 seconds on a 5 Mbps WAN [9]. Also, it may cause a substantial service delay [8].

In this paper, by utilizing the service request forwarding technique, a dynamic MEC resource provisioning scheme is proposed for a mobile network operator (MNO) as a solution to the MSM problem. In general, each ES is installed with the service execution images of some service types, but not all due to the limited local computing and storage resource. Therefore, the occurrence of missing requests is inevitable, and it becomes important to optimally forward the missed requests to serving ESs so that the expected quality of service (QoS) is fulfilled while reducing the forwarding cost. In contrast to the previous studies focusing only on either minimizing total delay [10] or maximizing the number of forwarded missed requests [11], we We consider both the priority of each request

T. Kim is with the School of Computer Science and Engineering, Pusan National University, Busan 46241 South Korea (e-mail: taewoon@pusan.ac.kr).

J.-W. Lin and C.-T. Hsieh are with the Department of Computer Science and Information Engineering, Fu Jen Catholic University, New Taipei City, 24205 Taiwan (e-mail: jwlin@csie.fju.edu.tw, 410226048@csie.fju.edu.tw).

* Corresponding author: Prof. Jenn-Wei Lin (E-mail: jwlin@csie.fju.edu.tw, Tel: +886-2-29053855, Fax: +886-2-2902-3550, Address: No. 510, Zhongzheng Rd., Xinzhuang Dist., New Taipei City 24205, Taiwan, R.O.C)

This research was supported in part by the National Science and Technology Council, Taiwan, R.O.C, under Grant NSTC 110-2221-E-030-002 and 111-2221-E-030-011, and in part by the National Research Foundation of Korea (NRF) grant funded by the Korea government (MSIT) (No. 2021R1F1A1059109).

(or the QoS requirements) and the transmission delay from forwarding to make the proposed solution more practical. Also, by introducing a tunable objective function considering the two at the same time, the MNO is able to easily adjust the scheduling decision according to its preference.

To optimally solve the missed request allocation/forwarding problem, we first formulate the MSM problem as a variable multiple-knapsack (VMK) problem. The single knapsack problem is a well-known combinatorial optimization problem [12]–[16], whereas the VMK problem is an extension with multiple knapsacks of different capacities. The VMK formulation of the missed request forwarding problem is then transformed into an integer linear program (ILP), with which the optimal solution can be computed with a computer solver. However, the computation complexity of classic knapsack problem is NP-hard [12]–[16], and the VMK problem is even more difficult. Although advanced techniques, such as relaxation and branch-and-bound/cut, can reduce the complexity of the ILP significantly, ILP may not be feasible as an online method especially when the problem size increases (see Section IV-D). Many efforts have been made to reduce the problem complexity by using techniques such as relaxation [11], [23]–[26]. However, they are centralized approaches, and thus they may not scale well with the problem size. To significantly reduce the complexity, we propose a distributed request forwarding scheduling method. In particular, the Lagrangian relaxation (LR) method [17] as well as Decomposition Theory [18] is used in this work to solve the missed request forwarding problem efficiently.

The main contributions of this paper are summarized as follows.

- We study the MSM problem to effectively handle the missed requests in a mobile MEC network, and propose practical missed request forwarding solutions that can run online by assuming ESs have different capacities and the workload (i.e., users requests) is heterogeneous.
- As a solution to the MSM problem, we formulate the optimal missed request forwarding problem by using the VMK framework and then, transform it into ILP. The objective is tunable, and is designed to jointly consider priority and transmission delay of the forwarded missed requests. By using the tunable weight parameters, MNO can make a balance between QoS (priority) or delay.
- To reduce the computation complexity and to make the proposed solution suitable for online processing, we present a systematic approach for a low-complex and real-time scheduling algorithms by using Lagrangian relaxation and decomposition theory.
- We present a distributed MEC management method that effectively selects ES managers and collects the required information to solve the forwarding problem in a decentralized manner. Also, we propose the message structures to efficiently share such information among ESs.
- To show the effectiveness of the proposed solutions, we make a performance comparison of the proposed distributed approach against the optimal solution, conventional algorithms, and state-of-the-art approaches.

TABLE I: Frequently used abbreviations and acronyms

BS	Base Station
CC	Cloud Computing
ES	Edge Server (or MEC Server)
ESM	Edge Server (or MEC Server) Manager
ILP	Integer Linear Programming/Problem
MEC	Multi-access Edge Computing
MNO	Mobile Network Operator
MSM	MEC Service Missing
QoS	Quality of Service
RAN	Radio Access Network
VMK	Variable Multiple Knapsack

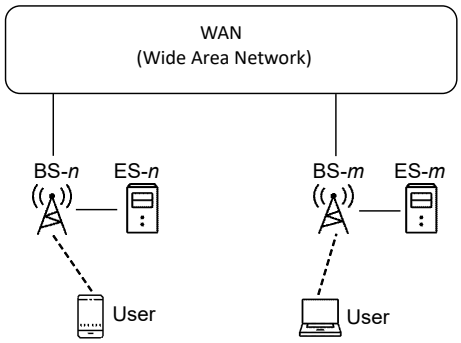


Fig. 1: The assumed system model where an edge server is collocated with a base station.

- We show the reduced complexity and convergence of the proposed approaches, and analyze the results.

The rest of the paper is organized as follows. The assumed system model and related work are introduced in Section II. The proposed missed request forwarding scheme is presented in Section III. In Section IV we evaluate the performance of the proposed approaches and compare them to the optimal, conventional, and state-of-the-art methods. Finally, Section V concludes this paper. Table I summarizes the abbreviations and acronyms that are frequently used throughout this paper.

II. Preliminaries

This section presents the system model of the assumed mobile MEC system. Then, the selected existing studies related to the MSM problem considered in this paper are discussed.

A. System Model

The system model considered in this paper is depicted in Fig. 1. An MNO provides data offloading services of different service types over a large geographic area for its customers (users). The area under service is decomposed into a number of wireless coverage areas, each of which is deployed with a BS to provide wireless connectivity to the users visiting the area. A BS is also equipped (or collocated) with an ES which processes the service requests from the users associated with the BS. We assume the one-to-one relation between BS and its co-located ES. However, when a BS does not have a co-located ES, it forwards the received service requests to the closest ES in terms of the transmission delay. In such case, an ES is shared by multiple BSs. A user follows the closest-BS-first rule for association [21], and the association is assumed

not to change for the period of our interest. BSs as well as the corresponding ESs are connected with each other by a wide area network (WAN).

B. Related Work

There has been a substantial amount of studies on the MSM problem. In [4], the authors proposed a content-centric networking-based MEC service deployment and discovery protocol, consisting of the following two stages. The 1) service initiation stage deploys services in each ES. Thereafter, each of the service requests from users is processed by a 2) discovery procedure to determine an appropriate ES which may be the local ES, a remote ES in the same region, or a remote ES in other neighboring region. The discovery procedure chooses the best ES to forward the received service request to, and it is carried out by checking whether the measured round-trip time (RTT) is within the delay constraint. The resource availability at each candidate ES is important to prevent ESs from being overflow and large queuing/processing delays, but it is not discussed therein. Also, considering different preferences of MNOs, QoS needs to take multiple factors into consideration, but there is no such study in their work.

Sun et al. [19] studied task assignment and offloading for multimedia MEC networks aiming at minimizing power consumption. In their assume system, local processing at the user device and MEC processing (i.e., offloading) mode are supported. Their proposed solution decides on the processing mode and the related resource assignment. Then, the authors proposed a stochastic optimization model, and utilized the Lyapunov technique to decompose the problem for complexity. However, the proposed solution is multimedia application-specific, and only a single BS-ES site is considered in their system model, leaving the scalability issue unresolved. In addition, due to the limited resources in an ES, the overflow problem may occur, which is not discussed therein.

In [10], authors jointly considered the computation offloading, content caching, and network resource assignment for delay-sensitive tasks. In their work, the content caching is that the requested contents (e.g., program codes) can be cached from the Internet to the ES, which help avoid duplicate transmissions of the same content. Their objective is to minimize the total latency of all computation tasks. They proposed a mixed integer nonlinear programming (MINLP) problem formulation, and to obtain the solution the authors proposed an asymmetric search tree and modified branch-and-bound method. In addition, a low-complex solution is proposed by using the generalized benders decomposition method. However, the authors assumed a centralized single BS-ES system model, and thus a further study is required before applying their proposed solution to a practical, large-scale network. While the authors focused only on minimizing the total delay, it may not be the only concern for MNOs, and thus it might be required to incorporate multiple factors into the objective function.

In MEC systems, balancing the computation load among ESs is important to prevent service outage or large queuing/processing delays, and the authors in [20] analytically studied different load balancing approaches. The considered load sharing is carried out by re-directing users requests to other ESs in either random or shortest-queue-length-first manner. The analysis results show that the balanced system can reduce blocking probability and waiting time. However, all the considered methods are designed based on theoretical queues (or multiple-knapsack problems with identical items and knapsacks), and do not fully reflect the physical constraints that a general MEC may encounter such as limited resource at ES, request re-direction cost, etc. Moreover, due to the simplicity of the considered methods, an in-depth study on the optimality of the methods is missing therein.

In [21], authors studied the delay-sensitive MEC resource provisioning and workload assignment problem, which is divided into the following three tasks. The first task decides on the number and the placement of ESs, the second task determines the number and the placement of application instances to deploy on the ESs, and the last task assigns (or forwards) the service requests from users to suitable ESs that can meet the processing and response time constraint. The authors formulated the joint problem as a mixed integer program minimizing the edge server deployment cost, and then decompose it into the delay-aware load assignment and the mobile ES dimensioning problems for complexity reduction. Despite of the reduced complexity by decomposition, their proposed approach is still centralized, and may suffer from significant complexity burden as the network size increases.

Beraldi et al. [22] proposed a cooperative strategy among nearby ESs to overcome their resource limitation. The proposed CooLoad is a load sharing scheme aiming at minimizing both the request blocking probability and the execution delay. In short, upon receiving a service request when its buffer is full, an ES passes the request to a nearby ES. The authors modeled the network as a finite quasi birth and death Markov process, and then studied the advantage of the proposed CooLoad to the cooperation-forbidden counterpart. However, the authors assumed a simple network with only two ESs, and thus, their findings may not be applicable as it is to the general scenarios with more ESs. Also, the authors assumed that the transmission delay between two ES is negligible, which may not be the case in a real network.

More recently, many efforts have been made to jointly optimize service placement and request routing [11], [23]–[26]. Poularakis et al. [23], [24] proposed a joint service placement and request routing problem. If a received service request can be processed by nearby ESs, it is routed to the ESs. However, if there is no such ESs having the corresponding service image or available resource, the user request is forwarded to the centralized cloud. The overall goal is to minimize the access to the centralized cloud. The optimization problem is formulated with combinatorial variables, and then to reduce time complexity, linear relaxation and randomized rounding technique is used to obtain approximated solution. Yuan et al. [25] proposed a low-complex solution to the joint optimization problem by using a two time-scale framework. The less frequent service placement and frequent request routing are carried out by greedy-based approximated method and linear relaxation-based heuristic scheme, respectively.

TABLE II: Commonly used notations and variables

$i \in I$	A particular missed request i in the total set of the total missed requests I
$k \in K$	A particular ES k in the entire set of ESs K
v_{ik}	Forwarding reward v_{ik} from forwarding missed request i to ES k
w_1, w_2	Weights on QoS and delay from forwarding missed requests
x_{ik}	Forwarding decision of missed request i to ES k
r_i	Number of resource units required to process an instance of missed request i
π_k	ES k's resource budget
s_i	Number of instances of missed request i to be scheduled for forwarding

Farhadi et al. [11] applied a similar time-scaled approach with greedy service placement algorithm and linear relaxation-based request scheduling algorithm. Lastly, Chen et al. [26] proposed a Lyapunov-based approach to decompose the joint optimization problem, and then applied the linear relaxation and randomized rounding for complexity reduction. Despite the close-to-optimality performance of the above methods and their decomposition or relaxation-based complexity reduction, they still are centralized method. As a result, a large amount of information may need to be transferred to the decision-making entity. Also, as the problem size grows, the CPU time for optimization may increase as well, preventing real-time scheduling.

The proposed method in this paper overcomes the limitations of the aforementioned studies. First, we propose a low-complex distributed framework for request forwarding that reduces the amount of message exchange for optimization, and runs in parallel, resulting in the huge reduction in CPU time. Second, we propose a tunable objective so that MNO can weigh its preference on different aspects such as QoS and delays from forwarding particular requests. Lastly, we assume a practical-scale, generalized configuration in that ESs have different capacities and the workload (i.e., users requests) is heterogeneous.

III. Proposed Idea

In this section, we present a distributed missed request forwarding solutions to the MSM problem, aiming at maximizing the QoS with respect to the total serviced missed requests while minimizing the total redirection cost. To optimize the missed request forwarding and the two objectives together, we formulate a variant multiple-knapsack (VMK) problem with placement values reflecting the two factors. The VMK problem is the extension of the well-known knapsack problem [12]–[16] with the NP-hard time complexity. To avoid the high computation overhead, we also propose a relaxed, iterative algorithm that can run in parallel by leveraging the Lagrangian relaxation (LR) method [17]. The commonly used notations and variables are defined in Table II.

A. Definitions and Terminologies

Before elaborating the basic idea of the proposed scheme, we first introduce the following definitions.

Definition 1: Service type y_i and its execution image. A mobile MEC system provides a fixed number of services that are available to mobile users. Each service request of a particular service type can be processed only by the ESs with the corresponding service execution image installed.

Definition 2: Missing ES and the missed request of a service type y_i. Due to the resource limitation, each ES is installed with a few service execution images of different service types. When a mobile user transmits a service request of y_i to its associated BS, the co-located ES (referred to as responsible ES) may not have the corresponding service execution image. In such a case, called MSM problem, the ES is called a missing ES and the service request is referred to as missed request with respect to y_i.

Definition 3: Serving ES of service type y_i. An ES having the service execution image of service type y_i is called a serving ES with respect to y_i. For a missed request of y_i received at a missing ES to be properly processed, it should be redirected/forwarded to a serving ES of the particular service type.

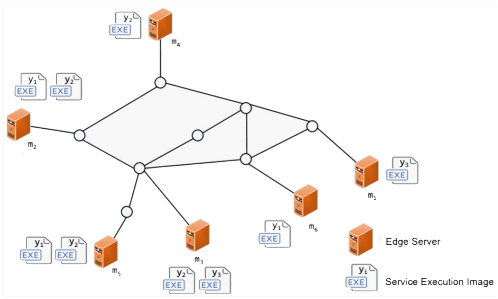
Definition 4: ES Manager, ESM. ESM schedules the forwarding of missed requests, and there can be one or more ESMs chosen by the proposed rule specified in Section III-D. The former is a centralized approach, while the latter is a distributed one. ESMs periodically collect the missed requests and the ES states (i.e., installed service execution images and available capacity) by using the proposed message format defined in Fig. 3.

B. Basic Idea: Variable Multiple Knapsack (VMK) Problem

The proposed approach utilizes the VMK framework to formulate the optimal forwarding problem. Thus, in this section, we briefly introduce VMK and how it is used to schedule the forwarding of missed requests.

Assume a set of items and multiple knapsacks. The goal is to maximize the total values of items placed inside knapsacks. The VMK problem is the extension of the well-known single knapsack problem [12]–[16], which has the following characteristics: 1) There are multiple knapsacks with different capacities, 2) Each item exists at a certain number of instances, and they can be put into different knapsacks. However, a single instance cannot be divided further, and 3) There may exist placement or packing constraints In the case of MSM problem, items and knapsacks in VMK correspond to missed requests and ESs, respectively. The action of packing an item into a knapsack in VMK becomes forwarding a missed request to a serving ES in MSM problem. Also, an ES has limited capacity with service constraints as knapsacks do in VMK.

Fig. 2a shows a small-scale mobile MEC network, where there are six ESs from m_1 to m_6, and they provide some of the three different service types, i.e., y_1, y_2, and y_3. The missing and serving ESs with respect to each service type at a particular time are listed in Fig. 2b. Note that the values therein are arbitrarily chosen. For the service type y_1, missed requests occurred at m_1, m_3, and m_4 since these ESs are not pre-installed with the execution image for y_1. On the other hand, ESs m_2, m_5, and m_6 can act as the serving ES of y_1 since



(a) A small-scale system model.

Service type		y_1	y_2	y_3
Size (no. of resource units) of each request		1	2	2
Missing ESs	ID	m_1, m_3, m_4	m_1, m_6	m_2, m_4, m_5, m_6
	Number of requests received	3, 4, 4	3, 1	5, 4, 5, 2
Serving ESs	ID	m_2, m_5, m_6	m_2, m_3, m_4, m_5	m_1, m_3
	Number of resource units available	5, 7, 1	5, 13, 9, 7	6, 13

(b) The missing and serving ESs in the system.

Fig. 2: An example scenario with six ESs and three service types.

Data structure for edge server state

Edge server ID	Corresponding edge server ID	Service-type provisioning set (i.e., installed imgs)	Capacity (no. available resource units)

Data structure for missed request

Missed request ID	Corresponding missing edge server ID	Associated service type	Number of Instances	Size of each instance

(a) Proposed data structure format.

Missed request ID	Corresponding missing ES ID	Associated service type	Number of instances	Size of each instance
i_1	m_1	y_1	3	1
i_2	m_3	y_1	4	1
i_3	m_4	y_1	4	1
i_4	m_1	y_2	3	2
i_5	m_6	y_2	1	2
i_6	m_2	y_3	5	2
i_7	m_4	y_3	4	2
i_8	m_5	y_3	5	2
i_9	m_6	y_3	2	2

Edge server ID	Corresponding ES ID	Service-type provisioning set (installed imgs)	Capacity size (no. avail. Resource units)
k_1	m_2	$\{y_1, y_2\}$	5
k_2	m_5	$\{y_1, y_2\}$	7
k_3	m_6	$\{y_1\}$	1
k_4	m_3	$\{y_2, y_3\}$	13
k_5	m_4	$\{y_2\}$	9
k_6	m_1	$\{y_3\}$	6

(b) Example of information integration by using the proposed data structures.

Fig. 3: Proposed data structure format and information integration.

they have the execution image for y_1. A missed request may be forwarded to an ES if it has enough available resources and it is installed with the service execution image for the particular request. To optimally forward missed requests, it is required for ESM to collects necessary information and makes decision. Each ESM manages a set of ESs and their request forwarding. Each ES passes i) the missed request information and ii) ES state (i.e., the resource availability and the installed service execution image information) to the corresponding ESM. An ESM can also be a missing/serving ES of a service type.

In short, the overall procedure of the proposed method consists of the following five steps. First, each ESM collects the received missed request, available resource, and available service information from the ES under its control (step 1). ESMs exchange collected information in part or completely with each other by using the proposed data structures in Fig. 3a (step 2). The missed request forwarding problem is formed by using the VMK structure (step 3). The VMK problem is then transformed into an ILP (step 4). Finally, an optimal missed request forwarding schedule is computed by either solving the ILP directly or by using the Lagrangian relaxation (step 5).

C. Detailed Description

In this subsection, we will elaborate on the detailed operations of the proposed scheme.

1) Data Structure Formation: To solve either VMK or MSM problem, the first step is to collect the items/capacities of knapsacks or missed request list/ES state information, respectively, where the ES state refers to the resources availability and the installed service execution image information collected by ESM. In this paper, we propose two data structures to integrate such information as shown in Fig. 3a. Note that for the remainder of this paper item and missed request are used interchangeably depending on the context, and so are knapsack and ES.

In the data structure for ES state, there are two ES IDs. The first ES ID is an shortened ID, arbitrarily assigned by ESM. On the other hand, the following corresponding ES ID is a unique and possibly long ID of the ES, such as IP [4] or MAC address. The short ES ID is used by ESM for easy identification during data integration and scheduling. However, once the forwarding decision is made and announced, each ES needs to forward missed requests to ESs according to the decision. Here is when the unique ES ID is required to actually transfer the missed requests over the network. Service-type provisioning set is the list of the installed service execution images, and the capacity tells the resource availability of the ES. In the data structure for missed request, missed request ID is a short, arbitrarily assigned ID. Corresponding missing ES ID is the unique ID of the ES which first received the request. Associated service type is the corresponding missed request's service type, and number of instances indicates how many instances exists in the request. Note that each instance has different size which is known by the size of each instance field.

Note that one of the ESM can be designated as the ESM leader to perform the information gathering procedure among ESMs. As shown in Fig. 2b and Fig. 3b, there are nine missed request records collected by the three manager ESs, labelled from i_1 to i_9. The first item i_1 corresponds to the missed request of the service type y_1, and the corresponding missing ES of this item is m_1. The number of instances is 3, meaning that there are three missed requests of the same type at the

same ES, or one request consisting of three instances. The instance size of this item is 1, which indicates the number of required resource units for processing each instance of the y_1-type request.

For the case of Fig. 2 with six ESs, there are six ES state records collected by the three manager ESs as shown in Fig. 3b. The first ES state record corresponds to the ES m_2 of which resources can be used for providing service to y_1 and y_2-type requests. With five available resource units in m_2, the capacity size is 5. Note that in Fig. 3b, the particular numbers given to items and knapsacks are arbitrarily chosen.

2) MSM Problem Formulation by VMK Framework: The information collected by using the above data structures can be used as an input to the VMK problem to eventually solve the MSM problem. To do so, an *item-knapsack* bipartite graph, called BG, is formed to represent the placement relationship between the items (missed requests) and knapsacks (serving ES) in the VMK problem by using the following definitions.

Definition 5: A bipartite graph $BG = (I, K, E)$ is a graph whose nodes are divided into two disjoint sets I and K, representing the items (missed requests) and knapsacks (ESs), respectively. Each edge $e \in E$ connects a node $i \in I$ to a node $k \in K$ if the following *collectable condition* is met.
In the context of MSM problem, $i \in I$ indicates a particular missed request i in the entire set I at the moment, $k \in K$ is a particular ES k in the entire set K available at the moment, and $e \in E$ refers to a particular request forwarding decision e among the all feasible decisions available E at the moment.

Definition 6: The collectable condition between i and k. In a BG, if there exists an edge between i and k, item i can be put into knapsack k. In such a case, i is called a collectable item of k. In the context of MSM, an edge indicates that the missed request i can be forwarded to serving ES k for further processing.

Similar to traditional single knapsack problem, the goal of VMK is to maximize the total value of the packed items. For the proposed MSM problem, we apply the same approach, and thus, it is important to define the value of placing a missed request onto a serving ES. In general MEC applications, one important objective is to maximize the QoS by forwarding missed requests. At the same time, minimizing the forwarding cost (e.g., transmission delay) needs to be carefully considered. In this regard, we propose a value function that jointly considers both QoS and forwarding cost as below.

Definition 7: For an edge $e(i,k) \in BG$, an associated edge weight represents the placement value or forwarding reward v_{ik} between missed request i and serving ES k, which is defined as follows.

$$v_{ik} = f_{pri}(pri(i)) \cdot w_1 + f_{trans}(trans(i,k)) \cdot w_2, \quad (1)$$

where f_{pri} and f_{trans} are the normalization or unit-scaling functions, and a higher priority with a lower transmission cost results in a higher forwarding reward. The function $f_{pri}(\cdot)$ normalizes the priority $pri(i)$ of the associated service type of i by $(pri(i) - pri_{min})/(pri_{max} - pri_{min})$, where pri_{max} and pri_{min} are the maximum and minimum priority values among all service types, respectively. Similarly, $f_{trans}(\cdot)$ normalizes the transmission cost $trans(i,k)$ between the corresponding missing ES of i and the corresponding serving ES k as $(trans_{max} - trans(i,k))/(trans_{max} - trans_{min})$, where $trans_{max}$ and $trans_{min}$ are the maximum and minimum transmission costs between any two ESs, respectively. In general, transmission cost is regarded as transmission delay. The parameters $w_1, w_2 \in [0,1]$, where $w_1 + w_2 = 1$, are the contribution ratios or weights associated with priority and transmission cost, respectively.

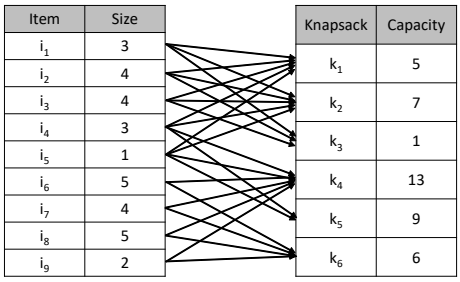


Fig. 4: An example of item-knapsack bipartite graph, where item and knapsack in the context of VMK refer to missed request and ES, respectively, for the MSM problem.

Each forwarding reward v_{ik} takes either a value in $[0,1]$ if k is the serving ES of missed request i (i.e., collectable condition is met) or N/A otherwise (non-collectable). In the evaluation to be shown in Section IV, the forwarding rewards are modified such that ϵ is added to a valid forwarding reward to prevent it from being zero:

$$v_{ik} = \begin{cases} -\infty, & \text{if } v_{ik} \text{ is N/A,} \\ v_{ik} + \epsilon, & \text{otherwise,} \end{cases} \quad (2)$$

where $\epsilon \in \mathcal{R}^+$ is set to 1 in this work, while it can be any strongly positive number. The goal of the proposed MSM (or VMK) problem is to optimally place/forward missed requests in a way that the total forwarding reward is maximized.

Using the given item (missed request) and knapsack (ES) information in Fig. 3b and the above definitions, an item-knapsack bipartite graph is formed as shown in Fig. 4. In the figure, item (missed request) i_1 has the collectable relationship with knapsacks (ES) k_1, k_2 and k_3. As it can be seen in Fig. 3b, the corresponding ESs (m_2, m_5, m_6) of the three knapsacks contain the service image of the associated service type (y_1) of i_1. Therefore, i_1 has three edges to k_1, k_2, and k_3, respectively. The computed edge weights (forwarding rewards) are given in Fig. 5 using the parameter values extracted from the example MEC system in Fig. 2. In Fig. 5, we arbitrarily set $w_1 = 0.9$ and $w_2 = 0.1$ to make the priority (QoS) more important than the transmission cost.

3) ILP Formulation of the MSM Problem: Next step is to convert the VMK formulation of the MSM problem into an equivalent ILP problem to obtain the optimal solution by using a computer solver [15], [16]. The ILP model of a classical (single) knapsack problem has been studied in [12]–[16]. Unlike the single knapsack problem, the VMK problem has the following three distinct points: multiple knapsacks, one or more instances exits per each item, and placement constraints. Thus, we extend the ILP model of a single knapsack problem

Service priority

Service type	1	2	3
Priority	1	2	3
Min. priority	1		
Max. priority	3		

Transmission costs (hop count)

ES	m_1	m_2	m_3	m_4	m_5	m_6
m_1	0	5	4	4	5	3
m_2	5	0	3	3	4	4
m_3	4	3	0	4	3	3
m_4	4	3	4	0	5	4
m_5	5	4	3	5	0	4
m_6	3	4	3	4	4	0
Min. cost	3					
Max. cost	5					

(a) The assumed priority and transmission cost values.

Edge server ID / Missed Req. ID	$k_1(m_2)$	$k_2(m_5)$	$k_3(m_6)$	$k_4(m_3)$	$k_5(m_4)$	$k_6(m_1)$
i_1	0	0	0.1	N/A	N/A	N/A
i_2	0.1	0.1	0.1	N/A	N/A	N/A
i_3	0.1	0	0.05	N/A	N/A	N/A
i_4	0.45	0.45	N/A	0.5	0.5	N/A
i_5	0.5	0.5	N/A	0.55	0.5	N/A
i_6	N/A	N/A	N/A	1	N/A	0.9
i_7	N/A	N/A	N/A	0.95	N/A	0.95
i_8	N/A	N/A	N/A	1	N/A	0.9
i_9	N/A	N/A	N/A	1	N/A	1

(b) The resulting forwarding rewards, where $\epsilon = 0$ to show the raw values.

Fig. 5: The forwarding reward calculation for the missed request forwarding problem derived from the item-knapsack bipartite graph.

for it to be applicable to solving the VMK problem as follows (called P. 3).

$$\max_{\mathbf{x}} \sum_{\forall i \in I} \sum_{\forall k \in K} x_{ik} \times v_{ik} \tag{3a}$$

subject to :

$$\forall k \in K : \sum_{i \in I} x_{ik} \times r_i \leq \pi_k, \tag{3b}$$

$$\forall i \in I : \sum_{k \in K} x_{ik} \leq s_i, \tag{3c}$$

$$\forall i \in I, \forall k \in K : x_{ik} \geq 0, \text{ and integer.} \tag{3d}$$

The objective Eq. 3a is to maximize the total forwarding reward of all missed request instances that are scheduled to be forwarded. Each missed request may have one or more instances, and the decision variable $\mathbf{x} \in \mathbf{Z}^{|I| \times |K|}$ determines the number of instances of missed request i to forward to serving ES k by x_{ik}. It takes a value from 0 to the total instance count s_i of missed request i (see Eq. 3d). The v_{ik} is the corresponding forwarding reward as defined in Eq. 1. The Eq. 3b and Eq. 3c are the missed request and ES constraints. For each missed request i, the number of the forwarded instances cannot be larger than its total instance count s_i. It is allowed that instances of a missed request can be forwarded to different serving ESs. The ES's resource budget constraint (Eq. 3b) prevents the overflow (or over-packing in VMK) situation, and the total amount of the missed request instances to be forwarded to serving ES k cannot exceed the ES's capacity π_k. The r_i is the number of resource units required to process an instance of missed request i.

Decision variable	VMK solution			ILP solution		
x_{ik}	Item	Knapsack	Number of instances placed	Missing ES	Serving ES	No. of processed requests
x_{12}=2	1	2	2	m_1	m_5	2
x_{13}=1	1	3	1	m_1	m_6	1
x_{22}=4	2	2	4	m_3	m_5	4
x_{31}=4	3	1	4	m_4	m_2	4
x_{45}=3	4	5	3	m_1	m_4	3
x_{55}=1	5	5	1	m_6	m_4	1
x_{64}=1	6	4	1	m_2	m_3	1
x_{76}=1	7	6	1	m_4	m_1	1
x_{84}=5	8	4	5	m_5	m_3	5
x_{96}=2	9	6	2	m_6	m_1	2

Fig. 6: The optimal forwarding decision of the missed requests derived from the VMK solution, where the decision variables that are zero are omitted.

The proposed problem formulation does not explicitly express the service type constraints, i.e., a missed request of a particular service type can only be processed at the corresponding serving ES. However, the constraint is in active and hidden in the forwarding reward, v_{ik}. If ES k is not the serving ES of missed request i, v_{ik} takes a value of $-\infty$ by Eq. 2. The optimization problem is formulated to maximize the objective function, and thus, such x_{ik}s will never be selected. In general, the resources related to MEC are multi-dimensional as stated in [26]. For example, a task offloaded to an ES may require CPU, memory, storage, networking, and/or power resources at the ES. However, the proposed ILP is formulated only with a single ES resource constraint, Eq. 3b. Although the constraint will be taken as a CPU cycle limit in Section IV, it is a generalized representation of a resource constraint, and one can easily extend the proposed ILP to the multi-dimensional resource problem by adding constraints similarly formed as Eq. 3b.

With the ILP model P. 3, the optimal solution to the bipartite item-knapsack graph in Fig. 4 derived from the sample MEC network in Fig. 2 with the forwarding rewards in Fig. 5b is given in Fig. 6. Note that by interpreting item and knapsack as missed requests and ES, respectively, the optimal solution can be used for knapsack packing and request forwarding interchangeably. The Fig. 6 shows how the VMK solution from the ILP model is converted to the optimal missed request forwarding decision. For example, $x_{12} = 2$ is an action of putting item $i = 1$ (i_1) into knapsack $k = 2$ (k_2). By inspecting the collected information at ESM, it is found that item $i = 1$ (i_1) corresponds to missed request i_1 which was buffered at ES m_1 with three instances of size 1 and service type y_1. Similarly, knapsack $k = 2$ (k_2) is the ES m_5 with installed y_1 and y_2 services and with 7 resources available. As a result, from the request forwarding perspective, $X_{12} = 2$ is a forwarding schedule of missed request i_1 for 2 instances from missing ES m_1 to serving ES m_5.

4) De-centralization by Lagrangian Relaxation and Decomposition: Unfortunately, the proposed ILP in P. 3 is intractable in general, and the complexity sharply increases with the network size and the number of the missed requests. To reduce the CPU time and to make the solution available in real-

time, we propose to decompose the original ILP P. 3 into low-complex subproblems by using Lagrangian relaxation and Decomposition Theory. The final outcome of this section is a general distributed request forwarding scheduling method that can run in parallel and can be applicable to a range of similar problems with little modification.

The first step is to decompose P. 3 into low-complex per-ES problems that can be solved in parallel. Then, it will be grouped into a fewer per-ESM problems for efficient data gathering and forwarding optimization. However, one cannot decompose P. 3 into per-ES/knapsack subproblems as it is due to the coupling constraint in Eq. 3c. By relaxing the constraint, one can form the Lagrangian as follows.

$$\max_{\mathbf{x}} \ \sum_i \sum_k x_{ik} \times v_{ik} - \sum_i \lambda_i (\sum_k x_{ik} - s_i) \quad (4a)$$

$$\text{subject to : Eq. 3b and 3d}$$

$$\forall i: \ \lambda_i \geq 0, \quad (4b)$$

where $\lambda \in \mathbf{R}_+^{|I|}$. By rearranging Eq. 4a, one can get:

$$L(\mathbf{x}; \lambda) = \sum_k \sum_i (v_{ik} - \lambda_i) x_{ik} + \sum_i \lambda_i s_i. \quad (5a)$$

Also, let

$$L_k(\mathbf{x}_k; \lambda) \ = \ \sum_i (v_{ik} - \lambda_i) x_{ik}, \quad (6)$$

where $\mathbf{x}_k \in \mathbf{Z}^{|I|}$. As it can be seen in Eq. 6, for each ES k, λ_i is subtracted from v_{ik}, and it can be interpreted as the price (or cost) for handling item i. The value of λ_i will be iteratively updated as shown below, by which the distributed solution can yield an effective sub-optimal solution.

The coupling constraint has been relaxed, and thus, the entire problem in P. 4 can be decomposed into two-level problems: one master problem (i.e., higher-level problem) and $|K|$ subproblems (i.e., lower-level problems), where K is the set of ESs. At the lower level, each knapsack k solves the following ILP for the given λ (called P. 7).

$$\mathbf{x}_k^*(\lambda) = \ \arg\max_{\mathbf{x}_k} L_k(\mathbf{x}_k; \lambda) \quad (7a)$$

$$\text{subject to : } \sum_i x_{ik} \times r_i \leq \pi_k \quad (7b)$$

$$\forall i: \ x_{ik} \geq 0, \text{integer}. \quad (7c)$$

Then, the master dual problem solves the following problem (called P. 8):

$$\min_{\lambda} \ g(\lambda) = \sum_k g_k(\lambda) + \sum_i \lambda_i s_i \quad (8a)$$

$$\text{subject to: } \lambda_i \geq 0, \forall i, \quad (8b)$$

where $g_k(\lambda) = L_k(\mathbf{x}_k^*, \lambda)$.

The lower problem P. 7 is a traditional single-knapsack problem, and the well-known dynamic programming approach can efficiently find the optimal solution in $O(S \times T)$ where S is the number of elements and T is the capacity of the knapsack. The master problem can be efficiently solved with a subgradient method with the following update rule for λ.

$$\lambda_i^{t+1} = [\lambda_i^t - \alpha^t (s_i - \sum_k x_{ik}^*(\lambda^t))]^+, \quad (9)$$

where t is the iteration counter, α is the step-size, and $[\cdot]^+ = \max(0, \cdot)$. Although there are various ways to update the step-size α such as $\alpha^t = \frac{1+m}{t+m}$ for $m \in \mathbb{R}_+$ [18], the following rule that has proven to be effective in practice [27] is used in this work.

$$\alpha^t = \beta^t (g(\lambda^t) - f^*) / \sum_i ||e_i||^2, \quad (10)$$

where $\beta^t \in (0, 2]$, f^* is the best known objective value of the original problem, and $e_i = s_i - \sum_k x_{ik}^*(\lambda^t)$. In general, β^t is set to 2 at the beginning, and then divided by two if $g(\lambda^t)$ does not change for several iterations. The study on the convergence with the update rule Eq. 10 is given in [28].

The formulations P. 7 and P. 8 represent the fully distributed (referred to as FD) approach since every ES on the network computes its own solution. However, this is not a desirable approach for the following two reasons. First, FD incurs heavy message exchanges for updating λ. Second, FD is equivalent to local greedy algorithm, and it is likely to yield either impossible (e.g., trying to forward more items than there really are) or inefficient solution for not considering other ESs when computing the solution.

As aforementioned, there are ESMs on the assumed network, and we propose to partition the problems in a way that each ESM solve a subset of the entire problem in a distributed, parallel manner. In other words, after clustering ESs into multiple groups, and each ESM (i.e., the corresponding cluster head) makes forwarding decision. Let H be the index set of the ESM on the network with $h \in H$ and $|H| \leq |K|$, where $|K|$ is the number of ESs on the network. Let $\eta(h)$ be the index set of ESs that are under control of the ESM h. To reformulate the FD into the per-ESM problems, let us rewrite the Eq. 6 as follows.

$$L_{\eta(h)}(\mathbf{x}_{k \in \eta(h)}; \lambda) \ = \ \sum_{k \in \eta(h)} \sum_i (v_{ik} - \lambda_i) x_{ik}. \quad (11)$$

Likewise, both P. 7 and P. 8 are rewritten such that a group of per-ES problems are combined into one, resulting in $|H|$ number of lower problems instead. Note that the Lagrangian relaxation-based approach (called LR hereafter) results in sub-optimal solutions, although there are some cases where the optimality gap is zero as discussed in Section IV-B.

D. Manager Selection

As aforementioned in Defn. 4, the ESM is in charge of collecting both the missed request and ES state information from the set of ESs under its control. ESMs can be arbitrarily chosen, but we propose to choose such ESs that the mean transmission delay from the chosen ES to the rest ESs are minimized. The $|H|$ is the number of ESMs determined by the MNO and $|K|$ is the number of ESs with $|H| \leq |K|$. Once ESMs are chosen, the set of ESs to be under control

of an ESM is determined by the shortest transmission delay between ES and ESM.

Let $D = [d_{ij}]$ be the one-hop transmission delay matrix between adjacent ESs, where the delay between the two adjacent ES i and j is denoted by d_{ij}. By using the Dijkstra's shortest path algorithm, the matrix $C = [c_{ij}]$ representing the delay of the shortest path from source ES i to destination ES j can be computed. The i-th row, c_{i*}, represents the delays from ES i to the rest ESs on the network with c_{ii} being 0. The mean delay from ES i to the rest is $\bar{c}_i := \sum_j c_{ij}/(|K| - 1)$. Finally, the proposed method chooses 1-st, 2-nd, $\cdots$, $|H|$-th smallest values among $\bar{c}_i, \forall i$, and the corresponding ESs become the ESMs. Note that if the number of managers is one, `LR` becomes the Lagrange dual of `ILP`, called `LR`(c) or centralized `LR`. On the other hand, if there are more than one managers, it is called `LR`(d) or distributed `LR`. In the evaluation, we assumed that the ESM with the smallest index takes charge of the master dual problem, i.e., updating λ. If there is only a single ESM, it will take care of both lower and master problems.

IV. Evaluation

To evaluate the performance of the proposed method, we have implemented the simulation environment and the proposed methods with Matlab [29], CVX [30], and MOSEK [31]. The Matlab is used to construct and emulate the network and to generate service requests from users. CVX is used for modelling/formulating the proposed optimization problems, and the MOSEK solver computes the optimal solutions. Note that MOSEK solves various optimization problems including mixed integer programs. A high-performance workstation with Intel(R) Core(TM) i9-10940X CPU and 128GB RAM is used for simulation and performance evaluation.

A. Configurations

We assume an R-by-R grid network consisting of $R \times R$ BS routers regularly located at the intersections [24]. Each link connecting two adjacent routers has a unitless random delay of Uniform$[1, 2]$. To be specific, a link delay is a sum of 1 and Uniform$[0, 1]$. The former captures the relatively stable nature of both propagation and transmission delays. As stated in [4], the delay can be configured based on the physical distance. On the other hand, the latter accounts for the randomness in queueing and processing delays at the router. Due to the evenly-spaced configuration of routers as well as the complicated dynamics in channel condition, network congestion and processing load, such delay modelling can be a reasonable approximation of the delays for the assumed system (see [4] for the detailed modeling of transmission delays). The number of ESs $|K|$ and their locations on the grid are randomly chosen. If an ES is located at the location of a particular router, both are co-located and the in-between delay is assumed to be zero. The list of services types $\{1, 2, \cdots, Q_{type}\}$ to be supported in the network are known to all ESs and the users, and a randomly chosen set of service executions images is installed on each ES. Each randomly placed user [10] is associated with the closest BS/ES [21], and sends a random number of requests associated with randomly chosen service types. The arrivals of user requests follow Poisson distribution [25], Poi(Uniform$[0, Q_{rate}]$). For the users under the coverage of the BS without equipping an ES, their requests are forwarded to the closest ES. The priority of each service type (QoS) is randomly chosen in $\{1, 2, \cdots, Q_{pri}\}$. Throughout this section, Q_{pri} is set to 3 to reflect the classification of priority into low, medium, and high defined in [32].

The instance size (i.e,. the number of resource block required to process an instance) of a request of each service type is randomly chosen in $\{1, 2, \cdots, Q_{sz}\}$. The available capacity of each ES is randomly chosen in $\{1, 2, \cdots, Q_{capa}\}$. As aforementioned in Section III-C3, the resource budget constraint Eq. 3b is a generalized representation, and it can be easily extended to multiple budget constraints if multi-dimensional resources are considered. In this section, we assume processing-centric MEC environment where user requests and the corresponding responses are negligibly small in bit length. Thus, available capacities and required resources in this section indicate the CPU cycles available at ESs and the required CPU cycles to process each instance of missed request, respectively. CPU budgets are discretized into equal-sized blocks, and thus, they can be represented by integers.

In what follows, each subsection considers different network layouts, and the configuration values to be used for each layout therein will be specified at the beginning of each subsection. Note that the most configurations and parameters defined above are reflecting the physical characteristics [4], widely used values [10], [24], [25], or borrowed from related standards [32]. The others are randomly selected assuming general use cases.

There are nine approaches that are implemented and compared with each other as below. `ILP`(c) and `LR`(c,d), are the proposed ones, whereas the rest are for performance comparison. Among the proposed approaches, we propose to use `LR`(d) for its low-complexity and a small optimality gap. Note that (c) and (d) refers to centralized and distributed, respectively. Centralized is the case when there is a single entity with the global information makes a global decision, whereas the distributed is the case when a small number of entities with limited information make local decisions concurrently and individually. Obviously, the centralized approaches will yield better solutions compared to the distributed counterparts. However, the centralized `ILP`(c), `LR`(c) and `LRRS`(c) solve an optimization problem, and their computation complexity can be high as the network size increases. Also, centralized approaches require the global knowledge, which may incur heavy and frequent data exchange. On the other hand, `LR`(d) solves partitions of the problem in parallel by ESMs. The reduced problem size as well as the parallelism helps reduce the computation time. After each iteration, `LR`(d) exchanges both optimal solutions of sub-problems and λ, but their size is small.

1. `ILP`(c): The proposed centralized optimal method using integer programming.
2. `LR`(c), `LR`(d): The proposed sub-optimal solutions using the Lagrangian relaxation and problem decomposi-

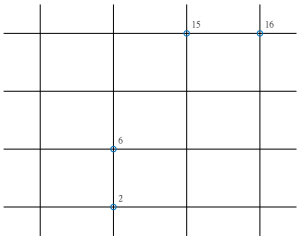


Fig. 7: A 4-by-4 small-scale network layout with four ESs.

tion. The dual variables λ are randomly initialized by Uniform$[0, 1]$ at the beginning. Note that LR iteratively updates the dual variables, and it is configured to run for a limited number of iterations.

3. RND(c), RND(d): Random approaches that assigns each missed request to a randomly chosen ES if it can process.
4. GRD(c), GRD(d): Greedy approaches that processes the missed request yielding the highest forwarding reward first.
5. LRRS(c): The request forwarding approach proposed in [11]. The authors proposed integer programming based request scheduling method, which is followed by linear relaxation and rounding for complexity reduction. Their objective is to maximizes the number of processed requests by forwarding.
6. LRRS(c)-v: A modified version of LRRS(c) for fair comparison. Instead of maximizing the request forwarding, LRRS(c)-v is modified to maximizes the total forwarding reward as ILP(c) and LR(c,d) do.

The weights are set to $w_1 = 0.5, w_2 = 0.5$ for ILP(c), LR(c,d), GRD(c,d), LRRS(c)-v to keep a balance between priority and transmission cost, while RND(c,d) and LRRS(c) has nothing to do with the weights during operation. However, when comparing the total forwarding reward, the same weight will be assumed for RND(c,d) and LRRS(c). In what follows, each subsection assumes a particular network layout with a particular set and placement of ESs. Then, the evaluation is carried out for 10 times with 10 different *scenarios*, i.e., 10 different request arrivals from users. Although each subsection assumes a particular network deployment, due to the randomness in ES deployment, user deployment and request arrivals, the results therein can be taken as representative across similar network sizes based on our experience.

B. Sub-Optimality Analysis on a Small-Scale Network

We first begin with a small-sized network as in Fig. 7 to study the sub-optimality of LR(c,d) compared to ILP. On the network, there are four ESs labeled 2, 6, 15 and 16. Although these are the IDs of routers to which each ES is directly attached, they also will be used as ES IDs and corresponding ES IDs for convenience. For LR(d), it is configured that all ES are managers. Although this is an extreme, fully-distributed case, due to the small number of ESs existing on the network, the optimality gap does not increase much as shown below.

TABLE III: The missed request versus ES forwarding reward matrix V and the number of instances of each missed request received at the missing ES for the first scenario.

V				no. of instances to schedule
$-\infty$	$-\infty$	$-\infty$	1.4500	2
$-\infty$	1.9923	1.9182	$-\infty$	3
1.0923	$-\infty$	1.0460	1.0279	1
$-\infty$	$-\infty$	1.9460	$-\infty$	1
$-\infty$	$-\infty$	$-\infty$	1.4779	5
$-\infty$	1.0460	$-\infty$	$-\infty$	2
$-\infty$	$-\infty$	$-\infty$	1.55	5
$-\infty$	$-\infty$	2.0000	$-\infty$	3
$-\infty$	1.9279	2.0000	$-\infty$	3

TABLE IV: Optimal solution of both ILP(c) and LR(c,d) for the first scenario, where each row and column indicate missed request and serving ES, respectively, and the entries with the value of 0 are left empty.

	3		
1			
		1	
			4
			5
		3	
	2	1	

Also, Q_{type}, Q_{rate}, Q_{pri}, Q_{sz}, and Q_{capa} is configured to 5, 3, 3, 3, 15, respectively.

Consider the forwarding reward matrix $V \in \mathbb{R}^{|I|\times|K|}$ in Table III, where nine rows and four columns indicate nine missed requests and four ESs (i.e., ES 2, 6, 15 and 16 corresponding to the column indices 1, 2, 3, 4, respectively), respectively. For example, $V(1, 4)$ is the forwarding reward when missed request 1 is forwarded to ES 16 to process. On other hand, missed request 2 cannot be forwarded to ES 2, resulting in $-\infty$ in V since the corresponding service image is not installed on the ES. Due to the simplicity of the problem, both ILP(c) and LR(c,d) took a trivial amount of time to compute the optimal solution. In this particular problem setting, ILP(c) and LR(c,d) result in the same objective value of 34.53 and the same optimal decisions as shown in Table IV. From the number of instances to schedule in Table III and the optimal decision in Table IV, it can be seen that the Lagrangian relaxation based methods do not violate the constraints (Eq. 3c) by not allocating any extra (i.e., non-existing) instances. In addition, some request instances are not to be forwarded to other ESs. For example, none of the the missed requests at the first and sixth rows in Table III is forwarded, whereas for the missed requests at the fifth row, only four instances out of five are re-directed to other serving ES. Under the practical scenarios where ESs are resource-limited, such selective no-forwarding (or partial-forwarding) decisions are unavoidable especially when the available resources at ESs are not sufficient to process the entire missed request instances.

In particular, LR(c,d) is a sub-optimal solution and thus, its solution is not as good as ILP(c) in general. However, in this particular scenario, the optimality gap was zero. The reason is that the majority of the high-rewarding missed requests did not

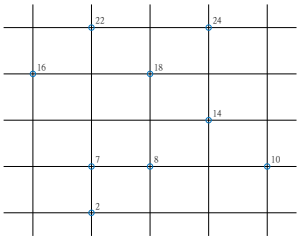
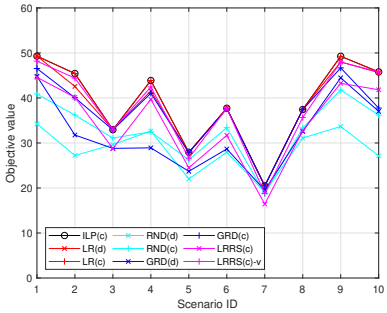
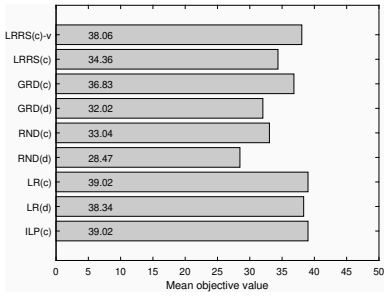


Fig. 8: A 5-by-5 small-scale network layout with nine ESs.



(a) Per-scenario objective values for the small-scale network scenarios



(b) Mean objective values for the small-scale network scenarios

Fig. 9: Comparison of the objective values on the small-scale network scenarios.

overlap among ESs. For example, for the ES 16 corresponding to the fourth column in V, the most high-rewarding missed request is the one at the 7th-row in V, which is not the case to the rest ESs. Thus, it decided to process all instances of it. The second high-rewarding missed request is the one at the 5th-row in V, which is also not the case to the rest ESs. Thus, it decided to process it as much as it can.

On the other hand, the case for ES 6 and 15 corresponding to the second and third column in V, respectively, is quite complicated. For ES 6, the most high-rewarding missed request is the one at the second row in V and thus, it decided to process all instances of it. For the remaining capacities, it decided to process the last missed request as much as it can, resulting in taking two instance of it. However, the last missed request is one of two most valuable missed request to ES 15; the last two missed requests have the same forwarding reward of 2 which is the highest among all for ES 15. Thanks to the λ update, however, the last missed request is penalized more than the second to the last missed request. Thus, to ES 15, the second to the last missed request has become the most high-rewarding. In this way, by updating λ, the most high-rewarding missed request to one ES has become less valuable to the rest ESs. Thus, each ES's individual decision making process also resulted in the global optimal solution that is equivalent to that of ILP(c).

In general, however, LR-based approaches result in sub-optimal solutions compared to ILP(c). We have carried out simulations on 10 different request arrival scenarios on the same network configuration, and the mean optimality gap of LR(d) was 0.86 with standard deviation of 0.80, which amounts to 3.31% of the mean optimal objective value of ILP. However, the optimality gap between LR(c) and ILP(c) was close to zero for all scenarios considered.

C. Performance Comparison on Small-Scale Networks

In this section, both the network size and the number of ES are slightly increased as shown in Fig. 8. The number of ESMs are set to three (ES 7, 8 and 18) by the rule mentioned in Section III-D. The other network parameters remain the same as above, and the iteration limit for LR(c,d) is 10. The objective values computed for the ten different different scenarios (i.e., different request arrivals) are depicted in Fig. 9.

Excluding ILP(c), the LR-based approaches outperformed the rest, and LR(c) yielded the optimal solution. The optimality gap of LR(d) was on average 0.68, which is only 1.74% lower than ILP(c). Such a small performance degradation proves the effectiveness of the proposed LR(d). Besides, in the considered scenarios, GRD(c) has shown only a small amount of performance degradation compared to the optimal solution on average. However, in some scenarios, such as Scenario #10, the difference was large. The rest methods, GRD(d) and RND(c,d), recorded a large amount of performance degradation, indicating their inadequacy as a solution to MSM.

From the viewpoint of the total forwarding reward maximization, LRRS(c) performs worse than the proposed ones since it is designed to maximize the number of request forwarding with no consideration of QoS or forwarding delay (i.e., forwarding reward). For a fair comparison, LRRS(c)-v is used after a minor modification to LRRS(c). LRRS(c)-v outperforms the heuristic algorithms, but the mean performance is strictly lower than LR(c) and comparable to LR(d). Note that LRRS(c)-v is a centralized approach, meaning that although both LRRS(c)-v and LR(d) have recorded a similar performance to each other, LR(d) has advantages such as low computational complexity and reduced information exchange which are not evident in Fig. 9.

Let *Distribution Price* (DP) be the difference in the mean objective values between a centralized approach and its distributed counterpart, and it quantifies the performance loss due to being distributed for a method. Obviously, the centralized approach always outperforms for having the global knowledge while incurring heavy message exchange and large computation complexity. The DP for LR is only 0.68 (1.74%), whereas that for GRD and RND were relatively large, i.e., 4.81 (13.06%) and 4.57 (13.83%), respectively. This shows that despite of the absence of the global information, the proposed LR(d) can

TABLE V: Performance comparison for the small-sized network averaged over 10 scenarios.

Algo.	Priority	Delay	Fwd. MRs
ILP(c)	36.50	3.05	23.80
LR(d)	36.00	3.10	23.40
LR(c)	36.50	3.05	23.80
RND(d)	27.50	4.44	18.80
RND(c)	32.60	4.56	21.60
GRD(d)	31.60	3.83	19.60
GRD(c)	34.70	3.00	22.20
LRRS(c)	34.40	4.55	22.20
LRRS(c)-v	35.50	2.95	23.10

achieve similar performance to LR(c). However, the limited local view restrains the performance improvement for RND and GRD much.

Table V summarizes the performance with respect to the following criteria:

- Priority: Sum of the priority values of the processed/ forwarded missed requests averaged over 10 scenarios.
- Delay: Mean delay of forwarding a chosen missed request averaged over 10 scenarios.
- Fwd. MRs (Forwarded Missed Request): Total count of the missed requests that are chosen to be forwarded averaged over 10 scenarios.

The larger values are the better for both Priority and Fwd. MRs, while the lesser is the better for Delay.

Both ILP(c) and LR(c) show the identical and optimal performance in all criteria, which proves that combining with the results shown in Fig. 9 the two approaches have resulted in the identical forwarding decision. The optimality gap of LR(d) we have seen in Fig. 9 becomes clear by the table Table V in that LR(d) is slightly outperformed by both ILP(c) and LR(c) in all three criteria. LRRS(c) is much outperformed by the proposed approaches since it focuses only on maximizing the number of missed request forwarding. On the other hand, the modified LRRS(c)-v has shown similar performance to the proposed approaches in all three criteria. Although LRRS(c)-v has achieved the shortest delay, it is because of the smaller number of forwarded requests compared to the proposed approaches, resulting in a slightly lower performance. The reason for the lower performance is due to the rounding and sequential algorithm used in LRRS. LRRS solves the linear-relaxed optimization problem, and them applies rounding operations. Then, determines the actual number of requests to forward by visiting each entry in the decision one by one. The rounding operation can be harmful especially when there are multiple forwarding candidates with the same reward. Also, the sequential operation may cause performance degradation when the order of visiting matters. GRD(c) has yielded a short delay as well, but its improvement compared to ILP(c) and LR(c,d) is not significant. Also, in all other criteria, GRD(c) is outperformed by ILP(c) and LR(c,d). The randomized approaches, RND(c) and RND(d), recorded the lowest performance in all criteria among the centralized and distributed algorithms, respectively.

Due to the problem simplicity, all considered algorithms took a negligibly small amount of time to solve. A close look at CPU time and convergence will be studied in the following subsection for a larger network layout with more ESs.

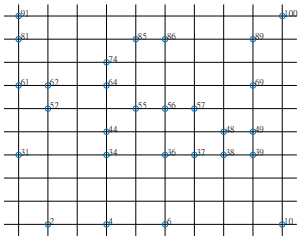
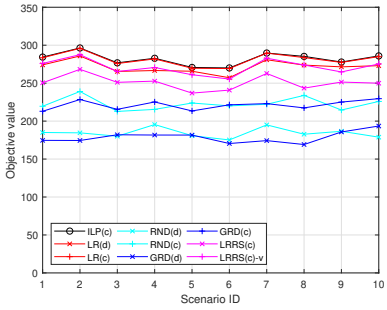
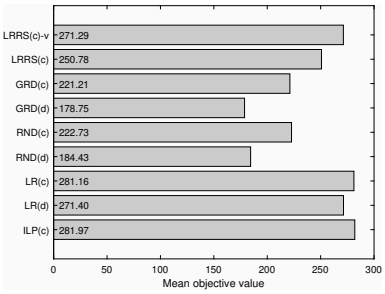


Fig. 10: A 10-by-10 large-scale network layout with 28 ESs.



(a) Per-scenario objective values for the large-scale network scenarios



(b) Mean objective values for the large-scale network scenarios

Fig. 11: Comparison of the objective values on the large-scale network scenarios

D. Performance Comparison on Large-Scale Networks

A larger network of a 10-by-by grid is considered in this section with 28 randomly generated ESs as shown in Fig. 10. The number of ESMs are 3, i.e., ES 55, 56 and 64. The iteration limit for LR(c) remains to be 10 as before, while that for LR(d) is tripled (i.e., 30). Such limits are found by repeated empirical studies, and they can vary with the problem size. Q_{type} is doubled (i.e., 10), while the rest parameters remain the same as before. The objective values computed across 10 different scenarios are shown in Fig. 11.

Similar to the case of the small-scale network, LR(c,d) outperforms the rest, except ILP(c). Although LR(c) has yielded a sub-optimal solution this time, the optimality gap is trivial (i.e., 0.29%). On the other hand, the sub-optimality of LR(d) has become noticeable, and the degradation in the objective value is observed in all of the considered scenarios. The LR(d) lets each ESM solve a partition of the entire problem with limited

TABLE VI: Performance comparison for the large-sized network averaged over 10 scenarios.

Algo.	Priority	Delay	Fwd. MRs
ILP(c)	310.50	4.65	174.90
LR(d)	290.40	4.32	169.10
LR(c)	310.20	4.75	174.60
RND(d)	218.00	9.06	121.50
RND(c)	253.30	8.86	148.40
GRD(d)	237.60	4.37	101.60
GRD(c)	284.40	4.13	127.20
LRRS(c)	282.20	9.17	168.90
LRRS(c)-v	298.40	4.76	168.70

TABLE VII: Mean CPU time (the value of zero indicates that the measured time is negligibly small).

	Time (seconds)	
Algo.	Mean	Std. dev.
ILP(c)	35.33	90.57
LR(d)	0.01	0.00
LR(c)	0.08	0.00
RND(d)	0.00	0.00
RND(c)	0.01	0.00
GRD(d)	0.00	0.00
GRD(c)	0.02	0.00
LRRS(c)	0.16	0.01
LRRS(c)-v	0.16	0.01

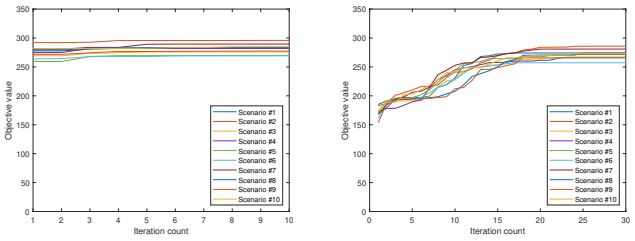
information. Thus, it can happen that some ESs are under-/over-utilized. The increased problem size can aggravate such inefficiency in solution, but the performance degradation from such cases is not significant. LRRS(c) is much outperformed by the proposed methods and LRRS(c)-v for having a different objective. On the other hand, LRRS(c)-v has resulted in a lower performance than ILP(c) and LR(c) and a slightly lower objective value than LR(d). Although the performance gap between LRRS(c)-v and LR(d) is insignificant, considering the advantages of being distributed, LR(d) is more applicable for practical use. An interesting finding is that unlike the previous small-scale network cases, the objective values of GRD(c,d) have become similar to that of RND(c,d). It implies that greedy approaches become inefficient as the network size increases.

The DP for LR is 3.47%, while that for GRD and RND was 23.75% and 17.19%, respectively. In all of the considered approaches, DP has increased compared to the small-scale network cases, meaning that the absence of the global information has degraded the performance of the distributed approaches much larger. Table VI summarizes the performance of the considered algorithms with respect to total priority, per-missed request forwarding delay, and the number of forwarded missed requests. As it has been observed in Fig. 11, LR(c) has yielded a sub-optimal solution, and thus, for both priority and forwarded missed requests, ILP(c) outperforms LR(c). Although LR(d) has shown an enhancement in delay compared to ILP(c), it cannot be solely taken to evaluate the effectiveness of the solution. Although LRRS(c) focuses only on forwarding as many missed requests as possible at the expense of delay, it is outperformed by the proposed approaches in that particular metric. LRRS(c)-v, on the other hand, has achieved better performance than LRRS(c). The different characteristics between GRD and RND have become clearer in the large-scale network with more missed requests. The GRD tends to assign high-rewarding missed requests first, and thus, it resulted in assigning a smaller number of missed requests with high priority and low delay. On the other hand, RND randomly and exhaustively assigns missed requests. As a result, the delay has become larger, and it has yielded a smaller the total priority while forwarding more missed requests.

The Table VII shows the CPU time to solve the considered problems. As expected, ILP(c) recorded the highest CPU time to solve the optimization problem. The average value is 35.33 seconds, and thus, it cannot be used for online processing. Also, the runtime depends largely on the problem size yielding a large standard deviation, which was not the case to the rest approaches. The two LRRS approaches have recorded 0.16 seconds of CPU time to compute the optimal solutions. Although the runtime is larger than that of the proposed Lagrangian-based methods, LRRS can still be used as an online scheduling method in some applications. As the network size increases, however, the complexity of the centralized LRRS methods also increases, resulting in the increased CPU time. In addition, LRRS has one-by-one iterative procedures which consumes much time especially when there are a large number of ESs and missed requests. On the other hand, both GRD and RND which simply iterate over all missed request-ES pairs have terminated quickly in all scenarios. The LR(c,d) approaches can also find solution in a short amount of time, suggesting the possibility of utilizing them as online algorithms. In addition, each ESM in LR(d) collects only a part of the information, and takes a smaller amount of time to return the solution due to the reduced problem size.

E. Complexity and Convergence

Both RND and GRD are iterative, polynomial algorithm whose complexity is proportional to $O(|I| \times |K|)$ RND(c) and GRD(c), where as that of distributed ones are smaller due to the reduce problem space. LRRS(c) and LRRS(c)-v are linear programming, and both have the identical complexity. Assuming the interior point method as a solution method, typical linear programmings have the complexity $O(n^{3.5}L)$, where $n = dim(X) = |I| \times |K|$, and L is the length of the binary-coded input data [33], [34]. The proposed LR solutions consist of a master problem and multiple lower problems. The master problem can be solved by a simple calculation in polynomial time as stated in Eq. 9. Assuming ESs are clustered and each ESM solves the forwarding schedules for multiple ESs, the lower problem P. 7 is the reduced NP-hard [6] assignment problem, whereas the ILP(c) in P. 3 is the full-sized, complete ILP problem. As aforementioned in Section III-C4, (multiple) knapsack problems can be efficiently solved in quasi-polynomial (pseudo-polynomial) time [35], but we have solved optimization problems with integer variables with MOSEK solver which uses relaxation, branch-and-bound, and cutting plane approaches. As a result, the complexity of each problem instance is reduced to that of interior method for linear programs, and the search space for LR(d) is significantly diminished for clustering.



(a) LR(c) (b) LR(d)

Fig. 12: Convergence of the primal objective values for the large-scale network scenarios.

Besides, it is important for iterative LR-based approaches to assure the convergence. The Fig. 12 shows the convergence behaviour of LR(c,d) for the large-scale network introduced in Section IV-D. As it can be seen in Fig. 12a, LR(c) converges much faster than its distributed counterpart for having the global knowledge. On average, the convergence was reached within the first five runs of iteration. On the other hand, LR(d) took longer to converge as shown in Fig. 12b for lacking information and knowledge disparity, and this is clearly a disadvantage of the distributed approach compared to its centralized counterpart. However, the convergence behaviour of LR(d) can be further enhanced by different step-size update rules, different initialization policies for λ and x_{ik}.

V. Conclusion

In this paper, we have studied the MEC service missing problems which arise when an ES cannot provide service to the received request from users. To effectively forward such requests to the ESs that are installed with the corresponding service images, we have formulated the request forwarding problem by using the variant multiple-knapsack framework. The resulting integer linear problem can yield an optimal request forwarding schedules, but it may suffer from a high computation complexity as the problem size increases. By the evaluations on different network sizes and with different request arrival scenarios, we have shown that the proposed centralized and distributed Lagrange dual solutions can result in high-rewarding solutions with small runtime. The centralized approach can yield near-optimal solution within only a few iterations, but it requires the global knowledge for the problem to be solved correctly. Considering the heavy message exchanges required to gather the network-wide information, the decomposed Lagrangian dual approach can be used at the expense of the total reward. Also, it can further reduce the computation time, making it more suitable for online processing. A study on the convergence of both centralized and distributed Lagrange dual approaches has been also carried out. The centralized approach has converged within a small number of iterations, while the distributed counterpart required more iterations to converge.

References

[1] W. Shi, J. Cao, Q. Zhang, Y. Li and L. Xu, "Edge computing: Vision and challenges," *IEEE Internet Things J.*, vol. 3, no. 5, pp. 637–646, Oct. 2016.

[2] N. Abbas, Y. Zhang, A. Taherkordi, and T. Skeie, "Mobile edge computing: a survey," *IEEE Internet Things J.*, vol. 5, no. 1, pp. 450–465, Feb. 2018.

[3] Y. C. Hu, M. P., D. Sabella, N. Sprecher, and V. Young, "Mobile edge computing-A key technology towards 5G," ETSI, Sophia Antipolis, France, White Paper, vol. 11, pp. 1-16, 2015.

[4] T.-D. Nguyen, E.-N. Huh, and M. Jo, "Decentralized and revised content-centric networking-based service deployment and discovery platform in mobile edge computing for IoT devices," *IEEE Internet Things J.*, vol. 6, no. 3, pp. 4162—4175, Jun. 2019.

[5] W.-C. Chang and P.-C. Wang, "Adaptive replication for mobile edge computing," *IEEE J. Sel. Areas Commun.*, vol. 36, no. 11, pp. 2422–2432, Nov. 2018.

[6] N. Kherraf, H. A. Alameddine, S. Sharafeddine, C. M. Assi and A. Ghrayeb, "Optimized provisioning of edge computing resources with heterogeneous workload in IoT networks," *IEEE Trans. Netw. Service Manage.*, vol. 16, no. 2, pp. 459–474, Jun. 2019.

[7] D. Zeng, L. Gu, S. Guo, Z. Cheng, and S. Yu. "Joint optimization of task scheduling and image placement in fog computing supported software-defined embedded system," *IEEE Trans. Comput.*, vol. 65, no. 12, pp.3702–3712, Feb. 2016.

[8] T. Kim, M. Al-Tarazi, J.-W. Lin, and W. Choi, "Optimal container migration for mobile edge computing: Algorithm, system design and implementation," *IEEE Access*, vol. 9, pp. 158074–158090, Nov. 2021.

[9] L. Ma, S. Yi, N. Carter and Q. Li, "Efficient live migration of edge services leveraging container layered storage," *IEEE Trans. Mobile Comput.*, vol. 18, no. 9, pp. 2020–2033, Sep. 2019.

[10] J. Zhang, X. Hu, Z. Ning, E. C.-H. Ngai, L. Zhou, J. Wei, J. Cheng, B. Hu, and V. C. M. Leung, "Joint resource allocation for latency-sensitive services over mobile edge computing networks with caching," *IEEE Internet Things J.*, vol. 6, no. 3, pp. 4283–4294, Jun. 2019.

[11] V. Farhadi, F. Mehmeti, T. He, T. F. L. Porta, H. Khamfroush, S. Wang, K. S. Chan, and K. Poularakis, "Service placement and request scheduling for data-intensive applications in edge clouds," *IEEE/ACM Trans. Netw.*, vol. 29, no. 2, pp. 779–792, Feb. 2021.

[12] S. Dasgupta, C. H. Papadimitriou, and U. V. Vazirani, Algorithms. New York, NY, USA: McGraw-Hill, 2008.

[13] V. V. Vazirani, Approximation Algorithms. Springer-Verlag, 2001.

[14] H. Kellerer, U. Pferschy, and D. Pisinger, Knapsack problems. Springer-Verlag Berlin Heidelberg, 2004.

[15] A. Schrijver, Theory of linear and integer programming. John Wiley & Sons, 1998.

[16] K. Mehlhorn and P. Sanders, Algorithms and data structures: The basic toolbox. Springer, 2007.

[17] M. L. Fisher, "The lagrangian relaxation method for solving integer programming problems," *Manage. Sci.*, vol. 50, no. 12, pp. 1861–1871, Dec. 2004.

[18] D. P. Palomar and M. Chiang, "A tutorial on decomposition methods for network utility maximization," *IEEE J. Sel. Areas Commun.*, vol. 24, no. 8, pp. 1439–1451, Aug. 2006.

[19] Y. Sun, T. Wei, H. Li, Y. Zhang, and W. Wu, "Energy-efficient multimedia task assignment and computing offloading for mobile edge computing networks," *IEEE Access*, vol. 8, pp. 36702–36713, 2020.

[20] L. Liu, S. Chan, G. Han, M. Guizani, and M. Bandai, "Performance modelling of representative load sharing schemes for clustered servers in multi-access edge computing," *IEEE Internet Things J.*, vol. 6, no. 3, pp. 4880–4888, Nov. 2018.

[21] N. Kherraf, H. A. Alameddine, S. Sharafeddine, C. M. Assi, and A. Ghrayeb, "Optimized provisioning of edge computing resources with heterogeneous workload in IoT networks," *IEEE Trans. Netw. Service Manage.*, vol. 16, no. 2, pp. 459-–474, Jun. 2019.

[22] R. Beraldi, A. Mtibaa, and H. Alnuweiri, "Cooperative load balancing scheme for edge computing resources," in Proc. 2nd Int. Conf. Fog Mobile Edge Comput. (FMEC), Valencia, Spain, May 8-11, 2017.

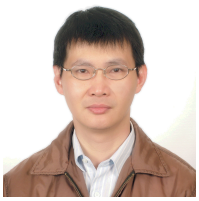
[23] K. Poularakis, J. Llorca, A. M. Tulino, I. Taylor, and L. Tassiulas, "Joint service placement and request routing in multi-cell mobile edge computing networks." in Proc. IEEE Conference on Computer Communications (INFOCOM), Paris, France, Apr. 29-May 02, 2019.

[24] K. Poularakis, J. Llorca, A. M. Tulino, I. Taylor, and L. Tassiulas, "Service placement and request routing in MEC networks with storage, computation, and communication constraints," *IEEE/ACM Trans. Netw.*, vol. 28, no. 3, pp. 1047–1060, Jun. 2020.

[25] B. Yuan, S. Guo, and Q. Wang, "Joint service placement and request routing in mobile edge computing," *Ad Hoc Netw.*, vol. 120, no. 1, pp. 1–1 Sept. 2021.

[26] X. Chen et al., "Dynamic service migration and request routing for microservice in multi-cell mobile edge computing," *IEEE Internet Things J.*, early access, Jan. 2022.
[27] M. L. Fisher, "An applications oriented guide to Lagrangian relaxation," *Interfaces*, vol. 15, no. 2, pp. 10–21, Mar.-Apr. 1985.
[28] M. Held, P. Wolfe, and H. P. Crowder, "Validation of subgradient optimization," *Mathematical Programming*, vol. 6, no. 1, pp. 62–88, Dec. 1974.
[29] MathWorks. Inc., Matlab, [Online]. Available: www.mathworks.com
[30] Michael Grant and Stephen Boyd, "CVX: Matlab software for disciplined convex programming, Version 2.1," [Online]. Available: cvxr.com/cvx
[31] MOSEK. (v8.0), Mosek. [Online]. Available: www.mosek.com/
[32] European Telecommunications Standards Institute (ETSI), "Multi-Access Edge Computing (MEC); Traffic Management APIs," ETSI GS MEC 015 V2.1.1, Jun. 2020-06.
[33] N. Karmarkar, "A new polynomial-time algorithm for linear programming." in Proc. ACM symposium on Theory of computing, 1984.
[34] F. A. Potra and S. J. Wright, "Interior-point methods," *J. Comput. Appl. Math.*, vol. 124, no. 1-2, pp. 281–302, 2000.
[35] C. Chekuri and S. Khanna, "A polynomial time approximation scheme for the multiple knapsack problem," *SIAM J. Comput.*, vol. 35, no. 3, pp. 713–728, Feb. 2006.

Taewoon Kim received the B.S. degree in Computer Science and Engineering from Pusan National University, Republic of Korea, in 2008, the M.S. degree in Information and Mechatronics from Gwangju Institute of Science and Technology, Republic of Korea, in 2010, and the Ph.D. degree in Computer Engineering from Iowa State University, Ames, IA, in 2018. He is currently an assistant professor in the School of Computer Science and Engineering, Pusan National University, Republic of Korea. From 2018 to 2022, he was an assistant professor with the Division of Software, College of Information Science, Hallym University, Republic of Korea, and from 2010 to 2013, he was a research engineer with the Telecommunications Technology Association, Republic of Korea. His research interest includes modeling, optimization and protocol design for wireless networking systems, such as WLAN, IoT/sensor networks, HetNets and C-RANs.



Jenn-Wei Lin is a full professor in the Department of Computer Science and Information Engineering, Fu Jen Catholic University, Taiwan. He received the Ph.D. degree in electrical engineering from National Taiwan University, Taiwan, in 1999. His research interests are edge computing, cloud computing, mobile computing and networks, distributed systems, and fault-tolerant computing.

Chi-Ting Hsieh is a research assistant in the Department of Computer Science and Information Engineering, Fu Jen Catholic University, Taiwan. She received the bachelor's degree in the Department of Computer Science and Information Engineering from Fu Jen Catholic University, Taiwan, in 2021. Her research interests are cloud computing and mobile computing and networks.