Stable Service Caching in MECs of Hierarchical Service Markets With Uncertain Request Rates

Zichuan Xu, *Member, IEEE*, Qiufen Xia, *Member, IEEE*, Lin Wang, *Student Member, IEEE*, Pan Zhou, *Member, IEEE*, John C. S. Lui, *Fellow, IEEE*, Weifa Liang, *Senior Member, IEEE*, Wenzheng Xu, *Member, IEEE*, and Guowei Wu

Abstract—Multi-access edge computing (MEC) enables extreme low-latency AI services, such as Augmented Reality (AR) and Virtual Reality (VR), by deploying cloudlets in locations close to users. Meanwhile, a 5G hierarchical service market is emerging with both large-scale and small-scale network service providers competing for both computing and network bandwidth resources of an infrastructure provider. In this paper, we investigate the problem of caching services originally deployed in remote clouds to cloudlets in an MEC network in a hierarchical service market. For the service caching problem, we first propose a novel approximation-restricted framework that guarantees the stability of the 5G service market. Under the proposed framework, we first propose an approximation algorithm with a provable approximation ratio for the problem with non-selfish network service providers. We then design an efficient Stackelberg congestion game with selfish network service providers, and analyze the Price of Anarchy (PoA) of the proposed Stackelberg congestion game to measure the efficiency loss of the game due to selfishness of network service providers. Considering that the request rate of each service may not be given in advance, we study the service caching problem with the uncertainlity of request rates, and propose an approximation algorithm and a Stackelberg game via leveraging the randomized rounding technique. We finally evaluate the performance of the proposed algorithms and mechanisms by both simulations and implementations in a real test-bed. Results show that the performance of our proposed mechanisms achieve around 9.2% less cost than those of existing approaches.

Index Terms—Mobile edge computing, service caching, stackelberg congestion game, approximation algorithms, price of anarchy

1 Introduction

5G services are on rise due to the fast deployment of 5G infrastructures. According to a recent report on COVID-19 implications [1], the 5G service market is expected to grow from $49.7 billion in 2019 to about $68.6 billion in 2020, as the demand for high-speed and better quality services is increasing because people are spending more time indoors for both work and leisures. Since most of such services are originally deployed in remote clouds, placing the services into a multi-access edge computing (MEC) network has been envisioned as a key approach to enable a stable 5G service market with high-quality network services. For example, Moving AR/VR services from data centers to cloudlets at an edge location (such as museums and sport stadium) or 5G base-stations within the proximity of users can significantly reduce the motion-to-photon latency experienced by VR users [17], [47], [57].

In this paper, we consider the scenario that network service providers hope to strategically and progressively move services that are originally deployed in remote clouds to cloudlets in a two-tiered MEC network [34], [39], [44], as shown in Fig. 1. Unlike the traditional service placement in MEC networks, we focus on placing existing services in remote clouds to cloudlets, and we refer it as the *service caching*. Due to the resource capacity constraints on cloudlets, services are only temporarily cached and their original services are still kept in remote clouds for later use when the cached service is destroyed from cloudlets.

The MEC network is operated by an infrastructure provider through leasing its resources to various network service providers. We consider a *hierarchical service market* with its first layer consisting of large-scale network service providers and its second layer having small-scale network service providers. Both large- and small-scale network service providers compete for both computing and bandwidth

- *Zichuan Xu, Lin Wang, and Guowei Wu are with the Key Laboratory for Ubiquitous Network and Service Software of Liaoning Province, School of Software, Dalian University of Technology, Dalian 116621, China. E-mail: {z.xu, wgwdut}@dlut.edu.cn, l.wang@mail.dlut.edu.cn.*
- *Qiufen Xia is with the Key Laboratory for Ubiquitous Network and Service Software of Liaoning Province, the International School of Information Science & Engineering, Dalian University of Technology, Dalian 116621, China. E-mail: qiufenxia@dlut.edu.cn.*
- *Pan Zhou is with the School of Cyber Science and Engineering, Huazhong University of Science and Technology, Wuhan, Hubei 430074, China. E-mail: panzhou@hust.edu.cn.*
- *John C. S. Lui is with the Department of Computer Science & Engineering, The Chinese University of Hong Kong, Shatin, N.T, Hong Kong. E-mail: cslui@cse.cuhk.edu.hk.*
- *Weifa Liang is with the Department of Computer Science, City University of Hong Kong, Kowloon, Hong Kong. E-mail: weifa.liang@cityu.edu.hk.*
- *Wenzheng Xu is with the School of Computer Science, Sichuan University, Chengdu, Sichuan 610017, China. E-mail: wenzheng.xu@scu.edu.cn.*

Manuscript received 31 Mar. 2021; revised 16 Jan. 2022; accepted 28 Jan. 2022. Date of publication 10 Feb. 2022; date of current version 5 June 2023.
The work of Zichuan Xu and Qiufen Xia was supported in part by the National Natural Science Foundation of China (NSFC) under Grants 62172068, 62172071, 61802048, and 61802047 and the "Xinghai scholar" program. The work of Weifa Liang was supported in part by the City University of Hong Kong under Grant 9380137/CS. The work of Wenzheng Xu was supported in part by NSFC under Grant 61602330. The work of Pan Zhou was supported in part by NSFC under Grant 61972448.
(Corresponding author: Qiufen Xia.)
Digital Object Identifier no. 10.1109/TMC.2022.3149870

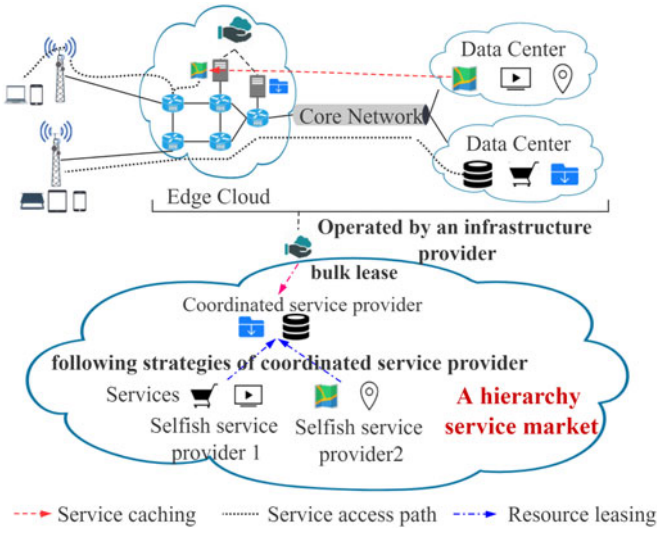


Fig. 1. An example of the two-tiered cloud network.

resources in the MEC network. Specifically, large-scale network service providers have bulk lease contracts with the infrastructure provider while small-scale network service providers lease on-demand resources to meet their instant demands when caching their services. An example of such hierarchical service market is the Alibaba edge node service (ENS) [2]. ENS enables large-, small- and medium-scale service providers to provide efficient and low-latency services to their users without the necessity of building their own infrastructures.

Significant challenges exist in the service caching in a hierarchical service market of an MEC network: (1) the cached instance of a service in the MEC and its original instance in the cloud have to be synchronized to ensure the correct operation of future cached service instances. As such, the states of the cached and original instances need to be updated quite often. How to jointly cache services, assign requests to the cached instances, and update the states of the cached instances to be consistent with their original instances are fundamentally challenging; (2) If the network service providers in a service market are not well coordinated, they may cache their services to the cloudlets with lowest costs and delays. This leads to significant congestion and performance degradation on some cloudlets in the MEC. Also, selfish network service providers may jeopardize the social benefit of all players [27], leading to an unstable market in which no network service provider wants to participate. Any mechanisms for the market have to be stable and close to the social optimum; and (3) a network service provider may not know the future request rates of its services. How to design a near-optimal adaptive mechanism by incorporating such request rate uncertainty is another challenge.

While studies on computing offloading and service placement [9], [12], [13], [16], [19], [20], [26], [28], [32], [37], [41], [42], [43], [44], [45], [46], [49], [51], [56] have been conducted, service caching from remote clouds to cloudlets with states updating in an MEC network has been hitherto overlooked. Also, due to limited computing resource capacities on cloudlets, the MEC network may experience congestions when too many services are cached in it, thereby reducing the benefits it brought to. Such congestion-aware service caching is largely ignored by existing studies. To the best of our knowledge, we are the first to incorporate congestion levels of cloudlets, request rate uncertainty for the service caching problem in a two-tiered MEC network, by proposing near-optimal and stable mechanisms with guaranteed gaps against their social optimum.

The main contributions of this paper are summarized as follows.

- We propose a novel *approximation-restricted* strategy with theoretical performance guarantees for the service caching problem.
- Under the approximation-restricted strategy, we devise an approximation algorithm with an approximation ratio for the service caching problem without selfish network service providers. We design a novel Stackelberg congestion game for the problem in an MEC network with resource capacities, and analyze the Price of Anarchy (PoA) of the game.
- For the service caching problem with uncertain request rates, we devise an approximation algorithm and a Stackelberg strategy for the service caching problem, based on the randomized rounding technique.
- We evaluate the performance of the proposed mechanisms in both simulation environments and a real test-bed. Results show that the proposed algorithms outperform existing studies by around 9.2% in terms of the service caching cost delivered by the algorithms.

The remainder of the paper is arranged as follows. Section 2 summarizes state of the arts on service caching in MEC networks. Section 3 introduces the system model, notations and problem formulation. The proposed Stackelberg congestion game for the service caching problem is described in Section 4. The approximation algorithm and Stackelberg strategy for the service caching problem with request rate uncertainty are designed in Section 5. Section 6 provides experimental results on the performance of the proposed algorithms in both the simulation environment and a real test-bed. The paper is concluded in Section 7.

2 Related Work

The studies on service caching can be related to either computation offloading, service provisioning, or edge content caching in MEC networks.

For studies on computation offloading, most existing studies either did not consider service placement/caching or ignored the impact of congestion of cloudlets during task offloading [14], [28], [51], [52], [56]. For example, Yu *et al.* [51] considered an application provisioning and data routing problem, with bandwidth and delay guarantees for data sources. Misra *et al.* [28] recently studied the task offloading problem in a software-defined network, where Internet-of-Things (IoT) devices are connected to fog computing nodes by multi-hop IoT access-points (APs). Zhou *et al.* [56] studied the joint task offloading and scheduling problem, by considering wireless network connections and mobile device mobility. Yang *et al.* [50] proposed a multi-task learning model to obtain an optimal computation offloading strategy for an MEC network.

For investigations on service provisioning, most of them did not consider a hierarchical service market with an infrastructure provider and various network service providers. In addition, they did not jointly consider the caching of services from the data centers to cloudlets in the MEC network and the state updating between cached and original services [6], [8], [13], [16], [19], [20], [26], [32], [41], [45], [51], [55]. For example, Farris *et al.* [13] devised methods for service replication and migration for mobile users with an objective to minimize the degradation of the quality of experience (QoE) and the service replication cost. Xu *et al.* [45] investigated a problem of service placement in MEC-enabled cellular networks, and proposed algorithms based on Lyapunov optimization and Gibbs sampling, to reduce the computation latency experienced by users. Wang *et al.* [41] presented a problem of provisioning a social VR application in MEC networks to serve a number of users to minimize the social cost for placing services in cloudlets. Ndikumana *et al.* [29] proposed a joint communication, computation, caching, and control framework for big data processing in MEC networks, to minimize a weighted sum of bandwidth consumption and latency. Zhu *et al.* [58] aimed to maximize the computation completion ratio for wireless MEC networks, by proposing an algorithm to jointly optimize time allocation and computation scheduling for mobile devices. Deng *et al.* [11] proposed an approach to find optimal deployment of micro-services in MEC networks, with the aim to minimize the deployment cost subject to response time requirements of requests. Jin *et al.* [21] investigated the service chaining problem in MEC networks, to simultaneously reuse computing and bandwidth resources while meeting latency requirements. Poularakis *et al.* [31] studied the problem of service placement and request routing in MEC-enabled multi-cell networks with multiple types of resources.

There are also several related studies on the content placement or caching in MEC networks [16], [20], [26], [44]. However, content caching is fundamentally different from service caching. Specifically, service caching needs caching services that consume computing resources, while content caching requires sufficient storage to store contents. Also, these studies neither consider the cache cost [16], [20] nor consider updating activities between local edge servers and remote clouds [26]. For instance, Hou *et al.* [16] investigated the problem of content caching in mobile networks and devised efficient algorithms to predict content popularity. Similarly, Jiang *et al.* [20] considered a content caching and delivery problem by placing popular contents in base stations and user equipments, such that the access latency of users is minimized, subject to the capacity constraints of base stations. Li *et al.* [26] considered a caching scheme for mobile networks, where each base station can cache popular contents to minimize delay experienced by users. Xiong *et al.* [44] investigated the interactions among content service providers under a novel 'sponsored content' scheme, by proposing a Stackelberg game.

In contrast to the mentioned studies on computation offloading, service placement, and content caching, we consider service caching in a two-tiered MEC network of a service market with both an infrastructure provider and multiple network service providers. In addition, the communications between origin services at remote clouds to the cached service at local edge servers are considered too. It must be mentioned that this paper is an extended version of its conference version in [48]. We have significantly extended the conference version by considering another closely related optimization problem - the service caching problem with request rate uncertainty, and for which we propose a new approximation algorithm with request rate uncertainty.

3 Preliminary

In this section, we introduce the system model and notations, and we also define the optimization problems of service caching precisely.

3.1 System Model

We consider a two-tiered MEC network $G = (\mathcal{CL} \cup \mathcal{DC}, E)$ that consists of a set $\mathcal{CL}$ of cloudlets deployed in locations within the proximity of users, and a set $\mathcal{DC}$ of remote clouds located in the core network, as shown in Fig. 1. Let CL_i be a cloudlet in $\mathcal{CL}$. The computing resource in each cloudlet is virtualized as Virtual Machines (VMs) or containers. Mobile services and their related databases/libraries can be implemented in the VMs of cloudlets in G. Each cloudlet has a limited capacity of bandwidth resource to transfer data traffic from/to itself. Let $C(CL_i)$ and $B(CL_i)$ be the computing and bandwidth resource capacities of each cloudlet CL_i, respectively. We do not consider the resource capacity constraint on each data center in $\mathcal{DC}$ since they have abundant resources. E is a set of links that interconnect the cloudlets and data centers in $\mathcal{CL} \cup \mathcal{DC}$.

3.2 Service Caching

Various large- and small-scale network service providers exist in the hierarchical service market of an MEC network. Denote by N a set of network service providers. Let sp_q be the qth network service provider in N, where $1 \leq q \leq |N|$. Each network service provider sp_q wants to cache a set of services to the MEC network. Let $\mathcal{S}_q$ be the set of services offered by sp_q, and $SV_{q,l}$ be a service in $\mathcal{S}_q$, where $1 \leq l \leq |\mathcal{S}_q|$. We refer to an implementation of service $SV_{q,l}$ in a VM of a remote cloud as its *original instance*. The requests of $SV_{q,l}$ will be directed to the *cached instance* of $SV_{q,l}$ for processing if an instance of $SV_{q,l}$ has been cached at a cloudlet in G already; otherwise, the *original instance* of $SV_{q,l}$ in the remote cloud will process its user requests. Denote by $r_{q,l}$ the request rate (i.e., the number of requests per time unit) that should be served by service $SV_{q,l}$. We consider a proportional resource consumption model. That is, the amount of resource demanded by each service $SV_{q,l}$ is $a_q \cdot r_{q,l}$, where a_q is the amounts of computing resource that network service provider assigns to each unit of request rate [33], [40], [54]. In addition, transmitting data from/to the cached instance of $SV_{q,l}$ consumes the bandwidth resource. Therefore, each network service provider allocates a certain amount of bandwidth resource to its user requests to guarantee service performance. Let b_q be the bandwidth resource assigned to a unit of request rate of each $SV_{q,l}$, the bandwidth consumption of $SV_{q,l}$ thus is $b_q \cdot r_{q,l}$.

The cached instances of each service $SV_{q,l} \in \mathcal{S}_q$ of network service provider sp_q may be de-allocated from the cloudlet due to the capacity constraint on the cloudlet. Therefore, the original service instance of $SV_{q,l}$ in its remote cloud will not be de-activated even if there are its cached instances in the MEC network.

3.3 Cost Model of Service Caching

The cost of caching a service in a cloudlet consists of a service instantiation cost due to the instantiation of a VM, the processing cost due to the processing of data traffic, and the update cost between the cached service and its original service in a remote cloud [3], [15]. These costs are categorized into the *service caching cost* and the *bandwidth consumption cost*, respectively.

Caching a service to the MEC network means instantiating a VM and running a service instance in the VM. This usually incurs a service instantiation cost due to software setup for the cached service instance. Let $c_{q,l}^{ins}$ be the cost of instantiating an instance of service $SV_{q,l}$ of network service provider sp_q. After service $SV_{q,l}$ being cached at a cloudlet, its user requests are forwarded to the cloudlet for processing. This incurs costs since data processing requires usage of computing resource.

The cached services in cloudlet CL_i are sharing the computing resource in CL_i. Although VMs and containers can provide resource isolation, inter-VM interferences still exist in cloud computing platforms including the state-of-the-art serverless platforms [24]. Such interference usually leads to performance degradation, thereby generating costs of violating service quality requirements of users. Intuitively, the inter-service cost becomes higher when there are a higher number of services cached in a cloudlet. We thus assume that the cost of caching a service in CL_i depends on the workload (i.e., the congestion or the number of services) of cloudlet CL_i. The cost due to congestions should be non-decreasing with the congestion levels. We adopt a proportional congestion model, following existing studies in [10], [18]. Let σ_i be the set of services with instances cached in cloudlet CL_i. The total service caching cost for the services in σ_i that are cached in CL_i can be calculated by

$$\alpha_i |\sigma_i| + \sum_{SV_{q,l} \in \sigma_i} c_{q,l}^{ins}, \tag{1}$$

where α_i is a given constant that captures the influence of congestion of a service on service caching costs. Note that the derivation technique in the later section does not rely on the assumption of the proportional cost model. Instead, it relies only on the non-decreasing cost with congestion levels. Therefore, the proportional congestion cost model can be easily extended to other more complicated non-decreasing cost models.

Recall that service $SV_{q,l}$ is temporarily cached at a cloudlet, and its original instance is maintained in the remote cloud. However, to ensure the seamless service transition between the cached and original service instances, the processed data of the cached instance needs to be updated and synchronized to its original instance in the remote cloud. The cost incurred by such updates is mainly due to the bandwidth resource consumption of cloudlet CL_i. Recall that each network service provider is assigned with an amount of $b_q \cdot r_{q,l}$ bandwidth resource. Hence, there is a fixed bandwidth consumption cost for each network service provider sp_q in cloudlet CL_i, referred to as $c_{q,i}^{bdw}$. In addition, the bandwidth consumption cost is affected by other network service providers using the bandwidth resource of CL_i, i.e., the congestion of cloudlet CL_i. The total update cost of all cached instances in σ_i is calculated as

$$\beta_i |\sigma_i| + |\sigma_i| \cdot c_{q,i}^{bdw}, \tag{2}$$

where β_i is a given constant that captures the impact of congestion of bandwidth resource of CL_i on the cost. Notice that the specific synchronization methods of such updating/synchronization depend on specific services. We thus consider that the design of synchronization methods is beyond the scope of this paper.

3.4 Hierarchical Service Markets

As shown in Fig. 1, the infrastructure provider of the hierarchical service market has full control of its resources of the MEC network. Its actions can have influence on the behaviors of other players in the service market. We thus can consider it as a *leader* in the market. The hierarchical service market usually has both large-scale and small-scale network service providers with diverse resource demands.

Large-scale network service providers have different network services serving a large number of users. Such network service providers usually have high demands from users. They thus prefer to have bulk-lease contracts with the infrastructure provider to lower their operational cost. The infrastructure provider thus can coordinate them as long as requirements in the bulk-lease contracts are met. Therefore, the network service providers with bulk-lease contracts are referred to as *coordinated network service providers*.

Small-scale start-ups offer services for a relative small group of users, and usually lease computing resources on-demand to process their requests instantly. Since small-scale network service providers just entered the market, they usually observe the behaviors of coordinated network service providers to make their own strategies. As such, their behaviors (by increasing or reducing and withholding of a specific type of services) are normally affected by the coordinated network service providers. We refer to them as *followers*. Since they are interested in their own revenues, we refer to such network service providers as *selfish network service providers*. Let S be the set of coordinated network service providers. $N \backslash S$ then denotes the set of selfish network service providers.

3.5 Stackelberg Congestion Game and its PoA

In the hierarchical service market of an MEC network, we consider a Stackelberg congestion game, where the leader is the infrastructure provider while the followers are the network service providers. In particular, the infrastructure provider leads and guides the behaviors of network service market by coordinating large-scale network service providers. As congestions of cloudlets play a vital role in the cost of service caching, this will indirectly lead the small-scale network service providers, even though they selfishly minimize their own costs. Without loss of generality, we

consider a symmetric Stackelberg where the strategy space of each network service provider is identical. The Stackelberg game is denoted by $\Gamma(N, \mathcal{CL}, (\sigma_l)_{sp_l \in N}, (c_i)_{CL_i \in \mathcal{CL}})$, where $\sigma_l \in 2^{|\mathcal{CL}|} \setminus \{\emptyset\}$ is the strategy space of each network service provider, and c_i is a non-negative and non-decreasing cost function associated with caching services in each cloudlet CL_i. Then, $c_i = \sum_{SV_{q,l} \in \sigma_i} c_{q,l,i}$. Denote by $c_q(\sigma_q)$ the cost of network service provider sp_q with strategy space σ_q, then

$$c_q(\sigma_q) = \sum_{CL_i \in \sigma_q} \sum_{SV_{q,l} \in \mathcal{S}_q} x_{q,l,i} \cdot c_{q,l,i}, \tag{3}$$

where $x_{q,l,i}$ is a binary indicator variable showing whether sp_q chooses CL_i to cache an instance of its service $SV_{q,l}$.

A *Stackelberg strategy* is an algorithm that chooses a subset of players and assigns them a prescribed strategy with the purpose of mitigating the detrimental effect of selfish behaviors of the remaining uncoordinated players.

To measure inefficiency caused by the selfishness of network service providers, we adopt a popular metric called the *Price of Anarchy* (PoA). It is defined as the proportion between the worst social utility from a *Nash equilibrium* (no players have incentives to deviate from their current strategies) and the optimal social utility in which players are not selfish, not necessarily from a Nash equilibrium.

3.6 Problem Definition

We consider an MEC network $G = (\mathcal{CL} \cup \mathcal{DC}, E)$ managed by an infrastructure provider. Its resources are shared by both large-scale (coordinated) and small-scale (selfish) network service providers. We consider that large-scale network service providers all prefer to have bulk contracts. Small-scale network service providers prefer on-demand leases. We further assume that such preferences does not change within a time horizon T. It must be mentioned that the network service providers' preference may change. For example, each coordinated network service provider will have a chance of deciding whether to continue their contracts when the contracts expire. On the other hand, a selfish network service provider may prefer a bulk-lease contract when its request rate increases dramatically. Once such preference changes, the strategy adjustment of the market will start in the beginning of a coming time horizon T. We define the following optimization problems in time horizon T.

Problem 1. Assuming that the request rate of each service $SV_{q,l}$ provided by network service provider $sp_q \in N$ is given, *the service caching problem* in an MEC network of a service market is to cache the services of the N network service providers, by selecting and coordinating a subset of network service providers from N, and allowing the rest of network service providers to perform selfishly, such that the social cost is minimized, subject to the computing and bandwidth resource capacities on each cloudlet of the MEC network, where the *social cost* is defined as the cost of all network service providers in N, i.e., $\sum_{sp_q \in N} c_q(\sigma_q)$.

Problem 2. In real life scenarios, a network service provider sp_q may not know the request rate of its service $SV_{q,l}$ requested by its user. Given a set of coordinated network service providers and a set of selfish network service provider, *the service caching problem with request rate uncertainty* thus is to adaptively cache the services of the N network service providers with uncertain information of request rate of each service, such that the social cost of the mobile service market is minimized while no players have incentives to deviate from their current strategies, subject to both computing and bandwidth resource capacities on each cloudlet.

All the symbols used in this paper are listed in Table 1.

4 An Efficient Mechanism for the Service Caching Problem

We now devise an efficient mechanism for the service caching problem, based on a novel Stackelberg game. We first propose a novel strategy called *approximation-restricted* in the mechanism design of a Stackelberg game, such that the performance of the proposed mechanism is not far from the social optimum when not all players are selfish. We then devise an approximation algorithm with an approximation ratio for the service caching problem with non-selfish players. Based on the approximation strategy and the approximation algorithm, we propose a Stackelberg strategy for the service caching problem with the consideration of both non-selfish and selfish players.

4.1 Approximation-Restricted Strategy

To avoid the performance degradation due to selfishness, a feasible method is to coordinate a subset of network service providers with near-optimal solutions, and then influence the small-scale network service providers (i.e., followers). We thus design a Stackelberg strategy with a guaranteed performance gap from the social optimum. The infrastructure provider of the MEC network then can find a subset of network service providers and coordinate them to avoid significant performance deviation caused by the rest of network service providers.

To guarantee the performance of the proposed mechanism, we focus on the *optimal-restricted* Stackelberg strategy. A Stackelberg strategy is optimal-restricted if the strategy assigned to the coordinated players coincides with the one they adopt in the social optimum solution OPT. However, the social optimum solution cannot be obtained in polynomial time due to the NP-Hardness of the problem. We thus find an approximate solution for the problem when all players are coordinated to approach the social optimum OPT. Then, we use the obtained approximate solution OPT' to guide the strategies of the coordinated players. We refer to this method as an *approximation-restricted* strategy.

Our idea is to design an approximation algorithm with an approximation ratio for the service caching problem in an MEC network with non-selfish network service providers. In the following, we reduce the problem to a generalized assignment problem (GAP) [36].

For the sake of clarity, we first describe the GAP problem [36]. Given n items and m knapsacks, each item itm_j can be assigned to a knapsack bin_i at a cost of c_{ij}. The weight of the item is w_{ij} if it is assigned to knapsack bin_i. The accumulative weight of all items assigned to bin_i cannot exceed its capacity CAP_i. The objective of the GAP problem is to assign as many items as possible to the m knapsacks such that the assignment cost is minimized.

TABLE 1
Symbols

Symbols	Meaning
$G = (\mathcal{CL} \cup \mathcal{DC}, E)$	an MEC network with a set $\mathcal{CL}$ of cloudlets, a set $\mathcal{DC}$ of remote clouds deployed in $\mathcal{CL}$ and a set of links E that interconnect the cloudlets and data centers
CL_i	a cloudlet in $\mathcal{CL}$
$C(CL_i)$ and $B(CL_i)$	the computing and bandwidth resource capacities of cloudlet CL_i
N and S	a set of network service providers N and a set of coordinated network service providers S
sp_q and S_q	the qth network service provider in N that offers a set of services S_q
$SV_{q,l}$ and $r_{q,l}$	service $SV_{q,l}$ and its request rate $r_{q,l}$
a_q and b_q	the amount of computing and bandwidth resources that are assigned to each unit of request rate $r_{q,l}$
$c_{q,l}^{ins}$	the cost of instantiating an instance of service $SV_{q,l}$ of network service provider sp_q
$c_{q,i}^{bdw}$	a fixed bandwidth consumption cost for each network service provider sp_q in the cloudlet CL_i
σ_i	a set of services that have instances cached in cloudlet CL_i
α_i and β_i	given constants that captures the influence of congestion on service caching and bandwidth costs of CL_i, respectively
$c_q(\sigma_q)$	the cost of network service provider sp_q with strategy space σ_q
$x_{q,l,i}$	a binary indicator variable showing whether sp_q chooses CL_i to cache an instance of its service $SV_{q,l}$
T	a finite time horizon
n, m, itm_j, bin_i, c_{ij}	a GAP problem that contain n items and m knapsacks, each item itm_j can be assigned to a knapsack bin_i at a cost of c_{ij}
w_{ij} and CAP_i	the weight of the item if it is assigned to knapsack bin_i with capacity CAP_i
a_{max} and a_{min}	the maximum and minimum demands of computing resources of a service $SV_{q,l}$
b_{max} and b_{min}	the maximum and minimum demands of bandwidth resources of a service $SV_{q,l}$
n_i and $CL'_{k,i}$	the number of virtual cloudlets in each cloudlet (CL_i) and $CL'_{k,i}$ is a virtual cloudlet of cloudlet CL_i
n'_{max}	the number of services that each virtual cloudlet maximally can cache
ξ	the percentage of the network service providers that are coordinated by the leader in the hierarchical service market
C, OPT and C'	the optimal cost due to algorithm `Appro`, the optimal cost and the social cost with non-selfish network service providers
ζ	the obtained approximation solution due to algorithm `Appro`
s and ϕ	any approximation-restricted Stackelberg caching of services and the worst pure NE induced by s
ω	the service caching that leads to the worst cost due to selfish behavior of the rest $(1-\xi)\|N\|$ network service providers
N_s	the set of coordinated network service providers determining the approximation-restricted Stackelberg strategy
$y_{q,l,i}$	an indicator variable that shows whether service $SV_{q,l}$ is cached in cloudlet CL_i
$E(c_{q,l,i})$ and $E(r_{q,l})$	the expected cost and request rates of each service $SV_{q,l}$
n	the fixed number of virtual cloudlets that each cloudlet is divided in algorithm `Appro_ADA`
z^*	a fractional solution z^* of the **LP**
$z_{q,l,i,k}$ and $z^*_{q,l,i,k}$	the real-valued variable and fractional solution showing the probability that service $SV_{q,l}$ is assigned to index k of virtual cloudlets
$\|\mathcal{S}_i\|$	the number of services that are randomly assigned to CL_i
z' and $z'_{q,l,i,k}$	the fractional solution obtained by solving **LP** and the probability of assigning service $SV_{q,l}$
y^* and z^*	the optimal solution to the **ILP** and the optimal solution to integral version of **LP**
$y^*_{q,l,i,k}$ and $s^*_{q,l,r}$	the indicator variables that show whether service $SV_{q,l}$ is admitted by virtual cloudlet $CL'_{i,k}$ of bs_i in solution y^* and $SV_{q,l}$ has request rate of r
$X_{q,l}$	the random variable indicating the most recent service
C_o	the amount of computing resource occupied by the services that are admitted in the previous rounds with index smaller than k

4.2 An Approximation Algorithm for the Problem With Non-Selfish Players

We reduce the service caching problem in an MEC network with non-selfish players to the GAP problem. The main difference between them lies in the cost model. The cost of caching an instance of service in a cloudlet is related to the 'congestion' of the cloudlet, i.e., the number of cached service instances. However, in the GAP problem the cost of assigning an item to a knapsack only depends on the item itself. We thus need to map the congestion-aware cost model to the flat cost model in the GAP problem.

Our basic strategy is to split each cloudlet into a set of *virtual cloudlets*, with each virtual cloudlet being restricted to cache a single service instance only. The rationale behind is to ignore the 'congestion' in the cost model first, and consider it later. Specifically, let a_{max} and b_{max} be the maximum demands of computing and bandwidth resources of a service $SV_{q,l}$, i.e., $a_{max} = \arg\max_{q,l}(a_q \cdot r_{q,l})$ and $b_{max} = \arg\max_{q,l}(b_q \cdot r_{q,l})$. Similarly, $a_{min} = \arg\min_{q,l}(a_q \cdot r_{q,l})$ and $b_{min} = \arg\max_{q,l}(b_q \cdot r_{q,l})$ be the minimum demands of computing and bandwidth resources of a service. We further assume that $\frac{a_{max}}{a_{min}}$ and $\frac{a_{max}}{a_{min}}$ usually are fixed constants. Note that such ratios depend on the number of service types, which can be obtained from historical information of services. We then split each cloudlet CL_i into $n_i = \min\{\lfloor C(CL_i)/a_{max} \rfloor, \lfloor B(CL_i)/b_{max} \rfloor\}$ virtual cloudlets with each virtual cloudlet being able to cache a single service SV_l. Let $\{CL'_{1,i}, \ldots, CL'_{k,i}, \ldots, CL'_{n_i,i}\}$ be the set of virtual

cloudlets for cloudlet CL_i. Each virtual cloudlet CL'_k has a capacity $\max\{a_{max}, b_{max}\}$ such that any service in N can be cached in it.

We now reduce the problem into the GAP problem, by considering each virtual cloudlet as a knapsack, with its capacity being set to $\max\{a_{max}, b_{max}\}$. Clearly, each virtual cloudlet maximally can cache

$$n'_{max} = \max\Big\{\frac{\max\{a_{max}, b_{max}\}}{a_{min}}, \frac{\max\{a_{max}, b_{max}\}}{b_{min}}\Big\}. \quad (4)$$

services in N. The cost of caching a service $SV_{q,l}$ in virtual cloudlet $CL'_{k,i}$ is set to $\alpha_i + \beta_i + c^{ins}_{q,l} + c^{bdw}_{q,i}$, which means that the contribution of other services in $CL'_{k,i}$ is not considered. That is, the cost of using resource by $SV_{q,l}$ in virtual cloudlet $CL'_{k,i}$ solely depends on the service itself and the location of CL_i.

We then solve the GAP problem by adopting the approximation algorithm in [36]. The obtained solution assigns each service to a virtual cloudlet. However, this is not a feasible solution to the original service caching problem. To obtain a feasible solution, we assign all the services that are assigned in the virtual cloudlets in $\{CL'_{1,i}, \ldots, CL'_{k,i}, \ldots, CL'_{n_i,i}\}$ to cloudlet CL_i. The detailed steps of the proposed algorithm are described in algorithm 1, which is referred to as algorithm `Appro`.

Algorithm 1. `Appro`

Input: An MEC network G and a number of network service providers wishing to cache their services in G.

Output: A cloudlet to cache each service $SV_{q,l}$ of each network service provider sp_q.

1: Split each cloudlet into a set of n_i virtual cloudlets, with each virtual cloudlet having the ability of caching only a limited number of services;
2: Consider each virtual cloudlet as a knapsack in the GAP problem [36], and use the cost function that does not incorporate the 'congestion', i.e., $\alpha_i + \beta_i + c^{ins}_{q,l} + c^{bdw}_{q,i}$;
3: Invoke the approximation algorithm for the GAP problem in [36];
4: Move all the network service providers that are assigned to virtual cloudlets in $\{CL'_{1,i}, \ldots, CL'_{k,i}, \ldots, CL'_{n_i,i}\}$ to cloudlet CL_i;

4.3 The Stackelberg Strategy of the Proposed Mechanism

We describe the Stackelberg strategy to deliver an approximate solution for the service caching problem with non-selfish network service providers. Specifically, we coordinate a subset of network service providers in N by assigning strategies in the approximate solution to them. We allow the uncoordinated network service providers to selfishly select cloudlets that would minimize their costs.

Let ξ be the percentage of the network service providers that are coordinated by the leader in the mobile service market (i.e., the infrastructure provider), where the $0 < \xi < 1$. The proposed Stackelberg strategy has two steps. In the first step, a number of $\lfloor \xi|N| \rfloor$ network service providers are selected to be coordinated by the infrastructure provider. Since the services of network service providers have different resource demands, they have different impacts on the social cost. To enlarge the influence of coordinated network service providers, we select $\lfloor \xi|N| \rfloor$ network service providers that incur the highest costs of caching their services. We refer to this strategy as Largest Cost First (LCF). The second step of the Stackelberg game is to allow the rest network service providers to selfishly select cloudlets that incur the lowest costs for them. The proposed Stackelberg strategy is shown in algorithm 2, which is called `LCF`.

Algorithm 2. `LCF`

Input: An MEC network G and a number of network service providers wishing to cache their services in the cloudlets of the cloud.

Output: A cloudlet to cache each service SV_l of each network service provider.

1: Find an approximate solution of the service caching problem in an MEC network with non-selfish players according to algorithm `Appro`;
2: Find a number $\xi|N|$ of network service providers with the maximum cost of caching their services in a cloudlet of the MEC network;
3: Let N_s be the set of such coordinated network service providers;
4: **for** each network service provider $sp_q \in N_s$ **do**
5: Use the location in the approximate solution to cache its services in $\mathcal{S}_q$;
6: **for** each network service provider $sp_q \in N \setminus N_s$ **do**
7: Use the location that could incur a minimum cost for each of its service $SV_{q,l} \in \mathcal{S}_q$;

4.4 Analysis

The rest is to analyze the performance of the proposed algorithms `Appro` and `LCF` in the following.

Lemma 1. *The solution obtained by algorithm* `Appro` *is feasible for the service caching problem in an MEC network with non-selfish network service providers, assuming that the resource capacities are far greater than the maximum resource demands of any service of service providers in N.*

Proof. The solution feasibility of algorithm `Appro` is to show that each service SV_l of network service provider is cached into a cloudlet while the computing and bandwidth capacities of each cloudlet $C(CL_i)$ and $B(CL_i)$ can be met.

In algorithm `Appro`, each cloudlet CL_i is divided into a set of virtual cloudlets, i.e., $\{CL'_{1,i}, \ldots, CL'_{k,i}, \ldots, CL'_{n_i,i}\}$. The services of network service providers in N then are assigned to those virtual cloudlets, using the approximate solution to the GAP problem. A feasible solution then is obtained, by assigning all the services that are assigned to virtual cloudlets in $\{CL'_{1,i}, \ldots, CL'_{k,i}, \ldots, CL'_{n_i,i}\}$ to cloudlet CL_i. Therefore, each service is cached to a single cloudlet only.

We now show that both computing and bandwidth capacities of cloudlets are met. For the computing capacity constraint $C(CL_i)$ of cloudlet CL_i, each cloudlet is divided into a number of virtual cloudlets according to the maximum demand (either computing or bandwidth) of a service. This guarantees that each virtual cloudlet has enough resource to cache a service. Considering that

each knapsack in the GAP problem has its capacity is not violated by any feasible solution, the capacity of each virtual cloudlet is not violated too. Having obtaining the solution due to algorithm in [36], we move the services cached into virtual cloudlet $CL'_{k,i}$ to cloudlet CL_i. Since the computing capacity of $CL'_{k,i}$ is not violated, the computing capacity of each cloudlet CL_i is not violated. Similar derivation can be performed for the bandwidth capacity of each cloudlet CL_i. The solution thus is feasible. □

Lemma 2. *Given an MEC network G and a set of non-selfish network service providers, there is an approximation algorithm, i.e., `Appro`, for the service caching problem with non-selfish network service providers. Its approximation ratio is $2 \cdot \delta \cdot \kappa$, where $\delta = \frac{C(CL_i)}{a_{max}}$ and $\kappa = \frac{B(CL_i)}{b_{max}}$.*

Proof. We analyze the approximation ratio of the proposed algorithm `Appro` as follows.

Denote by C' the obtained social cost for the problem with non-selfish network service providers under cost function $\alpha_i + \beta_i + c_{q,l}^{ins} + c_{q,i}^{bdw}$ for caching service $SV_{q,l}$ in cloudlet CL_i. Let OPT' be the optimal cost. Similarly, let C be the social cost due to algorithm `Appro` and let OPT be the optimal cost for the service caching problem in a two-tiered MEC network. The approximation ratio of algorithm in [36] for the GAP problem is 2. This means

$$C'/OPT' = 2. \tag{5}$$

We then find the relationship between the social cost C' under the cost function $\alpha_i + \beta_i + c_{q,l}^{ins} + c_{q,i}^{bdw}$ and the social cost under the cost function in the original problem. It must be mentioned that the social cost C' does not include the cost due to 'congestion' of a cloudlet. After invoking the approximation algorithm due to [36], there are at most n'_{max} services that are assigned to each virtual cloudlet $CL'_{k,i}$, since the capacity of each virtual cloudlet is set to $max\{a_{max}, b_{max}\}$. Recall that there are at most n_i virtual cloudlets for each cloudlet CL_i. All the services assigned to these n_i virtual cloudlets will be moved to a single cloudlet CL_i. The cost due to such movement increases as it increases the congestion of cloudlet CL_i. There are at most $n_i \cdot n'_{max}$ services involving in the movement, that is

$$\begin{aligned} C &\leq \sum_{CL_i \in \mathcal{CL}} (n_i \cdot n'_{max} \cdot (\alpha_i + \beta_i) + c_{q,l}^{ins} + c_{q,i}^{bdw}) \\ &= n_i \cdot n'_{max} \cdot \sum_{CL_i \in \mathcal{CL}} \left(\alpha_i + \beta_i + \frac{c_{q,l}^{ins} + c_{q,i}^{bdw}}{n_i \cdot n'_{max}} \right) \end{aligned}$$

Clearly, it can be seen that $n_i > 1$ and $n'_{max} > 1$. We then have

$$C < n_i n'_{max} \sum_{CL_i \in \mathcal{CL}} (\alpha_i + \beta_i + c_{q,l}^{ins} + c_{q,i}^{bdw}) = n_i \cdot n'_{max} \cdot C'.$$

Assuming that $\frac{C(CL_i)}{a_{max}}$ and $\frac{B(CL_i)}{b_{max}}$ are small constants, let $\frac{C(CL_i)}{a_{max}} = \delta$ and $\frac{B(CL_i)}{b_{max}} = \kappa$. We have $C < \delta \cdot \kappa \cdot C' = 2 \cdot \delta \cdot \kappa \cdot OPT'$, due to Eq. (5). Since OPT considers the congestion of each cloudlet, we have $OPT' < OPT$, which means that $C < 2 \cdot \delta \cdot \kappa \cdot OPT$. The approximation ratio of the proposed algorithm `Appro` thus is $2 \cdot \delta \cdot \kappa$. □

We analyze the performance of the proposed Stackelberg strategy `LCF`, by stating the main results as follows.

Lemma 3. *Given an MEC network G and a set of network service providers, assuming that some network service providers can be coordinated by the infrastructure provider of G while some other network service providers behave selfishly, there exists a Stackelberg game, i.e., `LCF`, which can achieve at least one Nash equilibrium (NE) of the proposed game.*

Proof. The proposed Stackelberg game deals with a set of coordinated service providers and another set of selfish service providers. Recall that the coordinated service providers follow the decisions made by algorithm `Appro`, and their decisions will not be affected by the selfish network service providers. We only need to show that a NE could be achieved for the selfish network service providers. Recall that the cost of caching a service $SV_{q,l}$ in cloudlet CL_i is calculated by $\alpha_i + \beta_i + c_{q,l}^{ins} + c_{q,i}^{bdw}$, which is an affine function. Following existing results in [30], an affine congestion game admits at least one NE. This concludes the proof. □

We establish the PoA of the proposed Stackelberg game under the `LCF` strategy in the following theorem.

Theorem 1. *Given an MEC network G and a set of network service providers, assuming that some network service providers can be coordinated by the infrastructure provider of G while some other network service providers behave selfishly, we then have the PoA of the proposed strategy is $\frac{2\delta\kappa}{1-v}(\frac{1}{4v} + 1 - \xi)$ with $v \in (0,1)$, where $\delta = \frac{C(CL_i)}{a_{max}}$ and $\kappa = \frac{B(CL_i)}{b_{max}}$.*

Proof. Let ζ be the obtained approximation solution due to algorithm `Appro`, let s be any approximation-restricted Stackelberg caching of services of an MEC network, and let ω be the service caching that leads to the worst cost due to selfish behavior of the rest $(1-\xi)|N|$ network service providers. Denote by N_s the set of coordinated network service providers determining the approximation-restricted Stackelberg strategy. Let ϕ be the worst pure NE induced by s. We then know that $\omega = s + \phi$.

Recall that ϕ is a NE with respect to the cost function $\alpha_i|\sigma_i| + \beta_i|\sigma_i| + c_{q,l}^{ins} + c_{q,i}^{bdw}$ for each of the $(1-\xi)|N|$ network service providers. We have

$$\begin{aligned} &(\alpha_{\phi(q,l)} + \beta_{\omega(q,l)})|\sigma_{\omega(q,l)}| + c_{q,l}^{ins} + c_{q,\omega(q,l)}^{bdw} \\ &< \alpha_{\zeta(q,l)}|\sigma_{\zeta(q,l)} + 1| + \beta_{\zeta(q,l)}|\sigma_{\zeta(q,l)} + 1| + c_{q,l}^{ins} + c_{q,\zeta(q,l)}^{bdw}, \end{aligned} \tag{6}$$

where $\omega(q,l)$ and $\zeta(q,l)$ are the cloudlets that are used to cache service $SV_{q,l}$ of network service provider sp_q of the NE and the approximate solution due to algorithm `Appro`.

If we sum up the inequalities (6) for all uncoordinated network service providers in $N \setminus N_s$, we have

$$\sum_{sp_l \in N \setminus N_s} (\alpha_{\omega(q,l)} + \beta_{\omega(q,l)})|\sigma_{\omega(q,l)}| + c^{ins}_{q,l} + c^{bdw}_{q,\omega(q,l)} < \sum_{sp_l \in N \setminus N_s} (\alpha_{\zeta(q,l)} + \beta_{\zeta(q,l)})|\sigma_{\zeta(q,l)} + 1| + c^{ins}_{q,l} + c^{bdw}_{q,\zeta(q,l)}.$$

We use $\sigma_{i,\omega}$ to denote the number of services that are assigned to cloudlet CL_i of the service caching that leads to the worst cost due to selfish behavior of the rest $(1-\xi)|N|$ network service providers. We have

$$\sum_{CL_i \in \mathcal{CL}} \sigma_{i,\omega}\big((\alpha_i + \beta_i)|\sigma_{i,\omega}| + c^{ins}_{q,l} + c^{bdw}_{q,i}\big) < \sum_{CL_i \in \mathcal{CL}} (\sigma_{i,\zeta} - \sigma_{i,s})\big((\alpha_i + \beta_i)|\sigma_{i,\omega} + 1| + c^{ins}_{q,l} + c^{bdw}_{q,i}\big).$$

Adding the cost $\sigma_{i,s}\big((\alpha_i + \beta_i)\sigma_{i,\omega} + c^{ins}_{q,l} + c^{bdw}_{q,i}\big)$ of coordinated players to both sides of the above inequality, we obtain

$$C(\omega) = \sum_{CL_i \in \mathcal{CL}} (\sigma_{i,\phi} + \sigma_{i,s}) \cdot \big((\alpha_i + \beta_i)\sigma_{i,\omega} + c^{ins}_{q,l} + c^{bdw}_{q,i}\big) < \sum_{CL_i \in \mathcal{CL}} \big((\alpha_i + \beta_i)\sigma_{i,\omega}\sigma_{i,\zeta} + c^{ins}_{q,l}\sigma_{i,\zeta} + c^{bdw}_{q,i}\sigma_{i,\zeta} + (\sigma_{i,\zeta} - \sigma_{i,s})(\alpha_i + \beta_i)\big). \quad (7)$$

Given the inequality $xy \le vx^2 + \frac{1}{4v}y^2$ that is valid for all $x, y \in \mathtt{R}$ and $v \in (0,1)$, we then derive the following inequality

$$\begin{aligned}
&\sum_{CL_i \in \mathcal{CL}} \big((\alpha_i + \beta_i)\sigma_{i,\omega}\sigma_{i,\zeta} + c^{ins}_{q,l}\sigma_{i,\zeta} + c^{bdw}_{q,i}\sigma_{i,\zeta}\big)\\
&< v \sum_{CL_i \in \mathcal{CL}} (\alpha_i + \beta_i)(\sigma_{i,\omega})^2 + \frac{1}{4v} \sum_{CL_i \in \mathcal{CL}} (\alpha_i + \beta_i)(\sigma_{i,\zeta})^2 + \sum_{CL_i \in \mathcal{CL}} (c^{ins}_{q,l} + c^{bdw}_{q,i})\sigma_{i,\zeta}\\
&= vC(\omega) - v \sum_{CL_i \in \mathcal{CL}} (c^{ins}_{q,l} + c^{bdw}_{q,i})(\sigma_{i,\omega}) + 1/(4v) \sum_{CL_i \in \mathcal{CL}} (\alpha_i + \beta_i)(\sigma_{i,\zeta})^2 + \sum_{CL_i \in \mathcal{CL}} (c^{ins}_{q,l} + c^{bdw}_{q,i})\sigma_{i,\zeta}\\
&= vC(\omega) + 1/(4v) \cdot vC(\zeta) + \sum_{CL_i \in \mathcal{CL}} (c^{ins}_{q,l} + c^{bdw}_{q,i})\sigma_{i,\zeta} - \big(v + 1/(4v)\big) \sum_{CL_i \in \mathcal{CL}} (c^{ins}_{q,l} + c^{bdw}_{q,i})\sigma_{i,s} - 1/(4v) \sum_{CL_i \in \mathcal{CL}} (c^{ins}_{q,l} + c^{bdw}_{q,i})(\sigma_{i,\zeta} - \sigma_{i,s}), \text{ since } \sigma_{i,\omega} \ge \sigma_{i,s}\\
&\le vC(\omega) + 1/(4v)C(\zeta) + (1 - 1/(4v)) \sum_{CL_i \in \mathcal{CL}} (c^{ins}_{q,l} + c^{bdw}_{q,i})(\sigma_{i,\zeta} - \sigma_{i,s}), \quad \text{since v} + 1/(4\text{v}) \ge 1. \quad (8)
\end{aligned}$$

Combining inequalities (7) and (8), we have

$$\begin{aligned}
&(1-v)C(\omega)\\
&\le \frac{1}{4v}C(\zeta) + (1 - \frac{1}{4v}) \sum_{CL_i \in \mathcal{CL}} (c^{ins}_{q,l} + c^{bdw}_{q,i})(\sigma_{i,\zeta} - \sigma_{i,s}) + \sum_{CL_i \in \mathcal{CL}} (\sigma_{i,\zeta} - \sigma_{i,s})(\alpha_i + \beta_i).\\
&\le 1/(4v)C(\zeta) + \sum_{CL_i \in \mathcal{CL}} (c^{ins}_{q,l} + c^{bdw}_{q,i})(\sigma_{i,\zeta} - \sigma_{i,s}) + \sum_{CL_i \in \mathcal{CL}} (\alpha_i + \beta_i)(\sigma_{i,\zeta} - \sigma_{i,s})\\
&\le 1/(4v)C(\zeta) + \sum_{CL_i \in \mathcal{CL}} \big((\alpha_i + \beta_i)(\sigma_{i,\zeta} - \sigma_{i,s})^2 + (c^{ins}_{q,l} + c^{bdw}_{q,i})(\sigma_{i,\zeta} - \sigma_{i,s})\big), \quad (10)\\
&\text{since } (\sigma_{i,\zeta} - \sigma_{i,s}) \le (\sigma_{i,\zeta} - \sigma_{i,s})^2 \text{ for non-negative integers}\\
&\le 1/(4v)C(\zeta) + \sum_{CL_i \in \mathcal{CL}} \big((\alpha_i + \beta_i)\sigma_{i,\zeta} + (c^{ins}_{q,l} + c^{bdw}_{q,i})\big)\\
&(\sigma_{i,\zeta} - \sigma_{i,s}), \text{since } (\sigma_{i,\zeta} - \sigma_{i,s})^2 \le \sigma_{i,\zeta}(\sigma_{i,\zeta} - \sigma_{i,s})\\
&\le 1/(4v)C(\zeta) + (1-\xi)C(\zeta) = (1/(4v) + 1 - \xi)C(\zeta)\\
&\le 2\delta\kappa(1/(4v) + 1 - \xi)OPT, \text{due to Lemma 2 .}
\end{aligned}$$

The PoA of the proposed approximation-restricted Stackelberg strategy thus is $\frac{2\delta\kappa}{1-v}(\frac{1}{4v} + 1 - \xi)$. □

5 Algorithm for the Service Caching Problem With Request Rate Uncertainty

In this section, we propose algorithms for the service caching problem with request rate uncertainty in an MEC network.

5.1 Overview

To harness the performance degradation due to request rate uncertainty, we adaptively admit the services of all coordinated network service providers instead of admitting them at once. Specifically, although the request rates are uncertain, we still can obtain the historical information, such as the expected request rate of each type of services. By adopting the similar design rationale of algorithms `Appro` and `LCF`, we propose an approximation algorithm for the problem with non-selfish network service providers, and then devise another algorithm with both selfish and non-selfish network service providers. However, given that the request rates are uncertain, we adopt a randomized rounding method to obtain an approximate solution. Further, the uncertain request rates require fully adaptive algorithms to address the unexpected changes of the network. We then propose another adaptive algorithm to enable dynamic changes of request rates of different services, based on the proposed randomized rounding method.

5.2 Approximation Algorithm for the Problem With Request Rate Uncertainties and Non-Selfish Network Service Providers

We first formulate the service caching problem with request rate uncertainty into an Integer Linear Program (ILP). Recall that the problem is to minimize the expected cost of caching the services of all network service providers into the MEC network. Let $y_{q,l,i}$ be an indicator variable that shows whether service $SV_{q,l}$ is cached in cloudlet CL_i. Denote by $E(c_{q,l,i})$ the expected cost of assigning service $SV_{q,l}$ to

cloudlet CL_i, and let $E(r_{q,l})$ be the expected request rates of each service $SV_{q,l}$. The problem thus can be formulated by

$$\textbf{ILP:} \quad \min \sum_{q=1}^{N} \sum_{SV_{q,l} \in \mathcal{S}_q} y_{q,l,i} \cdot E(c_{q,l,i}), \tag{11}$$

subject to

$$\sum_{CL_i \in \mathcal{CL}} y_{q,l,i} \leq 1, \quad \forall \ \mathrm{SV}_{q,l} \tag{12}$$

$$\sum_{q=1}^{N} \sum_{l=1}^{\mathcal{S}_q} y_{q,l,i} \cdot a_q \cdot E(r_{q,l}) \cdot \leq C(CL_i), \forall\ CL_i \tag{13}$$

$$\sum_{q=1}^{N} \sum_{l=1}^{\mathcal{S}_q} y_{q,l,i} \cdot b_q \cdot E(r_{q,l}) \cdot \leq B(CL_i), \forall\ CL_i \tag{14}$$

$$y_{q,l,i} \in \{0, 1\}, \tag{15}$$

where Constraint (12) means that each service $SV_{q,l}$ can only be assigned to a single cloudlet in $\mathcal{CL}$. Constraints (13) and (14) say that the computing and bandwidth capacities of each cloudlet CL_i cannot be violated. Constraint (15) makes sure $y_{q,l,i}$ is a binary indicator variable.

To obtain an approximate solution to the service caching problem, we adopt a randomized rounding method. Specifically, we first propose a feasible relaxation to the **ILP**. We then obtain a fractional solution to the relaxed linear program, based on which we finally devise the approximation algorithm.

LP relaxation. To have a feasible relaxation to the **ILP**, we first divide each cloudlet into virtual cloudlets, with each virtual cloudlet representing a portion of the computing resource of the cloudlet. We then index the virtual cloudlets of each cloudlet, where two virtual cloudlets with the consecutive portions of computing resource of the cloudlet are assigned to consecutive indexes. Let n be the number of virtual cloudlets that each cloudlet is divided. Unlike algorithm `Appro`, we here assume that n is a given parameter. Each virtual cloudlet $CL'_{k,i}$ thus has an amount $\lfloor \frac{C(CL_i)}{n} \rfloor$ of computing resource, and an amount $\lfloor \frac{B(CL_i)}{n} \rfloor$ of bandwidth resource. Note that for each virtual cloudlet $CL'_{k,i}$, k naturally becomes its index.

We relax the **ILP**, denoted by **LP**. Specifically, let $z_{q,l,i,k}$ be a real-valued variable showing the probability that service $SV_{q,l}$ is assigned to index k of virtual cloudlets. The objective is then

$$\textbf{LP:} \ \min \sum_{q=1}^{N} \sum_{SV_{q,l} \in \mathcal{S}_q} z_{q,l,i,k} E(\alpha_i + \beta_i + c_{q,l}^{ins} + c_{q,i}^{bdw}), \tag{16}$$

subject to

$$\sum_{CL_i \in \mathcal{CL}} \sum_{k=1}^{n} z_{q,l,i,k} \leq 1, \quad \forall \ \mathrm{SV}_{q,l} \tag{17}$$

$$\sum_{q=1}^{N} \sum_{l=1}^{\mathcal{S}_q} \sum_{k'=1}^{k} z_{q,l,i,k'} \cdot a_q \cdot E\left(\min\left\{ r_{q,l}, \frac{C(CL_i)}{n \cdot a_q} \right\} \right) \leq 2 \cdot k \cdot \lfloor (C(CL_i))/n \rfloor, \text{ for each } \mathrm{CL}'_{i,k} \tag{18}$$

$$\sum_{q=1}^{N} \sum_{l=1}^{\mathcal{S}_q} \sum_{k'=1}^{k} z_{q,l,i,k'} \cdot b_q \cdot E\left(\min\left\{ r_{q,l}, \frac{B(CL_i)}{n \cdot b_q} \right\} \right) \leq 2 \cdot k \cdot \lfloor B(CL_i)/n \rfloor, \text{ for each } \mathrm{CL}'_{i,k} \tag{19}$$

$$0 \leq z_{q,l,i,k} \leq 1. \tag{20}$$

Randomized Rounding. Given the relaxation of **ILP**, we can obtain a fractional solution z^* of the **LP**. We treat each $z^*_{q,l,i,k}$ of the fractional solution as a probability of assigning service $SV_{q,l}$ to the kth virtual cloudlet of CL_i. Based on these probabilities, we then assign the services to virtual cloudlets randomly. There may be multiple services to be assigned to a single cloudlet. We adaptively admit the services that are randomly assigned to each index k of virtual cloudlets. Specifically, for kth virtual cloudlet of CL_i, there may be multiple services that are randomly assigned to it. These services may not fit into it due to its capacity constraints. The reason is that the request rate of each service is not known before its admission. However, after the admission of each service, it may reveal its demand information to the system. We thus adaptively admit the assigned services one-by-one in each cloudlet according to the revealed information of each service. That is, we admit the services that are assigned to each cloudlet CL_i in n iterations. Let $|\mathcal{S}_i|$ be the number of services that are randomly assigned to CL_i. For each iteration k (i.e., $1 \leq k \leq n$), we consider the kth service with the smallest expected request rates. The service is admitted to CL_i if and only if the servcies that are already admitted to CL_i occupy no more than $k \cdot \lfloor \frac{C(CL_i)}{n} \rfloor$ amount of computing resource and $k \cdot \lfloor \frac{B(CL_i)}{n} \rfloor$ amount of bandwidth resource. Detailed steps of the proposed approximation algorithm are shown in algorithm 3, which is referred to as algorithm `Appro_ADA`.

Algorithm 3. `Appro_ADA`

Input: An MEC network $G = (\mathcal{CL} \cup \mathcal{DC}, E)$ and a number of network service providers wishing to cache their services in the cloudlets of the cloud.

Output: A cloudlet to cache each service SV_l of each network service provider.

1: Obtain a fraction solution y by solving the **LP**;
2: Assign service $SV_{q,l}$ of network service provider sp_q randomly to the kth virtual cloudlet $CL'_{k,i}$ of cloudlet CL_i with probability $z_{q,l,i,k}$;
3: **for** $k \leftarrow 1, \ldots, n$ **do**
4: **for** each cloudlet $CL_i \in \mathcal{CL}$ **do**
5: Consider the service $SV_{q,l}$ with the kth smallest expected request rate $r_{q,l}$;
6: **if** the services admitted so far in CL_i occupy at most $k \cdot \lfloor \frac{C(CL_i)}{n} \rfloor - 2E(r_{q,l} \cdot a_q)$ amount of computing resource and $k \cdot \lfloor \frac{B(CL_i)}{n} \rfloor - 2E(r_{q,l} \cdot b_q)$ amount of bandwidth resource **then**
7: Admit $SV_{q,l}$ to cloudlet CL_i;
8: **else**
9: Re-assign service $SV_{q,l}$ of network service provider sp_q randomly to the k'th virtual cloudlet $CL'_{k',i'}$ of cloudlet $CL_{i'}$ with probability $z_{q,l,i',k'}$, where $k' \geq k$ and $i' \neq i$;

5.3 The Stackelberg Strategy for the Service Caching Problem With Request Rate Uncertainty

Given algorithm *Appro_ADA*, we now remove the assumption of non-selfish network service providers. It must be mentioned that the request rate of each service is not known in advance. However, request rate is usually a primary metric to determine the scale of network service providers. Some network service providers may prefer to behave selfishly when they have small request rates of their services. On the other hand, some large-scale network service providers can be coordinated. When the request rate of each service is not known, we may not known the percentage ξ of network service providers that can be coordinated. As such, we need to dynamically learn the percentage ξ. We thus propose a $\epsilon-$greedy adaptive Stackelberg strategy to adaptively learn a proper value of ξ, by dynamically deviating from its optimal decision in each learning step.

The proposed Stackelberg strategy consists of two steps. In the first step, we select $\lfloor \xi|N| \rfloor$ network service providers that incur the highest expected cost of caching their services. We refer this strategy as the Largest Expected Cost First (LECF). The second step of the Stackelberg game is to allow the rest network service providers to selfishly select cloudlets that incur the lowest cost for them. With probability ϵ, the algorithm randomly selects a network service provider. The proposed Stackelberg strategy is shown in algorithm 4, which is referred to as `LECF`.

Algorithm 4. `LECF`

Input: An MEC network G and a number of network service providers wishing to cache their services in the cloudlets of the MEC network.

Output: A cloudlet to cache each service SV_l of each network service provider.

1: Find an approximate solution of the service caching problem in an MEC network with non-selfish players according to algorithm `Appro_ADA`;
2: **for** $q \leftarrow 1 \cdots \xi|N|$ **do**
3: Either find a network service providers with the qth maximum expected cost of caching their services in a cloudlet of the MEC network with probability $1-\epsilon$, or select a network service provider randomly with probability ϵ;
4: Let N_s be the set of such coordinated network service providers;
5: **for** each network service provider $sp_q \in N_s$ **do**
6: Use the location in the approximate solution to cache its services in $\mathcal{S}_q$;
7: **for** each network service provider $sp_q \in N \setminus N_s$ **do**
8: Use the location that could incur a minimum expected cost for each of its service $SV_{q,l} \in \mathcal{S}_q$.

5.4 Algorithm Analysis

We show the feasibility and performance of the proposed algorithm `Appro_ADA`.

Lemma 4. *The solution obtained by algorithm `Appro_ADA` is feasible to the service caching problem with request rate uncertainty.*

Proof. The solution feasibility of the proposed approximation algorithm is to show that both computing and bandwidth resources of the MEC network G are met by the solution delivered by algorithm `Appro_ADA`.

Let z' be the fractional solution obtained by solving **LP**. Constraints (18) and (19) show that z' ensures that the demand of the admitted services by the kth virtual cloudlet is no greater than 2 times of the accumulative capacity of the previous k virtual cloudlets, i.e., $2 \cdot k \cdot \lfloor \frac{C(CL_i)}{n} \rfloor$ and $2 \cdot k \cdot \lfloor \frac{B(CL_i)}{n} \rfloor$. We know that the computing and bandwidth resources may be violated by the fraction solution. However, in algorithm `Appro_ADA`, we further consider fractional solution $z'_{q,l,i,k}$ as a probability of assigning service $SV_{q,l}$.

To ensure that both computing and bandwidth resources of each CL_i are met, we admit service $SV_{q,l}$ that is randomly assigned to CL_i as long as the services admitted so far occupy at most $k \cdot \lfloor \frac{C(CL_i)}{n} \rfloor$ computing resource of cloudlet CL_i. After the admission of service $SV_{q,l}$, the resource capacities however may be violated, because each service only reveals its real demand once admitted. This means service $SV_{q,l}$ is the only service whose demand causes the violation of the computing capacity of CL_i.

To show the probability of violating the computing capacity by the admission of $SV_{q,l}$, we use C_o to denote the amount of computing resource occupied by the services that are admitted in the previous rounds with index smaller than k. Showing the probability thus is to show $Pr(k\lfloor C(CL_i)/n \rfloor - C_o \leq r_{q,l} \cdot a_q)$, i.e.,

$$\begin{aligned} &Pr(k\lfloor C(CL_i)/n \rfloor - C_o \leq r_{q,l} \cdot a_q) \\ &\leq Pr[E(r_{q,l} \cdot a_q)/(k\lfloor C(CL_i)/n \rfloor - C_o)] \\ &\leq 1/2, \text{due to Constraint (18)}, \end{aligned} \tag{21}$$

following the Markov's inequality and the admission policy shown in Step (6). □

Theorem 2. *Given an MEC network G and a set of non-selfish network service providers, assuming that the network service providers can be coordinated by the infrastructure provider of G and the request rate $r_{q,l}$ of each service $SV_{q,l}$ of network service provider sp_q is not known in advance, there is an approximation algorithm, i.e., `Appro_ADA`, for the service caching problem with request rate uncertainty. The approximation ratio of the obtained solution by algorithm `Appro_ADA` is $2 \cdot \delta \cdot \kappa$, where $\delta = C(CL_i)/a_{max}$ and $\kappa = B(CL_i)/b_{max}$.*

Proof. We show the approximation ratio of the proposed algorithm `Appro_ADA` as follows. We denote by y^* the optimal solution to the **ILP**, i.e., the service caching problem with request rate uncertainty and non-selfish network service providers. Let z^* be the optimal solution to integral version of **LP**, that is when $z_{q,l,i,k} \in \{0,1\}$. We know that $z^* \leq 2 \cdot \delta \cdot \kappa \cdot y^*$, due to Lemma 2.

The rest is to show that **LP** is a feasible relaxation of **ILP**. To this end, we need to show that solution y^* meets the constraints of (17), (18), and (19). Clearly, y^* meets constraints (17) because $y_{q,l,i} = \sum_{k=1}^{n} z_{q,l,i,k}$. To show that y^* also meets constraint (18) and (19), we consider some virtual cloudlet k of cloudlet CL_i and a choice of

TABLE 2
Parameter Settings

parameter	value
datacenters	5
switch nodes	[50,100]
# of VMs	[15, 30]
bandwidth resource	[800, 1000] Mbps
costs of transmitting 1 GB data	[$0.05, $0.12]
costs of processing 1 GB data	[$0.15, $0.22]
request rate	[50,200]
traffic volume	[1000,2000]
α and β	[0, 1]

cloudlets in the optimal solution y^*. Recall that the request rate of each service $SV_{q,l}$ is not known in advance. The optimal solution y^* thus is a randomized policy. Consider a run of the optimal solution, we use $y^*_{q,l,i,k}$ and $s^*_{q,l,r}$ to denote the indicator variables that show whether service $SV_{q,l}$ is admitted by virtual cloudlet $CL'_{i,k}$ of bs_i in solution y^* and $SV_{q,l}$ has request rate of r, respectively. Let $X_{q,l}$ be the random variable indicating the most recent service that is assigned to the virtual cloudlets with index smaller than k.

Considering that each service $SV_{q,l}$ occupies an amount of computing resource of each cloudlet CL_i, service $X_{q,l}$ in y^* is the only service that may have its demanded computing resource exceeds that of virtual cloudlet $CL'_{i,k}$. This means that

$$\sum_{SV_{q,l}\neq X_{q,l}} \sum_{k'\leq k} \sum_{r\cdot a_q \leq C(CL_i)} y^*_{q,l,i,k} s^*_{q,l,r} r \leq \frac{k\cdot C(CL_i)}{n\cdot a_q}. \tag{22}$$

If we include service $X_{q,l}$ in inequality (22) and truncating its request rate by $\frac{k\cdot C(CL_i)}{n\cdot a_q}$, we can get

$$\sum_{SV_{q,l}\in\mathcal{S}} \sum_{k'\leq k} \sum_{r=1}^{r_{max}} y^*_{q,l,i,k}\cdot s^*_{q,l,r}\cdot \min\left\{r, \frac{k\cdot C(CL_i)}{n\cdot a_q}\right\} \leq (2\cdot k\cdot C(CL_i))/(n\cdot a_q). \tag{23}$$

Recall that we consider algorithms that admit a service before knowing its request rate. Obviously, $y^*_{q,l,i,k}$ and $s^*_{q,l,r}$ are independent variables. We then can re-write the LHS of the above inequality as

$$\begin{aligned}
&\sum_{SV_{q,l}\in\mathcal{S}} \sum_{k'\leq k} \sum_{r=1}^{r_{max}} y^*_{q,l,i,k}\cdot s^*_{q,l,r}\cdot \min\{r, \frac{k\cdot C(CL_i)}{n\cdot a_q}\} \\
&= \sum_{SV_{q,l}\in\mathcal{S}} \sum_{k'\leq k} Pr[y^*_{q,l,i,k}=1] \sum_{r=1}^{r_{max}} Pr[s^*_{q,l,r}=1] \\
&\quad \cdot \min\{r, (k\cdot C(CL_i))(n\cdot a_q)\} \\
&= \sum_{SV_{q,l}\in\mathcal{S}} \sum_{k'\leq k} y^*_{q,l,i,k} E(\min\{r, (k\cdot C(CL_i))/(n\cdot a_q)\}),
\end{aligned}$$

where $Pr[Y]$ denotes the probability that event Y is true. Constraint (18) is met by the optimal solution y^*. Similarly, we can also show that constraint (19) is also met. Therefore, **LP** is a feasible relaxation of **ILP**. We thus have $z' \leq z^* \leq 2\cdot\delta\cdot\kappa\cdot y^*$. □

6 Experiments

We evaluate the performance of the proposed algorithms by both simulations and implementations in a real test-bed.

6.1 Parameter Settings

We use GT-ITM [53] to generate the topology of a two-tiered MEC network with its size varying from 50 to 400 switch nodes. 30% of the switch nodes are attached with cloudlets and 5 datacenters are deployed into the network. We also use a real network topology AS1755 [25]. The number of VMs provided by each cloudlet is randomly generated from [15, 30]. The bandwidth resource of all system is randomly drawn from the range of [800, 1000] Mbps. The costs of transmitting and processing 1 GB of data are set within [$0.05, $0.12] and [$0.15, $0.22], respectively, following typical charges in Amazon EC2 and S3 [4], [5]. The request rate of each service is varied from 50 to 200 requests per time unit, and the traffic volume of each service is randomly drawn from $[1000, 2000]$ Megabytes [7]. The values for α_i and β_i of each cloudlet CL_i are randomly drawn in the range of $[0, 1]$. The data volume of consistency updating from a cached instance to the original instance of a service is set to 10% of the service's data volume. The running time of each algorithm is obtained based on a machine with a 3.70GHz Intel i7 Hexa-core CPU and 16 GiB RAM. Unless otherwise specified, these parameters will be adopted in the default setting. The parameter settings are also shown in Table 2.

We compare the proposed mechanisms `LCF` and `LECF` with the following algorithms:

- *Optimal-restricted solution:* Recall that we adopt algorithm `Appro` in `LCF` to find an approximate solution to the service caching problem in an MEC network with non-selfish network service providers. Instead, in an optimal-restricted solution, we adopt the following optimal solution `OPT` in *step 1* of algorithm `LCF`, which is referred to as algorithm `LCF-OPT`. Recall that $x_{q,l,i}$ is an indicator variable that shows whether service $SV_{q,l}$ is cached into cloudlet CL_i. We then have

$$\texttt{OPT:} \quad \min \sum_{q=1}^{N} \sum_{SV_{q,l}\in\mathcal{S}_q} x_{q,l,i}\cdot c_{q,l,i}, \tag{24}$$

subject to

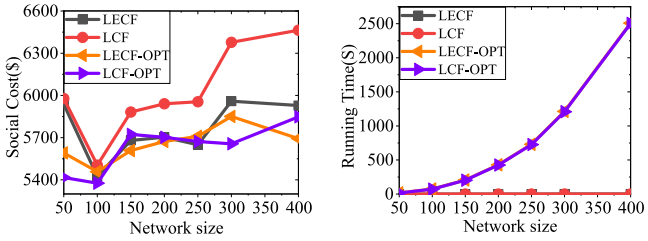
$$\sum_{CL_i\in\mathcal{CL}} x_{q,l,i} \leq 1, \quad \forall \ \mathrm{SV}_{q,l} \tag{25}$$

$$\sum_{q=1}^{N} \sum_{l=1}^{\mathcal{S}_q} x_{q,l,i} a_q r_{q,l} \leq C(CL_i), \forall\ CL_i \tag{26}$$

$$\sum_{q=1}^{N} \sum_{l=1}^{\mathcal{S}_q} x_{q,l,i} b_q r_{q,l} \leq B(CL_i), \forall\ CL_i \tag{27}$$

$$x_{q,l,i} \in \{0, 1\}, \tag{28}$$

where Constraint (25) means that service $SV_{q,l}$ can only be assigned to a single cloudlet in $\mathcal{CL}$.



(a) The social cost of all network service providers (b) The running times

Fig. 2. Algorithm performance comparing with the optimal solution.

Constraints (26) and (27) say that the computing and bandwidth capacities of cloudlet CL_i cannot be violated. Constraint (28) makes sure $x_{q,l,i}$ is a binary indicator variable.

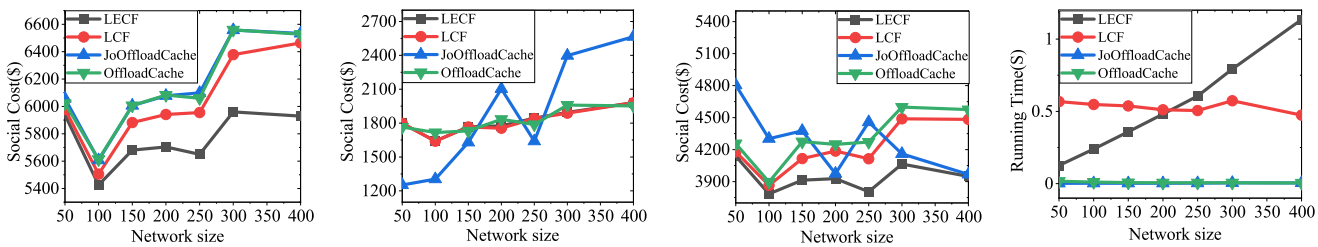
- *The optimal solution to the service caching problem with request rate uncertainty:*We also compare the performance of algorithm LECF against the exact solution with expected costs and request rates, shown in **ILP**, which is referred to as LECF-OPT. It must be mentioned that the real costs and requests can not be obtained in advance. As such, LECF-OPT is an estimation of the optimal solution to the service caching problem with uncertain request rates.
- *Service caching without data updating:* The third benchmark is the algorithm in [45], which provides efficient solution to the problem of service caching in MEC networks. One difference of this study from ours is that we consider a two-tier MEC network that jointly considers task offloading, service caching, and data updating. The data updating however is not considered in [45]. In addition, since the algorithm in [45] does not consider the mobile service market with multiple network service providers, we consider that each network service provider runs the algorithm in [45], without communicating with each other. For simplicity, we refer such joint offloading and caching algorithms as JoOffloadCache.
- *A greedy algorithm:* the fourth benchmark is a greedy algorithm [38], in which each network service provider considers offloading and service placement/cache separately. Specifically, the algorithm simply selects the cloudlets for each request that could achieve an optimal offloading cost. Based on the assignment, services are instantiated in the cloudlets with their requests (or nearby cloudlets). This algorithm is referred to as OffloadCache.

Note that LCF-OPT and LECF-OPT may not be obtained for large problem sizes. In such cases, we use the relaxed versions of LCF-OPT and LECF-OPT as the estimated optimal solutions. This estimation is conservative because the relaxed solutions are much lower than the optimal ones.

6.2 Simulations

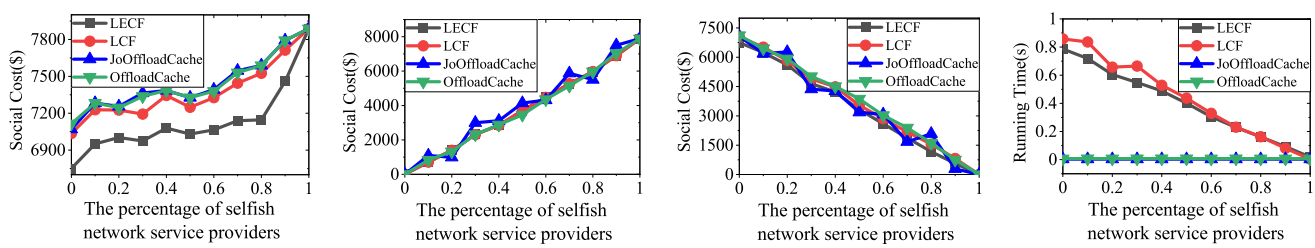
We first evalauted the performance of the proposed algorithms LCF and LECF with algorithms LCF-OPT and LECF-OPT. The results are shown in Fig. 2, from which we can see that algorithms LCF and LECF both have high social costs compared with algorithms LCF-OPT and LECF-OPT. Specifically, algorithm LCF has a higher gap from the optimal solution LCF-OPT while algorithm LECF has a lower gap, because LECF benefits from the randomized rounding technology of algorithm LECF. The running times of algorithms LCF, LECF, LCF-OPT and LECF-OPT are shown in Fig. 2b. Algorithms LCF and LECF have extremely smaller running times than their optimal solutions LCF-OPT and LECF-OPT.

We then evaluated performance of the proposed algorithms LCF and LECF against those of JoOffloadCache



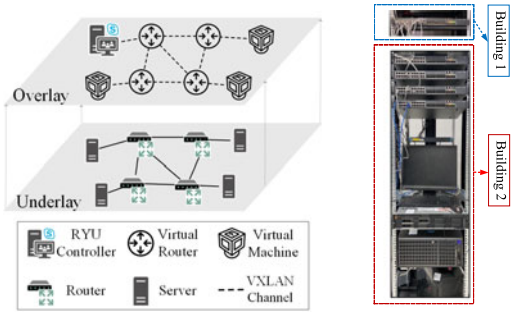
(a) The social cost of all network service providers (b) The social cost of the selfish network service providers (c) The social cost of the coordinated network service providers (d) The running times

Fig. 3. Algorithm performance in GT-ITM generated networks with sizes varied from 50 to 400.



(a) The social cost of all network service providers (b) The social cost of the selfish network service providers (c) The social cost of the coordinated network service providers (d) The running times

Fig. 4. The impact of $(1-\xi)$ on the algorithm performance with size 200.

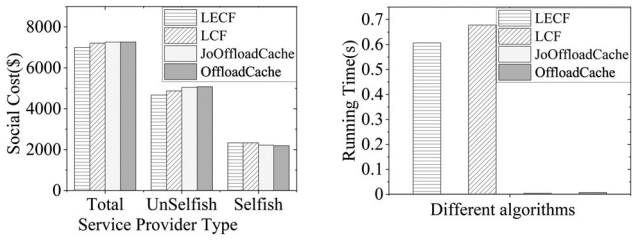


(a) The underlay and overlay of the test-bed

(b) The physical deployment of the hardware switches

Fig. 5. A test-bed with both hardware switches and virtual resources.

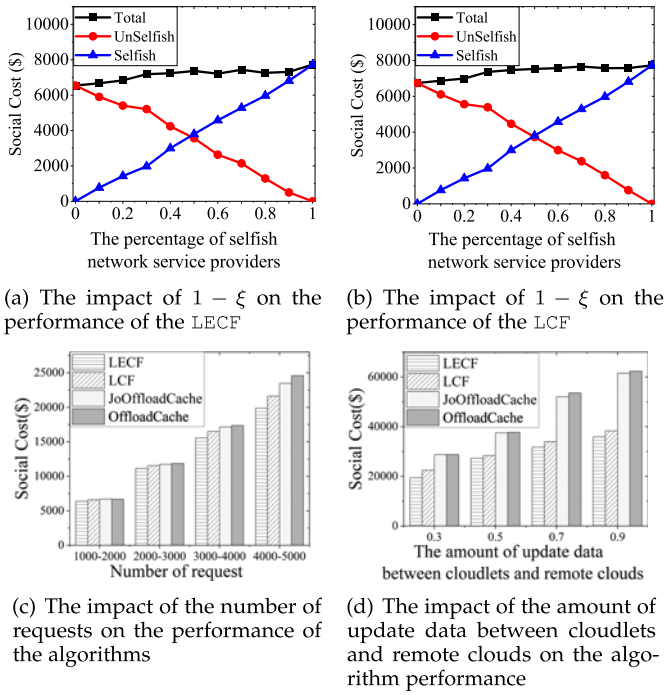
and OffloadCache, by varying the network size from 50 to 400 with a number of 100 network service providers and fixing $(1-\xi)$ to 0.3. Fig. 3 shows the results in terms of the social cost of all network service providers, the social cost of selfish network service providers, the social cost of coordinated network service providers, and the running times. From Fig. 3a, we can see that algorithms LCF and LECF consistently deliver the lower social costs than those of algorithms JoOffloadCache and OffloadCache, while algorithm OffloadCache has the highest social cost. The reason is that algorithms LCF and LECF coordinate a number of coordinated network service providers to avoid significant performance degradation, while algorithms JoOffloadCache and OffloadCache allow each network service provider to make decisions selfishly. In addition, algorithms LECF and LCF optimize the data uploading cost that is ignored by the rest two algorithms. Notice that the cost of LECF first decreases when the network size grows from 50 to 100, because that LECF can adapt request rates to improve computing resource utilization. It then increases afterwards when the network size is larger than 100. The reason is that a larger network usually means longer transmission paths, which incurs higher transmission costs. From Figs. 3b and 3c, we can see that the social cost reduction is mainly by coordinated network service providers. The running times of algorithms LECF, LCF, JoOffloadCache, and OffloadCache are shown in Fig. 3d. It can be seen that algorithms LCF and LECF have slightly higher running times than those of JoOffloadCache, and OffloadCache.



(a) The social cost of all network service providers

(b) The running times

Fig. 6. Performance in the testbed with both physical underlay and virtual overlay, where Total, UnSelfish, and Selfish are used to denote the social costs of all, coordinated, and selfish network service providers, respectively.



(a) The impact of $1-\xi$ on the performance of the LECF

(b) The impact of $1-\xi$ on the performance of the LCF

(c) The impact of the number of requests on the performance of the algorithms

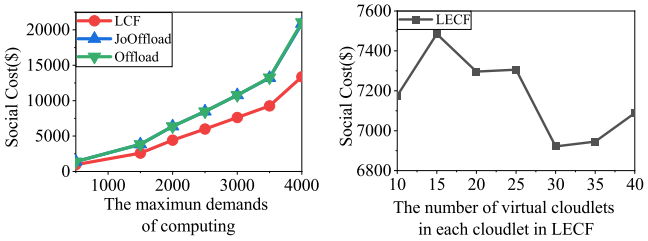
(d) The impact of the amount of update data between cloudlets and remote clouds on the algorithm performance

Fig. 7. The impact of $(1-\xi)$ and the number of requests in the test-bed, where Total, UnSelfish, and Selfish are used to denote the social costs of all, coordinated, and selfish network service providers, respectively.

We finally study the impact of the ratio of the number of selfish network service providers and the total number of network service providers, i.e., $(1-\xi)$, by fixing the network size to 200 and varying the value of $(1-\xi)$ from 0 to 1. Results are shown in Fig. 4. From Fig. 4a, we can see that the social cost of all network service providers by algorithm LCF and LECF increases with the growth of $(1-\xi)$. The reason is that with the increase of selfish players, less network service providers can be coordinated by the approximation solution obtained due to algorithm Appro. This can also be evidenced by Figs. 4b and 4c, where the social cost of selfish network service providers increases while the social cost of coordinated network service providers decreases, with the growth of $(1-\xi)$. We can also see from Fig. 4a that the social cost by LECF is smaller than algorithms JoOffloadCache and OffloadCache until $(1-\xi)$ is increased to 0.8. This is due to a relative high percentage of selfish network service providers. Also, algorithms JoOffloadCache and OffloadCache do not consider selfishness of network service providers at all. The running times of the algorithms are shown in Fig. 4d, from which we can see that the running times of algorithms LCF and LECF keep decreasing with the growth of $(1-\xi)$. The reason behind is that LCF and LECF adopts approximation-restricted strategy to obtain a near-optimal solution by adopting an approximation algorithm Appro. As $(1-\xi)$ grows, less network service providers can be coordinated, and this means that the problem size of Appro becomes smaller, thereby reducing the time due to Appro.

6.3 Implementations in a Test-Bed

Testbed Settings. We build a test-bed consisting of both an underlay with hardware switches and an overlay with



(a) The impact of a_{max} on the performance of the algorithms

(b) The impact of the number n of virtual cloudlets on the performance of the algorithms LECF

Fig. 8. Results of the impact of maximum demands of computing and bandwidth resource demand in the test-bed.

virtual switches, as shown in Fig. 5. The physical underlay consists of five Huawei S5720-32C-HI-24S-AC switches in two on-campus buildings. Each switch is connected to at least two switches. There are also five servers with i7-8700 CPU and 16G RAM. Netconf and SNMP protocols are used to manage the switches and the links interconnecting them. The underlay can be seen as a resource pool with computing and bandwidth resources which can be used to build overlay networks. We use VXLAN to virtualize these switches and servers, by building an overlay that spans the hardware in the two buildings. Specifically, we virtualize an overlay network with a number of Open vSwitch (OVS) [23] nodes and VMs. The overlay network is built following the real topology AS1755. Its OVS nodes and VMs are controlled by a Ryu [35] controller. The proposed algorithms are implemented as Ryu applications. All the rest settings are the same as those of the simulations in the previous subsection.

Performance Results. We investigated the performance of the algorithms in the test-bed, by fixing $(1-\xi)$ to 0.3. The results are shown in Fig. 6, where `Total`, `UnSelfish`, and `Selfish` denote the social costs of all, coordinated, and selfish network service providers, respectively. From Fig. 6a, we can see that algorithm `LECF` has the lowest social cost among algorithms `LCF`, `JoOffloadCache` and `OffloadCache`. The running times of the comparison algorithms are shown in Fig. 6b.

We then study the impact of various parameters on the performance of algorithms `LCF`, `LECF`, `JoOffloadCache` and `OffloadCache` in the test-bed. Figs. 7a and 7b show the impact of $1-\xi$, from which we can see that the social costs of algorithm `LECF` and `LCF` increase with the growth of $(1-\xi)$. This is because the higher percentage of selfish network service providers worsens the social optimum, considering that each network service provider only cares about its own cost instead of the social cost. From the Fig. 7c, we can see that a higher number of requests means a higher social cost. Fig. 7d depicts the impact of the amount of update data between cloudlets and remote clouds on the algorithm performance. It can be seen that a larger amount of to-be-updated data incurs a higher social cost due to the higher bandwidth consumption.

We finally investigate the impact of maximum values of computing resource demands, i.e., a_{max}, on the performance of algorithms `LCF`, `LECF`, `JoOffloadCache` and `OffloadCache` in the test-bed. Fig. 8a shows the impact of a_{max}, from which we can see that the obtained cost is increasing with the growth of a_{max}. The rationale behind is that in the Stackelberg game each cloudlet is partitioned into n_i $(= \{\lfloor\frac{C(CL_i)}{a_{max}}\rfloor, \lfloor\frac{B(CL_i)}{b_{max}}\rfloor\})$ virtual cloudlets, and the number n of virtual cloudlets is decreasing if a_{max} grows. This means that the algorithm has a higher probability to reject some requests in the adjustment procedure. Also, this verifies the correctness of Lemma 2. In addition, we also study the impact of the number of virtual cloudlets on performance of algorithm `LECF`, by varying n from 10 to 40. Fig. 8b shows the impact of n, from which we can see that the obtained cost fluctuates with the growth of n, and reaching the lowest cost when $n = 30$.

7 Conclusion and Future Work

In this paper, we investigated the service caching problem in a two-tiered MEC network for a mobile service market. If the request rate of each service is given, we developed an approximation-restricted optimization framework that can guarantee stable yet near-optimal operations of the service market in the MEC network. Within the framework, we devised an approximation algorithm with an approximation ratio for the problem with non-selfish players. We also designed an efficient, stable Stackelberg congestion game with a provable Price of Anarchy (PoA). Otherwise if the request rate of each service is unknown, we considered the problem with request rate uncertainty by devising an approximation algorithm and a Stackelberg game. We finally evaluated the performance of the proposed algorithms and mechanism by simulations and experiments in a real test-bed. Results showed that the proposed mechanisms obtain a social cost which is 9.2% lower than the ones by existing approaches.

The future potential studies built upon the work in this paper include: (1) the cost of service caching is determined by the congestion levels of cloudlets, As different cloudlets may have time-varying impacts of congestions on service caching costs, which further is determined by the behaviors of cached service instances. That is, how to capture the dynamics of α_i on the congestion cost is challenging and worth further exploration. (2) The data freshness of AI applications plays a vital role in guaranteeing the prediction performance. For example, in an application of real-time object recognition, a moving object needs to be identified as soon as possible. How to consider such responsiveness requirement of services by analyzing the most fresh data is a fundamental challenge in the two-tiered MEC network; and (3) the resources in MEC networks are provisioned in an agile way, such that the cost of service provisioning can be further reduced. A promising way is to adopt finer-grained resource allocation in serverless edge computing (SEC). As such, we will investigate fundamental methods and technologies that enable the hierarchical service market of an agile and low-cost MEC environment.

Acknowledgments

We appreciate the three anonymous referees and the associate editor for their constructive comments and valuable suggestions, which helped us improve the quality and presentation of the paper greatly.

REFERENCES

[1] 5G Services Market Report 2020–2030 - COVID-19 Implications and Growth Assessment. Accessed: Mar. 2021. [Online]. Available: https://www.prnewswire.com/news-releases/5g-services-market-report-2020–2030—covid-19-implications-and-growth-assessment-301078727.html

[2] Alibaba Cloud. Extending the boundaries of the cloud with edge computing, Accessed: Mar. 2021. [Online]. Available: https://www.alibabacloud.com/blog/extending-the-boundaries-of-the-cloud-with-edge-computing_594214

[3] Azure Pricing. Accessed: Mar. 2021. [Online]. Available: https://azure.microsoft.com/en-in/pricing/

[4] Accessed: Mar. 2021. [Online]. Available: https://aws.amazon.com/ec2/

[5] Accessed: Mar. 2021. [Online]. Available: https://aws.amazon.com/s3/

[6] D. Bega, M. Gramaglia, A. Banchs, V. Sciancalepore, and X. Costa-Pérez, "A machine learning approach to 5G infrastructure market optimization," *IEEE Trans. Mobile Comput.*, vol. 19, no. 3, pp. 498–512, Mar. 2020.

[7] T. Braud, P. Zhou, J. Kangasharju, and P. Hui, "Multipath computation offloading for mobile augmented reality," in *Proc. IEEE Int. Conf. Pervasive Comput. Commun.*, 2020, pp. 1–10.

[8] X. Cao, G. Tang, D. Guo, Y. Li, and W. Zhang, "Edge federation: Towards an integrated service provisioning model," *IEEE/ACM Trans. Netw.*, vol. 28, no. 3, pp. 1116–1129, Jun. 2020.

[9] M. Chen, B. Liang, and M. Dong, "Joint offloading and resource allocation for computation and communication in mobile cloud with computing access point," in *Proc. IEEE Conf. Comput. Commun.*, 2019, pp. 1–9.

[10] A. Drucker, F. Kuhn, and R. Oshman, "On the power of the congested clique model," in *Proc. ACM Symp. Princ. Distrib. Comput.*, 2014, pp. 367–376.

[11] S. Deng *et al.*, "Optimal application deployment in resource constrained distributed edges," *IEEE Trans. on Mobile Comput.*, vol. 20, no. 5, pp. 1907–1923, May 2021.

[12] N. Eshraghi and B. Liang, "Joint offloading decision and resource allocation with uncertain task computing requirement," in *Proc. IEEE Conf. Comput. Commun.*, 2019, pp. 1414–1422.

[13] I. Farris, T. Taleb, M. Bagaa, and H. Flick, "Optimizing service replication for mobile delay-sensitive applications in 5G edge network," in *Proc. IEEE Int. Conf. Commun.*, 2017, pp. 1–6.

[14] B. Gao, Z. Zhou, F. Liu, and F. Xu, "Winning at the starting line: Joint network selection and service placement for mobile edge computing," in *Proc. IEEE Conf. Comput. Commun.*, 2019, pp. 1459–1467.

[15] Google Cloud Pricing, Accessed: Mar. 2021. [Online]. Available: https://cloud.google.com/pricing/

[16] T. Hou, G. Feng, S. Qin, and W. Jiang, "Proactive content caching by exploiting transfer learning for mobile edge computing," in *Proc. IEEE Glob. Commun. Conf.*, 2017, pp. 1–6.

[17] W. Hu, Y. Gao, K. Ha, J. Wang, B. Amos, and Z. Chen, "Quantifying the impact of edge computing on mobile applications," in *Proc. 7th ACM SIGOPS Asia-Pacific Workshop Syst.*, 2016, pp. 1–8.

[18] J. W. Hegeman, G. Pandurangan, S. V. Pemmaraju, V. B. Sardeshmukh, and M. Scquizzato, "Toward optimal bounds in the congested clique: Graph connectivity and MST," in *Proc. ACM Symp. Princ. Distrib. Comput.*, 2015, pp. 91–100.

[19] M. Jia and W. Liang, "Delay-sensitive multiplayer augmented reality game planning in mobile edge computing," in *Proc. 21st ACM Int. Conf. Model. Anal. Simul. Wireless Mobile Syst.*, 2018, pp. 147–154.

[20] W. Jiang, G. Feng, and S. Qin, "Optimal cooperative content caching and delivery policy for heterogeneous cellular networks," *IEEE Trans. Mobile Comput.*, vol. 16, no. 5, pp. 1382–1393, May 2017.

[21] P. Jin, X. Fei, Q. Zhang, F. Liu, B. Li, "Latency-aware VNF chain deployment with efficient resource reuse at network edge," in *Proc. IEEE Conf. Comput. Commun.*, 2020, pp. 267–276.

[22] OpenFlow. Accessed: Mar. 2021. [Online]. Available: https://www.opennetworking.org

[23] Open vSwtich, Accessed: Mar. 2021. [Online]. Available: https://www.openvswitch.org

[24] K. Kaur, S. Garg, G. Kaddoum, S. H. Ahmed, and M. Atiquzzaman, "KEIDS: Kubernetes-based energy and interference driven scheduler for industrial IoT in edge-cloud ecosystem," *IEEE Internet Things J.*, vol. 7, no. 5, pp. 4228–4237, May 2020.

[25] S. Knight, H. X. Nguyen, N. Falkner, R. Bowden, and M. Roughan, "The internet topology zoo," *IEEE J. Sel. Areas Commun.*, vol. 29, no. 9, pp. 1765–1775, Oct. 2011.

[26] X. Li, X. Wang, S. Xiao, and V. C. M. Leung, "Delay performance analysis of cooperative cell caching in future mobile networks," in *Proc. Int. Conf. Commun.*, 2015, pp. 5652–5657.

[27] Q. Ma, E. Yeh, and J. Huang, "How bad is selfish caching?," in *Proc. 20th ACM Int. Symp. Mobile Ad Hoc Netw. Comput.*, 2019, pp. 11–20.

[28] S. Misra and N. Saha, "Detour: Dynamic task offloading in software-defined fog for IoT applications," *IEEE J. Sel. Areas Commun.*, vol. 37, no. 5, pp. 1159–1166, May 2019.

[29] A. Ndikumana *et al.*, "Joint communication, computation, caching, and control in big data multi-access edge computing," *IEEE Trans. Mobile Comput.*, vol. 19, no. 6, pp. 1359–1374, Jun. 2020.

[30] N. Nisan, T. Roughgarden, E. Tardos, and V. V. Vazirani, *Algorithmic game theory*. Cambridge, U.K.: Cambridge Univ. Press, 2007.

[31] 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.

[32] L. Pu, L. Jiao, X. Chen, L. Wang, Q. Xie, and J. Xu, "Online resource allocation, content placement and request routing for cost-efficient edge caching in cloud radio access networks," *IEEE J. Sel. Areas Commun.*, vol. 36, no. 8, pp. 1751–1767, Aug. 2018.

[33] Z. Qin *et al.*, "Task selection and scheduling in UAV-enabled MEC for reconnaissance with time-varying priorities," *IEEE Internet Things J.*, vol. 8, no. 24, pp. 17 290–17 307, Dec. 2021.

[34] H. Ren *et al.*, "Efficient algorithms for delay-aware NFV-enabled multicasting in mobile edge clouds with resource sharing," *IEEE Trans. Parallel Distrib. Syst.*, vol. 31, no. 9, pp. 2050–2066, Sep. 2020.

[35] Ryu SDN Controller, Accessed: Mar. 2021. [Online]. Available: https://ryu-sdn.org/

[36] D. B. Shmoys and É. Tardos, "An approximation for the generalized assignment problem," *Math. Program.*, vol. 62, pp. 461–474, 1993.

[37] Q. Sun, S. Ren, and C. Wu, "A provably-efficient online algorithm for re-utilizing unused VM resources for edge providers," in *Proc. IEEE Int. Conf. Commun.*, 2019, pp. 1–6.

[38] J. R. Thomsen, M. L. Yiu, and C. S. Jensen, "Effective caching of shortest paths for location-based services," in *Proc. ACM SIGMOD Int. Conf. Manage. Data*, 2012, pp. 313–324.

[39] T. X. Tran, A. Hajisami, P. Pandey, and D. Pompili, "Collaborative mobile edge computing in 5G networks: New paradigms, scenarios, and challenges," *IEEE Commun. Mag.*, vol. 55, no. 4, pp. 54–61, Apr. 2017.

[40] J. Wang, D. Feng, S. Zhang, A. Liu, and X. Xia, "Joint computation offloading and resource allocation for MEC-enabled IoT systems with imperfect CSI," *IEEE Internet Things J.*, vol. 8, no. 5, pp. 3462–3475, Mar. 2021.

[41] L. Wang, L. Jiao, T. He, J. Li, and M. Mühlhäuser, "Service entity placement for social virtual reality applications in edge computing," in *Proc. IEEE Conf. Comput. Commun.*, 2018, pp. 468–476.

[42] R. Xia, H. Dai, J. Zheng, R. Gu, X. Wang, and G. Chen, "SAFE: Service availability via failure elimination through VNF scaling," in *Proc. 48th Int. Conf. Parallel Process.*, 2019, pp. 1–10.

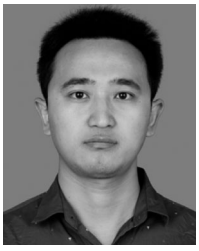
[43] Q. Xia, Z. Xu, W. Liang, S. Yu, S. Guo, and A. Y. Zomaya, "Efficient data placement and replication for QoS-aware approximate query evaluation of big data analytics," *IEEE Trans. Parallel Distrib. Syst.*, vol. 30, no. 12, pp. 2677–2691, Dec. 2019.

[44] X. Xiong, S. Feng, D. Niyato, P. Wang, A. Leshem, and Z. Han, "Joint sponsored and edge caching content service market: A game-theoretic approach," *IEEE Trans. Wireless Commun.*, vol. 18, no. 2, pp. 1166–1181, Feb. 2019.

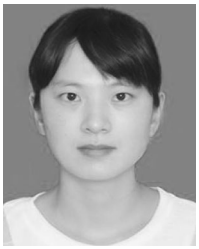
[45] J. Xu, L. Chen, and P. Zhou, "Joint service caching and task offloading for mobile edge computing in dense networks," in *Proc. IEEE Conf. Comput. Commun.*, 2018, pp. 207–215.

[46] Z. Xu, W. Liang, M. Jia, M. Huang, and G. Mao, "Task offloading with network function services in a mobile edge-cloud network," *IEEE Trans. Mobile Comput.*, vol. 18, no. 11, pp. 2672–2685, Nov. 2019.

[47] Z. Xu *et al.*, "Online learning algorithms for offloading augmented reality requests with uncertain demands in MECs," in *Proc. IEEE 41st Int. Conf. Distrib. Comput. Syst.*, 2021, pp. 1064–1074.
[48] Z. Xu *et al.*, "To cache or not to cache: Stable service caching in mobile edge-clouds of a service market," in *Proc. IEEE 40th Int. Conf. Distrib. Comput. Syst.*, 2020, pp. 421–431.
[49] Z. Xu, L. Zhou, S. Chi-Kin Chau, W. Liang, Q. Xia, and P. Zhou, "Collaborate or separate? Distributed service caching in mobile edge clouds," in *Proc. IEEE Conf. Comput. Commun.*, 2020, pp. 2066–2075.
[50] B. Yang, X. Cao, J. Bassey, X. Li, and L. Qian, "Computation offloading in multi-access edge computing: A multi-task learning approach," *IEEE Trans. Mobile Comput.*, vol. 20, no. 9, pp. 2745–2762, Sep. 2021.
[51] R. Yu, G. Xue, and X. Zhang, "Application provisioning in fog computing-enabled internet-of-things: A network perspective," in *Proc. IEEE Conf. Comput. Commun.*, 2018, pp. 783–791.
[52] S. Zang, W. Bao, P. L. Yeoh, B. Vucetic, and Y. Li, "Filling two needs with one deed: Combo pricing plans for computing-intensive multimedia applications," *IEEE J. Sel. Areas Commun.*, vol. 37, no. 7, pp. 1518–1533, Jul. 2019.
[53] E. W. Zegura, K. Calvert, and S. Bhattacharjee, "How to Model an Internetwork," in *Proc. IEEE Conf. Comput. Commun.*, 1996, pp. 594–602.
[54] L. Zhang and N. Ansari, "Latency-aware IoT service provisioning in UAV-aided mobile-edge computing networks," *IEEE Internet Things J.*, vol. 7, no. 10, pp. 10573–10580, Oct. 2020.
[55] Q. Zhang, Q. Zhu, M. F. Zhani, and R. Boutaba, "Dynamic service placement in geographically distributed clouds," in *Proc. 32nd Int. Conf. Distrib. Comput. Syst.*, 2012, pp. 526–535.
[56] B. Zhou, A. V. Dastjerdi, R. N. Calheiros, and R. Buyya, "An online algorithm for task offloading in heterogeneous mobile clouds," *ACM Trans. Internet Technol.*, vol. 18, no. 2, 2018, Art. no. 23.
[57] P. Zhou, K. Wang, J. Xu, and D. Wu, "Differentially-private and trustworthy online social multimedia big data retrieval in edge computing," *IEEE Trans. Multimedia*, vol. 21, no. 3, pp. 539–554, Mar. 2019.
[58] T. Zhu, J. Li, Z. Cai, Y. Li and H. Gao, "Computation scheduling for wireless powered mobile edge computing networks," in *Proc. IEEE Conf. Comput. Commun.*, 2020, pp. 596–605.



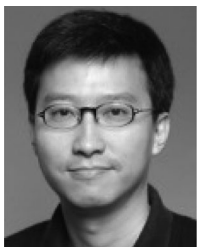
Zichuan Xu (Member, IEEE) received the BSc and ME degrees in computer science from the Dalian University of Technology, China, in 2011 and 2008, respectively, and the PhD degree from Australian National University in 2016. From 2016 to 2017, he was a research associate with the Department of Electronic and Electrical Engineering, University College London, U.K. He is currently a full professor and PhD advisor with the School of Software, Dalian University of Technology. He is also a xinghai scholar with the Dalian University of Technology. His research interests include cloud computing, mobile edge computing, network function virtualization, software-defined networking, algorithmic game theory, and optimization problems.



Qiufen Xia (Member, IEEE) received the BSc and ME degrees in computer science from the Dalian University of Technology, China, in 2012 and 2009, respectively, and the PhD degree from Australian National University in 2017. She is currently an associate professor with the Dalian University of Technology. Her research interests include mobile cloud computing, query evaluation, big data analytics, big data management in distributed clouds, and cloud computing.

Lin Wang (Student Member, IEEE) received the BS degree from the Taiyuan University of Technology, Taiyuan, China, in 2016, and the MS degree from Inner Mongolia University, Hohhot, China, in 2019. He is currently working toward the PhD degree with the School of Software, Dalian University of Technology. His main research interests include mobile edge computing, network storage, task offloading, deep learning, and optimization problems.

Pan Zhou (Member, IEEE) received the BS degree in advanced class from the Huazhong University of Science and Technology (HUST), Wuhan, China, in 2006, and the PhD degree from the School of Electrical and Computer Engineering, Georgia Institute of Technology, Atlanta, GA, USA, in 2011. He is currently an associate professor with the School of Electronic Information and Communications, HUST. From 2011 to 2013, he was a Senior Technical Member with Oracle, Inc., American, Boston, MA, USA. His current research interests include security and privacy, machine learning and Big Data analytics, and information networks.



John C. S. Lui (Fellow, IEEE) received the PhD degree in computer science from the University of California, Los Angeles, Los Angeles, CA, USA, in 1991. He is currently a professor with the Department of Computer Science and Engineering, Chinese University of Hong Kong, Hong Kong, where he was the chairman of the Department from 2005 to 2011. His current research interests include communication networks, network/system security (such as, cloud security and mobile security), network economics, network sciences (such as, online social networks and information spreading), cloud computing, large-scale distributed systems, and performance evaluation theory. He is an elected member of the IFIP WG 7.3. He is on the editorial board of the *IEEE/ACM Transactions on Networking, IEEE Transactions on Computers*, *IEEE Transactions on Parallel and Distributed Systems*, *Journal of Performance Evaluation*, and *International Journal of Network Security*. He is a fellow of ACM and a croucher senior research fellow.

Weifa Liang (Senior Member, IEEE) received the BSc degree in computer science from Wuhan University, China, in 1984, the ME degree in computer science from the University of Science and Technology of China in 1989, and the PhD degree in computer science from Australian National University in 1998. He is currently a professor with the Department of Computer Science, City University of Hong Kong, Hong Kong, prior to that position, he was a professor with Australian National University. His research interests include design and analysis of energy efficient routing protocols for wireless ad hoc and sensor networks, the Internet of Things, mobile edge computing (MEC), network function virtualization (NFV), software-defined networking (SDN), design and analysis of parallel and distributed algorithms, approximation algorithms, combinatorial optimization, and graph theory. He is currently an associate editor for the *IEEE Transactions on Communications*.

Wenzheng Xu (Member, IEEE) received the BSc, ME, and PhD degrees in computer science from Sun Yat-sen University, Guangzhou, China, in 2008, 2010, and 2015, respectively. He is currently an associate professor with Sichuan University. He was also a visitor with the Australian National University and Chinese University of Hong Kong. His research interests include wireless ad hoc and sensor networks, mobile computing, approximation algorithms, combinatorial optimization, online social networks, and graph theory.

Guowei Wu received the PhD degree from Harbin Engineering University in 2003. He is currently a professor with the School of Software, Dalian University of Technology, China. He has authored or coauthored more than 100 papers in journals and conferences. His research interests include embedded real-time systems, cyber-physical systems, and smart edge computing.