# Near-Optimal and Collaborative Service Caching in Mobile Edge Clouds

Zichuan Xu, *Member, IEEE,* Lizhen Zhou, Sid Chi-Kin Chau, *Member, IEEE,* Weifa Liang, *Senior Member, IEEE,* Haipeng Dai, *Member, IEEE,* Lixing Chen, *Member, IEEE,* Wenzheng Xu, *Member, IEEE,* Qiufen Xia, *Member, IEEE,* and Pan Zhou, *Member, IEEE*

**Abstract**—With the development of 5G technology, mobile edge computing is emerging as an enabling technique to reduce the response latency of network services by deploying cloudlets at 5G base stations to form mobile edge cloud (MEC) networks. Network service providers now shift their services from remote clouds to cloudlets of MEC networks in the proximity of users. However, the permanent placement of network services into an MEC network is not economic due to limited computing and bandwidth resources imposed on its cloudlets. A smart way is to cache frequently demanded services from remote clouds to cloudlets of the MEC network.
In this paper, we study the problem of service caching in an MEC network under a service market with multiple network service providers competing for both computation and bandwidth resources in terms of Virtual Machines (VMs) in the MEC network. We first propose an Integer Linear Program (ILP) solution and a randomized rounding algorithm, for the problem without VM sharing among different network service providers. We then devise a distributed and stable game-theoretical mechanism for the problem with VM sharing among network service providers, with the aim to minimize the social cost of all network service providers, through introducing a novel cost sharing model and a coalition formation game. We also analyze the performance guarantee of the proposed mechanism, Strong Price of Anarchy (SPoA). We thirdly consider the cost- and delay-sensitive service caching problem with temporal VM sharing, and propose a mechanism with provable SPoA. We finally evaluate the performance through extensive simulations and a real world test-bed implementation. Experimental results demonstrate that the proposed algorithms outperform existing approaches by achieving at least $40\%$ lower social cost via service caching and resource sharing among different network service providers.

**Index Terms**—Service caching; mobile edge clouds; resource sharing; coalition formation; strong price of anarchy; game theory.

## 1 Introduction

In the past decade, with the development of cloud, communication and AI technologies, various multimedia applications have been attracting much attentions by network service and infrastructure providers. For example, Virtual Reality (VR) services have been deployed in data centers located in core networks for the real-time processing of 8K video data collected from VR headsets. Such VR services consume vast amounts of computing and bandwidth resources in data centers for rendering and bandwidth resources to receive 8K video data from headsets. However, network service providers are facing difficulties in meeting delay requirements of mobile users, due to the large volume of data to be processed and ever-increasingly congested core networks. 5G-enabled mobile edge computing is a promising solution to address this issue, by deploying cloudlets (e.g., edge servers) in locations (e.g., 5G base stations) close to mobile users and providing VR services within the proximity of the mobile users. Network service providers can cache their services into cloudlets of mobile edge clouds (MEC) to meet various QoS requirements of users.

In this paper, we consider a fundamental problem of *collaborative service caching* in an MEC network that allows multiple network service providers to collaboratively cache their services from remote data centers to cloudlets of an MEC network. There usually is a service market of the MEC network with different role players. Specifically, an infrastructure provider owns and operates the MEC network. Meanwhile, medium- and small-scale network service providers who do not own physical resources lease Virtual Machines (VMs) from the infrastructure provider [37]. These network service providers then have to bear the costs of using the leased VMs in cloudlets owned to them. To reduce their costs, the network service providers may share their leased VMs with each other, and these sharing VMs service providers form a coalition.

Collaborative service caching in an MEC network with VM sharing among different network service providers


- *Z. Xu and L. Zhou are with the School of Software, Dalian University of Technology, and the Key Laboratory for Ubiquitous Network and Service Software of Liaoning Province, P. R. China. E-mails: z.xu@dlut.edu.cn, zhou_lizhen@mail.dlut.edu.cn.*
- *S. C. Chau is with School of Computing, the Australian National University, Canberra, ACT 2601, Australia. E-mail: sid.chau@anu.edu.au.*
- *W. Liang is with the Department of Computer Science, City University of Hong Kong, P. R. China. Email: weifa.liang@cityu.edu.hk.*
- *H. Dai is with the State Key Laboratory for Novel Software Technology, Nanjing University, Nanjing, Jiangsu 210023, P. R. China. E-mail: haipengdai@nju.edu.cn.*
- *L. Chen is with the Institute of Cyber Science and Technology, Shanghai Jiao Tong University, Shanghai China, Email: lxchen@sjtu.edu.cn.*
- *W. Xu is with the Department of computer network and communication, Sichuan University, Chengdu, Sichuan, P. R. China, 610000. Email: wenzheng.xu@scu.edu.cn.*
- *Q. Xia (Corresponding author) is with the International School of Information Science and Engineering, Dalian University of Technology, and the Key Laboratory for Ubiquitous Network and Service Software of Liaoning Province, P. R. China. E-mail: qiufenxia@dlut.edu.cn.*
- *Pan Zhou is with Hubei Engineering Research Center on Big Data Security, School of Cyber Science and Engineering, Huazhong University of Science & Technology, Wuhan, Hubei 430074, China. E-mail: panzhou@hust.edu.cn.*

poses several fundamental challenges: (1) Network service providers are selfish and only care about their own revenues. It is impossible to centrally coordinate them towards the social optimum via a centralized mechanism. A distributed mechanism for the selection of caching locations of 5G services is needed. Thus, how to design a distributed mechanism for collaborative service caching so that each network service provider has an incentive to participate in the service market. Also, how to guarantee such a service market is stable, i.e., no players can increase its utility by deviating from its current coalition. (2) To reduce the cost of collaborative service caching, a network service provider may choose to share its leased VMs (with computing and bandwidth resources) with others when its leased VMs are under-utilized. On one hand, if a group of network service providers sharing the same resource is considered as a *coalition*, how to design an efficient cost sharing mechanism for coalitions to mitigate the system performance degradation due to resource sharing. On the other hand, within each coalition, how to enable temporal sharing of VMs to improve the system resource utilization is challenging; (3) The selfish behaviours of network service providers give rise to outcomes that deviate from the social optimum. A near-optimal solution with a bounded gap from the social optimum thus is needed.

Although there are studies of task offloading and service placement in content centric networks (CCNs) or MEC networks, they are essentially different from collaborative service caching in an MEC network in this paper. First, the research on CCNs focused on caching contents in nodes with storage capacities when contents are requested. Services that process the contents are usually ignored. Second, studies on task offloading and service placement usually assume that services are only deployed in an MEC [21], [22], [27], [41], [43], [50], [58]. This however is not applicable to service caching from remote data centers to cloudlets of an MEC network with multiple network service providers.

To the best of our knowledge, we are the first to formulate the problem of collaborative service caching in an MEC network for a service market with multiple network service providers. The novelties of our study lie in the following: (1) We consider a temporary caching of services from remote data centers to cloudlets in the MEC network. (2) We propose a randomized rounding algorithm with a provable approximation ratio and high probability for the problem. (3) We propose the very first distributed mechanism with a guaranteed gap of the solution obtained to the optimal one, for the service caching problem with or without VM sharing among different network service providers.

The main contributions of this paper are as follows.

- We formulate an Integer Linear Program (ILP) solution to the service caching problem without VM sharing, and devise a randomized algorithm with a good approximation ratio at the expense of moderate bounded resource violations.
- We design a novel coalition formation game for the problem with VM sharing. We aim to minimize the total cost of all network service providers.
- We devise a mechanism for the coalition formation game with a provable Price-of-Anarchy (PoA), which guarantees the worst-case performance gap between the obtained social cost and the optimal one. We also design another mechanism for a variant of the problem that allows temporal VM sharing among different network service providers in each cloudlet based on the proposed mechanism.
- We show how to extend the proposed coalition formation games to consider user mobility and network uncertainties.
- We evaluate the performance of the proposed algorithms via both experimental simulations and implementations in a real test-bed. Experimental results show that the performance of the proposed algorithms outperform existing approaches by achieving at least $40\%$ lower social cost via service caching and resource sharing among different network service providers.

The remainder of the paper is arranged as follows. Section 2 summarizes state-of-the-arts on this topic. Section 3 introduces the system model, notations and problem formulations. Section 4 presents an ILP solution and a randomized approximation algorithm for the cost-sensitive service caching problem. Section 5 proposes a distributed mechanism for the coalition formation game of the cost- and delay-sensitive service caching problem. Section 6 devises a mechanism for the cost- and delay-sensitive service caching problem with temporal VM sharing. Section 7 considers the user mobility and network uncertainties in the proposed coalition formation games. Section 8 evaluates the performance of the proposed algorithms and mechanisms, and Section 9 concludes the paper.

## 2 Related Work

Service caching has attracted much attention [27], [41], [58], since caching services MEC networks shows great potential in reducing service latency. According to the 'entity' that can be cached or offloaded to edge clouds, existing studies can be classified into three categories: (1) content/data caching; (2) task/computation offloading; and (3) service placement and caching, respectively.

For content/data caching, most studies focused on novel architectures, methods, and algorithms for content centric networks [2], [32], [40], [55], [56], conventional cloud networks [23], or cellular networks [3], [19], [33]. For example, to improve efficiency of content caching, lots of efforts have been devoted to optimize the path selection [30], server placement [36] and content duplication strategy [5]. These studies focused on the placement of contents and data only, and the placement of services used to process data has not been considered.

Closely related to the service caching problem of this paper is the works on task offloading and service placement [7], [12], [15], [27], [47], [52], [53], [54], [57], [58]. For task offloading, Misra *et al.* [27] studied task offloading 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.* [58] investigated the joint task offloading and scheduling problem, by considering wireless network connectivity and mobile device mobility. In addition, the service placement problem is well investigated. For example, He *et al.* [15] investigated the problem of joint

service placement and request scheduling in mobile edge computing systems under communication, computation, and storage constraints. Zhang *et al.* [57] studied the problem of service placement to minimize service hosting costs while ensuring performance requirements.

There are a handful of studies on service caching [17], [26], [29], [38], [41], [45], [47], [48], [59], and some of them focused on caching based on popularity of services [17], [59]. For example, Huang *et al.* [17] proposed algorithms to cache more popular services based on CPU, RAM, and disk resources in an edge cloud. Zhang *et al.* [59] proposed a novel scheme of caching popular services in an overlay network, by jointly considering the size capacity of switches and delay requirements of users. Moreover, others focued on the interplay of service providers [26], [48]. Xiong *et al.* [48] investigated the interplay between the network infrastructure provider, content service providers, and mobile users under a three-stage Stackelberg game. In particular, Liang *et al.* [26] studied the service entity caching problem from the utility perspective, which can be generalized to the service caching problem by modifying the 'utility'. Collaborative caching has also been investigated in [29], [41]. For example, The cooperative service caching and workload scheduling in MEC networks has been studied in [29], with the aim to minimize the service response time and the outsourcing traffic to central clouds. In [49], a joint service pricing scheme is proposed to coordinate with service caching and task offloading. However, in these mentioned studies, there is only one network service provider and resource sharing among different network service providers has not been considered. Most existing works did not consider service caching by joint consideration of both the MEC network with cloudlets and remote data centers, and ignored data updating between cached services in MEC and their original version in remote data centers.

In summary, we are the first to consider service caching in an MEC network for a service market with multiple network service providers, by leveraging the coalition formation technique to enable efficient and effective collaborations among different network service providers, such that their leased VMs can be fully utilized through resource sharing and the total (social) cost of the market is minimized. It must be mentioned that this is an extension of the conference paper [42].

## 3 Preliminary

In this section, we first introduce the system model, notions and notations, and then define the problems precisely.

### 3.1 System Model

We consider an MEC network $G = (\mathcal{CL} \cup \mathcal{DC}, E)$ with a set $\mathcal{CL}$ of cloudlets deployed in locations close to users, a set $\mathcal{DC}$ of data centers with abundant computing resource in remote locations, and a set $E$ of links that interconnect cloudlets and data centers in $\mathcal{CL} \cup \mathcal{DC}$. Let $CL_i$ be a cloudlet in $\mathcal{CL}$. Each cloudlet $CL_i \in \mathcal{CL}$ has limited computing and bandwidth resources to implement various services in VMs. We focus on collaborations among different network service providers, for whom the management and resource allocations for VMs are transparent. They only see what types of VMs are available in each cloudlet of the MEC network. We thus assume that each cloudlet maintains a pool of VMs, which is normally maintained by an infrastructure provider. The number of available VMs in each cloudlet is released to network service providers periodically, and the network service providers collaborate to share the limited number of VMs in each cloudlet. In addition, according to the mechanisms of most cloud platforms [1], each VM is usually assigned an amount of bandwidth resource to guarantee the data transmission rate from or to the VM. For example, a "r5.12xlarge" VM instance in Amazon EC2 has network bandwidth resource of 10 Gbps [1]. Denote by $C(CL_i)$ and $B(CL_i)$ the computing and bandwidth capacities of each cloudlet $CL_i$. Each data center in $\mathcal{DC}$ hosts a set of to-be-cached services.

In this paper, we assume that the MEC network is built and operated by an infrastructure provider. The resources in the MEC network are leased to small- and medium network service providers in terms of VMs. It must be mentioned that this is a widely adopted market model of IT services, which releases network service providers from the work of managing physical resources [1]. As such, the network service providers execute their service requests in the leased or sharing VMs in the MEC network, as shown in Fig. 1. Specifically, each network service provider is self-interest and cares for only its own utility. However, different network service providers have different resource demands. They may collaborate together to share resources in a cloudlet and thus share the cost of using the resources. Particularly, most infrastructure providers virtualize their computing resource in each cloudlet into different types of VMs. Since an infrastructure provider normally does not know the exact resource demand of a service of a network service provider, the amount of resource assigned to each VM may not match the actual demand of a service. This means that a VM may be shared with other network service providers if its own network service provider wants to reduce the cost when the VM is idle. In addition, each network service provider usually has a stable set of loyal users who would not shift to the other network service providers in a short term [34], if the overall service quality of the provider is stable. Fig. 1 shows an example of service caching in an MEC network $G$.

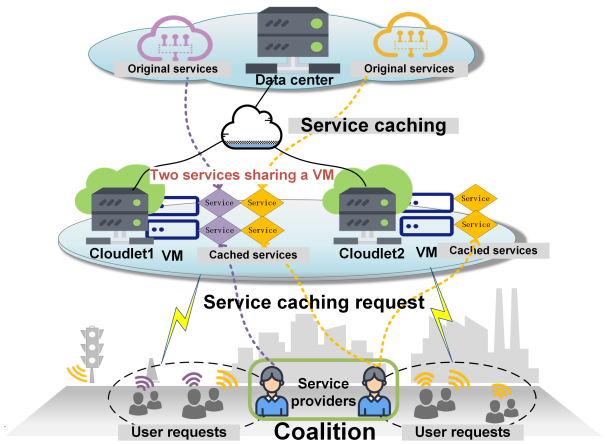



Fig. 1. An example of an MEC network and service caching.

### 3.2 Service Caching from Remote Data Centers to Cloudlets with State Updating

We consider a number of network service providers who already have services originally deployed in remote data centers. They seek opportunities of caching the services from remote data centers to cloudlets in $\mathcal{CL}$ of the MEC network to minimize the delay experienced by its users, which is referred to as *service caching*.

Assume that there are $L$ network service providers in an MEC network, and let $sp_l$ be a network service provider with $1 \leq l \leq L$. Each network service provider $sp_l$ has a service, denoted by $S_l$, which is deployed in a remote data center. Each service $S_l$ serves multiple mobile users by implementing user requests. If network service provider $sp_l$ caches a service instance into a cloudlet, its user requests will be re-directed to its *cached instance* in the cloudlet. Otherwise, the *original instance* of $S_l$ in a data center continues serving its user requests. Let $\rho_l$ be the request rate of service $S_l$ in terms of the number of requests arrived per time unit. Example services provided by such network service providers include data processing services with queries arriving at a certain rate, VR services with requests of rendering a stream of images, and various network functions with requests of processing network flows.

Considering that cloudlets in MEC networks have limited computing and bandwidth resources, it is not economic to cache all services permanently in cloudlets. Therefore, the cached instance of $S_l$ in a cloudlet $CL_i$ may be revoked for the caching of other services, and its occupied VM will be used for the caching of other services. To enable such flexible caching of services, the original service instances in data centers have to be kept in case of possible removals of its cached instances. Furthermore, considering that the instances of $S_l$ are cached in the MEC network temporarily, the data that is processed by its cached instances may be needed by the service in future. This means that the states generated by the cached instance must be forwarded to its original service; otherwise, the original service may be no longer functional when its cached instance is destroyed from cloudlets.

Each network service provider $sp_l$ has a preference of cloudlets to cache its service, considering that some cloudlets have necessary data or software required by $S_l$. Let $\mathcal{CL}(S_l)$ be the set of cloudlets that are preferred by $S_l$. Only the cloudlets in $\mathcal{CL}(S_l)$ can cache instances of $S_l$, and assume that set $\mathcal{CL}(S_l)$ is given as a priori. Notice that the model and algorithms of this paper can be easily extended to consider different types of services, which is discussed in Section 8.

### 3.3 Default and Collaboration Costs

Caching services in cloudlets of an MEC network incurs cost of using VMs. Ideally, each cached service instance is implemented in a single VM of a cloudlet. The cost incurred in such an implementation is referred to as the *default cost* of a network service provider, which consists of the usage costs of computing and bandwidth resources of a VM. Note that the default cost is the cost of using a VM to implement service $S_l$ solely; that is, the VM is not shared with the other network service providers. On the contrary, if a network service provider shares its VMs with others, the incurred resource usage cost is referred to as *the collaboration cost*.

**Default cost**: The VM hosting the cached service instances in cloudlet $CL_i$ consumes both computing and bandwidth resources to implement user requests. Denote by $c^p_{l,i}$ the cost of using a unit of computing resource in $CL_i$ by service $S_l$. Following existing studies [8], [9], [21], [22], the usage cost of computing resource usage of service $S_l$ in $CL_i$ is

$$c^p_{l,i} \cdot C^{vm}_{l,i}, \tag{1}$$

where $C^{vm}_{l,i}$ is the amount of computing resource allocated to a VM in cloudlet $CL_i \in \mathcal{CL}(S_l)$ for service $S_l$. In addition, bandwidth resource is needed to transfer data from/to cloudlet $CL_i$. We consider that the usage cost of bandwidth resource is proportionally shared among the cached instances in $CL_i$ [8], [9], [21], [22]. Let $c^b_{l,i}$ be the cost of using a unit of bandwidth resource of cloudlet $CL_i$ by service $S_l$, then the bandwidth resource usage cost is

$$c^b_{l,i} \cdot B^{vm}_{l,i}, \tag{2}$$

where $B^{vm}_{l,i}$ is the amount of bandwidth resource that is allocated to a VM in cloudlet $CL_i \in \mathcal{CL}(S_l)$ for service $S_l$.

Denote by $c_{l,i}$ the default cost of caching an instance of service $S_l$ in a VM of cloudlet $CL_i$, then,

$$c_{l,i} = c^p_{l,i} \cdot C^{vm}_{l,i} + c^b_{l,i} \cdot B^{vm}_{l,i}. \tag{3}$$

**Collaboration cost**: A network service provider may share its VMs with other network service providers when its VMs are idle if there is no security breaches or privacy leakage. In this case, the cost will be shared among the network service providers as well.

### 3.4 Default and Collaboration Delays

The delay experienced by implementing a user request in a cached instance of a service is mainly due to the waiting time in queuing.

A network service provider has a *default delay* if it does not share its idle VMs with others. This means that its service $S_l$ will use its VM solely, and its service requests will queue at its VMs. The default delay of processing requests of $sp_l$ thus consists of the queuing delay at the VM [28], which is actually an M/M/1 queue, i.e.,

$$d^{dft}_{l,i} = \frac{1}{\mu_i - \rho_l}, \tag{4}$$

where $\mu_i$ is the request processing rate by services in cloudlet $CL_i$, and $\rho_l$ is the request rate of $S_l$.

On the other hand, if the network service provider shares its VMs in a coalition. Its requests can be processed by the VMs of the other network service providers in its coalition. We thus assume that there is a M/M/c queue in each coalition which accepts the requests of the network service providers of the coalition. Once a network service provider selects a coalition, each of its request will stay in the queue. Let $g_i$ be the coalition with a number $c_i$ of VMs that can be shared by its network service providers. As such, in the M/M/c queue $c = c_i$. The collaboration delay $d^{col}_{l,i}$ thus can be calculated by

$$d^{col}_{l,i} = \frac{E(c_i, (\sum_{sp_l \in g_i} \rho_l)/(c_i \cdot \mu_i))}{c_i \cdot \mu_i - \sum_{sp_l \in g_i} \rho_l} + \frac{1}{\mu_i}, \tag{5}$$

where $E(c_i, (\sum_{sp_l \in g_i} \rho_l)/(c_i \cdot \mu_i))$ is the probability that an arriving request will need to queue as opposed to immediately being served, which is known as the Erlang's C formula [28] with $c_i$ VMs (corresponding to servers of the queue system).

Let $d_{l,i}$ be the average delay experienced by user requests of service $S_l$ if its requests are served in a cached service instance in cloudlet $CL_i$. Therefore, we have

$$d_{l,i} = \begin{cases} d_{l,i}^{col}, & \text{if network service providers share VMs} \\ d_{l,i}^{dft}, & \text{otherwise.} \end{cases} \quad (6)$$

Let $d_l^{DC}$ be the average delay experienced by user requests of service $S_l$ if its requests are served in its original instance in a data center. For each service $S_l$, $d_l^{DC}$ is usually no less than $d_{l,i}$ for each $CL_i \in \mathcal{CL}$, which can be obtained from historical information. Since data centers are usually located in remote areas, the delay of cached service instances is far smaller than that of original service instances, i.e., $d_{l,i} \ll d_l^{DC}$. For example, data center access latency via can be 16 times of cloudlet access latency [18].

### 3.5 Default Utility

We consider that each network service provider aims to shorten the response delay of its services. We assume that the delay experienced by a request of service $S_l$ is the major component of the utility of network service provider $sp_l$, since prohibitively long delays may incur cost penalties. That is, the utility of $sp_l$ is a function of the improved delay of serving users in cached service instances over that of serving users in the original instances in data centers. Notice that we do not make use of conventional utility models based on resource usages, because we consider the service caching from the perspectives of network service providers who do not own physical resources in MEC networks.

Let $u_l^{dft}$ be the *default utility* of network service provider $sp_l$ with a default cost, i.e., the VM caching the service of $sp_l$ is not shared with other network service providers, which can be formulated as

$$u_l^{dft}(CL_i) = (v_l \cdot (d_l^{DC} - d_{l,i}) - c_{l,i}), \quad (7)$$

if service $S_l$ has a cached instance in a VM of cloudlet $CL_i \in \mathcal{CL}(S_l)$. Notice that $v_l$ is a private value of network service provider $sp_l$ that represents the utility it can obtain by reducing a unit delay that is experienced by requests of $S_l$. Since we consider delay-sensitive services, we assume that $u_l^{dft}(CL_i)$ is always positive for any $CL_i \in \mathcal{CL}_l$.

### 3.6 Coalition Formation and Collaboration Utility

We consider network service providers as *players*. From the perspective of a network service provider, it is vital to determine with whom to share its VM. Sharing its VM with others having high request rates may not be a good choice. Instead, network service providers benefit from sharing VMs if their services have complementary request rates. To this end, the network service providers that agree to share VMs naturally form into a group with a common interest, i.e., 'minimizing their cost incurred by idle VMs'. As such, network service providers that share VMs in the same cloudlet are considered as a coalition. Let $g_i$ be the coalition in cloudlet $CL_i$. Since both computing and bandwidth resources are limited, we assume that the coalition $g_i$ of each cloudlet $CL_i$ has a capacity in terms of the maximum number of network service providers that can join. We refer to each coalition $g_i$ as *a capacitated coalition*, and its capacity is denoted by $K_i$, which depends on the computing and bandwidth resources of the cloudlet of $g_i$.

Denote by $p_l(g_i)$ the payment that its network service provider $sp_l$ has to pay by staying in coalition $g_i$. If $sp_l$ chooses to stay in coalition $g_i$, it obtains a revenue due to the collaboration with other network service providers in the coalition. Such revenue is referred to as the *collaboration utility*. Let $u_l^{coll}(g_i)$ be the collaboration utility in coalition $g_i$ of cloudlet $CL_i$, then,

$$u_l^{coll}(g_i) = v_l \cdot (d_l^{DC} - d_{l,i}) - p_l(g_i). \quad (8)$$

The *utility* of network service provider $sp_l$ obtained through collaborating with other network service providers, by sharing its VM is defined as the difference between its collaboration utility and its default utility. Assume that $sp_l$'s default utility can be maximized in cloudlet $CL_{i'}$ ($\in \mathcal{CL}(S_l)$). Network service provider $sp_l$ may prefer to cache its service in $CL_i$ if its collaboration utility can be maximized by sharing a VM with other network service providers in $CL_i$. Denote by $u_l(g_i)$ the utility obtained by network service provider $sp_l$ by staying in coalition $g_i$, which is given as

$$\begin{aligned} u_l(g_i) &= u_l^{coll}(g_i) - u_l^{dft}(CL_{l'}) \\ &= c_{l,i'} - \Big(p_l(g_i) - v_l \cdot d_{l,i'} + v_l \cdot d_{l,i}\Big). \end{aligned} \quad (9)$$

Considering that each network service provider $sp_l$ has a default utility with resource usage cost $c_{l,i'}$ without sharing, it may choose to share with others if its utility can be further improved. This means that $p_l(g_i) - v_l \cdot d_{l,i'} + v_l \cdot d_{l,i}$ in Eq. (9) can be minimized. We refer to such a cost as the *collaboration cost*, denoted by $c_l^{coll}(g_i)$, which is defined by

$$c_l^{coll}(g_i) = p_l(g_i) - v_l \cdot d_{l,i'} + v_l \cdot d_{l,i}. \quad (10)$$

### 3.7 Problem Definitions

Given an MEC network $G$ and a set of services $\mathcal{S} = \{S_l \mid 1 \leq l \leq L\}$ to be cached in $G$, each network service provider $sp_l$ requires to cache an instance of its service $S_l$ in its preferred set $\mathcal{CL}(S_l)$ of cloudlets. We aim to investigate the benefit of collaborative service caching in the MEC network with multiple network service providers. To this end, we first study the problem of *cost- and delay-sensitive service caching without VM sharing*. Considering the solution to the problem without VM sharing as a baseline, we then investigate the impact of VM sharing by allowing network service providers to collaborate. Note that different infrastructure providers may allow network service providers to make decisions in different granularities. This means that the network service providers may or may not be allowed to choose a specific VM in a cloudlet. Therefore, each network service provider forms coalitions based on the information of cloudlets, if each network service provider is allowed to choose cloudlets only. Actual VM assignment is done by the infrastructure provider of the MEC network. Otherwise, a network service provider can choose a specific VM of a cloudlet when making decisions of coalition formation. Due to such considerations,

we also study the problems of *cost- and delay-sensitive service caching with VM sharing* and *cost- and delay-sensitive service caching with temporal VM sharing*, respectively.

**Problem 1:** *The cost- and delay-sensitive service caching problem without VM sharing* is to cache services of network service providers to cloudlets of an MEC network, such that the total default cost of caching services is minimized, while meeting the bandwidth and computing resource capacity constraints on each cloudlet and the delay requirement $d_l^{req}$ of each service request $S_l$. Note that the total default cost is the sum of default costs of all network service providers.

To quantify the solution quality of **Problem 1**, we adopt the concept of *approximation ratio* defined as the ratio of a feasible solution to an optimal one of the problem.

**Problem 2:** *The cost- and delay-sensitive service caching problem with VM sharing* is to form a collection of stable coalitions (for VM sharing) of network service providers by caching their services into cloudlets in an MEC network, such that the social cost is minimized, subject to the capacity on the coalition of each cloudlet, where the *social cost* is the total collaboration cost of all network service providers through collaborating with the other network service providers. Let $C$ be the collection of stable coalitions, and the social cost can be represented by $c^{coll}(C)$, which is defined as the total collaboration cost of all network service providers.

**Problem 3:** In real scenarios, each network service provider may allow to not only choose cloudlets but also VMs in chosen cloudlets to cache their services. For example, some network service providers may have some idle VMs to be shared with others to reduce their costs. In this way, cached services in the same VM will share their usage cost. Specifically, the usage of VMs is counted by time slots, and each cached service instance can specify its starting time and duration of using the VM. Let $f_{mi}$ be a VM $m$ in cloudlet $CL_i$. The cached services at the same time slot share the cost of using VM $f_{mi}$ of cloudlet $CL_i$ at that time slot. *The cost- and delay-sensitive service caching problem with temporal VM sharing* is to form a collection of stable coalitions of network service providers by caching instances of their services into VMs of cloudlets in a finite time horizon, with an objective to minimize the social cost of all network service providers, subject to the capacity of the coalition of each cloudlet.

For **Problem 2** and **Problem 3**, we aim to design a mechanism with good performance. We define the *Strong Price of Anarchy* (SPoA) as the ratio of the social cost of a stable coalition structure to a social optimum over any feasible cost. We thus have

$$SPoA = c^{coll}(C)/c(C^*), \tag{11}$$

where $c(C^*)$ is the optimal social cost in a social optimum solution. In contrast to the traditional price of anarchy (PoA), SPoA quantifies the loss incurred due to the selfishness of network service providers and lack of coordinations among network service providers, which isolates the loss originated due to selfishness from that obtained due to lack of coordination.

For clarity, the symbols used in this paper are summarized in Table 1.

# 4 Exact and Approximation Algorithms for the Cost- and Delay-Sensitive Service Caching Problem without VM Sharing

In this section, we provide exact and approximate solutions to the cost- and delay-sensitive service caching problem without VM sharing.

## 4.1 Exact Solution

Let $x_{li}$ be a binary variable that indicates whether service $S_l$ of network service provider $sp_l$ is cached in cloudlet $CL_i$. The problem can be formulated as an ILP as follows.

$$\textbf{ILP}: \quad \min \sum_{l=1}^{L} \sum_{i=1}^{|\mathcal{CL}(S_l)|} x_{li} \cdot c_{l,i}, \tag{12}$$

subject to

$$\sum_{i=1}^{|\mathcal{CL}(S_l)|} x_{li} = 1, \qquad \forall S_l \in \mathcal{S} \tag{13}$$

$$\sum_{l=1}^{L} x_{li} \cdot C_{l,i}^{vm} \leq C(CL_i), \qquad \forall CL_i \in \mathcal{CL}(S_l) \tag{14}$$

$$\sum_{l=1}^{L} x_{li} \cdot B_{l,i}^{vm} \leq B(CL_i), \qquad \forall CL_i \in \mathcal{CL}(S_l) \tag{15}$$

$$x_{li} \cdot d_{l,i}^{dft} \leq d_l^{req}, \qquad \forall S_l \text{ and } \forall CL_i \in \mathcal{CL}(S_l) \tag{16}$$

$$x_{li} \in \{0, 1\}, \tag{17}$$

where Constraint (13) says that each service $S_l$ has to be cached into a cloudlet $CL_i$. Given the fact that the computing resource of cloudlets is limited, we consider that each service $S_l$ can only be cached into a single cloudlet and cannot be 'split' into multiple cloudlets. This however can be easily be extended if multiple service instances can be cached for each service in the MEC network. Constraint (14) ensures that the computing resource capacity of each $CL_i$ is not violated, while Constraint (15) ensures that the bandwidth resource capacity of each cloudlet $CL_i$ is met. Constraint (16) guarantees that the delay requirement of each network service provider is not violated.

## 4.2 Randomized Algorithm

We now describe the algorithm. We first relax Constraint (17) into

$$0 \leq x_{li} \leq 1. \tag{18}$$

Then, the **ILP** is relaxed into an **LP** with the objective shown in (12), subject to Constraints (13), (14), (15), (16), and (18).

The optimal solution to the **LP** can be obtained in polynomial time [4], which however may not be feasible to the original problem. To develop a feasible solution, we need to round the fractional solution to an integer solution, by utilizing a randomized rounding technique [24], [28]. For each service $S_l$, we use $X_{li}$ to denote an i.i.d. event that service $S_l$ is assigned to cloudlet $CL_i$ with probability $\frac{1}{2}x_{li}$. The detailed algorithm is given in **Algorithm** 1, which is referred to as `ApproRR`.

## 4.3 Algorithm Analysis

We now analyze the solution feasibility and performance.

***Lemma 1.*** Assuming that $C(CL_i) \geq 12 \ln |\mathcal{CL}|$ and $B(CL_i) \geq 12 \ln |\mathcal{CL}|$, the obtained solution by **Algorithm** 1 is a feasible solution with the computing and

TABLE 1
Symbols

| Symbols | Meaning |
|---|---|
| $G = (\mathcal{CL} \cup \mathcal{DC}, E)$ | an MEC network with a set $\mathcal{CL}$ of cloudlets and a set $\mathcal{DC}$ of remote data centers. |
| $CL_i$, $E$, and $e$ | a cloudlet in $\mathcal{CL}$, a set of links that interconnect cloudlets and data centers in $\mathcal{CL} \cup \mathcal{DC}$, and a link in $E$. |
| $C(CL_i)$ and $B(CL_i)$ | the computing and bandwidth resource capacities of each cloudlet $CL_i$. |
| $L$ and $sp_l$ | the number of network service providers in the MEC network and a network service provider $sp_l$ with $1 \leq l \leq L$. |
| $\rho_l$ | the request rate of service $S_l$ in terms of the number of requests arrived per time unit. |
| $S_l$ and $\mathcal{CL}(S_l)$ | the service of network service provider $sp_l$ and a set of cloudlets that are preferred by service $S_l$. |
| $c^p_{l,i}$ and $c^b_{l,i}$ | the costs of using a unit of computing resource and bandwidth resource in $CL_i$ by service $S_l$. |
| $C^{vm}_{l,i}$ and $B^{vm}_{l,i}$ | the amounts of computing resource and bandwidth resource allocated to a VM for service $S_l$ in cloudlet $CL_i$. |
| $c_{l,i}$ | the default cost of caching an instance of service $S_l$ in a VM of cloudlet $CL_i$. |
| $d^{DC}_l$ and $d_{l,i}$ | the average delays experienced by users of service $S_l$ when its requests are served in its original service instance in a data center and a cloudlet, respectively. |
| $\mu_i$ | the request processing rate by services in cloudlet $CL_i$. |
| $d^{dft}_{l,i}$ and $d^{col}_{l,i}$ | the default delay and the collaboration delay of processing the requests of $sp_l$ by its service. |
| $d^{req}_l$ | the delay requirement of each network service provider. |
| $u^{dft}_l$ | the default utility of network service provider $sp_l$ with a default cost. |
| $v_l$ | a private value of network service provider $sp_l$. |
| $g_i$ and $K_i$ | a coalition in cloudlet $CL_i$ and the maximum number of network service providers that can stay in coalition $g_i$. |
| $p_l(g_i)$ | the payment that its network service provider $sp_l$ has to pay by staying in coalition $g_i$. |
| $u^{coll}_l(g_i)$ | the collaboration utility in coalition $g_i$ of cloudlet $CL_i$. |
| $u_l(g_i)$ | the utility obtained by network service provider $sp_l$ by staying in coalition $g_i$. |
| $c^{coll}_l(g_i)$ | collaboration cost. |
| $C$ | the collection of stable coalitions. |
| $c^{coll}(C)$ and $c(C^*)$ | the social cost and the optimal cost in a social optimum solution. |
| $f_{mi}$ | a VM $m$ in cloudlet $CL_i$. |
| $x_{li}$ | a binary variable that indicates whether service $S_l$ of network service provider $sp_l$ is cached in cloudlet $CL_i$. |
| $Z_{li}$ | a caching decision for each network service provider. |
| $X_{li}$ and $c(X_{li})$ | an i.i.d. event that service $S_l$ is assigned to cloudlet $CL_i$ and the cost of associated with this event. |
| $\sigma$ | a constant with $\sigma$>0. |
| $c(g_i)$ | the cost of using the resources in $CL_i$. |
| $\mathcal{G} = (N_k, E_k)$ | a directed graph with nodes in $N_k$ denoting the coalitions and edges in $E_k$ denoting the preference. |
| $\mathcal{C}$ | a maximal subset of sinks in directed graph $\mathcal{G}$. |
| $\mathcal{SP}$ | the set of network service providers that are covered by the coalitions in $\mathcal{C}$. |
| $H_1, \cdots, H_{K_i}$ | a collection of subsets, such that each $H_s = \{sp_{l_s}, \cdots, sp_{l_{K_i}}\}$. |
| $c_{ls}(H_1)$ | the cost of a network service provider that does not collaborate with anyone else. |
| $i(H_s)$ | the index of cloudlet that hosts coalition $H_s$. |
| $C^* = \{g^*_1, \cdots, g^*_h\}$ | the optimal coalition structure that has a social optimum welfare. |
| $d_{max}$ | the maximum difference of delays of caching the service of $sp_l$ in any two cloudlets. |
| $\hat{s}$ | the smallest integer such that $\sum_{t=\hat{s}+1}^{K_i} c_{l,i(H_t)} \leq 1$. |
| $t_s$ and $t_e$ | the start and end time slots that network service provider $sp_l$ caches its service $S_l$ to VM $f_{mi}$ of $CL_i$. |
| $S^t_{mi}$ | the set of services cached in VM $f_{mi}$ in time slot $\tau$. |
| $S_i$ and $X(S_i)$ | a subset of network service providers in coalition $g_i$ and the cost of VMs that are solely used by the network service providers in $S_i$. |
| $n_{s,l}$ | the number of switches of coalitions of network service provider $sp_l$. |
| $\vartheta$ | the threshold on the number $n_{s,l}$ of switches of coalitions of a network service provider. |
| $\beta_l$ | the discount factor for network service provider $sp_l$. |

**Algorithm 1** ApproRR

**Input:** $G = (\mathcal{CL} \cup \mathcal{DC}, E)$, a set of network service providers, each network service provider $sp_l$ needs to cache its service $S_l$ in $G$.
**Output:** A caching decision for each network service provider.
1: Relax Constraint (17) of **ILP** into Constraint (18) and obtain an **LP**;
2: Obtain a fraction solution $x$ by solving the **LP**;
3: **for** each service $S_l$ **do**
4: Choose a single cloudlet $CL_i$ for service $S_l$ by setting $X_{li} = 1$ with probability $\frac{1}{2}x_{li}$, no cloudlet is chosen with probability $1 - \frac{1}{2}x_{li}$;
5: **if** all $X_{li}$ defines a feasible solution **then**
6: $Z_{li} \leftarrow X_{li}$;
7: **else**
8: $Z_{li} \leftarrow 0$ for all $l$ and $i$;
9: **return** $Z_{li}$;

bandwidth resource capacities $C(CL_i)$ of each cloudlet being violated with probability of $1/|\mathcal{CL}|^2$.

Please see Appendix for the detailed proof.

***Lemma 2.*** Assuming that $d^{req}_l \geq 12 \ln d^{req}_{min}$, the obtained solution by **Algorithm** 1 is a feasible solution with the delay requirement $d^{req}_l$ of each network service provider being violated with probability of $\frac{1}{(d^{req}_{min})^2}$, where $d^{req}_{min} = \min_{1 \leq l \leq L}\{d^{req}_l\}$.

Please see Appendix for the detailed proof.

***Theorem 1.*** Assuming that $C(CL_i) \geq 12 \ln |\mathcal{CL}|$ and $L \geq \frac{24 \ln |\mathcal{CL}|}{c_{min}}$, the approximation ratio of **Algorithm** 1 is within twice the optimal solution with a high probability of $(1 - \frac{1}{|\mathcal{CL}|^2})$, where $c_{min} = \arg\min_{i,l} c_{l,i}$, and $L$ is the number of network service providers.

Please see Appendix for the detailed proof.

# 5 Distributed Coalition Formation Game for the Cost- and Delay-Sensitive Service Caching Problem with VM Sharing

In this section, we propose an efficient distributed coalition formation mechanism for the cost- and delay-sensitive service caching problem with VM sharing.

## 5.1 Overview and Pricing Strategy

Network service providers whose VMs are in the same cloudlet may collaborate with each other to boost their utilities, through reducing the delays of their requests (a major component of their utilities) and the cost of resource usages. To this end, each network service provider needs to find the right coalition to join. Since each network service provider has its own preference (in terms of values) regarding to a cloudlet that caches its services, we consider distributed coalition formation by allowing each network service provider to make its own decision based on its own value. Each network service provider makes decisions solely on whether its utility will be worse-off. If yes, the network service provider chooses to not join a coalition, while a coalition can also decide whether a network service provider is allowed to join.

To ensure that none of the network service providers in each coalition has an incentive to deviate from its current coalition, the pricing method needs to ensure the existence of a stable coalition structure. Specifically, we assume that the payment of network service provider $sp_l$ due to resource usage in a cloudlet is proportional to its default cost (defined in Eq. (3)). Recall that $p_l(g_i)$ is the payment of network service provider $sp_l$ if it stays in coalition $g_i$ of cloudlet $CL_i \in \mathcal{CL}(S_l)$, then

$$p_l(g_i) = \frac{c_{l,i}}{\sum_{sp_{l'} \in g_i} c_{l',i}} c(g_i), \tag{19}$$

where $c(g_i)$ is the cost of using the resources in $CL_i$, i.e.,

$$c(g_i) = c^p_{l,i} \cdot C(CL_i) + c^b_{l,i} \cdot B(CL_i). \tag{20}$$

## 5.2 A Distributed Coalition Formation Mechanism

The basic idea of the proposed algorithm follows the deferred acceptance algorithm for stable matching [35]. The algorithm forms a stable coalition structure for cloudlets iteratively. Initially, there are $|\mathcal{CL}|$ coalitions with a coalition $g_i$ for each cloudlet $CL_i \in \mathcal{CL}$. Each network service provider identifies the most preferable coalition (i.e., cloudlet) from set $\mathcal{CL}(S_l)$, and sends a 'joining request' to the coalition. Each network service provider selects a coalition $g_i$ that could achieve the highest utility by collaborating with other network service providers in $g_i$. Recall that the delay experienced by the requests of a network service provider is determined by the status of the queue in its coalition. We assume that the request arrival rate and processing rate of each queue are known.

Each coalition $g_i$ responses the joining request of each network service provider. The coalition $g_i$ in $CL_i$ thus checks whether its capacity $K_i$ is violated by admitting the network service provider. If not, the coalition $g_i$ will select one network service provider that could achieve the minimum social cost of its members. The selected network service provider will be considered as 'pre-assigned' to coalition $g_i$. In subsequent iterations, the network service providers that are not pre-assigned to any coalition will keep sending joining requests to the coalitions that they have not been considered yet. The agent of each coalition may break up with its assigned network service providers, if there is a better choice. The coalition joining procedure continues until all network service providers that are pre-assigned to coalitions. The network service providers of each coalition $g_i$ then pays $p_l(g_i)$ (Eq. (19)) for staying in coalition $g_i$. Its service $S_l$ is then cached in cloudlet $CL_i$. The detailed mechanism is given in **Algorithm** 2, referred to as `Coalition`. An illustrative example of the proposed algorithm is shown in Fig. 2.

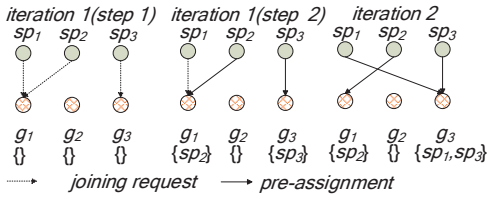



Fig. 2. An example of algorithm `Coalition` with $g_1$, $g_2$, and $g_3$ three coalitions and three network service providers, where the capacities of $g_1$, $g_2$, and $g_3$ are 1, 2, and 2, respectively.

**Algorithm 2** `Coalition`

**Input:** $G = (\mathcal{CL} \cup \mathcal{DC}, E)$, a set of network service providers, each network service provider $sp_l$ needs to cache its service $S_l$ in a cloudlet.

**Output:** A set of coalitions of the network service providers where each coalition has a set of network service providers sharing the resources in a cloudlet.

1: **while** there is a network service provider that is not pre-assigned to any coalition **do**
2: Each $sp_l$ sends 'joining request' to the coalition that is not considered before and could achieve the highest revenue;
3: Each coalition $g_i$ considers the joining requests sent to itself, and select a network service provider that could achieve the minimum social cost without violating the capacity of $g_i$ and jeopardizing the revenue of other members in the coalition;
4: $g_i$ may break up with another network service provider that is already in $g_i$ and choose the current network service provider that sends the joining request, if a lower social cost can be achieved;
5: The network service providers that are 'pre-assigned' to each coalition $g_i$ pay a payment of $p_l(g_i)$ for staying in the coalition;

## 5.3 Mechanism Analysis

We analyze the economic properties and SPoA of the proposed algorithm `Coalition`.

***Lemma 3.*** Assuming that the delay and cost of each network service provider are independent of the location of its cached instance in the MEC network, there exists a stable coalition structure where no network service provider can promote its own revenue by deviating from the current coalition structure.

Please see Appendix for the detailed proof.

***Theorem 2.*** Given an MEC network $G = (\mathcal{CL} \cup \mathcal{DC}, E)$, a set of network service providers, each network service provider $sp_l$ needs to cache its service $S_l$ in a cloudlet, there exists a distributed mechanism, i.e., algorithm `Coalition`, and its SPoA is $O(K \log_{10} K)$ with $K = \max_{g_i} K_i$.

Please see Appendix for the detailed proof.

# 6 Coalition Formation Game for the Cost- and Delay-Sensitive Service Caching Problem with Temporal VM Sharing

In the following we deal with the cost- and delay-sensitive service caching problem with temporal VM sharing among multiple network service providers.

## 6.1 Basic Idea

The basic idea of the proposed algorithm is similar to the one for the cost- and delay-sensitive service caching problem with VM sharing. Specifically, we design a pricing strategy to ensure that network service providers make their own decisions, and we hope to obtain a stable coalition structure in the end. Network service providers are allowed to share each VM in a finite time horizon. The services that are cached in VM $f_{mi}$ of cloudlet $CL_i$ can use its computing and bandwidth resources in different time slots. We thus consider a usage based pricing strategy. Let $t_s$ and $t_e$ be the start and end time slots that network service provider $sp_l$ caches its service $S_l$ to VM $f_{mi}$ of $CL_i$.

The payment of network service provider $sp_l$ is determined by both the duration of its service $S_l$ and the number of other network service providers sharing the costs at their time slot $\tau$. Let $\mathcal{S}_{mi}^{\tau}$ be the set of services cached in VM $f_{mi}$ in time slot $\tau$. The payment of network service provider $sp_l$ is

$$p_l(g_i) = \sum_{\tau'=1}^{t_e-t_s+1} (c(g_i)/|\mathcal{S}_{mi}^{\tau'}|). \quad (21)$$

## 6.2 Mechanism

The adopted mechanism is similar to the one in the proposed **Algorithm** 2. The only difference is that each network service provider can obtain the information of VM availability in each cloudlet. This enables each network service provider to make decision of forming coalitions by a fine-grained evaluation of the cost of using VMs. To this end, the agent of each coalition $g_i$ needs to find a pre-allocation of services to VMs of a cloudlet $CL_i$ so that the cost of such 'pre-assignment' to $CL_i$ can be obtained. Recall that the coalition $g_i$ of each cloudlet $CL_i$ has a capacity in terms of the maximum number of network service providers that can join. Each agent of $g_i$ only needs to greedily find a VM that achieves the minimum cost, and 'pre-allocate' the service to the found VM. All the rest procedures are the same as **Algorithm** 2, as shown in **Algorithm** 3 that is referred to as `CoalitionVMS`. Fig. 3 shows an example of the difference between **Algorithm** 2 and **Algorithm** 3.

The rest is to analyze the performance ratio of **Algorithm** 3. The proof for the existence of a stable coalition structure is similar to that of **Algorithm** 2 as shown in Lemma 3. We thus only derive the lower bound of the SPoA of **Algorithm** 3.

***Theorem 3.*** Given an MEC network $G = (\mathcal{CL} \cup \mathcal{DC}, E)$, a set of network service providers, each network service provider $sp_l$ needs to cache its service $S_l$ in a cloudlet, the SPoA of the proposed mechanism in algorithm 3 is $\Omega(K')$ with $K' = \min_{g_i} K_i$

Please see Appendix for the proof.

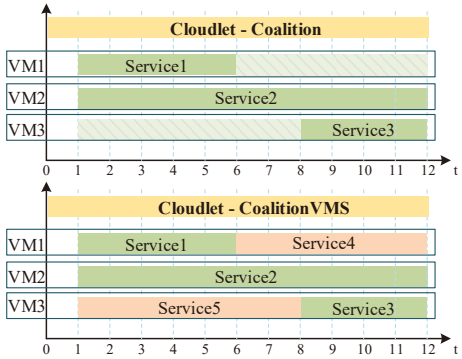



Fig. 3. An example of the difference between **Algorithm** 2 and **Algorithm** 3.

**Algorithm 3** `CoalitionVMS`

**Input:** $G = (\mathcal{CL} \cup \mathcal{DC}, E)$, a set of network service providers, each network service provider $sp_l$ needs to cache its service $S_l$ in a cloudlet.

**Output:** A set of coalitions of the network service providers where each coalition has a set of network service providers sharing the resources in a cloudlet.

1: **while** If there is a network service provider that is not pre-assigned to any coalition **do**
2: Each network service provider $sp_l$ sends 'joining request' to the coalition that is not considered before and could achieve the highest revenue;
3: **for** each cloudlet $CL_i$ **do**
4: The agent of its coalition $g_i$ greedily finds a VM in cloudlet $CL_i$ that achieves the minimum cost for each of the joining requests;
5: The agent of $g_i$ selects a network service provider that could achieve the minimum social cost without jeopardizing the revenue of other network service providers in the coalition;
6: The agent of $g_i$ may break up with another network service provider that is already in $g_i$ and choose the current network service provider that sends the joining request, if a lower social cost can be achieved;
7: The network service providers that are 'pre-assigned' to each coalition $g_i$ pay a payment of $p_l(g_i)$ for staying in the coalition;

# 7 Discussions

The rest is to address how to incorporate service security and privacy, and network dynamics.

## 7.1 Network Dynamics and Uncertainty

We show how to extend the proposed algorithms by incorporating network dynamics and uncertainty. The proposed algorithms for coalition formation have an inherent feature of supporting network dynamics and uncertainty, which can be seen that the coalition formation procedure itself is a dynamic process, and network service providers can make decisions dynamically according to their current situations, such as user mobility or network uncertainties.

A user request of each service may change its resource demand dynamically. The network service provider needs to respond to the dynamics so that the delay experienced by its users is minimized. The proposed algorithms can be easily extended to consider such a case. Specifically, each network service provider has a private valuation $v_l$, representing the utility it can obtain by reducing a unit delay experienced by requests of $S_l$. Network dynamics and uncertainty impact the valuation of each network service provider $sp_l$. For example, if a mobile user requesting service $S_l$ move across a wide geographical area with a high rate, its valuation usually is smaller. The rationale behind is that the caching of a

service may not be able to trade-off the cost incurred by the user movement. In addition, in some cases, the users of a service may suddenly move from places to places. The current coalition of the service may not be an ideal coalition for the service in the future. The network service provider then has the incentive to deviate from its current coalition.

To consider user mobility and network dynamics, we can simply allow the network service provider to adjust its private valuation due to service caching and to deviate its current coalition, once dynamics happen. This however may make the service market unstable if network service providers keep changing their private valuations frequently. To avoid this, we keep track of the number of switches to other coalitions of each network service provider. If the number of switches exceeds a given threshold, we assign the network service provider a discount on its price $p_l(g_i)$ to incentive it staying in current coalition $g_i$. Denote by $n_{s,l}$ the number of switches of coalitions of network service provider $sp_l$ and $\vartheta$ the threshold. The discount factor for network service provider is denoted by $\beta_l$. If $n_{s,l} \geq \vartheta$ in some time point, we set the price that each network service provider needs to pay for its staying in coalition $g_i$ to $\beta_l \cdot p_l(g_i)$. All the rest is the same as algorithms `Coalition` and `CoalitionVMS`. For the sake of clarity, we refer to this revised algorithm as algorithm `CoalitionDYM`.

***Theorem 4.*** Algorithm `CoalitionDYM` delivers a feasible solution to the cost- and delay-sensitive service caching with VM sharing, user mobility and network dynamics.

Please see appendix for the proof.

### 7.2 Security Requirements and Different Service Types

This paper aims to reduce the delay experienced by services that are originally deployed in remote data centers. Indeed, the security and privacy concerns of VM sharing is an important issue. There however have many existing studies to ensure the security and privacy of VM sharing among different service instances [11], [46], [51]. We thus consider that the design of efficient security and privacy protocols is out the scope of this paper. However, the proposed mechanisms can be easily extended. Specifically, we can maintain a set of trusted network service providers for each network service provider. Also, for each VM, we maintain a set of network service providers that used the VM. To consider security and privacy of services, each network service provider only needs to select a coalition that does not have untrusted providers and the VMs in the coalition are shared by the trusted providers.

The proposed algorithms can also be extended to consider different types of services with different request rates and resource demand. We can assume that the services are implemented in different VMs with various processing rates. To extend the proposed methods, we create a number of *virtual cloudlets* with each virtual cloudlet having a single type of VMs to cache one type of service. The proposed queuing model and algorithm still apply for this scenario. Specifically, we maintain a queue in each virtual cloudlet, and there is a coalition of each virtual cloudlet.

## 8 Implementations

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

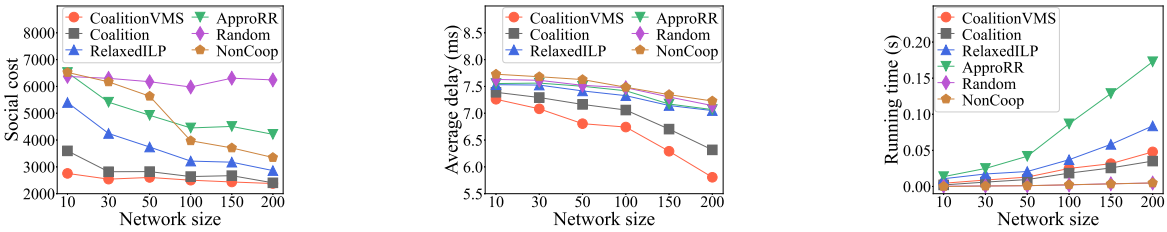
### 8.1 Parameter Settings

We consider an MEC network with network size varying from 10 to 200 nodes, where each network topology is generated using GT-ITM [13]. Each cloudlet has a computing capacity in the range 8,000 to 16,000 $MHz$ [44]. The bandwidth capacity of each cloudlet varies between 100 $Mbps$ and 1,000 $Mbps$ [25]. The computing capacity of each VM is randomly drawn from [1,000, 4,000] $MHz$ [44]. The bandwidth capacity of each VM is drawn from the range of [10,100] $Mbps$ [25]. The costs of using a unit amount of computing resource and bandwidth resource in a cloudlet are set within [$0.04, $0.08] and [$0.02, $0.06], respectively [20]. The request rate of each service is randomly withdrawn from the range of [5,10] requests per second, and the request processing rate by the services of a cloudlet varies from range [11,20] requests per second [8]. The average delay experienced in a data center varies from 30 to 100 milliseconds, and the delay requirement of each network service provider is a random value between 10 and 50 milliseconds [39]. Unless otherwise specified, we will adopt these default settings in our experiments. Each value in the figures is the mean of the results by applying each algorithm on 80 different network topologies.

We evaluate the proposed algorithms against two benchmark algorithms: (1) a non-cooperative mechanism (`NonCoop`): each network service provider does not cooperate with others. It greedily selects a cloudlet that only maximizes its own utility; (2) `Random`, that the network service provider randomly selects cloudlets.

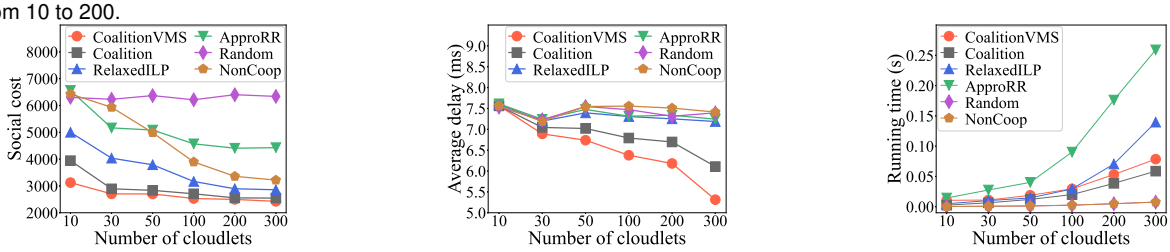
### 8.2 Performance Evaluation

We first evaluate the performance of the relaxed `ILP` referred to as `RelaxedILP`, algorithms `ApproRR`, `Coalition`, `CoalitionVMS` against `NonCoop` and `Random`, by varying the network size from 10 to 200, while fixing the ratio of the number of cloudlets and the number of switches to 0.7 and 0.3. We also set the number of network service providers to 100. The results can be seen in Fig. 4. From Fig. 4 (a) it can be seen that algorithm `Coalition` achieves a much lower social cost than algorithms `RelaxedILP`, `ApproRR`, `NonCoop` and `Random` when the network size varies from 10 to 200. The reason is that algorithm adopts an efficient cost sharing mechanism that allows multiple network service providers to share the resource in a cloudlet; instead algorithms `RelaxedILP`, `ApproRR`, `NonCoop` and `Random` do not allow resource sharing. Furthermore, we can see that algorithm `CoalitionVMS` achieves the lowest social cost, since algorithm `CoalitionVMS` allows a finer-grained sharing of resources of VMs of each cloudlet, which is ignored by algorithm `Coalition`. Specifically, we can see that when the network size increases, the social cost decreases, because the algorithms have more choices to cache services, thereby reducing costs. From Fig. 4 (b), we can see that with the growth of network size, the average delays of algorithms `Coalition` and `CoalitionVMS` decrease. This is due to that algorithm `Coalition` allows resource sharing



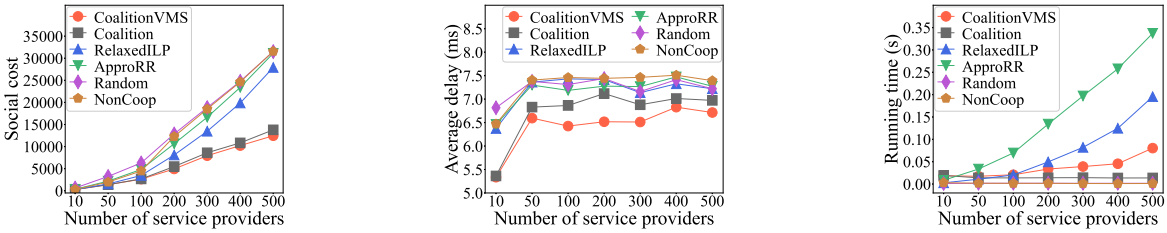

(a) Social cost. (b) Average delay. (c) Running time.

Fig. 4. The performance of algorithms `Coalition`, `CoalitionVMS`, `Non Coop`, `RelaxedILP`, `ApproRR`, and `Random` in networks with sizes varying from 10 to 200.



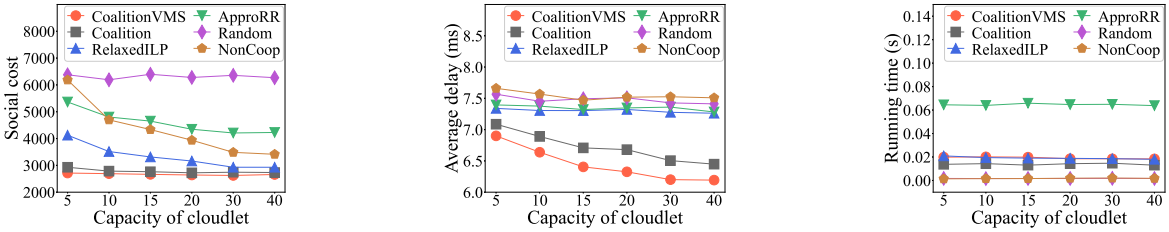

(a) Social cost. (b) Average delay. (c) Running time.

Fig. 5. The impact of the number of cloudlets on the performance of algorithms `Coalition`, `CoalitionVMS`, `NonCoop`, `RelaxedILP`, `ApproRR`, and `Random` in a real network AS1755.




(a) Social cost. (b) Average delay. (c) Running time.

Fig. 6. The impact of the number of network service providers on the performance of algorithms `RelaxedILP`, `ApproRR`, `Coalition`, `CoalitionVMS`, `NonCoop`, and `Random` in a real network AS1755.




(a) Social cost. (b) Average delay. (c) Running time.

Fig. 7. The impact of the capacity $M$ of each cloudlet on the performance of algorithms `RelaxedILP`, `ApproRR`, `Coalition`, `CoalitionVMS`, `NonCoop`, and `Random` in a real network AS1755.

in each coalition with a few network service providers in the same cloudlet. Furthermore, algorithm `CoalitionVMS` makes caching decisions based on the lifetime of services, which allows the idle VMs to be used by other incoming services at a certain moment. The increase in resource utilization in cloudlets allows more user requests to be processed, which greatly reduces service latency. Fig. 4 (c) shows the running time of the algorithms, from which it can be seen that with the growth of the network size, the running times of algorithms `RelaxedILP` and `Appro` increases too. In contrast, algorithms `Coalition` and `CoalitionVMS` only take slightly longer times to obtain feasible solutions than those of algorithms `NonCoop` and `Random`.

We then investigate the impact of the number of cloudlets on the performance of algorithms `RelaxedILP`, `ApproRR`, `Coalition`, `CoalitionVMS`, `NonCoop`, and `Random` in a real network topology AS1755, by varying the number of cloudlets from 10 to 300. From Fig. 5 (a), we can see that algorithm `CoalitionVMS` consistently delivers the lowest social cost compared with that of algorithms `Coalition`, `RelaxedILP`, `ApproRR`, `NonCoop` and `Random`. In addition, the social costs by algorithms `CoalitionVMS` and `Coalition` are decreasing with the growth of the number of cloudlets. This is because a larger number of cloudlets enable both algorithms to have a higher chance of selecting lower cost cloudlets. Also, user requests can be processed in cloudlets with higher processing rates, thereby reducing queuing delays. Similar trends can be found in Fig. 5 (b) for the average delay. However, the large number of cloudlets may take more time to find an optimal solution, which undoubtedly increases the running time of algorithms, which can be seen in Fig. 5 (c).

We now investigate the impact of the number of network service providers on the performance of algorithms `RelaxedILP`, `ApproRR`, `Coalition`, `CoalitionVMS`, `NonCoop`, and `Random` in a real network AS1755, by varying

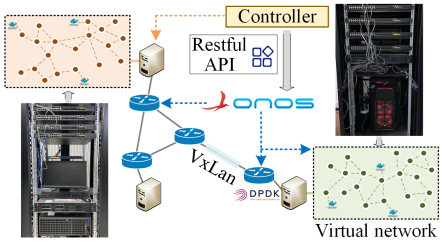
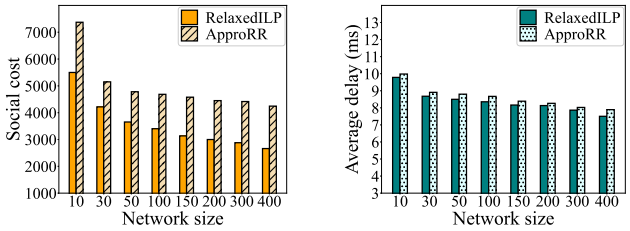
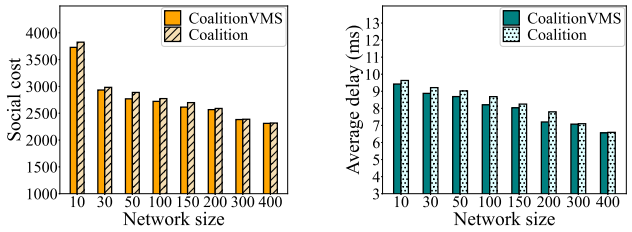



Fig. 8. A test-bed with both physical and virtual networks.



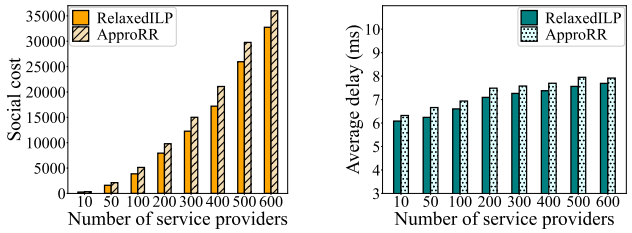

(a) Social cost. (b) Average delay.

Fig. 9. The performance of algorithms RelaxdILP and ApproRR in networks with sizes varying from 10 to 400.




(a) Social cost. (b) Average delay.

Fig. 10. The performance of algorithms Coalition and CoalitionVMS in networks with sizes varying from 10 to 400.




(a) Social cost. (b) Average delay.

Fig. 11. The performance of algorithms RelaxdILP and ApproRR by varying the number of network service providers from 10 to 600.

the number of network service providers from 10 to 500. Results on the social cost, average delay of each user request, and running time of the algorithms are shown in Fig. 6, from which it can be seen that the social costs obtained by algorithms RelaxedILP, ApproRR, Coalition, CoalitionVMS, NonCoop, and Random increase, with the growth of the number of network service providers. The reason behind is that with more network service providers allowing to share resources, the request processing cost also increases. Meanwhile, the queueing delay and running time may be increased to form coalitions, since some of them may be assigned to cloudlets further to their requests. This can also be evidenced in Fig. 6 (b) and Fig. 6 (c).

We finally study the impact of the capacity $M$ of each cloudlet on the performance of algorithms RelaxedILP, ApproRR, Coalition, CoalitionVMS, NonCoop, and Random. As shown in Fig. 7, we can see from Fig. 7 (a) that the social costs of algorithms Coalition and CoalitionVMS are still lower than the algorithms RelaxedILP, ApproRR, NonCoop, and Random. This is because that with the increase on the capacity of a cloudlet, more network service providers can be allowed to cache their services within the proximity of users. Similar trends for the average delay of base stations can be found in Fig. 7 (b). As network service providers form stable coalitions, the network bandwidth cost they consume tends to stabilise. We can also see from Fig. 7 (c) that algorithms Coalition and CoalitionVMS show a steady trend in running time with the growth of the capacity of cloudlets.

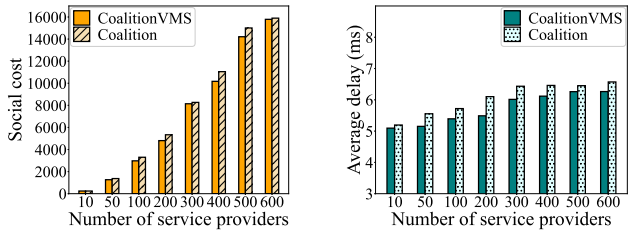
### 8.3 Performance Evaluation in a Test-bed

To evaluate both applicability in real environments and scalability of the proposed algorithms, we study the performance of algorithms Coalition, CoalitionVMS, RelaxedILP, and ApproRR in a test-bed. In the rest, we first describe the settings of our test-bed and then elaborate on the results obtained in the test-bed.

**Test-bed settings:** We build a hybrid testbed with both physical and virtual network elements to achieve unified control and management in a flexible, scalable and real test platform, as shown in Fig. 8. The physical core network consists of five H3C s5560x physical switches [14], with the support for VXLAN for virtual tunnel building and SDN capabilities. The virtual network uses Open vSwitch (OVS) [31] nodes as virtual forwarding devices that are deployed in 5 servers with each having Intel Xeon Gold 5118 dual CPUs and 256GB RAM. We also use DPDK (Data Plane Development Kit) [10] to optimize the transmission performance of OVS and reduce unnecessary computing overhead caused by multiple internal memory copies and excessive memory paging. To integrate the proposed algorithms in the testbed, we assume that the testbed is managed by the ONOS controller, which can control OVS or physical switches to forward data traffic. We realize the joint control of physical and virtual resources by running an agent service that implement the RestAPI in both physical server and virtual container. We use DPDK-based pktgen as a traffic generator to generate the traffic of the requests. The topology constructed by the testbed follows the real topology AS1755, and other settings are the same as the simulation settings.

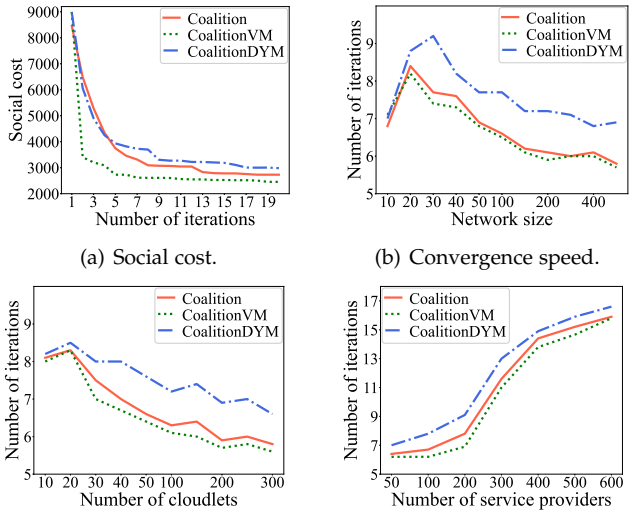
**Results:** To evaluate the effectiveness of resource sharing, we first evaluate the performance of RelaxedILP and ApproRR without resource sharing against that of Coalition and CoalitionVMS with resource sharing by varying the network size from 10 to 400. Fig. 9 and Fig. 10 show the results in terms of the social cost and average delay, from which we can see that the social cost and average delay of the algorithms decrease with the growth of the network size. The reason is that a cache location with the lowest service delay can always be found in a larger network scale, thereby reducing costs. However, as the network size continues to grow, caching locations with lower costs may not be found, because caching services to further locations with lower costs will violate the delay requirements of users. As such, the social cost and average delay tend to stabilize.

We then evaluate the performance of the mentioned four algorithms by varying the number of network service




(a) Social cost. (b) Average delay.

Fig. 12. The performance of algorithms `Coalition` and `CoalitionVMS` by varying the number of network service providers from 10 to 600.




(a) Social cost. (b) Convergence speed.

(c) Convergence speed. (d) Convergence speed.

Fig. 13. The convergence of algorithms `Coalition`, `CoalitionVMS` and `CoalitionDYM`.

providers from 10 to 600. From Fig. 11 and Fig. 12, we can see that the social cost and average delay increase with the growth of the number of network service providers. The rationale behind is that when the number of network service providers increases, more computing and bandwidth resources are needed to cache their services. This increases the cost of leasing computing resource of VMs in cloudlets. Furthermore, the network latency may increase when more network service providers establish coalitions, because more user requests will be assigned to each coalition, resulting in more congestions and high processing delays.

We finally evaluate the convergence of algorithms `Coalition`, `CoalitionVMS` and `CoalitionDYM`. Fig. 13 (a) shows that `CoalitionVMS` and `Coalition` converge to a stable solution very quickly in the fifth and seventh iterations, respectively. Algorithm `CoalitionDYM` requires nine iterations to converge, because the network dynamics and uncertainty affect the stability of the coalition formed by network service providers. Fig. 13 (b), (c) and (d) show the number of convergence iterations, by varying the network size, the number of cloudlets and the number of service providers, respectively. We can see that the number of convergence iterations decreases as the network size and the number of cloudlets increase. The reason is that each network service provider has more opportunities to join a stable coalition without reducing the obtained utility of other members in a coalition. However, as the number of service providers increases, the number of iterations when the algorithms tend to converge increases. This is because the network service providers of each coalition are likely to be allocated a lower utility, due to the addition joining of the increasing number of network service providers, thus leaving the current coalition.

## 9 Conclusion

In this paper, we investigated the problem of service caching with service resource sharing in an MEC network for a service market with multiple network service providers. We formulated three optimization problems. For the cost- and delay-sensitive service caching problem without VM sharing, we proposed an approximation algorithm via randomized rounding. We also analyzed the approximation ratio of the proposed approximation algorithm. For the cost- and delay-sensitive service caching problem with VM sharing, we devised an efficient and stable game-theoretical mechanism and showed its Strong Price of Anarchy (SPoA). We also proposed a mechanism with provable SPoA for the cost- and delay-sensitive service caching problem with temporal VM sharing. We finally evaluated the performance of the proposed algorithms and mechanisms by simulations and implementation in a real test-bed. Experimental results demonstrated that the proposed algorithms are promising, outperforming their existing counterparts by achieving at least 40% lower social cost via service caching and resource sharing among different network service providers.

The future works from this research include (1) proposing an algorithm to handle network dynamics and uncertainty by allowing network service providers to change their private valuations dynamically; and (2) investigating the impacts of the mobility and trajectory of each user on coalition game. We will consider the players of a game that consists of both network service providers and mobile users, and design stable mechanisms with bounded PoA to address the issues.

## Acknowledgement

We appreciate the six 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. The work by Zichuan Xu and Qiufen Xia is funded by the National Natural Science Foundation of China (NSFC) with grant numbers 62172068, 62172071, 61802048, 61802047, and the "Xinghai scholar" program. The work by Weifa Liang was supported by a grant from City University of Hong Kong with project No: 9380137/CS.

## References

[1] Amazon Pricing. https://aws.amazon.com/emr/pricing/, Accessed in 2021.
[2] W. Ali, S. M. Shamsuddin and A. S. Ismail. A survey of web caching and prefetching. *Int. J. Advance. Soft Comput. Appl*, Vol. 3, No. 1, pp.18–44, 2011.
[3] W. Ao and K. Psounis. Distributed caching and small cell cooperation for fast content delivery. *Proc. of MobiHoc*, ACM, 2015.
[4] A. Freville. The multidimensional 0-1 knapsack problem: An overview. *European Journal of Operational Research*, Vol. 155, No. 1, pp. 1–21, Elsevier, 2004.
[5] S. Borst, V. Gupta and A. Walid. Distributed caching algorithms for content distribution networks. *Proc. of INFOCOM*, IEEE, 2010.
[6] C-K Chau and K. Elbassioni. Quantifying inefficiency of fair cost sharing mechanisms for sharing economy. *IEEE Transactions on Control of Network Systems*, Vol. 5, No. 4, pp. 1809–1818, 2018.

[7] L. Chen, C. Shen, P. Zhou and J. Xu. Collaborative service placement for edge computing in dense small cell networks. *IEEE Transactions on Mobile Computing*, Vol. 20, No. 2, pp.377–390, 2021.

[8] X. Chen, Y. Zhang and Y. Chen. Cost-efficient request scheduling and resource provisioning in multiclouds for internet of things. *IEEE IoT Journal*, Vol. 7, No. 3, pp. 1594–1602, 2019.

[9] J. Chen, S. Chen, Q. Wang, B. Cao, G. Feng, and J. Hu. iRAF: A deep reinforcement learning approach for collaborative mobile edge computing IoT networks. *IEEE Internet of Things Journal*, Vol. 6, No. 4, pp.70117024, 2019.

[10] DPDK. https://www.dpdk.org/, Accessed in Oct. 2021.

[11] K. EI Makkaoui, A. Ezzati, A. Beni-Hssane and C. Motamed. Cloud security and privacy model for providing secure cloud services. *Proc. of CloudTech*, IEEE, 2016.

[12] B. Gao, Z. Zhou, F. Liu and F. Xu: Winning at the starting line: Joint network selection and service placement for mobile edge computing. *Proc. of INFOCOM*, IEEE, 2019.

[13] GT-ITM. http://www.cc.gatech.edu/projects/gtitm/, Accessed in 2021.

[14] H3C SDN Switches. 2019. Accessed: Feb. 2020. [Online]. Available: http://www.h3c.com/en/Product_Technology/Enterprise_Products/Switches/Campus_Switches/H3C_S5560X-EI/

[15] T. He, H. Khamfroush, S. Wang, T. La Porta and S. Stein. It's hard to share: joint service placement and request scheduling in edge clouds with sharable and non-sharable resources. *Proc. of ICDCS*, IEEE, 2018.

[16] M. Hoefer, D. Vaz and L. Wagner. Hedonic coalition formation in networks. *Proc. of AAAI*, 2015.

[17] C. K. Huang, S. H. Shen, C. Y. Huang, T. L. Chin and C. A. Shen. S-cache: toward an low latency service caching for edge clouds. *Proc. of the ACM MobiHoc Workshop on Pervasive Systems in the IoT Era*, ACM, 2019.

[18] W. Hu, Y. Gao, K. Ha, J. Wang, B. Amos and Z. Chen. Quantifying the impact of edge computing on mobile applications. *Proc. of APSys*, ACM, 2016.

[19] B. Jedari and M. Francesco. Auction-based cache trading for scalable videos in multi-provider heterogeneous networks. *Proc. of INFOCOM*, IEEE, 2019.

[20] M. Jia, W. Liang, M. Huang, Z. Xu and Y. Ma. Routing cost minimization and throughput maximization of NFV-enabled unicasting in software-defined networks. *IEEE Trans. on Network and Service Management*, Vol. 15, No. 2, pp. 732–745, 2018.

[21] Z. Xu, W. Liang, M. Jia, M. Huang and G. Mao. Task offloading with network function requirements in a mobile edge-cloud network. *IEEE Trans. on Mobile Computing*, Vol. 18, No. 11, pp. 2672–2685, 2018.

[22] M. Jia, W. Liang and Z. Xu. QoS-aware task offloading in distributed cloudlets with virtual network function services. *Proc of MSWiM*, ACM, 2017.

[23] V. Kantere, D. Dash, G. Francois, S. Kyriakopoulou and A. Ailamaki. Optimal service pricing for a cloud cache. *IEEE Trans. on Knowledge and Data Engineering*, Vol. 23, No. 9, pp. 1345–1358, 2011.

[24] T. Kesselheim. Randomized rounding. *Lecture notes for Randomized Algorithms*, ttps://www.mpi-inf.mpg.de/fileadmin/inf/d1/teacing/summer16/random/randomizedrounding.pdf, accessed in Oct. 2021.

[25] S. Knight, H. X. Nguyen, N. Falkner, R. Bowden and M. Roughan. The Internet topology zoo. *IEEE Journal on Selected Areas in Communications*, Vol. 29, No. 9, pp. 1765–1775, 2011.

[26] Y. Liang, J. Ge, S. Zhang, J. Wu, Z. Tang and B. Luo. A utility-based optimization framework for edge service entity caching. *IEEE Trans. on Parallel and Distributed Systems*, Vol. 30, No. 11, pp. 2384–2395, 2019.

[27] S. Misra and N. Saha. Detour: Dynamic task offloading in software-defined fog for IoT applications. *IEEE Journal on Selected Areas in Communications*, Vol. 37, No. 5, pp. 1159–1166, 2019.

[28] M. Mitzenmacher, E. Upfal. Probability and computing: randomized algorithms and probabilistic analysis. Cambridge University Press, 2005.

[29] X. Ma, A. Zhou, S. Zhang and S. Wang. Cooperative service caching and workload scheduling in mobile edge computing. *Proc. of INFOCOM*, IEEE, 2020.

[30] E. Nygren, R. K. Sitaraman and J. Sun. The Akamai network: a platform for high-performance internet applications. *ACM SIGOPS Operating Systems Review*, Vol. 44, No. 3, pp. 2–19, 2010.

[31] Open vSwtich. 2016. [Online]. Available: https://www.openvswitch.org

[32] G. S. Paschos, A. Destounis, L. Vigneri and G. Iosifidis. Learning to cache with no regrets. *Proc. of INFOCOM*, IEEE, 2019.

[33] K. Poularakis, J. Llorca, A. M. Tulino, I. Taylor and L. Tassiulas. Joint service placement and request routing in multi-cell mobile edge computing networks. *Proc. of INFOCOM*, IEEE, 2019.

[34] T. N. Quach, P. Thaichon and C. Jebarajakirthy. Internet service providers' service quality and its effect on customer loyalty of different usage patterns. *Journal of Retailing and Consumer Services*, Vol. 29, pp. 104–113, Elsevier, 2016.

[35] A. E. Roth. Deferred acceptance algorithms: history, theory, practice, and open questions. *International Journal of Game Theory*, Vol. 36, No.3-4, pp. 537–569, Springer, 2008.

[36] S. Scellato, C. Mascolo, M. Musolesi and J. Crowcroft. Track globally, deliver locally: improving content delivery networks by tracking geographic social cascades. *Proc. of WWW*, ACM, 2011.

[37] T. X. Tran, A. Hajisami, P. Pandey and D. Pompili. Collaborative mobile edge computing in 5G networks: New paradigms, scenarios, and challenges. *IEEE Communications Magazine*, Vol. 55, No. 4, pp. 54–61, 2017.

[38] T. X. Tran, K. Chan and D. Pompili. COSTA: Cost-aware service caching and task offloading assignment in mobile-edge computing. *Proc of SECON*, IEEE, 2019.

[39] S. Wang, Y. Guo, N. Zhang, P. Yang, A. Zhou and X. Shen. Delay-aware microservice coordination in mobile edge computing: A reinforcement learning approach. *IEEE Trans. on Mobile Computing*, Vol. 20, No. 3, pp. 939-951, 2019.

[40] Y. Wang, S. He, X. Fan, C. Xu and X. Sun. On cost-driven collaborative data caching: A new model approach. *IEEE Trans. on Parallel and Distributed Systems*, Vol. 30, No. 3, pp. 662–676, 2018.

[41] Q. Xie, Q. Wang, N. Yu, H. Huang and X. Jia. Dynamic service caching in mobile edge networks. *Proc. of MASS*, IEEE, 2018.

[42] Z. Xu, L. Zhou, S. C. K. Chau, W. Liang, Q. Xia and P. Zhou. Collaborate or separate? Distributed service caching in mobile edge clouds. *Proc. of INFOCOM*, IEEE, 2020.

[43] 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. on Mobile Computing*, Vol.18, No. 11, pp. 2672–2685, 2019.

[44] Z. Xu, W. Liang, M. Huang, M. Jia, S. Guo and A. Galis. Efficient NFV-enabled multicasting in SDNs. *IEEE Trans. on Communications*, Vol. 67, No. 3, pp. 2052–2070, 2018.

[45] Z. Xu, Y. Qin, P. Zhou, J. C. S. Lui, W. Liang, Q. Xia, W. Xu and G. Wu. To cache or not to cache: Stable service caching in mobile edge-clouds of a service market. *Proc. of ICDCS*, IEEE, 2020.

[46] X. Xu, X. Zhang, H. Gao, Y. Xue, L. Qi and W. Dou. BeCome: Blockchain-enabled computation offloading for IoT in mobile edge computing. *IEEE Trans. on Industrial Informatics*, Vol. 16, No. 6, pp. 4187–4195, 2019.

[47] J. Xu, L. Chen and P. Zhou. Joint service caching and task offloading for mobile edge computing in dense networks. *Proc. of INFOCOM*, IEEE, 2018.

[48] Z. 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. on Wireless Communications*, Vol. 18, No. 2, pp. 1166–1181, 2019.

[49] J. Yan, S. Bi, L. Duan and Y. J. A. Zhang. Pricing-driven service caching and task offloading in mobile edge computing. *IEEE Trans. on Wireless Communications*, Vol. 20, No. 7, pp. 4495–4512, 2021.

[50] B. Yang, W. K. Chai, Z. Xu, K. Katsaros and G. Pavlou. Cost-efficient NFV-Enabled mobile edge-cloud for low latency mobile applications. *IEEE Trans. on Network and Service Management*, Vol. 15, No. 1, pp. 475–488, 2018.

[51] H. Yang, Y. Liang, J. Yuan, Q. Yao, A. Yu and J. Zhang. Distributed blockchain-based trusted multi-domain collaboration for mobile edge computing in 5g and beyond. *IEEE Trans. on Industrial Informatics*, Vol. 16, No. 11, pp. 7094–7104, 2020.

[52] S. Yu, R. Langar, X. Fu, L. Wang and Z. Han. Computation offloading with data caching enhancement for mobile edge computing. *IEEE Trans. on Vehicular Technology*, Vol. 67, No. 11, pp. 11098-11112, 2018.

[53] S. Yu, X. Chen, Z. Zhou, X. Gong and D. Wu. When deep reinforcement learning meets federated learning: Intelligent multitimescale resource management for multiaccess edge computing in 5G ultradense network. *IEEE Internet of Things Journal*, Vol. 8, No. 4, pp. 2238–2251, 2021.

[54] 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 Journal on Selected Areas in Communications*, Vol. 37, No. 7, pp. 1518–1533, 2019.

[55] G. Zhang, Y. Li and T. Lin. Caching in information centric networking: A survey. *Computer Networks*, Vol. 57, No. 16, pp. 3128–3141, Elsevier, 2013.

[56] M. Zhang, H. Luo and H. Zhang. A survey of caching mechanisms in information-centric networking. *IEEE Communications Surveys & Tutorials*, Vol. 17, No. 3, pp. 1473–1499, 2015.

[57] Q. Zhang, Q. Zhu, M. F. Zhani, R. Boutaba and J. L. Hellerstein. Dynamic service placement in geographically distributed clouds. *IEEE Journal on Selected Areas in Communications*, Vol. 31, No. 12, pp. 762–772, 2013.

[58] 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, Article 23, 2018.

[59] W. Zhang, J. Xiong, L. Gui, B. Liu, M. Qiu and Z. Shi. Distributed caching mechanism for popular services distribution in converged overlay networks. *IEEE Trans. on Broadcasting*, Vol. 66, No. 1, pp. 66–77, 2019.

**Zichuan Xu** (M'17) received his PhD degree from the Australian National University in 2016, ME and BSc degrees from Dalian University of Technology in China in 2011 and 2008, all in Computer Science. From 2016 to 2017, he was a Research Associate at Department of Electronic and Electrical Engineering, University College London, UK. He is currently a full professor and PhD advisor in School of Software at Dalian University of Technology. He is also a "Xinghai Scholar' in Dalian University of Technology. His research interests include mobile edge computing, serverless computing, network function virtualization, Internet of Things, and algorithm design.

**Lizhen Zhou** received her B.E. degree in Computer Science and Technology from Tianjin University, China in 2019. She is currently pursuing her Ph.D. degree in Software Engineering at Dalian University of Technology. Her research interests include mobile edge computing, Internet of Things, and network function virtualization.

**Sid Chi-Kin. Chau** is with the Research School of Computer Science at the Australian National University. He was an Associate Professor with the Department of Computer Science at Masdar Institute, which was created in collaboration with MIT, and is a part of Khalifa University. His research interests are related to the computing systems and applications for sustainable smart cities by applying Internet-of-things, computational intelligence, advanced algorithms and big data analytics. He also researches in broad areas of algorithms, optimization, and Internet-of-things.

**Weifa Liang** (Senior Member, IEEE) received the PhD degree from the Australian National University in 1998, the ME degree from the University of Science and Technology of China in 1989, and the BSc degree from Wuhan University, China in 1984, all in Computer Science. He is a Professor in the Department of Computer Science at City University of Hong Kong. Prior to that, he was a Professor in the Australian National University. His research interests include design and analysis of energy efficient routing protocols for 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 currently serves as an Associate Editor in the Editorial Board of IEEE TRANSACTIONS ON COMMUNICATIONS, and he is a senior member of IEEE.

**Haipeng Dai** received the B.S. degree from Shanghai Jiao Tong University in 2010, and the Ph.D. degree from Nanjing University in 2014. He is an associate professor in the Department of Computer Science and Technology in Nanjing University. His research interests include Internet of Things, mobile computing, and data mining. His research papers have been published in many prestigious conferences and journals such as ACM MobiSys, ACM MobiHoc, ACM VLDB, ACM UbiComp, IEEE INFOCOM, IEEE JSAC, IEEE/ACM TON, IEEE TMC, IEEE TPDS, and etc. He is an IEEE and ACM member. He serves/ed as Poster Chair of the IEEE ICNP'14, Track Chair of the ICCCN'19, TPC member of the ACM MobiHoc'20, IEEE INFOCOM'20, IEEE ICDCS'20, IEEE ICNP'14, IEEE IWQoS'19, IEEE IPDPS'20, IEEE MASS'18-19, IEEE ICC'14-18, IEEE ICCCN'15-18 and IEEE Globecom'14-18. He received Best Paper Award from IEEE ICNP'15, Best Paper Award Runner-up from IEEE SECON'18, and Best Paper Award Candidate from IEEE INFOCOM'17.

**Lixing Chen** is an assistant professor at Shanghai Jiao Tong University. He received the Ph.D. degree in Electrical and Computer Engineering from the University of Miami in 2020, and the BS and ME Degrees from the College of Information and Control Engineering, China University of Petroleum, Qingdao, China, in 2013 and 2016, respectively. His primary research interests include mobile edge computing and machine learning for networks.

**Wenzheng Xu** received the BSc, ME and PhD degrees in computer science from Sun Yat-Sen University, Guangzhou, PR China, in 2008, 2010, and 2015, respectively. He currently is a Special Associate Professor at the Sichuan University and was a visitor at the Australian National University. His research interests include wireless ad hoc and sensor networks, mobile computing, approximation algorithms, combinatorial optimization, online social networks, and graph theory. He is a member of the IEEE.

**Qiufen Xia** received her PhD degree from the Australian National University in 2017, the ME degree and BSc degree from Dalian University of Technology in China in 2012 and 2009, all in Computer Science. She is currently a lecturer at 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.

**Pan Zhou** (S07M14-SM20) is currently a full professor and PhD advisor with Hubei Engineering Research Center on Big Data Security, School of Cyber Science and Engineering, Huazhong University of Science and Technology (HUST), Wuhan, P.R. China. He received his Ph.D. in the School of Electrical and Computer Engineering at the Georgia Institute of Technology (Georgia Tech) in 2011, Atlanta, USA. He received his B.S. degree in the Advanced Class of HUST, and a M.S. degree in the Department of Electronics and Information Engineering from HUST, Wuhan, China, in 2006 and 2008, respectively. He held honorary degree in his bachelor and merit research award of HUST in his master study. He was a senior technical member at Oracle Inc., America, during 2011 to 2013, and worked on Hadoop and distributed storage system for big data analytics at Oracle Cloud Platform. He received the Rising Star in Science and Technology of HUST in 2017. He is currently an associate editor of IEEE Transactions on Network Science and Engineering. His current research interest includes: security and privacy, big data analytics, machine learning, and information networks.