Collaborate or Separate? Distributed Service Caching in Mobile Edge Clouds

Zichuan Xu[†], Lizhen Zhou[†], Sid Chi-Kin Chau[‡], Weifa Liang[‡], Qiufen Xia[§*], and Pan Zhou[†]
[†] School of Software, Dalian University of Technology, Dalian, China.
[‡] Research School of Computer Science, Australian National University, Canberra, ACT 2601, Australia
[§] International School of Information Science & Engineering, Dalian University of Technology, Dalian, China.
[¶] School of Cyber Science and Engineering, Huazhong University of Science and Technology, Wuhan, China.
Emails: z.xu@dlut.edu.cn, zhou_lizhen@mail.dlut.edu.cn sid.chau@anu.edu.au, wliang@cs.anu.edu.au, qiufenxia@dlut.edu.cn, panzhou@hust.edu.cn.
[*]Corresponding author: Qiufen Xia.

***Abstract*—With the development of 5G technology, mobile edge computing is emerging as an enabling technique to promote Quality of Service (QoS) of network services. In particular, the response latency of network services can be significantly reduced by deploying cloudlets at 5G base stations in mobile edge clouds. Network service providers that usually deploy their services in remote clouds now shift their services from the remote clouds to the network edge in the proximity of users. However, the permanent placement of their services into edge clouds may not be economic, since computing and bandwidth resources in edge clouds are limited and relatively expensive. A smart way is to cache the services that are frequently requested by mobile users in edge clouds. In this paper, we study the problem of service caching in mobile edge network under a mobile service market with multiple network service providers completing for both computation and bandwidth resources of the edge cloud. We propose an Integer Linear Program (ILP) and a randomized rounding algorithm, for the problem without resource sharing among the network service providers. We also devise a distributed and stable game-theoretical mechanism for the problem with resource sharing among the network service providers, with the objective to minimize the social cost of all network service providers, by introducing a novel cost sharing model and a coalition formation game. We analyze the performance of the mechanism by showing a good guaranteed gap between the solution obtained and the optimal one, i.e., Strong Price of Anarchy (SPoA). We finally evaluate the performance of our algorithms by extensive simulations, and the obtained results show that the social cost of all players can be reduced significantly via allowing cooperation among network service providers in service caching.**

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

I. Introduction

In the past decade, with the development of cloud technologies, various multimedia applications are attracting much attention of many service and infrastructure providers. For example, various services for Virtual Reality (VR) services have been deployed in data centers for the real-time processing of 8K video data collected from VR headsets. Such services need to consume vast amounts of both computing resource for rendering and bandwidth resources to receive vast input video data from headsets. However, they are facing difficulties in meeting Quality of Services (QoS) requirements of mobile users, due to not only the large volume of data that needs to be processed but also the ever-increasingly congested core networks. The rapid development of mobile edge computing and 5G technology provides a promising solution to this problem, by deploying cloudlets on the side close to users and providing VR services within the proximity of users. The network service providers can cache their services or partial of their services into cloudlets of mobile edge clouds such that the QoS of users is promoted.

In this paper, we consider a fundamental problem of *service caching* that allows multiple network services providers to cache their services from remote data centers to cloudlets in a mobile edge cloud that is operated by an infrastructure provider. Network service providers can lease Virtual Machine (VMs) from the infrastructure provider [29]. Each network service provider has to bear the cost of using the resources of its leased VMs in cloudlets. To reduce its cost, the network service provider may share its leased VMs with other network service providers.

Service caching in the afore-mentioned mobile edge cloud with multiple network service providers faces many challenges: (1) Multiple network service providers in the service market compete for the limited resources of the edge cloud, and each of them only cares about their own revenue. It is thus impossible to centrally coordinate them towards the social optimum via a centralized mechanism. A distributed mechanism thus is needed. Specifically, how to design a distributed service caching mechanism so that each network service provider has an incentive to participate in the market. How to further guarantee that the market is stable, and no players can increase its utility by deviating from its current decision; (2) The selfish behavior of network service providers gives rise to outcomes that deviate from a social optimum. A near-optimal solution with a bounded gap from the social optimum thus is needed. Specifically, how to design a distributed service caching mechanism so that each service provider can obtain non-negative benefits, so that no players can increase its utility by deviating from its current decision, so as to ensure the stability of the mechanism, and (3) To further reduce their cost of service caching, network service providers may choose to

share their computing and bandwidth resources (that are leased from the infrastructure provider of the edge cloud) with other network service providers. Considering the group of network service providers sharing the same resource as a coalition, it is challenging to design efficient cost sharing mechanisms for each coalition that minimizes the system performance degradation due to sharing; Last but not least, (4) the selection of caching locations for 5G services, such as VR applications, can have a significant impact on the latency experienced by users. Naive caching of services can violate the latency requirement of users. How to devise strategic caching mechanisms of such delay-sensitive network services is another challenge.

Although there are studies on content centric networks (CCN), task offloading, and service placement, they are fundamentally different with the service caching problem in mobile edge clouds of this paper. First, the researches on CCN focused on caching contents in nodes with storage capacities while they are requested. Services that process those contents are usually ignored. Second, studies on task offloading and service placement usually assume that the services are only deployed in the mobile edge clouds [13], [17], [41], [31], [32], [33], [34], [36]. This however may not be suitable for service caching from remote data centers to cloudlets of a mobile edge cloud with multiple network service providers.

To the best of our knowledge, we are the first to formulate the problem of service caching from remote clouds to edge clouds in a service market with multiple network service providers. We propose a randomized rounding algorithm with a good approximation ratio with high probability for the problem. We also propose the very first distributed mechanism with a guaranteed gap of the obtained solution to the optimal one.

The main contributions of this paper are as follows.

- We formulate the service caching problem in a mobile edge cloud with and without resource sharing among multiple network service providers.
- We formulate an Integer Linear Program (ILP) solution for the problem without resource sharing, and devise a randomized algorithm with a good approximation ratio while maintaining moderate resource violations.
- We design a novel coalition formation game for the problem with resource sharing, with the 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 evaluate the performance of the proposed algorithm through experimental simulations, and the results show that the performance of the proposed algorithms outperform existing ones.

The remainder of the paper is arranged as follows. Section II summarizes the state-of-the-arts on this topic. Section III introduces the system model, notations and problem formulation. Section IV presents the proposed ILP and a randomized approximation algorithm. A mechanism for the coalition formation game of the delay-sensitive service caching problem is proposed in Section V. Section VI provides some experimental results on the performance of the proposed algorithm, and Section VII concludes the paper.

II. Related Work

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.

For content/data caching, most studies focus on efficient and effective architectures, methods, and algorithms for content centric networks [2], [21], [38], [39], conventional cloud networks [14], or cellular networks [3], [23], [12]. For example, to improve the content caching efficiency, many research efforts have been devoted to optimize the path selection [19], server placement [25] and content duplication strategy [4]. Wang [30] recently studied the problem of data sharing for network services via enabling data caching in a network. The research problems of the aforementioned studies only focused on the placement of contents and data, and the placement of services used to process such data are not considered.

Closely related to the service caching problem of this paper is the research on task offloading and service placement [7], [9], [17], [41], [37], [40]. For task offloading, Misra *et al.* [17] recently 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). Both exact and efficient heuristics are proposed. Zhou *et al.* [41] studied the joint task offloading and scheduling, by considering wireless network connections and mobile device mobility. In addition, the service placement problem is well investigated. For example, He *et al.* [9] investigated the problem of joint service placement and request scheduling in mobile edge computing systems under communication, computation, and storage constraints. Zhang *et al.* [40] studied the problem of service placement with an objective to minimize service hosting costs while ensuring critical performance requirements. Xie *et al.* [31] studied the dynamic service caching problem in mobile edge networks with base stations, and develop an efficient algorithm to improve the performance by utilizing the cooperative features of base stations in mobile edge clouds. They jointly considered the service placement and user association. However, in the above-mentioned studies, only a network service provider exists without considering resource sharing among different services by different network service providers. Most of them do not consider a mobile edge cloud with both cloudlets and data centers, and ignored the data updating between cached services and the original services.

III. Preliminary

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

A. System Model

We consider a mobile edge network $G = (\mathcal{CL} \cup \mathcal{DC}, E)$. It consists of not only cloudlets (in $\mathcal{CL}$) that are deployed

within the users' proximity but also powerful data centers (in $\mathcal{DC}$) with abundant computing resources in remote areas. Denote by CL_i a cloudlet in $\mathcal{CL}$. E is a set of links that interconnect cloudlets and data centers in $\mathcal{CL} \cup \mathcal{DC}$, and let $e \in E$ be a link in E. Each cloudlet $CL_i \in \mathcal{CL}$ has a limited amount of computing resource to implement various network services, such as data processing for Internet-of-Things applications. In addition, according to the mechanisms of most cloud platforms, VMs hosting cached service instances are usually assigned to 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 services that are to be cached in cloudlets. Each network service provider usually has a stable set of loyal users who would not shift to other network service providers in the short term [15], if the overall service quality of the provider is relative stable. The user association thus is out of the scope of this paper. Fig. 1 illustrates the two-tiered cloud network G.

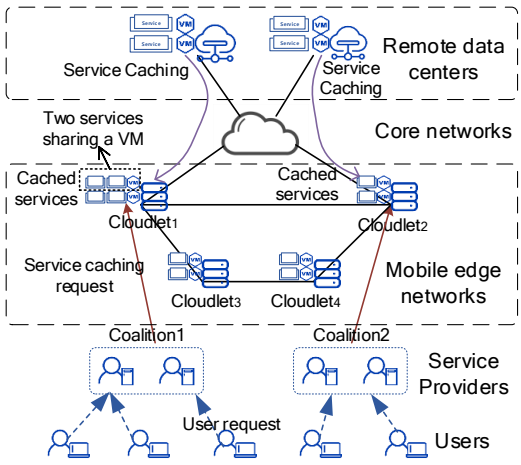


Fig. 1. An example of the two-tiered cloud network.

B. Service Caching for Multiple Network Service Providers

We focus on services that are originally deployed in data centers, such as services for VR applications, which need to be cached into the edge cloud to improve latency. We consider a number of network service providers aiming to cache such services to the cloudlets in $\mathcal{CL}$, with the aim to improve the QoS of their services. Such caching of services from remote data centers to local cloudlets is referred to as *service caching*. Each network service provider is self-interest and cares about its own utility when they are caching services. However, different network service providers may collaborate together to share resources in a single cloudlet and thus share the cost of using the resources, as shown in Fig. 1.

Let sp_l be a network service provider, where $1 \leq l \leq L$. Each network service provider sp_l has a service, denoted by S_l, which has already been deployed in one of the data centers in $\mathcal{DC}$. Service S_l is demanded by a set of users. We thus consider that each service has a set of user requests that needs to be processed by its service instances. If network service provider sp_l caches an instance of S_l into a cloudlet, user requests of S_l will be re-directed to its *cached service instance* in the cloudlet. Otherwise, the *original service instance* of S_l in a data center will continue serving its user requests. The cached instance of S_l in cloudlet CL_i may be destroyed and its occupied resource will be released back to the system. However, the data that is processed by its cached instance may be needed by the service in future. The user data processed by a cached service instance of S_l must be forwarded to its original service instance in a remote cloud.

Each network service provider sp_l usually has a preference of cloudlets to cache its service sp_l, considering that some cloudlets have necessary data or software required by S_l. Let $\mathcal{CL}_l$ be the set of cloudlets that are preferred by service S_l. Only the cloudlets in $\mathcal{CL}_l$ can cache service S_l and such set of cloudlets for services are given as a priori. The scenario when each sp_l has multiple services is discussed in Section VII.

C. Cost Model

Implementing services in cloudlets for network service providers incurs various resource usage costs. Specifically, if a cloudlet caches multiple service instances, the cost will be shared among the cached instances. Ideally, each service instance is implemented in a single VM of a cloudlet. It however may share the VM with other service providers if it does not use the VM at some time. The computing and bandwidth resource usage costs are defined as follows.

The cached service instances in a cloudlet CL_i consumes its computing resource. Denote by $c^p_{l,i}$ be the cost of using a unit of computing resource in CL_i by service S_l, the usage cost of computing resource usage of service S_l in CL_i is $c^p_{l,i} \cdot C^{vm}_i$, where C^{vm}_i is the amount of computing resource that is allocated to a VM in cloudlet $CL_i \in \mathcal{CL}_l$.

Bandwidth resource is needed to transfer data from/to a cloudlet. We consider that the cost of bandwidth resource usage is proportionally shared among the cached instances in CL_i. 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 $c^b_{l,i} \cdot B^{vm}_i$, where B^{vm}_i is the amount of bandwidth resource that is allocated to a VM in cloudlet $CL_i \in \mathcal{CL}_l$.

For clarity, the cost of network service provider sp_l without sharing a VM with others is referred to as the *default cost* of sp_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,

$$c_{l,i} = c^p_{l,i} \cdot C^{vm}_i + c^b_{l,i} \cdot B^{vm}_i. \quad (1)$$

Notice that the default cost is the cost of using a VM by service S_l solely; that is, the VM is not shared with others.

D. Delay and Utility Models

The delays experienced by implementing a user request in a cached service instance and an original service instance of service S_l vary significantly. Denote by d^{DC}_l and $d_{l,i}$ the average delays experienced by users of service S_l if its requests are served in its original service instance in a data center and

a cached service instance in cloudlet CL_i, respectively. For each S_l, d_l^{DC} is usually no less than $d_{l,i}$ for each $CL_i \in \mathcal{CL}$. The values for such delays can be obtained from historical information.

The delay experienced by requesting service S_l determines the utility of network service provider sp_l, since prohibitive long delays may cause cost penalties. We thus consider that 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.

Let u_l be the *default utility* of network service provider sp_l with a default cost, i.e., the VM that caches the service of sp_l without sharing with the other service providers, which can be formulated as

$$u_l^{dft}(CL_i) = (v_l \cdot (d_l^{DC} - d_{l,i}) - c_{l,i}), \tag{2}$$

if service S_l has a cached instance in a VM of cloudlet $CL_i \in \mathcal{CL}_l$. Notice that v_l is a private value of network service provider sp_l that represents the utility it can obtain by promoting a unit delay that is experienced by requests of S_l. 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 core network can be 16 times of cloudlet access latency via LTE [11]. We further consider that the services of each network service provider is delay-sensitive, and the utility obtained via shortening delay is higher than resource usage costs. This means that $u_l^{dft}(CL_i)$ is always positive for any $CL_i \in \mathcal{CL}_l$.

E. Cooperative Game Theory and Coalition Formation

In cooperative games, players may form a group to jointly determine their actions. Such groups of agents are usually referred to as "coalitions". We consider network service providers as players. The network service providers that have their services cached in the same cloudlet is considered as a 'coalition'. By staying in a coalition, they will be allowed to share both computing and bandwidth resources in the cloudlet, by paying a payment. Denote by g_i a coalition in cloudlet CL_i. Specifically, network service providers in coalition g_i will benefit from resource sharing if they have complementary resource demands. However, since both computing and bandwidth resources in each cloudlet are limited, the coalition of the cloudlet thus has limits on the maximum number of services that can be cached in it. We refer each coalition as *a capacitated coalition*. Let K be the capacity of each coalition, specifying the maximum number of service providers that can form a coalition.

Denote by $p_l(g_i)$ the payment that its service provider sp_l has to pay by staying in coalition g_i. The revenue via unconditionally collaborate with other network service providers 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,

$$u_l^{coll}(g_i) = v_l \cdot (d_l^{DC} - d_{l,i}) - p_l(g_i). \tag{3}$$

The *utility* of a service provider sp_l obtained through collaborating with other 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 $CL_{i'}$ ($\in \mathcal{CL}_l$). It however 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 can be given as

$$\begin{aligned} u_l(g_i) &= u_l^{coll}(g_i) - u_l^{dft}(CL_{l'}) \\ &= c_{l,i'} - \Big(p_l(g_i) - v_l \cdot d_{l,i'} + v_l \cdot d_{l,i})\Big). \end{aligned} \tag{4}$$

Considering that each 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 $\Big(p_l(g_i) - v_l \cdot d_{l,i'} + v_l \cdot d_{l,i})\Big)$ of Eq. (4) can be minimized. We refer to such cost as the *collaboration cost*, denoted by $c_l^{coll}(g_i)$, which is defined by

$$c_l^{coll}(g_i) = p_l(g_i) - v_l \cdot d_{l,i'} + v_l \cdot d_{l,i}. \tag{5}$$

F. Problem Definitions

Given a mobile edge 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 thus requires to cache an instance of its service S_l in its preferred set $\mathcal{CL}(S_l)$ of cloudlets. It must be mentioned that the proposed algorithms in this paper can be easily extended to the scenario when multiple instances of S_l can be cached into the edge cloud, as discussed in Section VII.

We consider the following two optimization problems.

Problem 1: *The cost-sensitive service caching problem* without share resource of each VM in a cloudlet is to cache the services of network service providers, such that the total cost of caching services in $\mathcal{S}$ is minimized, while meeting the capacity constraint of each cloudlet. Note that since each VM is not shared, the capacity constraint of each cloudlet is the number of VMs in it. Let M be the capacity of each cloudlet.

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

Problem 2: *The cost- and delay-sensitive service caching problem* is to form a collection of stable coalitions (for resource sharing) of network service providers by caching their service instances into cloudlets in G, such that their social cost is minimized, subject to the capacity constraint of each cloudlet. 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 $c^{coll}(C)$.

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

$$SPoA = c^{coll}(C)/c(C^*), \tag{6}$$

where $c(C^*)$ is the optimal cost in a social optimum solution.

IV. Exact and Approximation Algorithms for the Cost-Sensitive Service Caching Problem without Resource Sharing

We here provide exact and approximate solutions to the cost-sensitive service caching problem without resource sharing.

A. 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 then can be formulated as an ILP as follows:

$$\textbf{ILP}: \quad \min \sum_{l=1}^{L} \sum_{i=1}^{|\mathcal{CL}(S_l)|} x_{li} \cdot c_{l,i}, \tag{7}$$

subject to,

$$\sum_{i=1}^{|\mathcal{CL}(S_l)|} x_{li} = 1, \qquad \forall S_l \in \mathcal{S} \tag{8}$$

$$\sum_{l=1}^{L} x_{li} \leq M, \qquad \forall CL_i \in \mathcal{CL}(S_l) \tag{9}$$

$$x_{li} \in \{0,1\}, \tag{10}$$

where Constraints (8) say that each service S_l has to be cached into a cloudlet CL_i. Constraints (9) ensure the capacity of each CL_i is not violated, i.e., at most M network service providers can be assigned to a single CL_i.

B. Randomized Algorithm

We now describe the algorithm. We first relax Constraint (10) into

$$0 \leq x_{li} \leq 1. \tag{11}$$

Then the **ILP** is relaxed into an **LP** with the objective shown in (7), subject to Constraints (8), (9), and (11).

The optimal solution to the **LP** can be obtained in polynomial time. It however may not be a feasible solution to the original problem due to the fraction value of x_{li}. To make the solution feasible, we need to round the fractional solution to an integer solution, by utilizing a randomized rounding technique. 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. We assign service S_l to cloudlet CL_i with probability $\frac{1}{2}x_{li}$. The detailed algorithm are is given in **Algorithm** 1, which is referred to as `ApproRR`.

Algorithm 1 `ApproRR`

Input: $G = (\mathcal{CL} \cup \mathcal{DC}, E)$, a set of L service providers, each service provider sp_l needs to cache its service S_l in a cloudlet.
Output: A caching decision for each network service provider.
1: Relax Constraint (10) of **ILP** into Constraint (11) 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};

C. Algorithm Analysis

We now analyze the solution feasibility and approximation ratio.

Lemma 1: Assuming that $M \geq 12 \ln |\mathcal{CL}|$, the obtained solution by **Algorithm** 1 is a feasible solution with the capacity M of each cloudlet being violated with probability of $\frac{1}{L^2}$.

Proof Clearly, each service will be cached in a single cloudlet. In the following we show that the capacity of each cloudlet is violated with a small probability. Recall that in algorithm 1, service S_l is assigned to cloudlet CL_i with probability $\frac{1}{2}x_{li}$. This means that $X_{li} = 1$ with probability $\frac{1}{2}x_{li}$ and $X_{li} = 0$ with probability $1 - \frac{1}{2}x_{li}$. Let $X_i = \sum_{l=1}^{L} X_{li}$ and its expectation $E(X_i)$ is

$$E(X_i) = \sum_{l=1}^{L} E(X_{li}). \tag{12}$$

We first bound the expectation of event X_i. Assume that a service $S_{l'}$ is cached to cloudlet $CL_{i'}$. Since all other events are independent, we have for each cloudlet $CL_i \in \mathcal{CL}(S_l)$ that

$$E(X_i \mid X_{i'l'} = 1) = \sum_{l=1}^{L} E(X_{li}) = \sum_{l=1}^{L} \frac{1}{2} x_{li} \leq \frac{M}{2}$$

Denote by $Pr[\cdot]$ the probability of an event. The capacity of each cloudlet CL_i is violated only if after taking out one cached service, the remaining number of services cached in this cloudlet is still at least M. Calculating the probability that the capacity of each cloudlet is violated is to calculate

$$Pr[X_i \geq M \mid X_{i'l'} = 1]. \tag{13}$$

By a Chernoff bound [18] with $\mu = E(X_i)$ and $\delta = 1$, we have

$$\begin{aligned} &Pr[X_i \geq M \mid X_{i'l'} = 1] = Pr[X_i \geq 2E(X_i) \mid X_{i'l'} = 1] \\ &\leq exp\big(-\frac{E(X_i)}{3}\big) \leq exp\big(-\frac{M}{6}\big) \leq exp\big(-\frac{12 \ln |\mathcal{CL}|}{6}\big) \\ &= 1/|\mathcal{CL}|^2. \end{aligned} \tag{14}$$

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

Proof We now show the approximation ratio of the proposed algorithm. Let OPT be the optimal solution of the **ILP**. Let Z be the obtained solution of the approximation algorithm.

Recall that X_{li} is the event that service S_l is cached in cloudlet CL_i. Let $c(X_{li})$ be the cost associated with event X_{li}, and its expectation is

$$E(c(X_{li})) = E(X_{li}) \cdot c_l^{col}(g_i) = (1/2) x_{li} \cdot c_{l,i}. \tag{15}$$

Recall that a solution is feasible only when the capacity of each cloudlet is met, and each service is cached into a single cloudlet. By a union bound of Inequality (14), the probability of an infeasible solution, i.e., $Z_{il} = 0$ for all l and i, is

$$Pr[Z_{il} = 0 \mid X_{i'l'} = 1] < |\mathcal{CL}| \cdot \frac{1}{|\mathcal{CL}|^2} = \frac{1}{|\mathcal{CL}|} \leq \frac{1}{2}. \tag{16}$$

This means that the lower bound of $Pr[Z_{il} = 1 \mid X_{i'l'} = 1]$ is

$$Pr[Z_{il} = 1 \mid X_{i'l'} = 1] \geq \frac{1}{2} Pr[X_{i'l'} = 1] = \frac{x_{i'l'}}{4}. \tag{17}$$

Let Y be the social cost of solution Z, we thus have

$$E(Y) = \sum_{l=1}^{L} \sum_{CL_i \in \mathcal{CL}(S_l)} Pr[Z_{il} = 1 \mid X_{i'l'} = 1] \cdot c_{l,i}. \tag{18}$$

We have

$$E(Y) = \sum_{l=1}^{L} \sum_{CL_i \in \mathcal{CL}(S_l)} Pr[Z_{il} = 1 \mid X_{i'l'} = 1] \cdot c_{l,i}$$
$$\geq \frac{1}{4} \sum_{l=1}^{L} \sum_{CL_i \in \mathcal{CL}(S_l)} x_{i'l'} \cdot c_{l,i} \geq \frac{1}{4} L \cdot c_{min}. \tag{19}$$

To show the approximation ratio of the proposed algorithm with high probability, we instead show that the solution above the approximation ratio with small probability. Specifically, we calculate the following probability

$$Pr[Y \geq (1+\sigma)OPT], \tag{20}$$

where σ is a constant with $\sigma > 0$.

Recall that the fractional solution to **LP** is a lower bound of the optimal solution. We have $OPT > E(Y)$. This means

$$Pr[Y \geq (1+\sigma)OPT] < Pr[Y \geq (1+\sigma)E(Y)], \tag{21}$$

since the property of the upper tail property of the Chernoff bound [18]. Then, apply the Chernoff bound with $\sigma = 1$, we have

$$Pr[Y \geq 2 \cdot OPT] < Pr[Y \geq 2E(Y)]$$
$$< e^{-\frac{1}{3}E(Y)} < e^{-\frac{1}{12}L \cdot c_{min}}, \text{due to lnequality (19)}$$
$$< e^{-\frac{24 \ln |\mathcal{CL}|}{12}} < 1/|\mathcal{CL}|^2. \tag{22}$$

Therefore, the approximation ratio of the proposed algorithm is within 2 with high probability of $(1 - \frac{1}{|\mathcal{CL}|^2})$.

V. Distributed Coalition Formation Game for the Cost- and Delay-Sensitive Service Caching Problem

In the following we propose an efficient distributed mechanism for the delay-sensitive service caching problem, where each network service provider is allowed to share its VM with others when the VM is idle.

A. Overview and Pricing Strategy

Since each network service provider has its own value regarding to the cloudlet that caches its services, it prefers to make its own decisions. We consider a distributed mechanism that allows network service providers to make decisions on which coalitions to join, based on its own value and information. Each network service provider makes decisions solely on whether its revenue will be worse-off. If yes, the network service provider will not join a coalition. The coalition can decide whether a network service provider is allowed to join.

Once the coalitions of all cloudlets are formed, to ensure the stability of the proposed mechanism, i.e., none of the network service providers in each coalition has an incentive to deviate from its current coalition. To this end, we propose a pricing method 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 the cloudlet is proportional to its default cost (defined in Eq. (1)). Recall that $p_l(g_i)$ is the payment of service provider sp_l if it stays in coalition g_i of cloudlet $CL_i \in \mathcal{CL}(S_l)$, then,

$$p_l(g_i) = \frac{c_{l,i}}{\sum_{sp_{l'} \in g_i} c_{l',i}} c(g_i), \tag{23}$$

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

$$c(g_i) = c_i \cdot C(CL_i) + c_i^b \cdot B(CL_i). \tag{24}$$

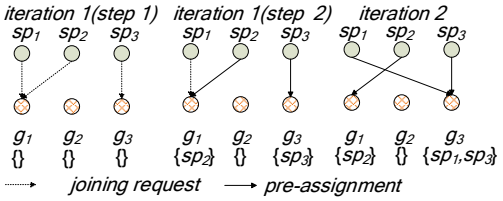


Fig. 2. An example of algorithm `Coalition` with g_1, g_2, and g_3 three coalitions and three network service providers, where the capacities of g_1, g_2, and g_3 are 1, 2, and 2, respectively.

B. Mechanism

The basic idea of our algorithm follows the deferred acceptance algorithm for the stable matching problem [24]. The algorithm iteratively forms a stable coalition structure for cloudlets. Initially, there are $|\mathcal{CL}|$ coalitions with a coalition g_i for each cloudlet $CL_i \in \mathcal{CL}$. Each network service provider then 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 usually selects a coalition g_i that could achieve the highest utility by collaborating with other network service providers in g_i. We further consider that there is an agent of each coalition that speaks for its network service providers. The agent of g_i responses the joining request of each network service provider. Specifically, it first checks whether the capacity K of g_i will be violated by admitting the network service provider. If not, it will select one network service provider that could achieve the minimum social cost. 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 network service providers that were pre-assigned to it in previous iterations, 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. (23)) 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`. A simple example of the algorithm is shown in Fig. 2.

C. Mechanism Analysis

We now analyze the economic properties of the proposed mechanism.

Lemma 2: There exists a stable coalition structure, where no network service provider can promote its own revenue by deviating from the current coalition structure.

Algorithm 2 `Coalition`

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

Output: A set of coalitions of the service providers where each coalition has a set of 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 sp_l sends 'joining request' to the coalition that is not considered before and could achieve the highest revenue;

3: The agent of 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: 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;

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;

Proof To show the existence of a stable coalition structure, we need showing that none of the network service providers prefers the other coalitions instead of the current one to promote its revenue. By contradiction, we define a *cyclic preference* as sequences $(l_1, \cdots, l_s)$ and $g_1, \cdots, g_s$, where $l_k \in g_k \cap g_{k+1}$ for all $k \leq s-1$, and $i_s \in g_s \cap g_1$, such that

$$u_{l_1}(g_1) > u_{l_1}(g_2), u_{l_2}(g_2) > u_{l_2}(g_3), \cdots, u_{l_s}(g_s) > u_{l_s}(g_1).$$

We show that if there is no such a cyclic preference, there always exists a stable coalition structure [5], [10]. Recall that different network service providers have different preferred sets of cloudlets. If every network service provider has a distinct set of preferred set of cloudlets, each network service provider stays in a cloudlet of $\mathcal{CL}(S_l)$ that has the maximum utility and will not deviate from its selection. Otherwise, we show that no cyclic preference exists in the following.

Given a directed graph $\mathcal{G} = (N_K, E_K)$ with nodes in N_K denoting the coalitions and edges in E_K denoting the preference. For two coalitions $g_1, g_2 \in N_K$, we use an edge $\langle g_1, g_2 \rangle$ to denote the fact that there exists a network service provider $sp_l \in g_1 \cup g_2$ such that $u_l(g_1) < u_l(g_2)$. This means that service provider $sp_l \in g_1 \cup g_2$ prefers coalition g_2 instead of coalition g_1, as staying in coalition g_2 leads to a higher utility. Otherwise, if there is no such preference, there will not be an edge from g_1 to g_2 in the directed graph $\mathcal{G}$. We then can identify a cyclic preference by finding a directed cycle in $\mathcal{G}$. This means that $\mathcal{G}$ is acyclic and has at least one sink. In the worst case, a network service provider may consider a sink as its best choice as it maximizes its utility. Let $\mathcal{C}$ be a maximal subsets of sinks in directed graph $\mathcal{G}$ such that any two distinct coalitions g and g' are disjoint. Let $\mathcal{SP}$ be the set of service providers that are covered by the coalitions in $\mathcal{C}$. We then remove all nodes in $\mathcal{G}$ that contains some service provider in $\mathcal{SP}$, and denote by $\mathcal{G}'$ the obtained network.

Let $\mathcal{C}'$ be a coalition structure among the service providers in $N_K \setminus \mathcal{SP}$. By induction, $\mathcal{C}'$ form a stable coalition structure as service providers in $\mathcal{C}' \cup \mathcal{C}$ is a stable coalition structure. Assume that a coalition g_1 breaks the status of the mentioned stable coalitions. We then have $g_1 \cup S \neq \emptyset$, since $\mathcal{C}'$ is not stable. This means that there must exist a service provider $sp_l \in \mathcal{SP}$ such that $u_l(g_1) > u_l(g_2)$, where g_2 is the coalition containing sp_l. Therefore, there should exist an edge $\langle g_2, g_1 \rangle$ in $\mathcal{G}$, which contradicts the fact that g_2 is a sink node. On the other hand, if $u_l(g_1) > u_l(g_2)$, we have

$$v_l \cdot (d_l^{DC} - d_{l,1}) - \frac{c_{l,1} \cdot c(g_1)}{\sum_{sp_{l'} \in g} c_{l',1}} > v_l \cdot (d_l^{DC} - d_{l,2}) - \frac{c_{l,2} \cdot c(g_2)}{\sum_{sp_{l'} \in g} c_{l',2}}, \quad (25)$$

which means

$$v_l \cdot d_{l,1} + \frac{c_{l,1} \cdot c(g_1)}{\sum_{sp_{l'} \in g} c_{l',1}} < v_l \cdot d_{l,2} + \frac{c_{l,2} \cdot c(g_2)}{\sum_{sp_{l'} \in g} c_{l',2}}$$

Therefore, if there exists a cyclic preference, we have

$$v_l \cdot d_{l,1} + \frac{c_{l,1} \cdot c(g_1)}{\sum_{sp_{l'} \in g} c_{l',1}} < v_l \cdot d_{l,2} + \frac{c_{l,2} \cdot c(g_2)}{\sum_{sp_{l'} \in g} c_{l',2}} < v_l \cdot d_{l,3} + \frac{c_{l,3} \cdot c(g_3)}{\sum_{sp_{l'} \in g} c_{l',3}} < \cdots < v_l \cdot d_{l,1} + \frac{c_{l,1} \cdot c(g_1)}{\sum_{sp_{l'} \in g} c_{l',1}}.$$

A contradiction that $a < a$ is obtained, where $a = v_l \cdot d_{l,1} + \frac{c_{l,1} \cdot c(g_1)}{\sum_{sp_{l'} \in g} c_{l',1}}$. The lemma then follows.

We then analyze the SPoA of the proposed mechanism.

Theorem 2: The SPoA of the proposed mechanism is $O(K \log K)$.

Proof Showing the SPoA of the mechanism is to estimate the upper bound of

$$c^{coll}(C)/c(C^*) = \sum_{l=1}^{L} c_l^{coll}(g) / \sum_{l=1}^{L} c_l(g^*). \quad (26)$$

To this end, we first define

$$\alpha(p_l(\cdot)_{1 \leq l \leq L}) = \max\{\sum_{s=1}^{K} c_{l_s}^{coll}(H_s)/c_{l_s}(H_1)\}, \quad (27)$$

where $c_{l_s}(H_1)$ is the cost of a network service provider that does not collaborate with anyone else (i.e., the VM that caches the service of each service provider sp_l is not shared with other service providers). $H_1, \cdots, H_K$ are a collection of subsets, such that each $H_s = \{sp_{l_s}, \cdots, sp_{l_K}\}$. For clarity, $i(H_s)$ is used to denote the index of cloudlet that hosts coalition H_s.

Denote by $C^* = \{g_1^*, \cdots, g_h^*\}$ the optimal coalition structure that has a social optimum welfare. Let C be the stable coalition structure derived from the proposed mechanism. Assume we have coalition $H_1^1 = g_1^*$. There exists a service provider $sp_{l_1^1} \in H_1^1$ and a coalition $g_1^1 \in C$, such that $c_{l_1^1}(H_1^1) \geq c_{l_1^1}^{coll}(g_1^1)$. Otherwise, all service providers in H_1^1 will form a coalition to strictly promote their utility, this contradicts the fact that C is a stable coalition structure as follows.

We assume that there is another coalition $H_2^1 = H_1^1 \setminus \{sp_{l_1^1}\}$. Similarly, we can find a network service provider $sp_{l_2^1} \in H_2^1$ and a coalition $g_2^1 \in C$ with $sp_{l_1^1} \in H_2^1$ and $c_{l_2^1}(H_2^1) \geq c_{l_2^1}^{coll}(g_2^1)$. Let $g_t^* = \{sp_{l_1^t}, \cdots sp_{l_K^t}\}$, for any $g_t^* \in C^*$. If we continue the aforementioned arguments, we obtain a collection of sets $\{H_s^t\}$, where each $H_s^t = \{sp_{l_s^t}, \cdots sp_{l_K^t}\}$ meets the condition: for any $t \in \{1, \cdots h\}$ and $g_t^* = \{sp_{i_1^t}, \cdots i_K^t\}$, there

exists $g_s^t \in C$, such that $sp_{l_s^t} \in g_s^t$ and $c_{l_s^t}(H_s^t) \geq c_{l_s^t}^{coll}(g_s^t)$.

Therefore, we can bound the SPoA by

$$\frac{\sum_{l=1}^{L} c_l^{coll}(g)}{\sum_{l=1}^{L} c_l(g^*)} = \frac{\sum_{t=1}^{h}\sum_{s=1}^{K} c_{l_s^t}^{coll}(g_s^t)}{\sum_{t=1}^{h} c(g_t^*)}$$
$$\leq \frac{\sum_{t=1}^{h}\sum_{s=1}^{K} c_{l_s^t}^{coll}(H_s^t)}{\sum_{t=1}^{h} c(H_1^t)} \leq \max_t \frac{\sum_{s=1}^{K} c_{l_s^t}^{coll}(H_s^t)}{c(H_1^t)}$$
$$= \alpha(p_l(\cdot)_{1\leq l\leq L}) = \max \frac{\sum_{s=1}^{K} c_{l_s}^{coll}(H_s)}{c_{l_s}(H_1)}$$
$$= \max \frac{1}{c_{l_s}(H_1)} \sum_{s=1}^{K} (p_l(H_s) - v_l \cdot d_{l,i'} + v_l \cdot d_{l,i(H_s)})$$

Let d_{max} be the maximum difference of delays of caching the service of sp_l in any two cloudlets, we then have

$$\frac{\sum_{l=1}^{L} c_l^{coll}(g)}{\sum_{l=1}^{L} c_l(g^*)} < \max \frac{1}{c_{l_s}(H_1)} \sum_{s=1}^{K} \Big(\frac{c_{l,i(H_s)} \cdot c(H_s)}{\sum_{sp_{l'}\in H_s} c_{l',i(H_s)}} + v_l \cdot d_{max}\Big).$$

Without loss of generality, we assume that $c(H_1) = 1$. We have $c_{l,i(H_s)} \leq c(H_s) \leq c(H_1) = 1$, by monotonicity. Assuming that $v_{max} = \arg\max_{1\leq l\leq L} v_l$, we then have

$$\frac{\sum_{l=1}^{L} c_l^{coll}(g)}{\sum_{l=1}^{L} c_l(g^*)} = K v_{max} d_{max} \cdot \max \sum_{s=1}^{K} \Big(\frac{c_{l,i(H_s)} \cdot c(H_1)}{\sum_{t=s}^{K} c_{l',i(H_t)}}\Big).$$

Let $\hat{s}$ be the smallest integer such that $\sum_{t=\hat{s}+1}^{K} c_{l,i(H_t)} \leq 1$. If $s \geq \hat{s}$, we have $\frac{c_{l,i(H_s)}\cdot c(H_1)}{\sum_{t=s}^{K} c_{l',i(H_t)}} \leq c_{l,i(H_s)}$. Otherwise, we have $\frac{c_{l,i(H_s)}\cdot c(H_1)}{\sum_{t=s}^{K} c_{l',i(H_t)}} \leq \frac{c_{l,i(H_s)}}{\sum_{t=s}^{K} c_{l',i(H_t)}}$. Thus, we obtain

$$SPoA$$
$$\leq K v_{max} d_{max} \max\Big(\sum_{s=\hat{s}}^{K} c_{l,i(H_s)} + \sum_{s=1}^{\hat{s}-1} \frac{c_{l,i(H_s)}}{\sum_{t=s}^{K} c_{l',i(H_t)}}\Big)$$
$$\leq K v_{max} d_{max} (2 + \sum_{s=1}^{\hat{s}-1} \frac{c_{l,i(H_s)}}{\sum_{t=s}^{K} c_{l',i(H_t)}}). \quad (28)$$

We then show an upper bound of $\sum_{s=1}^{\hat{s}-1} \frac{c_{l,i(H_s)}}{\sum_{t=s}^{K} c_{l',i(H_t)}}$ by utilizing the fact that for any positive numbers x, y, $\frac{x}{x+y} \leq \log(x+y) - \log x$. We thus have

$$\sum_{s=1}^{\hat{s}-1} \frac{c_{l,i(H_s)}}{\sum_{t=s}^{K} c_{l',i(H_t)}} \leq \log(\sum_{s=1}^{K} c_{l,i(H_s)}) - \log(\sum_{s=\hat{s}}^{K} c_{l,i(H_s)})$$
$$< \log K - \log(1) = \log K, \quad (29)$$

since $\sum_{s=1}^{K} c_{l,i(H_s)} < K$ and $\sum_{s=\hat{s}}^{K} c_{l,i(H_s)} \geq 1$. We have

$$SPoA \leq K v_{max} d_{max} (2 + \log K) = O(K \log K). \quad (30)$$

VI. Simulations

In this section, we evaluate the performance of the proposed algorithms by experimental simulations.

A. Parameter Settings

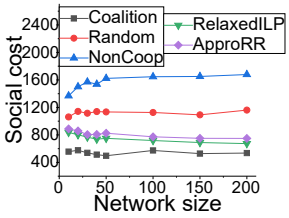
We consider a two-tiered cloud network with size varying from 10 to 200 switches, where each network topology is generated using GT-ITM [8]. Each cloudlet has a computing capacity in the range 8,000 to 16,000 MHz. The bandwidth capacity of each cloudlet varies between 100 $Mbps$ and 1,000 $Mbps$. The computing capacities of each VM is randomly drawn from [4,000, 8,000] MHz. The bandwidth capacity of each VM is drawn from the range of [10,100] $Mbps$. The costs of using a unit amount of computing resource and bandwidth resource in a cloudlet are set within [\$0.15, \$0.22] and [\$0.05, \$0.12], respectively. The average delay experienced by a user in a cached instance of cloudlet is a value between 10 and 50 milliseconds and the average delay experienced in a data center is a value between 50 and 100 milliseconds. 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 mentioned algorithm on 80 different topologies of the mobile edge cloud.

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

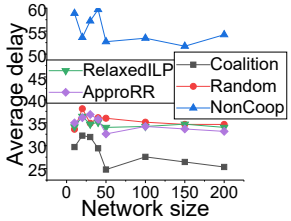
B. Performance Evaluation

We first evaluate the performance of the relaxed `ILP` referred to as `RelaxedILP`, algorithms `ApproRR`, `Coalition` against `NonCoop` and `Random`, by varying the network size from 10 to 200 and the fixing the ratio of the number of cloudlets and the number of switches to 0.3. The results can be seen in Fig. 3. From Fig. 3 (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 is varied from 10 to 200. The reason is that algorithm adopts an efficient cost sharing mechanism that allows multiple network service providers share the resource in a cloudlet; instead algorithms `RelaxedILP`, `ApproRR`, `NonCoop` and `Random` do not allow resource sharing. Resource sharing avoids the resource wastage when some of the services experience low resource utilization. In other words, algorithm `Coalition` enables a more flexible way of using resources of cloudlets in the mobile edge cloud. From Fig. 3 (b), we can see that algorithm `Coalition` obtained the smallest average delay for each request. This is because algorithm `Coalition` allows resource sharing in each coalition with a few network service providers sharing the same cloudlet and thus reduces the chances of implementing services in cloudlets instead of data centers. Fig. 3 (c) shows the running times of the three algorithms, from which it can be seen that algorithm `Coalition` only takes slightly longer time to obtain feasible solutions than algorithms `RelaxedILP`, `ApproRR`, `NonCoop` and `Random`.

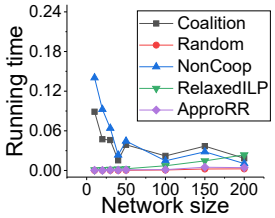
We then investigate the impact of the number of cloudlets on the performance of algorithms `RelaxedILP`, `ApproRR`, `Coalition`, `NonCoop`, and `Random` in a real network



(a) Social cost.

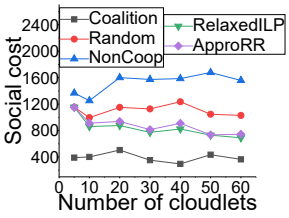


(b) Average delay.

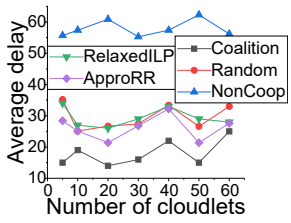


(c) Running time.

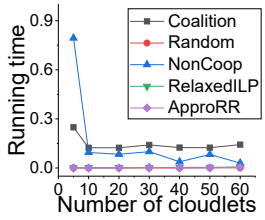
Fig. 3. The performance of algorithms `Coalition`, `NonCoop`, `RelaxedILP`, `ApproRR`, and `Random` in networks with sizes varying from 10 to 200.



(a) Social cost.



(b) Average delay.

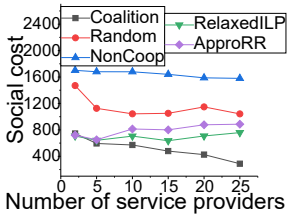


(c) Running time.

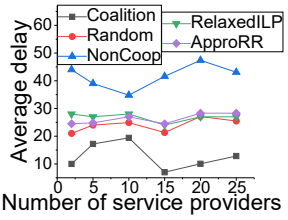
Fig. 4. The performance of algorithms `Coalition`, `NonCoop`, `RelaxedILP`, `ApproRR`, and `Random` in a real network AS1755.

AS1755, by varying the number of cloudlets from 5 to 60. From Fig. 4, we can see that algorithm `Coalition` consistently delivers the lowest social cost compared with that of algorithms `RelaxedILP`, `ApproRR`, `NonCoop` and `Random`. In addition, the social costs by algorithms `Coalition` and `NonCoop` are decreasing with the growth of the number of cloudlets. This is because a higher number of cloudlets make both algorithms have a higher chance of selecting cloudlets with lower costs. Also, the bandwidth cost can be reduced as well, because the cloudlets have a higher probability of being deployed closer to user requests. Similar trends can be found in Fig. 4 (b) for the average delay.

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



(a) Social cost.



(b) Average delay.

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

of network service providers from 2 to 25. Results on the social cost, the average cost of each user request, and the running time are shown in Fig. 5. From Fig. 5 (a) shows that the social cost obtained by algorithms `RelaxedILP`, `ApproRR`, `Coalition` and `NonCoop` increases when the number of network service providers growths from 2 to 5 and decreases afterwards when the number of network service providers keeps increasing. The reason is that with more network service providers being allowed to share resources, the request processing cost can be reduced. However, the network bandwidth cost may be increased since to form coalitions some of them may be assigned to cloudlets further to their requests. This can also be evidenced in Fig. 5 (b). When the number of network service providers keeps increasing, the benefit due to resource sharing traded-off the increase of network bandwidth cost. Therefore, the social cost decreases if the number of network service providers grows from 5 to 25.

VII. Conclusion

In this paper, we investigated the problem of service caching in a mobile edge network of a mobile service market with multiple network service providers. For the cost-sensitive service caching problem without resource sharing, we proposed an approximation algorithm via randomized rounding. We also analyze the approximation ratio of the approximation algorithm. For the cost- and delay-sensitive service caching problem, we devised an efficient and stable game-theoretical mechanism and showed its Strong Price of Anarchy (SPoA). We then evaluated the performance of our algorithm by simulations, and results show that the social cost of all network service providers can be reduced significantly if they collaborate with others.

Acknowledgement

We appreciate the three anonymous referees 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 61802048 and 61802047 and the "Xinghai scholar" program. The work by Weifa Liang and Sid Chi-Kin Chau is supported by the Australian Research Council Discovery Project (no. DP200101985).

REFERENCES

[1] Amazon Pricing. *https://aws.amazon.com/emr/pricing/*

[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] S. Borst, V. Gupta, and A. Walid. Distributed caching algorithms for content distribution networks. *Proc. of IEEE INFOCOM*, IEEE, 2010.

[5] 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, IEEE, 2018.

[6] Y. Gao *et al.* A multi-objective ant colony system algorithm for virtual machine placement in cloud computing. *Journal of Computer and System Sciences*, Vol. 79, No. 8, pp. 1230-1242, 2013.

[7] 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.

[8] GT-ITM. http://www.cc.gatech.edu/projects/gtitm/.

[9] T. He *et al.* It's hard to share: joint service placement and request scheduling in edge clouds with sharable and non-sharable resources. *Proc. of the IEEE 38th International Conference on Distributed Computing Systems* (ICDCS). IEEE, 2018.

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

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

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

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

[14] V. Kantere *et al.* Optimal service pricing for a cloud cache. *IEEE Transactions on Knowledge and Data Engineering*, Vol. 23, No. 9, pp. 1345-1358, 2011.

[15] 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.

[16] S. Mehrotra *et al.* Network system with cache offload service for flash storage. U.S. Patent No. 9,940,241. 10 Apr. 2018.

[17] 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, IEEE, 2019.

[18] M. Mitzenmacher, E. Upfal. Probability and Computing: Randomized Algorithms and Probabilistic Analysis. Cambridge University Press, 2005.

[19] 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.

[20] D. Oppenheimer *et al.* Service Placement in a Shared Wide-Area Platform. *Proc. of the USENIX Annual Technical Conference*, USENIX, 2006.

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

[22] P. Paul *et al.* Efficient service cache management in mobile P2P networks. *Future Generation Computer Systems*, Vol. 29, No. 6, pp. 1505–1521, 2013.

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

[24] 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.

[25] S. Scellato, C. Mascolo, M. Musolesi, and J. Crowcroft. Track globally, deliver locally: improving content delivery networks by tracking geographic social cascades. *Proc. of the 20th international conference on World wide web*. ACM, 2011.

[26] O. Skarlat *et al.* Towards qos-aware fog service placement. *Proc. of the 2017 IEEE 1st international conference on Fog and Edge Computing* (ICFEC), IEEE, 2017.

[27] M. Steiner *et al.* Network-aware service placement in a distributed cloud environment. *ACM SIGCOMM Computer Communication Review*, Vol. 42, No. 4, pp. 73-74, 2013.

[28] J. Tordsson *et al.* Cloud brokering mechanisms for optimized placement of virtual machines across multiple providers. *Future Generation Computer Systems*, Vol. 28, No. 2, pp. 358-367, 2012.

[29] 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, IEEE, 2017.

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

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

[32] Q. Xia, L. Bai, W. Liang, Z. Xu, L. Yao, and L. Wang. QoS-Aware Proactive Data Replication for Big Data Analytics in Edge Clouds. *Proc of 48th Intl Conf on Parallel Processing (ICPP19) Workshops*, 2019.

[33] Z. Xu, W. Liang, M. Jia, M. Huang, and G. Mao. Task offloading with network function services in a mobile edge-cloud network. *IEEE Transactions on Mobile Computing*, Vol.18, No. 11, pp. 2672–2685, IEEE, 2019.

[34] Z. Xu, Y. Zhang, W. Liang, Q. Xia, O. F. Rana, A. Galis, G. Wu, and P. Zhou. NFV-enabled multicasting in mobile edge clouds with resource sharing. *Proc of 48th Intl Conf on Parallel Processing* (ICPP19), 2019.

[35] L. Yang *et al.* Cost aware service placement and load dispatching in mobile cloud systems. *IEEE Transactions on Computers*, Vol. 65, No. 5, pp. 1440-1452, 2015.

[36] 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 Transactions on Network and Service Management*, Vol.15, No. 1, pp.475–488, IEEE, 2018.

[37] 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, IEEE, 2019.

[38] 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.

[39] 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.

[40] Q. Zhang *et al.* Dynamic service placement in geographically distributed clouds. *IEEE Journal on Selected Areas in Communications*, Vol. 31, No. 12, pp. 762-772, 2013.

[41] 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, ACM, 2018.