Dynamic Service Caching in Mobile Edge Networks

(Invited Paper)

Qingyuan Xie*, Qiuyun Wang*, Nuo Yu*†§, Hejiao Huang*, Xiaohua Jia‡
*School of Computer Science and Technology, Harbin Institute of Technology, Shenzhen, China
†School of Electrical Engineering, Anhui Polytechnic University, China
‡Department of Computer Science, City University of Hong Kong, China
§Corresponding author: yunuohit@gmail.com

***Abstract*—Caching application services at the edge of mobile networks can both reduce the traffic load in core networks and improve the quality of services. Since the capacity of a single BS is constrained, only a small number of service can be executed simultaneously by each BS. However, when the BSs are densely deployed in the network, the BSs that are close to each other can cooperatively cache the services to improve the performance of the system. Moreover, we should avoid frequent service switching when the users' service requests always change. In this paper, we study the dynamic service caching (DSC) problem in mobile edge networks. Our objective is to minimize the traffic load that needs to be forwarded to the cloud, as well as considering service switching cost of BSs. This DSC problem involves two important issues, which include cooperative service caching of adjacent BSs and service switching in adjacent time slots. To solve the DSC problem, we propose a dynamic service caching algorithm for the BSs to cooperatively cache the services in an online manner. The simulation results show that our algorithm can greatly reduce the forwarded traffic load without frequently changing the service caching of BSs.**

***Index Terms*—Service caching, mobile edge computing, cellular networks, task offloading.**

I. Introduction

Due to the limited battery power, memory and computational capacity of mobile devices, many mobile applications are facing challenges in executing delay and computational resource[1]. Mobile edge computing (MEC) is an ever-increasing popularity technology to solve this problem, which adopts MEC servers at base stations (BSs) in the vicinity of users[2] as shown in Fig. 1. Owing to the densely deployed MEC servers, the service traffic load can be greatly reduced in the backbone networks. At the same time, it can also help service providers to bring a better service quality for users.

Although many works have studied MEC problems, most of them concentrate on dealing with resource allocation problems based on computation offloading technology[3], [4], [5], while service caching has not drawn much attention. Actually, there are two crucial issues that should be carefully studied for the MEC service caching. First, restricted by the resource capacity of each individual BS, it can only execute a small number of services at the same time. Hence, we need to consider which types of service requests that each BS can handle at a time. Second, BSs are densely deployed in next generation networks, so the coverage areas of neighboring BSs are overlapped. This means a user is always covered by multiple BSs at the same time. Consequently, the BSs need to cooperatively decide which services to cache and which UEs to serve by each of them.

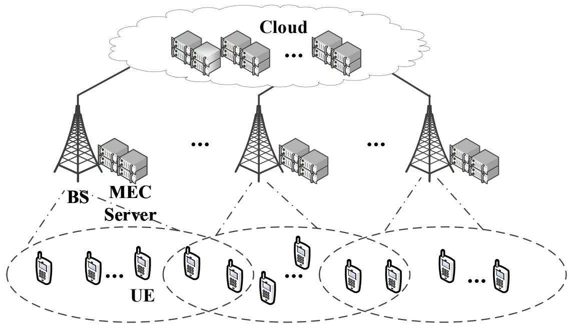

Fig. 1. A mobile edge network structure.

Furthermore, since the users' distributions and requirements are always changing, we need to dynamically cache services for each BS. In this scenario, the switching cost of activating/deactivating services should not be ignored. This makes the service caching problem be more complicated, since that the decisions of service caching for adjacent BSs are coupled both in time and over space. We need to optimize the serve caching of BSs in a continuous-time manner.

In this paper, we study the dynamic service caching problem in mobile edge networks, and develop an efficient algorithm to improve the MEC performance by utilizing the cooperative features of BSs. We aim to minimize the traffic load that needs to be forwarded to remote cloud, as well as considering the service switching cost of BSs. The main contributions of this paper are summarized as follows:

1) We formulate the dynamic service caching problem in mobile edge networks, by considering the switching cost of service caching. There are two important issues included in this problem: a) cooperative service caching of adjacent BSs; b) service switching cost in adjacent time slots. To the best of our knowledge, this problem has not been studied in existing works.

2) We propose a novel strategy for BSs to cooperatively cache services and associate users. Given the users' distributions and service requirements, we can use this strategy to minimize the traffic load that is forwarded to the cloud.

3) We adopt an online prediction algorithm to predict the

users' distributions in future time slots. And then, we combine the proposed service caching strategy with this prediction algorithm to dynamically make the decisions of service caching and user association for each BS. Thus, we can greatly reduce the forwarded traffic load, while do not change the service caching of BSs frequently.

The rest of paper is organized as follows. Section II surveys some related works. Section III describes the system model. Section IV formulates the dynamic service caching problem in mobile edge networks. Section V introduces the service caching algorithms that we propose. Section VI shows the simulation results and compares our algorithms with an existing method. Finally, Section VII concludes the paper.

II. RALATED WORK

MEC has drawn a wide range of attention these years whenever in academia or industrial community. Other than task offloading that is studied by many prior works [6], [7], service caching is another important issue that should be investigated to provide services at the edge of mobile networks.

Service caching has been studied to improve the quality of service in mobile edge networks. In [8], the service placement and load dispatching problem in mobile could system has been studied. It has proposed a centralized heuristic algorithm to minimize the average latency of UEs' service requests. However, it has a drawback that the cloudlets' coverage areas are not assumed to overlap with each other. In [9], service caching problem has been considered. The author proposed an algorithm by utilizing graphing coloring. But it is only based on a single time slot with fixed users and didn't consider the service switching problem. In [10], the authors have studied the collaborative service placement problem for mobile edge computing applications and proposed a distributed problem based on Matching Theory to solve it. These works demonstrated that service caching it is a promising technology in mobile edge network. However, none of these works has investigated the dynamic natures of both users' distributions and users' service requests.

There are some works refer to the resource allocation problems in a dynamic situation. In [11], a framework for dynamic service placement problem based on game-theoretic model has been proposed. But it is based on geographically distributed data centers rather than mobile edge systems. So these works can not be directly used to cache services for MEC application since mobile edge networks are much more complicated. In [12], the authors have considered dynamic changes of offloading request modes over time, and developed an effective prediction mechanism to release or create instances of network functions in different cloudlets for cost saving. It aims to maximize the number of request admitted and reduce the admission cost. In [13], [14], the author considered opportunistic task offloading and location-aware task offloading problem for mobile users. Both of them assumed that each offloaded task will be assigned the computing resources they need. In [15], although it considered the dynamic service placement problem in MEC and proposed an efficient online algorithm based on Gibbs sampling to reduce the computation latency. However, it didn't consider BS-UE association in problem formulation. No existing works has studied the dynamic service caching problem in mobile edge networks, while considering the cooperative features of adjacent BSs and the switching cost of services.

III. SYSTEM MODEL

A. System Description

The network we consider in this paper mainly consists of three components: BSs, UEs, and services. The BSs are densely deployed and hold a set of services. Each UE try to connect a BS in its communication range to get the service it needs. We term the set of BSs as $\mathcal{S}$, the set of UEs as $\mathcal{E}$, and the set of services as $\mathcal{V}$.

We consider a continuous period of time, which is divided into time slots with an equal size. The amount of UEs and the service that each UE requests change over time.

In each time slot, due to the BSs' dense deployment, a UE may within the coverage areas of multiple BSs. We assume that a UE can only be connected to a BS at a time. Also, a UE only requests one service in a time slot. The service request from a UE may be served by any BS in its communication range. If all the BSs that cover a UE can not satisfy its service request, then this request will be forwarded to the cloud by a BS. We assume that any request forwarded to the cloud will be served.

The major resources we consider for a BS are computing resource and radio resource. The computing resources are used to complete the computing tasks in the service request and the radio resource are used to transmit data between UEs and BSs.

B. Computing Resource and Radio Resource

To complete a computing task, many kinds of computing resources are needed, such as CPU, memory and network I/O [16]. In this paper, we only consider the computing capacity of the MEC server's processor. We term the maximum computing capacity of a BS $i \in \mathcal{S}$ as c_i^{max}.

Assume a BS $i \in \mathcal{S}$ servers a set $\mathcal{E}_i$ of UEs, the computing resource for BS i to provide service $k \in \mathcal{V}$ can be modeled as.

$$c_{i,k} = c_k^b + \sum_{j \in \mathcal{E}_i} m_{k,j} c_j. \tag{1}$$

In formulation (1), c_k^b is a constant for service k, which is defined as the baseline computing resource. Usually, this part of computing resource is for the base processing that is shared by all UEs it served. For the second part of computing resource of UEs that request service k in BS i, we use $\sum_{j \in \mathcal{E}_i} m_{k,j} c_j$ to represent. $m_{k,j} \in \{1, 0\}$ indicates whether the $j \in \mathcal{E}_i$ requests the service k or not, where $m_{k,j} = 1$ means UE j request service k, otherwise $m_{k,j} = 0$. Since a UE only request one MEC service at a time, $\sum_{k \in \mathcal{V}} m_{k,j} = 1, \forall j \in \mathcal{E}$. We use c_j to represent the computing resource for processing the computing task of UE j. The total required computing c_i for BS i to provide services is,

$$c_i = \sum_{k \in \mathcal{V}} c_{i,k}. \tag{2}$$

For a BS, radio resource is used to transmit data between UEs and itself. In fact, the output data size of MEC service is usually smaller than the raw data from UEs. Therefore, we only consider the radio resource for uploading data. We assume the network adopts an orthogonal frequency division multiple access (OFDMA) system [17]. The radio resource of a BS is organized in the form of resource blocks (RBs). we also term the maximum the number of RBs of a BS $i \in \mathcal{S}$ as n_i^{max}.

The received signal-to-interference-plus-noise ratio (SINR) per RB of BS i from UE j is defined as $\text{SINR}_{i,j}$. According to the Shannon equation, the received data rate per RB of BS i from UE j can be modeled as.

$$r_{i,j} = Wlog_2(1 + SINR_{i,j}), \tag{3}$$

where W is the bandwidth for each RB. We assume λ_j is the required data rate for UE j to get service from BS i. The number of RBs that should be allocated by BS i to UE j is calculated as,

$$n_{i,j} = \lceil \lambda_j / r_{i,j} \rceil. \tag{4}$$

To server all the UE in $\mathcal{E}_i$, the total amount n_a of RBs that are allocated by BS i is modeled as.

$$n_i = \sum_{j \in \mathcal{E}_i} n_{i,j}. \tag{5}$$

C. *Switching Cost of service*

For the service caching in MEC scene, if a BS $i \in \mathcal{S}$ decides to activate a service $k \in \mathcal{V}$, it will spend certain time to allocate various resources, such as computing resource for processing. Also, when a BS decides to deactivate the service, it will cost time to clean it up. For a BS, frequent service switching will degrade the quality of service. Therefore, we need to take into account the switching cost of services when making the decisions of service caching in each time slot.

Notice that, on one hand, new services need to be activated to adopt the sharp increase of new requests in the near future. On the other hand, old services need to be deactivated if they are not requested in the near future. For two adjacent time slots t and $t-1$, Let $\mathcal{V}_{t,i}$ denote the set of services are cached in BS i in time slot t and $\mathcal{V}_{t-1,i}$ denote the set of services are cached in the BS i in time slot $t-1$. The total number of service switching can be calculated as.

$$s = |\mathcal{V}_{t-1,i} - \mathcal{V}_{t,i})| + |\mathcal{V}_{t,i} - \mathcal{V}_{t-1,i}|. \tag{6}$$

IV. Problem Formulation

In a mobile edge network, the service request from a UE can be served by any BS within its communication range. If all the BSs within the UE's communication range can not serve this request, the request will be forwarded to the cloud. Compare to handling the service request locally, transmitting requests to the cloud brings additional transmission cost, which will incur the transmission delay, and greatly affect the quality of service. In this case, if we reduce the number of service request forwarded to the cloud, the service quality of the MEC system will be greatly improved.

In a time period T, which are divided into time slots t, given the locations and the service requirements of UEs, we need to answer two questions: 1) in each time slot, BS i should activate which services; 2) in each time slot, UE j should connect to which BS to get the service. We define two variables $x_{i,k,t} \in \{1,0\}$ and $y_{i,j,t} \in \{1,0\}$. $x_{i,k,t} = 1$ indicates that, in time slot t, BS i activates the service k, otherwise, BS i does not activate the service k. $y_{i,j,t} = 1$ indicates that, in time slot t, BS i serves UE j, otherwise, BS i does not serve UE j. By defining this two variables, we can model the dynamic service caching (DSC) problem as follows.

Definition 1. Dynamic Service Caching (DSC) Problem.

$$\underset{x_{i,k,t}, y_{i,j,t}, \forall i \in \mathcal{S}, \forall j \in \mathcal{E}, \forall k \in \mathcal{V}}{minimize} \sum_{t \in T} (\sum_{j \in \mathcal{E}_i} (1 - \sum_{i \in \mathcal{S}} y_{i,j,t}) \lambda_j) \tag{7}$$

$$\textbf{s.t.} \quad y_{i,j,t} \leq x_{i,k,t} m_{k,j}, \forall i \in \mathcal{S}, \forall j \in \mathcal{E}, \forall k \in \mathcal{V}, \forall t \in T \tag{8}$$

$$\sum_{i \in \mathcal{S}} y_{i,j,t} \leq 1, \forall j \in \mathcal{E}, \forall t \in T \tag{9}$$

$$\sum_{k \in \mathcal{V}} x_{i,k,t} c_{i,k} \leq c_i^{max}, \forall i \in \mathcal{S}, \forall t \in T \tag{10}$$

$$\sum_{j \in \mathcal{E}} y_{i,j,t} n_{i,j} \leq n_i^{max}, \forall i \in \mathcal{S}, \forall t \in T \tag{11}$$

The constraint (8) indicates that, in each time slot, for a UE that requests a certain service, only the BS which hold this service can satisfy its requirement. The constraint (9) indicates that, in each time slot, a UE can only connect to one BS. The constraint (10) and (11) indicate that, for a BS, the allocated computing and radio resources can not exceed the maximum capacity.

V. Algorithm Design

In this section, we first propose a strategy for a single BS to cache services and select UEs, aiming to minimize the forwarded service traffic load. Based on this strategy, we then propose two service caching algorithms to solving the DSC problem algorithms by considering the switching cost of services.

A. *Service Caching and UE selecting for a single BS*

For a single BS $i \in \mathcal{S}$, given all the service requirements of UEs within its coverage area, it's hard to choose which services to cache and which UEs to serve. Since the service caching and the UE associating are correlated for a BS. To solve this problem, we propose a solution to select services and UEs according to their priorities.

For each BS, we first choose the services that it needs to activate, then select the UEs to serve according to the activated services and the UEs' requirements. We define a variable $\Theta_{i,k}$

Algorithm 1: Service Caching and UE Selecting for a Single BS

```
  Input: $\mathcal{E}_i, \mathcal{E}_i^c, c_i^{max}, c_k^b, c_j, m_{k,j}, n_i^{max}, \lambda_j, SINR_{i,j}, \rho_j$,
         $\forall k \in \mathcal{V}, \forall j \in \mathcal{E}_i^c$
  Output: $\mathcal{E}_i$
1 Compute the set $\mathcal{V}_i^c$ of services that are requested by UEs in $\mathcal{E}_i^c$;
2 Compute the set $\mathcal{E}_{i,k}^c \subseteq \mathcal{E}_i^c$ of UEs that request service $k$, $\forall k \in \mathcal{V}_i^c$;
3 signal = 0;
4 while $\mathcal{V}_i^c \neq \emptyset$ and $singal == 0$ do
5     Select $k^* = argmax \Theta_{i,k}$, $\forall k \in \mathcal{V}_i^c$;
6     while $\mathcal{E}_{i,k^*}^c \subseteq \emptyset$ and $signal == 1$ do
7         Compute the set $\mathcal{E}'_{i,k^*} \subseteq \mathcal{E}_{i,k^*}^c$ of UEs that have the smaller $\rho_j|_{j \in \mathcal{E}_{i,k^*}^c}$;
8         while $\mathcal{E}'_{a,k^*} \subseteq \emptyset$ and $signal == 1$ do
9             Select $j^* = argmin \, n_{i,j}|_{j \in \mathcal{E}'_{i,k^*}}$ ;
10            Compute $c_i$ and $n_i$ assuming UE $j^*$ is served by BS $i$;
11            if $c_i \leq c_i^{max}$ and $n_i \leq n_i^{max}$ then
12                $\mathcal{E}_i = \mathcal{E}_i + \{j^*\}$;
13                $\mathcal{E}'_{i,k^*} = \mathcal{E}'_{i,k^*} - \{j^*\}$;
14            else
15                singal=0;
16            end
17        end
18    end
19 end
```

that represents the priority for each service k in BS $i \in \mathcal{S}$. We model the variable as,

$$\Theta_{i,k} = \frac{\sum_{j \in \mathcal{E}_i^c} m_{k,j} \lambda_j / \rho_j}{c_k^b + \sum_{j \in \mathcal{E}_i^c} m_{k,j} c_j / \rho_j}, \tag{12}$$

where the ρ_j means the amount of BSs that cover UE j. $\sum_{j \in \mathcal{E}_i^c} m_{k,j} \lambda_j / \rho_j$ is the expected total data rate by all UEs that request service k. $\sum_{j \in \mathcal{E}_i^c} m_{k,j} c_j / \rho_j$ is the expected total computing resource by all UEs that request service k. We consider that the less BSs that a UE j can reach, the more weight UE j can get in formulation (12). For a BS, at the cost of same computing capacity, activating a service with higher $\Theta_{i,k}$ can attract more transmission load locally. Therefore, we select services in descending order of their $\Theta_{i,k}$.

Next, we consider the problem of selecting UEs to be served for each selected service. Given $\mathcal{E}_i^c$ request the same service k, we first give high priority to the UEs with a small ρ_j. Since the UEs with a small ρ_j will have less chance to be served by other BSs. To minimize the number of UEs that are not served by BSs, we prefer to select this part of UEs first. Then, if two UEs request same service and have same ρ_j, we give high priority to the UE who consume less number of RBs. This is because in real scenario, the RBs is more scarce than computing resource.

Algorithm 2: Context Free Algorithm for DSC Problem in Each Time Slot

```
  Input: $\mathcal{S}$,$\mathcal{E}$,$\mathcal{V}$,$c_i^{max}$,$c_k^b$,$c_j$,$m_{k,j}$,$n_i^{max}$,$\lambda_j$,$SINR_{i,j}$,$\forall i \in \mathcal{S}, \forall j \in \mathcal{E}, \forall k \in \mathcal{V}$
  Output: $x_{i,k}, y_{i,j}, \forall i \in \mathcal{S}, \forall j \in \mathcal{E}, \forall k \in \mathcal{V}$
1 initialization $x_{i,k} = 0, y_{i,j} = 0, \forall i \in \mathcal{S}, \forall j \in \mathcal{E}, \forall k \in \mathcal{V}$;
2 while $\mathcal{S} \neq \emptyset$ and $\mathcal{E} \neq \emptyset$ do
3     Initialize the set $\mathcal{E}_i$ of UEs that are served by BS $i$, $\mathcal{E}_i = \emptyset, \forall i \in \mathcal{S}$;
4     Compute the number $\rho_j$ of BSs that cover UE $j$, $\forall j \in \mathcal{E}$;
5     Compute the set $\mathcal{E}_i^c \subseteq \mathcal{E}$ of candidate UEs that are coverd by BS $i$, $\forall i \in \mathcal{S}$;
6     foreach $i \in \mathcal{S}$ do
7         Compute $\mathcal{E}_i \subseteq \mathcal{E}_i^c$ by using Algorithm 1;
8     end
9     Compute $\lambda_i = \sum_{j \in \mathcal{E}_i} \lambda_j, \forall i \in \mathcal{S}$;
10    Select $i^* = argmax \lambda_i|_{i \in \mathcal{S}}$;
11    Compute the set $\mathcal{V}_{i^*}$ of services that are requested by UEs in $\mathcal{E}_{i^*}$;
12    Set $x_{i^*,k} = 1, y_{i^*,j} = 1, \forall k \in \mathcal{V}_{i^*}, \forall j \in \mathcal{E}_{i^*}$;
13    $\mathcal{S} = \mathcal{S} - \{i^*\}, \mathcal{E} = \mathcal{E} - \mathcal{E}_{i^*}$;
14 end
```

Algorithm 1 illustrates the details of our solution. For a single BS $i \in \mathcal{S}$, given the set of UEs within its coverage and UEs' service requirements, this algorithm will return the schedule of service caching and UE association. In each iteration (Lines 4-19), it selects the service k with largest $\Theta_{i,k}$, and selects the UEs which request service k. For each BS, The iteration ends when its computing resource or radio resource is exhausted or all UEs within its coverage are selected.

B. Context Free Algorithm

We first propose a context free algorithm, which is illustrated in Algorithm 2 to optimize the decisions of service placement and UE association in each time slot. This algorithm makes the decisions in each time slot without considering the UEs' service requests in other time slots. By using Algorithm 2 in all the time slots, we can get a solution for the DSC problem.

Algorithm 2 is a greedy heuristic solution to the DSC problem. In each iteration (Lines 2-14), it first constructs the set $\mathcal{E}_i^c \subseteq \mathcal{E}$ of candidate UEs for each BS $i \in \mathcal{S}$ that has not made the decision of service caching and UE association. Then, by using Algorithm 1, it computes the set of UEs $\mathcal{E}_i \subseteq \mathcal{E}_i^c$ that can be associated with each BS (Line 7), respectively. After that, the total data rate λ_i of associated UEs for each BS i is computed, and the BS i^* with the largest λ_i is selected. Finally, the set $\mathcal{V}_{i^*}$ of hosted services and the set of associated UEs $\mathcal{E}_{i^*}$ are decided for BS i^*. The above procedure is repeated until the decisions of all BSs have been made, or no more UEs' service requests need to be handled.

Algorithm 3: Service Caching and UE Selecting for a Single BS with Prediction

```
   Input: $\mathcal{E}_i, \mathcal{E}_i^c, c_i^{max}, c_k^b, c_j, m_{k,j}, n_i^{max}, \lambda_j, SINR_{i,j}, \rho_j$,
          $\forall k \in \mathcal{V}, \forall j \in \mathcal{E}_i^c$
   Output: $\mathcal{E}_i$
 1 Compute the set $\mathcal{V}_i^c$ of services that are requested by UEs in $\mathcal{E}_i^c$;
 2 Compute the set $\mathcal{E}_{i,k}^c \subseteq \mathcal{E}_i^c$ of UEs that request service $k$, $\forall k \in \mathcal{V}_i^c$;
 3 Compute the $\phi_{i,k}$, $\forall k \in \mathcal{V}_i^c$, select the set of services $\mathcal{V}_i^s$ in which $\phi_{i,k} \geq -0.5, \forall k \in \mathcal{V}_i^s$;
 4 signal = 0;
 5 while $\mathcal{V}_i^s \neq \emptyset$ and $singal == 0$ do
 6     Select $k^* = argmax\Theta_{i,k}, \forall k \in \mathcal{V}_i^c$;
 7     while $\mathcal{E}_{i,k^*}^c \subseteq \emptyset$ and $signal == 1$ do
 8         Compute the set $\mathcal{E}'_{i,k^*} \subseteq \mathcal{E}_{i,k^*}^c$ of UEs that have the smaller $\rho_j|_{j \in \mathcal{E}_{i,k^*}^c}$;
 9         while $\mathcal{E}'_{a,k^*} \subseteq \emptyset$ and $signal == 1$ do
10             Select $j^* = argmin\, n_{i,j}|_{j \in \mathcal{E}'_{i,k^*}}$ ;
11             Compute $c_i$ and $n_i$ assuming UE $j^*$ is served by BS $i$;
12             if $c_i \leq c_i^{max}$ and $n_i \leq n_i^{max}$ then
13                 $\mathcal{E}_i = \mathcal{E}_i + \{j^*\}$;
14                 $\mathcal{E}'_{i,k^*} = \mathcal{E}'_{i,k^*} - \{j^*\}$;
15             else
16                 singal=0;
17             end
18         end
19     end
20 end
```

C. Context Based Algorithm

In the previous subsection, by executing algorithm 2 in each time slot individually, we can minimize the traffic load forwarded to the cloud. However, in reality, there are still many factors that will affect the quality of service, such as the switching of services. Therefore, when UEs arrive in or depart from the system, the schedule of service caching and UE association at current time is correlated to the situation in the near future.

To measure the changes of services in adjacent time slots, we define the change-range ratio (CR ratio) of a service. For a service k in BS i, in time slot t and $t+1$, its CR ratio is defined as follows,

$$\phi_{i,k} = \alpha(\frac{\lambda_{k,t+1}}{\lambda_{i,t+1}} - \frac{\lambda_{k,t}}{\lambda_{i,t}}) + (1-\alpha)(\frac{\beta_{k,t+1}}{\beta_{i,t+1}} - \frac{\beta_{k,t}}{\beta_{i,t}}), \quad (13)$$

where $\lambda_{k,t+1}$ is the transmission load of service k in time slot $t+1$. $\lambda_{i,t+1}$ is the transmission load for BS i in time slot $t+1$. $\beta_{k,t+1}$ is the number of UEs request service k in time slot $t+1$ and $\beta_{i,t+1}$ is the number of UEs within BS i's coverage area in time slot $t+1$. We assume if $\phi_{i,k} \geq 0.5$, there will be a sharp increase for the number of UEs that request service k in next time slot, we should activate service k in current time. Also we assume if $\phi_{i,k} \leq -0.5$, the service k will be deactivated in current time. Specifically, we adopt a linear regression method to predict the number of UEs for each service in next time slot.

Algorithm 4: Context Based Algorithm for DSC Problem in Each Time Slot

```
   Input: $\mathcal{S}$,$\mathcal{E}$,$\mathcal{V}$,$c_i^{max}$,$c_k^b$,$c_j$,$m_{k,j}$,$n_i^{max}$,$\lambda_j$,$SINR_{i,j}$,$\forall i \in \mathcal{S}, \forall j \in \mathcal{E}, \forall k \in \mathcal{V}$
   Output: $x_{i,k}, y_{i,j}, \forall i \in \mathcal{S}, \forall j \in \mathcal{E}, \forall k \in \mathcal{V}$
 1 initialization $x_{i,k} = 0, y_{i,j} = 0, \forall i \in \mathcal{S}, \forall j \in \mathcal{E}, \forall k \in \mathcal{V}$;
 2 while $\mathcal{S} \neq \emptyset$ and $\mathcal{E} \neq \emptyset$ do
 3     Initialize the set $\mathcal{E}_i$ of UEs that are served by BS $i$, $\mathcal{E}_i = \emptyset, \forall i \in \mathcal{S}$;
 4     Compute the number $\rho_j$ of BSs that cover UE $j$, $\forall j \in \mathcal{E}$;
 5     Compute the set $\mathcal{E}_i^c \subseteq \mathcal{E}$ of candidate UEs that are coverd by BS $i$, $\forall i \in \mathcal{S}$;
 6     foreach $i \in \mathcal{S}$ do
 7         Compute $\mathcal{E}_i \subseteq \mathcal{E}_i^c$ by using Algorithm 3;
 8     end
 9     Compute $\lambda_i = \sum_{j \in \mathcal{E}_i} \lambda_j, \forall i \in \mathcal{S}$;
10     Select $i^* = argmax\lambda_i|_{i \in \mathcal{S}}$;
11     Compute the set $\mathcal{V}_{i^*}$ of services that are requested by UEs in $\mathcal{E}_{i^*}$;
12     Set $x_{i^*,k} = 1, y_{i^*,j} = 1, \forall k \in \mathcal{V}_{i^*}, \forall j \in \mathcal{E}_{i^*}$;
13     $\mathcal{S} = \mathcal{S} - \{i^*\}, \mathcal{E} = \mathcal{E} - \mathcal{E}_{i^*}$;
14 end
```

Algorithm 3 illustrates the details of our solution about service caching and UE selecting for a single BS with prediction mechanism. For a single BS $i \in \mathcal{S}$, in each time slot, this algorithm returns the schedule of service caching and UE association with the context provided by prediction. Firstly, it gets the number of UEs of each service according to the given information, then uses linear regression to predict the number of UEs for all kinds of services in next time slot. By the prediction information, we can calculate the $\phi_{i,k}$, $\forall k \in \mathcal{V}_i^c$ and select the set of services $\mathcal{V}_i^s$. Then, it makes the decisions of service caching and UE association just as Algorithm 1 does. Since it adopts the prediction of UEs' service requests in near future, the dynamics of UEs' service requests are considered by each BS.

Algorithm 4 give the details of our context based algorithm for the DSC Problem. Same to the context free algorithm, algorithm 4 is also a greedy heuristic solution to the DSC problem in each time slot. In each iteration (Lines 2-14), it first computes the set $\mathcal{E}_i^c \subseteq \mathcal{E}$ of candidate UEs that are coverd by BS i. Then, by using algorithm 3, it gets the set of UEs $\mathcal{E}_i \subseteq \mathcal{E}_i^c$ which can be associated with each BS. After computing the total data rate λ_i of each BS, the BS with the largest λ_i is selected. This procedure is repeated until all BSs have been selected or all UEs have been served. By using Algorithm 3, it takes the switching cost of services into consideration when caching the services and associating the UEs.

VI. SIMULATION RESULT

In this section, we evaluate the performance of our proposed algorithms for the DSC problem by simulation experiment.

A. *Experimental settings*

Consider a time period of 5 hours, which is divided into 20 time slots with 15 minutes for each. There are 25 BSs in the network, and the BSs are deployed with an inter-site distance of 300m. We assume that the BSs in the network are homogeneous. The total computing resources of each BS is set to 1 (normalized). We set the bandwidth of uplink channel of each BS to 10MHz and the bandwidth of each BS is 180kHz [18]. In each time slot, UEs are randomly distributed in the network. We assume there are 5 kinds of MEC services and each UE requests a random service in each time slot. The required base computing resource of each service is from 0.05 to 0.25 (normalized), and the required computing resource for each UE is from 0.005 to 0.025 (normalized). The require data rate for a UE is from 1Mbps to 5Mbps. The transmission power of each UE is 10dBm. We assume The uplink channel model follows the free space pathloss model,

$$32.44 + 20log_{10}d_{i,j}(km) + 20log_{10}f(MHz), \qquad (14)$$

where $d_{i,j}$ is the distance between the BS i and UE j, and the f is the carrier frequency. The simulation parameters are summarized in TABLE I.

TABLE I
SIMULATION PARAMETERS

Parameter	**Value**
Inter-site distance	300 m
Number of BS	25
Computer resource per BS	1 (normalized)
Base computing resource per service	$0.05 \sim 0.25$ (normalized)
Required computing resource per UE	$0.005 \sim 0.025$ (normalized)
Carrier frequency (f)	2 GHz
Uplink bandwidth per BS	10 MHz
Bandwidth per RU	180 kHz
Required data rate per UE	1 Mbps $\sim$ 5 Mbps
Transmission power per UE	10 dBm
Uplink channel model	Free Space Pathloss
Power spectral density of noise	-174 dBm/Hz [19]

B. *Performance evaluation*

We compare our algorithms with an Independent and Uncooperative (IU) algorithm. The IU algorithm does not consider the cooperation of adjacent BSs for service caching. And also it does not take the switching cost of service into consideration. In each time slot, each UE is connected to the BS with maximum SINR. The service caching for a BS is decided according to the descending order of each service's total data rate requirement. If a BS can not satisfy some of the service requests from the UEs that it connects, it will forward the service requests to the cloud.

Fig.2 illustrates the number of UEs in the network within a time period of 5 hours. It shows that the number of UEs is always changing.

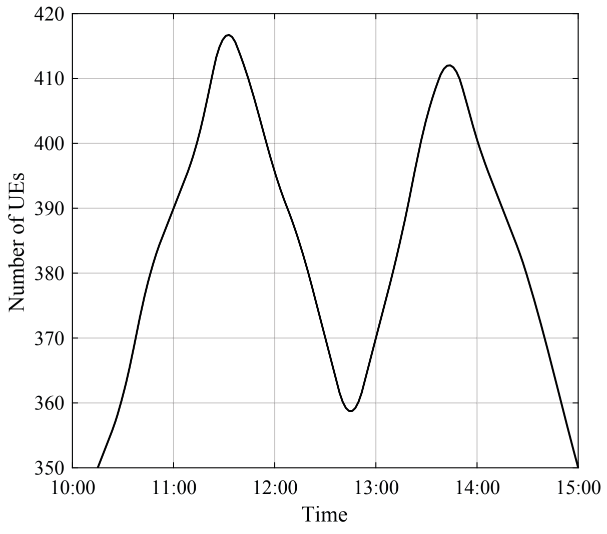

Fig. 2. The number of UEs within 5 hours.

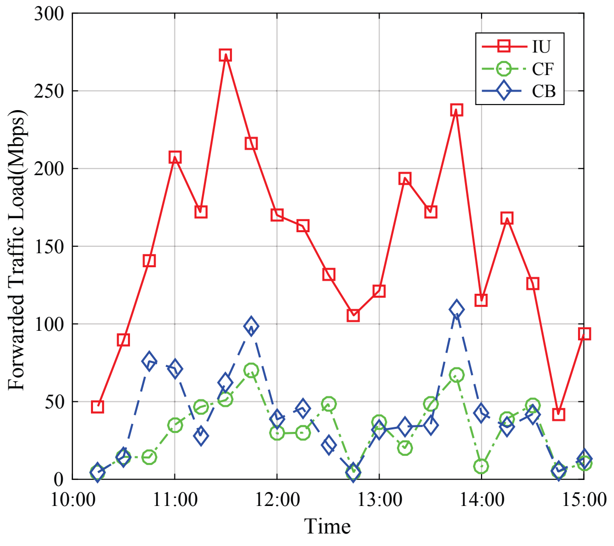

Fig. 3. Comparison of forwarded traffic load by using different algorithms.

Fig.3 compares the total forwarded traffic load in each time slot within 5 hours, by using IU, Context Free (CF) and Context Based (CB) algorithms. We can see that the forwarded traffic load by using IU is much larger than that by using CF and CB. This is because that there is no cooperation of BSs for IU algorithm, in which each UE only request service from the BS with the maximum SINR. Thus, for a dense network, even there are other nearby BSs can provide the requested service, a part of service requests will be forwarded to the cloud. On the contrary, CF and CB improve the performance by using the cooperative features of BSs. This figure also shows that CF algorithm can reduce a little more forwarded traffic load than CB algorithm do at most of the time.

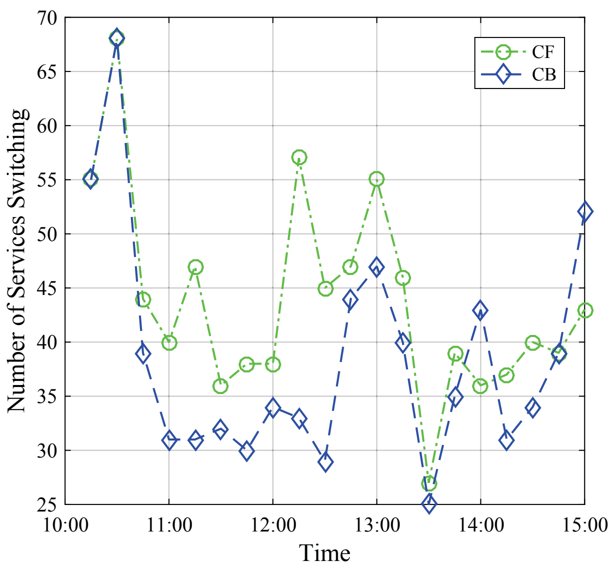

Fig. 4. Number of service switching by using CF and CB algorithms.

Fig.4 illustrates the number of service switching at each time slot by using CF and CB algorithms. It shows that the CB algorithm has a smaller service switching cost at most time slots. Combined with Fig.3, we can see that, by using the predict mechanism, we can reduce the number of service switching at the cost of a slight increasing of forwarded traffic load to the cloud.

VII. Conclusion

In this paper, we study the dynamic service caching problem in mobile edge networks. This problem aims to minimize the traffic load that needs to be forwarded to the cloud, as well as considering the service switching cost of BSs. We propose a dynamic service caching algorithm for the BSs to cooperatively cache the services in an online manner. When the number of UEs keeps changing within a continuous time period, our proposed algorithm can be used to relocate the service caching of BSs and re-associate the UEs to BSs. In this way, we can both cut down the traffic load that is forwarded to the remote cloud and improve the performance of mobile edge computing applications. We finally evaluate the performance of the proposed algorithm by simulations. Simulation results shows that our proposed algorithm can greatly reduce the forwarded traffic load without frequently changing the service caching schedules of BSs.

Acknowledgment

This work was supported in part by the National Key Research and Development Program of China under Grant 2016YFB0800804 and Grant 2017YFB0803002, and in part by the National Natural Science Foundation of China under Grant 61732022 and Grant 61672195.

References

[1] S. Ranadheera, S. Maghsudi, and E. Hossain, "Mobile edge computation offloading using game theory and reinforcement learning," *CoRR*, vol. abs/1711.09012, 2017. [Online]. Available: http://arxiv.org/abs/1711.09012

[2] T. Taleb, K. Samdanis, B. Mada, H. Flinck, S. Dutta, and D. Sabella, "On multi-access edge computing: A survey of the emerging 5g network edge cloud architecture and orchestration," *IEEE Communications Surveys Tutorials*, vol. 19, no. 3, pp. 1657–1681, thirdquarter 2017.

[3] C. You, K. Huang, H. Chae, and B. H. Kim, "Energy-efficient resource allocation for mobile-edge computation offloading," *IEEE Transactions on Wireless Communications*, vol. 16, no. 3, pp. 1397–1411, March 2017.

[4] X. Chen, L. Jiao, W. Li, and et al., "Efficient multi-user computation offloading for mobile-edge cloud computing," *[J]. IEEE/ACM Transactions on Networking and (5): 2795-2808.*, 2016.

[5] P. Mach and Z. Becvar, "Mobile edge computing: A survey on architecture and computation offloading," *CoRR*, vol. abs/1702.05309, 2017. [Online]. Available: http://arxiv.org/abs/1702.05309

[6] B.-G. Chun, S. Ihm, P. Maniatis, M. Naik, and A. Patti, "CloneCloud: Elastic execution between mobile device and cloud," in *Proceedings of the Sixth Conference on Computer Systems*, ser. EuroSys '11. New York, NY, USA: ACM, 2011, pp. 301–314.

[7] E. Cuervo, A. Balasubramanian, D.-k. Cho, A. Wolman, S. Saroiu, R. Chandra, and P. Bahl, "MAUI: Making smartphones last longer with code offload," in *Proceedings of the 8th International Conference on Mobile Systems, Applications, and Services*, ser. MobiSys '10. New York, NY, USA: ACM, 2010, pp. 49–62.

[8] L. Yang, J. Cao, G. Liang, and X. Han, "Cost aware service placement and load dispatching in mobile cloud systems," *IEEE Transactions on Computers*, vol. 65, no. 5, pp. 1440–1452, May 2016.

[9] L. Chen and J. Xu, "Collaborative service caching for edge computing in dense small cell networks," *CoRR*, vol. abs/1709.08662, 2017. [Online]. Available: http://arxiv.org/abs/1709.08662

[10] N. Yu, Q. Xie, Q. Wang, H. Du, H. Huang, and X. Jia, "Collaborative service placement for mobile edge computing applications," in *Proceedings of 2018 IEEE Global Communications Conference (GLOBECOM)*, 2018, pp. 1–6.

[11] Q. Zhang, Q. Zhu, M. F. Zhani, and R. Boutaba, "Dynamic service placement in geographically distributed clouds," in *2012 IEEE 32nd International Conference on Distributed Computing Systems*, June 2012, pp. 526–535.

[12] M.Jia, W.Liang, and Z.Xu, "QoS-Aware task offloading in distributed cloudlets with virtual network function services," *MSWiM '17 Proceedings of the 20th ACM International Conference on Modelling, Analysis and Simulation of Wireless and Mobile Systems*, pp. 109–116, November 2017.

[13] Q. Xia, W. Liang, and W. Xu, "Throughput maximization for online request admissions in mobile cloudlets," in *38th Annual IEEE Conference on Local Computer Networks*, Oct 2013, pp. 589–596.

[14] Q. Xia, W. Liang, Z. Xu, and B. Zhou, "Online algorithms for location-aware task offloading in two-tiered mobile cloud environments," in *Proceedings of the 2014 IEEE/ACM 7th International Conference on Utility and Cloud Computing*, ser. UCC '14. Washington, DC, USA: IEEE Computer Society, 2014, pp. 109–116.

[15] J. Xu, L. Chen, and P. Zhou, "Joint service caching and task offloading for mobile edge computing in dense networks," *CoRR*, vol. abs/1801.05868, 2018. [Online]. Available: http://arxiv.org/abs/1801.05868

[16] N. Yu, Z. Song, H. Du, H. Huang, and X. Jia, "Multi-resource allocation in cloud radio access networks," in *2017 IEEE International Conference on Communications (ICC)*, May 2017, pp. 1–6.

[17] Y. E and D. Z, "A survey on uplink resource allocation in OFDMA wireless networks," *IEEE Communications Surveys & Tutorials*, pp. 14(2): 322–337, 2012.

[18] N. Yu, Z. Song, H. Du, H. Huang, and X. Jia, "Dynamic resource provisioning for energy efficient cloud radio access networks," *IEEE Trans. on Cloud Comput.*, pp. 1–1, 2018.

[19] N. Yu, Y. Miao, L. Mu, H. Du, H. Huang, and X. Jia, "Minimizing energy cost by dynamic switching ON/OFF base stations in cellular networks," *IEEE Trans. on Wireless Commun.*, vol. 15, no. 11, pp. 7457–7469, Nov. 2016.