Winning at the Starting Line: Joint Network Selection and Service Placement for Mobile Edge Computing

Bin Gao[1] Zhi Zhou[2] Fangming Liu*[1] Fei Xu[3]
[1]Key Laboratory of Services Computing Technology and System, Ministry of Education,
School of Computer Science and Technology, Huazhong University of Science and Technology, China
[2]School of Data and Computer Science, Sun Yat-Sen University, Guangzhou, China
[3]School of Computer Science and Software Engineering, East China Normal University, Shanghai, China
[1]{gaobin.me, fangminghk}@gmail.com, [2]hustzhouzhi@gmail.com, [3]kudofaye@gmail.com

***Abstract*—Mobile Edge Computing (MEC) is an emerging computing paradigm in which computational capabilities are pushed from the central cloud to the network edges. However, preserving the satisfactory quality-of-service (QoS) for user applications is non-trivial among multiple densely dispersed yet capacity constrained MEC nodes. This is mainly because both the access network and edge nodes are vulnerable to network congestion. Previous works are mostly limited to optimizing the QoS through dynamic service placement, while ignoring the critical effects of access network selection on the network congestion. In this paper, we study the problem of jointly optimizing the access network selection and service placement for MEC, towards the goal of improving the QoS by balancing the access, switching and communication delay. Specifically, we first design an efficient online framework to decompose the long-term optimization problem into a series of one-shot problems. To address the NP-hardness of the one-shot problem, we further propose an iteration-based algorithm to derive a computation efficient solution. Both rigorous theoretical analysis on the optimality gap and extensive trace-driven simulations validate the efficacy of our proposed solution.**

I. Introduction

With the explosive growth of mobile devices over the years, our daily life is increasingly exposed to a plethora of mobile applications. The widely used cloud computing and technologies such as fourth/fifth generation mobile networks (4G/5G) [1] and the pervasive WiFi access provide centralized service support for mobile applications [2]. However, the centralization of services implies a long distance between end users and service-hosting clouds, which inevitably increases the average end-to-end latency. Especially for the recently emerging delay-sensitive applications, such as augmented reality (AR) [2], virtual reality (VR), and connected cars [3], the existing cloud computing paradigm is unable to meet the stringent timeliness requirement.

*The corresponding author is Fangming Liu (fmliu@hust.edu.cn). This work was supported in part by the National Key Research & Development (R&D) Plan under grant 2017YFB1001703, in part by NSFC under Grant 61722206 and 61761136014 (and 392046569 of NSFC-DFG) and 61520106005, in part by the Fundamental Research Funds for the Central Universities under Grant 2017KFKJXX009 and 3004210116, and in part by the National Program for Support of Top-notch Young Professionals in National Program for Special Support of Eminent Professionals.

To fulfill the requirements of such delay-sensitive applications, the novel paradigm of Mobile Edge Computing (MEC) [4] is proposed as an extension to augment the processing ability of centralized clouds. The key idea beneath MEC is to deploy storage and computation resources from the core network to network edges in the close proximity of users [5]. The elements that implement the edge clouds are micro data centers, and each edge is collocated with an access point (AP) (e.g., base station or WiFi hotspot) [3]. The users can require the mobile edge cloud services by selecting a nearby AP. The services of users are placed at a nearby edge cloud site rather than the remote cloud [6], so as to reduce the latency from end devices to the cloud-hosted service. Inspired by the recent work [7], [8], resources of MEC will be allocated at a fine granularity (e.g., per user) subject to the QoS requirements. The general service model of a MEC system is described as follows. The system divides each user's service applications into two parts: a client entry (CE) and a service entry (SE). The CE runs on the side of client device and communicates with the SE running in the edge cloud to get data or computing services. Here, the system starts independent SE for each user, and the SE can be started by a fine-grained VM or Container. A natural problem is how to distribute and place SEs to provide users with the satisfactory QoS, while achieving the economic efficiency for the system operators. Further, due to the independent and fine-grained system services, the system-wide optimization can be achieved through careful scheduling and placement of services and resources.

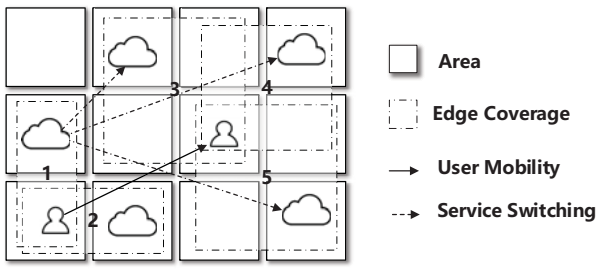

Fig. 1. User mobility and dynamic service migration.

One of the challenges that comes with the emergence of MEC is the dynamic placement and migration of services according to the mobility of users. One commonly used approach for handling users' mobility in MEC is to migrate services. In general, the service is migrated close to the user so as to reduce the communication delay. As an example shown in Fig. 1, we assume that the first time slice user is in the lower left corner area, and in the next slice the user moves from the lower left corner area to the second row and third column area. Before the moving occurs, the user is within the coverage of Edge 1 and Edge 2, and his SE is placed on Edge 1 to render a service. When the user moves to the next area, the coverage of the his surrounding edge cloud changes. The user leaves the coverage of Edge 1 and Edge 2 and switches to the extent that the Edge 3, 4, 5 collectively cover. After the user movement occurs, the system will have to make the following decisions for the user's service: (1) *whether* to switch the service accompanying the user movement; (2) if adapting switching, *when* to switch; (3) if adapting switching, *where* (*e.g.,* Edge 3, 4, 5) to switch. The MEC system operator makes these decisions above for users to acquire the QoS optimization.

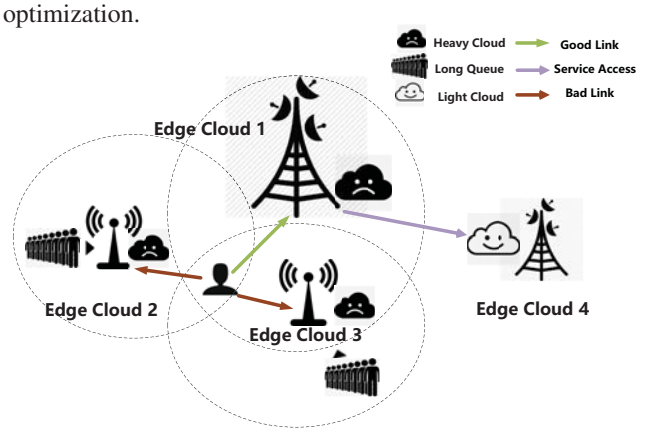

Fig. 2. Network selection and service placement.

However, previous works are limited to optimize the QoS through dynamic service placement, while ignoring the critical effects of access network selection on network congestion [9]. To illustrate the importance of network selection for user service QoS, we consider a typical edge cloud service scenario with four edge clouds shown in Fig. 2. Each edge cloud is composed of a network access point and a cloud. The network access points at Edge 2 and Edge 3 have long queues to process, which results in a long queue delay for users connecting to these two edge clouds. As shown in Fig. 2, if a user moving to this place needs to select which network to access, it is better to select empty network in Edge 1 with lower queuing delay for access. After that, choose the place where the service is placed for the user. Consider the situation in the figure, at this time, the edge cloud around the user placed a huge number of services, resulting in the cloud's heavy. At this point, we can choose to place the service on Edge 4 that is out of the user's coverage, but this procedure will suffer additional communication overhead between Edge 1 and Edge 4. It can be seen from the above analysis that optimizing the user's network selection process and service placement process can improve the service quality of users. Accordingly, to improve the QoS of a MEC system, the MEC system operators necessarily jointly optimize the network selection and service placement strategy for users to gain a system-wide optimization.

In addition, jointly optimizing service placement with network selection can achieve the following benefits. We can avoid the high latency caused by users continuously connecting to poor networks, such as those users who do not move often. Although they do not change their positions over a long period of time, the number of people connected to their nearby APs is constantly changing. If the dynamic selection of the access network is not performed, these users who do not move very often may always connect to the network access point with poor network conditions, and thus they will obtain unacceptable QoS of applications.

In this paper, we jointly consider the access network selection and service placement problem in MEC. We take into account the access delay, switching delay and communication delay to improve the QoS for MEC applications. In particular, the communication delay in this work is incurred when users access services in indirectly connected edge clouds. The access delay and switching delay are incurred by the queue waiting time for AP and the dynamic migration of services, respectively. As the formulated long-term optimization problem is stochastic by nature, we focus on designing an efficient online algorithm, which relies on the limited or zero information of the future, in order to satisfy the random mobility of the users. Specifically, we first divide the long-term optimization problem into a series of one-shot problems. Consider the NP-hardness of these one-shot problems, we apply an iterative-based algorithm named IA to derive an efficient and near-optimal solution. Furthermore, through the theoretical analysis of our algorithm, we can strictly get the competition ratio compared to the offline optimal. Extensive simulations based on the real-world trace demonstrate the effectiveness of our algorithm.

The rest of this paper is organized as follows. We discuss related work in Sec. II. In Sec. III, we introduce our system model and QoS model. In Sec. IV, we introduce our algorithm. In Sec. V, we analyze our algorithm to get a competitive ratio. The online framework is evaluated in Sec. VI and we conclude this paper in Sec. VII.

II. Related Work

The placement of services is a relatively common topic, and extensive research has been explored in cloud computing. The goal of service placement by optimizing the placement of services in cloud computing is to reduce service access delay, reduce communication delay, and achieve network balance, etc [10], [11]. However, the user mobility associated with mobile edge computing creates a focusing problem of service dynamic migration. Part of the dynamic service placement study is focused on user mobility predictions in mobile edge networks. The work in [12], [13] is to formulate the mobility

driven service migration problem as a Markov decision process (MDP) [14]. By using the MDP model to predict the flow of the user, the works [12], [13] make decisions whether to migrate services. However, there are cases where the Markovian assumption is not valid [15]. Besides, the work [7] is mainly to consider various types of costs, such as switching cost, configuration cost, etc., to develop an online service placement strategy with some predictions to minimize costs. However, these works only consider the cost associated with the service (e.g., service switching cost, service configuration cost), while ignoring the cost when the user accesses the network. Non-service related costs, such as queuing delays, are incurred when a user selects an access point to access the network without optimization. In particular, when the access point is overloaded [16], it will cause network congestion, which will become a major overhead part of the communication with the service.

A closely related work to ours is [17]. Without requiring the future user mobility as a priory knowledge, [17] develops an energy-aware mobility management scheme to optimize the total delay due to both communication delay and computation delay under the long-term energy consumption constraint of the user. Compared to [17], our work has the following advantages and improvements: (1) Our work considers the nonlinear network access latency, switching delays, and communication delays to minimize overall latency. (2) [17] only considers single-user's service placement, while we consider multiple users and we can minimize the average delay by coordinating all users in the entire edge network system. (3) We design an online algorithm to reduce frequent switching cost and balance the access delay and communication delay.

III. System Model and Problem Formulation

A. System Model

We consider a mobile edge computing system which consists of M access points (e.g., 4G, 5G or WiFi.) and N users. Each AP is equipped with several MEC servers and can be regarded as edge computing clouds. We denote the set of APs/edge computing clouds and the set of users by $\mathcal{M} = \{0, 1, ...M\}$ and $\mathcal{N} = \{0, 1, ...N\}$, respectively. Without loss of generality, the system works in a time-slotted fashion within a large time span and its timeline is discretized into time frames $t \in \mathcal{T} = \{0, 1, ..., T\}$. We assume that MEC platform running on edge clouds adopt light weight virtualization solutions (e.g., VM or Container.) [18] that allow a portable runtime of MEC services. For each user $k \in \mathcal{N}$, a service program is brought up on one of the edge clouds $i \in \mathcal{M}$ to offer all necessary computing environment (e.g., Computation, Storage or Database access.) desired by users. For edge cloud $i \in \mathcal{M}$, we let E_i indicate the total number of services that can be started in edge cloud i. At each time slot t, each user k has multiple means to access the edge clouds via neighboring APs [19], denoted by $\phi_k(t)$. Key parameter notations are listed in Table I for ease of reference.

TABLE I
List of Main Notations.

Notation	Definition
$\mathcal{N}$	Set of mobile users
$\mathcal{M}$	Set of access point/edge clouds
$\phi_k(t)$	Set of available access points of user k
$x_{ik}(t)$	Whether the service of user k is placed on edge cloud i (=1) or not (=0)
$y_{jk}(t)$	Whether the AP j is selected for user k to access the edge clouds (=1) or not (=0)
$x(t)$	Vector of the decision variable $x_{ik}(t)$
$y(t)$	Vector of the decision variable $y_{jk}(t)$
C_j	Capacity of each AP in MEC
E_i	The maximum capacity of edge cloud i
D^q	The total queuing delay to access the network
D^s	The total switching delay to adjust service placement
D^c	The total communication delay to access services
$l_{ij}(t)$	The communication delay between edge cloud i and j
$r_k(t)$	The resource demand for AP of user k
$s_k(t)$	The service resource demand for edge cloud of user k

B. Access Point Selection Model

At each time slot t, the operator makes the AP selection decision for all the users. Here we take a binary indicator $y_{jk}(t)$ to denote the dynamic access point selection decision variable, let $y_{jk}(t) = 1$ if the user $k \in \mathcal{N}$ selects access point $j \in \phi_k(t)$ to connect to the edge clouds at time slot t, and $y_{jk}(t) = 0$ otherwise. Note that at a given time slot, since each user is served by only one access point, we have the following constraints for $y_{jk}(t)$:

$$\sum_{j \in \phi_k(t)} y_{jk}(t) = 1, \forall k \in \mathcal{N}, \tag{1}$$

$$y_{jk}(t) \in \{0, 1\}, \forall j \in \phi_k(t), \forall k \in \mathcal{N}. \tag{2}$$

At any time t, given a service demand $r_k(t)$ for a given user, the system cannot exceed the resource limitation of the access point:

$$\sum_{k \in \mathcal{N}} r_k(t) y_{jk}(t) \leq C_j, \forall j \in \phi_k(t). \tag{3}$$

C. Service Placement Model

As above mentioned, each user accesses edge clouds via a suitable AP. Then for each user k, a service is brought up on one of the edge clouds i to provide demand service for user k. Specially, there is no necessary association between the access point selection and service placement for a single user. To be more specific, the service of user k can be placed on any edge cloud $i \in \mathcal{M}$, but only can select AP $j \in \phi_k(t)$ to access edge clouds at time slot t. We assume this because each user have limited communication distance in a MEC system, that is, user can only access the networks via the AP close to him. Similar

to the AP selection model, we denote the service placement model as follow:

$$\sum_{i\in\mathcal{M}} x_{ik}(t) = 1, \forall k \in \mathcal{N}, \tag{4}$$

$$\sum_{k\in\mathcal{N}} s_k(t)x_{ik}(t) \leq E_i, \forall i \in \mathcal{M}, \tag{5}$$

$$x_{ik}(t) \in \{0,1\}, \forall i \in \mathcal{M}, \forall k \in \mathcal{N}. \tag{6}$$

Constraint (4) states that each service must be allocated to exactly one of the edge clouds. Eq. (5) ensures that total number of service in each cloud do not exceed capacity limits. Eq. (6) indicates that whether placing service of user k on the edge cloud i or not.

D. QoS Model

1) Queuing delay: For an access point, the number of connected users varies over time. Occasionally, AP is selected preferentially according to the location, which may cause some APs to be overloaded. The increase in queuing delay will greatly affect the quality of user service quality. To analyze the delay performance of users, we model each access point as a M/M/1 queue [20]. Therefore, the total queuing delay of the whole AP set $\mathcal{M}$ at time slot t is given by

$$D^q(\vec{y}(t)) = \sum_{k\in\mathcal{N}}\sum_{i\in\mathcal{M}} y_{jk}(t)\frac{1}{C_j - \sum_{k\in\mathcal{N}} r_k(t)y_{jk}(t)}, \tag{7}$$

where C_j is the capacity of AP j. Note that each user only can select AP from his nearby AP set $\phi_k(t)$, so y_{jk} used in (7) should be 0 when $j \notin \phi_k(t)$. To ensure that the Eq. (7) makes sense, we will guarantee that the resource capacity C_j is greater than the demand for all resources at any time.

2) Switching Delay: In the paradigm of MEC, it is inevitable to consider user's mobility. When users roam around the APs, keeping the placement of their services invariant would greatly deteriorate the perceived latency of users. It behooves to optimize the user experience via dynamically replacing or migrating the services. Toggling services and edge clouds indeed incurs the switching delay, which can augment the service time due to the lead or the initialization time of booting new software resources, loading profiles, migrating states, and so on. The total switching delay of all users between time slot $t-1$ and time slot t is given:

$$D^s(\vec{x}(t), \vec{x}(t-1)) = \sum_{k\in\mathcal{N}}\sum_{i\in\mathcal{M}} s_i[x_{ik}(t) - x_{ik}(t-1)]^+, \tag{8}$$

where $[x_{ik}(t) - x_{ik}(t-1)]^+ = \max\{x_{ik}(t) - x_{ik}(t-1), 0\}$.

3) Communication Delay: In our model, the placement of the service is independent, and it does not have to be placed in the cloud adjacent to the user's communication access point. Doing so will alleviate the pressure of some hotspot edge clouds to reap the load balancing of the entire edge clouds. Correspondingly, users accessing services across the edge clouds will spend extra communication delays. Let l_{ij} denote the delay of the transmission path from the AP j to the edge cloud i. Thus, when considering the AP selection decision y_{jk} and service placement decision x_{ik}, the overall corresponding communication delay at time slot t in the system can be expressed as:

$$D^c(\vec{x}(t), \vec{y}(t)) = \sum_{k\in\mathcal{N}}\sum_{i\in\mathcal{M}}\sum_{j\in\phi_k(t)} y_{jk}(t)x_{ik}(t)l_{ij}(t). \tag{9}$$

E. Problem Formulation

By combining the queuing delay $D_k^q(t)$, the switching delay $D_k^s(t)$ and the communication delay $D_k^c(t)$, we formulate **the offline network selection and service placement optimization problem (NSSP)** as follow:

$$\begin{aligned} &\min \sum_{t=1}^{T} D(\vec{x}(t), \vec{y}(t)) = \sum_{t=1}^{T}(D^q(\vec{y}(t)) + \\ &D^s(\vec{x}(t), \vec{x}(t-1)) + D^c(\vec{x}(t), \vec{y}(t))) \\ &s.t. \quad (1)(3)(2)(4)(5)(6). \end{aligned} \tag{10}$$

In a long-term issue, it requires the future system information (ie., the user mobility pattern), so that MEC system operator can make the global optimal decisions for users to achieve better performance both in network selection and service placement. Unfortunately, it is difficult to get the advance system information. To address this, we can firstly divide the **NSSP** problem (10) into T one-shot optimization problems, **ONSSP**:

$$\begin{aligned} &\min\ D(\vec{x}(t), \vec{y}(t)) = D^q(\vec{y}(t)) + \\ &D^s(\vec{x}(t), \vec{x}(t-1)) + D^c(\vec{x}(t), \vec{y}(t)) \\ &s.t. \quad (1)(2)(3)(4)(5)(6), \end{aligned} \tag{11}$$

and then we can get solution for long-term problem with a competitive ratio by solving a series of short-term problems.

F. Challenges of network selection and service placement

In the above sections, we have formulated the optimization problem of network selection and service placement. Now we analyze the complexity of the problem. Obviously, problem (12) is a complex pure integer nonlinear programming **(PINLP)** problem for the binary variable x and the binary variable y. In the objective function (12), the term $D^q(\vec{y}(t))$ denotes the network selection and $D^s(\vec{x}(t), \vec{x}(t-1)) + D^c(\vec{x}(t), \vec{y}(t))$ denotes the service placement, respectively. Specially, the partial objective function $D^c(\vec{x}(t), \vec{y}(t))$ is a coupling term for network selection and service placement, which depends on both the network selection variable x and the service placement variable y. This coupling term creates difficulties for the problem. We can't just achieve optimal state by optimizing network selection or service placement. Well, this coupling term also illustrates the need for joint optimization of network selection and service placement. Only by optimizing the relationship between the two can the objective function be optimized overall.

IV. ALGORITHM

In the above section, we demonstrate the difficulties of our joint optimization problem. To address these difficulties, in this section, we firstly design an online algorithm for the long-term optimization. Then we design an iteration-based algorithm to gain an efficient solution to deal with the divided short-term problems.

A. Online Algorithm

Our objective is to find an online algorithm to minimize the total delay of the users in the MEC system. Intuitively, we can re-calculate the optimal solution in each time slot, but this comes the expense of frequent switching delay. For example, assume that user k moves to a light load access point j at time slot t, and one-shot optimization would select AP j for k to access network. But if crowded users flood to AP j at time slot $t+1$, the offline optimization may keep the connect state for user k without switching. This inspires us to carefully consider when to switch the service placement for the user and which network access point to choose for the user. The key idea of our algorithm is to tolerate as much as nonswitching delay (e.g., queuing delay and communication delay) as possible until it significantly exceeds the switching delay. Inspired by the work [21], we design an online lazy switching algorithm (*OLSA*) (Alg. 1) to deal with the AP selection and service placement problem.

In order to describe our algorithm more clearly, we divide the overall delay $D(\vec{x}(t),\vec{y}(t))$ incurred at time t into two parts: (1) switching delay $D^s(\vec{y}(t),\vec{y}(t-1))$ defined in (8) which is related to the decisions in $t-1$; (2) nonswtiching delay $D^{ns}(\vec{x}(t),\vec{y}(t))$ that only relies on the current information at t, that is, the sum of queuing delay and the communication delay:

$$D^{ns}(\vec{x}(t),\vec{y}(t)) = D^q(\vec{y}(t)) + D^c(\vec{x}(t),\vec{y}(t)), \tag{12}$$

hence, the total latency can be expressed as follows:

$$\sum_{t=1}^{T} D(\vec{x}(t),\vec{y}(t)) = \sum_{t=1}^{T}(D^{ns}(\vec{x}(t),\vec{y}(t)) + D^s(\vec{x}(t),\vec{x}(t-1)). \tag{13}$$

In the algorithm, we initialize the two decision vector x and y by assigning $T=1$. After that, we obtain the original switching delay and nonswitching delay deceived by above decision variables. Then, at each time instance t, *OLSA* chooses the proper access point and service placement for every user according to the following strategies. Let $\hat{t}$ state the last switching moment over the past time. *OLSA* firstly computes overall nonswitching delay $\sum_{v=\hat{t}}^{t-1} D^{ns}(\vec{x}(t),\vec{y}(t))$ in time $[\hat{t}, t-1]$. The algorithm checks whether the overall nonswitching delay is at least β times than the switching delay $D^s(\vec{y}(\hat{t}),\vec{y}(\hat{t}-1))$. If the condition is tenable, *OLSA* obtains the decision vector $x(t)$ and $y(t)$ by minimizing problem (12). Under the previous conditions, the algorithm decides to place or migrate the services at time t by judging whether a new service placement strategy is generated ($x(t) \neq x(t-1)$) at time t. Other than that, in all remaining cases, the algorithm keeps the service placed without any switching operation ($x(t) = x(t-1)$).

Algorithm 1: The Online Lazy Switching Algorithm (OLSA)

```
t = 1;
t̂ = 1; // The time cursor for the last service switching occurring;
Initialize AP selection decision vector x(1) and service placement decision vector y(1) by minimizing problem (10) where T = 1;
Compute D^ns(x(1), y(1)) and D^s(y(1), y(0));
while t ≤ T do
    if D^s(y(t̂), y(t̂ − 1)) ≤ (1/β) Σ_{v=t̂}^{t−1} D^ns(x(t), y(t)) then
        Obtain the vector x(t) and y(t) by minimizing problem (12) using Alg. 2;
        if x(t) ≠ x(t − 1) then
            Use the new service placement vector x(t);
            t̂ = t;
        end if
    end if
    if t̂ < t then
        x(t) = x(t − 1);
        If y(t) is not derived, compute it by solving problem (12) using Alg. 2;
    end if
    t = t + 1;
end while
```

Here we use $\beta > 0$ as an indicator to control the frequency of switching service. More specifically, a larger β signifies to tolerate more nonswitching cost in our algorithm. This can be observed from the judgement condition $D^s(\vec{y}(\hat{t}),\vec{y}(\hat{t}-1)) \leq \frac{1}{\beta}\sum_{v=\hat{t}}^{t-1} D^{ns}(\vec{x}(t),\vec{y}(t))$. We noticed that when we have an expected switching threshold, a larger β will tolerate a larger $\sum_{v=\hat{t}}^{t-1} D^{ns}(\vec{x}(t),\vec{y}(t))$, that is, more nonswitching costs can be tolerated. We can set β values for different types of applications to get the desired quality of service.

B. Iteration-based algorithm for solving nonswitching problem (12)

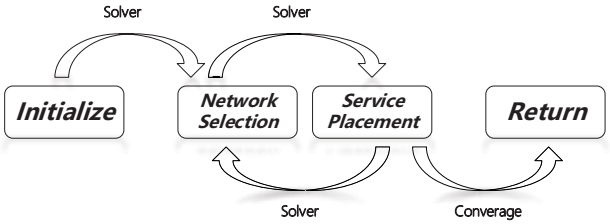

Fig. 3. Iteration-based algorithm for minimizing nonswitching cost.

A critical challenge of *OLSA* is how to solve the problem (12). Note that the difficulty in solving the above problem roots partially in that the decisions of network selection and service placement of top level stages are coupled. While if

we fix one decision, then optimizing another decision reduces to a linear programming or mixed-integer linear programming, both of which are greatly simplified. Motivated by this insight, we solve the pure integer nonlinear programming (**PINLP**) problem (12) by optimizing network selection and service placement, iteratively. Specially, when optimizing network selection, we fix the service placement decision set as $\hat{\vec{x}}$ and solve the integer nonlinear programming (**INLP**) problem:

$$\begin{aligned} \min \quad & \sum_{k\in\mathcal{N}}\sum_{i\in\mathcal{M}} y_{ik}(t)\frac{1}{C_i-\sum_{k\in\mathcal{N}} r_k(t)y_{ik}(t)}+ \\ & \sum_{k\in\mathcal{N}}\sum_{i\in\mathcal{M}}\sum_{j\in\Phi_k(t)} \vec{y}(t)\theta_{ik}(t) \\ \text{s.t.} \quad & (1)(2)(3), \end{aligned} \tag{14}$$

where $\theta_{ik}(t)$ can be obtained by $\hat{\vec{x}}$ and $l_{ij}(t)$.

Algorithm 2: The Iteration-based Algorithm (*IA*)

Input:
User Number, N;
Edge Cloud Number, M;
User's Available Edge Cloud Map, ϕ_k;
Precision tolerance, ϵ;
Max Iteration Number, $MaxIter$;
Output:
Service Placement, x;
Network Selection, y;

$lastDelay \leftarrow 0$,$currentDelay \leftarrow 0$;
$xt \leftarrow$ NULL,$yt \leftarrow$ NULL;
for $j = 1, 2, ..., MaxIter$ **do**
  $yt \leftarrow$ NetworkSelection.solve(N, M, ϕ_k);
  $xt \leftarrow$ ServicePlacement.solve(N, M, ϕ_k);
  $currentDelay \leftarrow$ ComputeDelay(xt,yt);
  if $lastDelay \leq currentDelay + \epsilon$ **then**
    $x \leftarrow xt$;
    $y \leftarrow yt$;
    break;
  else
    $lastDelay \leftarrow$ ComputeDelay(xt,yt);
  end if
end for
return x, y.

When optimizing the service placement, we fix the network selection as $\hat{\vec{y}}$, and obtain the mixed integer programming (**MIP**) problem (15):

$$\begin{aligned} \min \quad & \sum_{k\in\mathcal{N}}\sum_{i\in\mathcal{M}}\sum_{j\in\Phi_k(t)} \vec{x}(t)\delta_{ik}(t) \\ \text{s.t.} \quad & (4)(5)(6), \end{aligned} \tag{15}$$

where $\delta_{ik}(t)$ can be obtained by $\hat{\vec{y}}$ and $l_{ij}(t)$.

Both subproblem (14) and subproblem (15) can be proved to be NP-hard problems by polynomial time relaxation (to be detailed in Sec. V). Because of the NP-hard nature of the two problems, it is difficult to find algorithms that solve them at polynomial time. Fortunately, we can obtain a solution by the solver (e.g., Gurobi [22]).

Overall, our solution for optimizing the nonswitching cost problem (12) consists of optimizing network selection and service placement alternately via fixed-point iterations, as shown in Fig. 3 and Alg. 2. Starting with the initialize, we iteratively optimize network selection and service placement via solver optimizer, to optimize the solution of each subproblem and thus to optimize the unified objective. The iterative heuristic is terminated when no further reduction on the unified objective is possible or when an expected number of iterations are executed. It would be specially mentioned that our algorithm can be completed in a very limited number of iterations. In our study, the algorithm obtains a solution usually ended within two iterations (more details in Sec. VI).

V. Theoretical Analysis

In this section, we analyze the theoretical performance of *OLSA* algorithm for MEC. First we discuss the performance relationship between the swtiching delay and nonswitching cost. Then we compare *OLSA* with the offline optimal to discern the competitive ratio.

Theorem 1. *The one-shot optimization problem **ONSSP** (11) is NP-hard.*

Proof. We construct a polynomial-time reduction to **ONSSP** (11) from the generalized assignment problem (GAP), a classic combinatorial optimization problem which is known to be NP-hard [23]:

$$\begin{aligned} \min \quad & \sum_{i=1}^{m}\sum_{j=1}^{n} c_{ij}x_{ij} \\ \text{s.t.} \quad & \sum_{i=1}^{m} x_{ij} = 1, \forall j = 1, ..., n, \\ & \sum_{j=1}^{n} w_{ij}x_{ij} \leq t_i, \forall i = 1, ..., m, \\ \text{Var} \quad & x_{ij} \in \{0, 1\}, \forall i = 1, ..., m, j = 1, ..., n. \end{aligned} \tag{16}$$

Given an instance $A = (m, n, c_{ij}, w_{ij}, t_i)$ of the GAP, we map it to an instance of the one-shot optimization problem **ONSSP** (11) with $A' = (|\mathcal{M}| = m, |\mathcal{N}| = n, l_{ij} = c_{ij}, \vec{y} = 1, w_j = r_k, t_i = C_i)$. Clearly, the above mapping can be done in polynomial time. Then, if there exists an algorithm that solves the cost-performance tradeoff problem A', it solves the corresponding GAP A as well. As a result, the GAP can be treated as a special case of **ONNSP**. Given the NP-hardness of GAP, the one-shot optimization problem **ONNSP** (11) must be NP-hard as well. □

Theorem 2. *The network selection subproblem (14) is NP-hard.*

Proof. The proof process is similar to Theorem 1. Given an instance $A = (m, n, c_{ij}, w_{ij}, t_i)$ of the GAP problem (16), we map it to an instance of the network selection subproblem (14) with $A' = (|\mathcal{M}| = m, |\mathcal{N}| = n, \theta_{ik} = c_{ij}, t_i = C_i)$. At the

same time, we relax the expression about the queue delay to 0. These relaxations can be finished in polynomial time. Hence, network selection subproblem (14) is NP-hard as well. □

Theorem 3. *The service placement subproblem (15) is NP-hard.*

Proof. Given an instance $A = (m, n, c_{ij}, w_{ij}, t_i)$ of the GAP problem (16), we map it to an instance of the service placement subproblem (15) with $A' = (|\mathcal{M}| = m, |\mathcal{N}| = n, \delta_{ik} = c_{ij}, t_i = E_i)$. This relaxation can be finished in polynomial time. Hence, subproblem (15) is NP-hard as well. □

A. Online Algorithm Performance Analysis

1) Switching Cost Analysis:

Lemma 1. *In a time slice $[1, T]$, given any control parameters β, the $OLSA$ algorithm provides a determined bound for overall switching delay as follows:*

$$\sum_{t=1}^{T} D^s(\vec{y}(t-1), \vec{y}(t)) \leq \frac{1}{\beta} \sum_{t=1}^{T} D^{ns}(\vec{x}(t), \vec{y}(t)).$$

Proof. First define $\hat{t}_i$ as the time point for i-time switching occurring. Before $(i+1)$-time switching comes, according to judgement condition of the online algorithm, we have the bound that the nonswitching delay is at most β than the switching cost at each time slot t in $[\hat{t}_i, \hat{t}_{i+1}]$. Hence, we have:

$$\sum_{t=1}^{T} D^s(\vec{y}(t-1), \vec{y}(t)) \leq \frac{1}{\beta} \sum_{t=1}^{T-1} D^{ns}(\vec{x}(t), \vec{y}(t))$$
$$\leq \frac{1}{\beta} \sum_{t=1}^{T} D^{ns}(\vec{x}(t), \vec{y}(t)).$$

□

2) No Switching Cost Performance Analysis:

Lemma 2. *Let $(\vec{x}^*(t), \vec{y}^*(t))$ denote the optimal solution to problem (10). In a time slice $[1, T]$, the overall nonswtiching delay is at most ϵ times the total offline optimal, that is:*

$$\sum_{t=1}^{T} D^{ns}(\vec{x}(t), \vec{y}(t)) \leq \epsilon \sum_{t=1}^{T} D(\vec{x}^*(t), \vec{y}^*(t)).$$

where $\epsilon = \max_{t \in [1,T]} \frac{\max D^{ns}(\vec{x}(t), \vec{y}(t))}{\min D^{ns}(\vec{x}(t), \vec{y}(t))}$.

Proof. We prove this lemma by the definition of ϵ. The meaning of ϵ is the ratio of the maximum nonswitching delay to the minimum switching delay during the time period $[1, T]$. On this ground, the nonswitching delay incurred at each time slot is ϵ times the optimal nonswitching delay and we have $D^{ns}(\vec{x}(t), \vec{y}(t)) \leq \epsilon D^{ns}(\vec{x}^*(t), \vec{y}^*(t))$. Therefore, the lemma can be derived by the following steps:

$$\sum_{t=1}^{T} D^{ns}(\vec{x}(t), \vec{y}(t)) \leq \epsilon \sum_{t=1}^{T} D^{ns}(\vec{x}^*(t), \vec{y}^*(t))$$
$$\leq \epsilon \sum_{t=1}^{T} \{D^{ns}(\vec{x}^*(t), \vec{y}^*(t)) + D^s(\vec{y}^*(t-1), \vec{y}^*(t))\}$$
$$\leq \epsilon \sum_{t=1}^{T} D(\vec{x}^*(t), \vec{y}^*(t)).$$

□

B. Competitive Analysis

Theorem 4. *The $OLSA$ produces a solution with a competitive ratio of $\epsilon(1 + \frac{1}{\beta})$.*

Proof. By applied Lemma 1 and Lemma 2 to equation (13), we have:

$$\sum_{t=1}^{T} D(\vec{x}(t), \vec{y}(t)) = \sum_{t=1}^{T} (D^{ns}(\vec{x}(t), \vec{y}(t))$$
$$+ D^s(\vec{y}(t), \vec{y}(t-1)))$$
$$\leq \sum_{t=1}^{T} ((1 + \frac{1}{\beta}) D^{ns}(\vec{x}(t), \vec{y}(t)))$$
$$\leq \epsilon(1 + \frac{1}{\beta}) \sum_{t=1}^{T} (D(\vec{x}^*(t), \vec{y}^*(t))).$$

hence, we prove that the $OLSA$ can gain a competitive ratio of $\epsilon(1 + \frac{1}{\beta})$ compared to the offline optimum. □

VI. Performance Evaluation

In this section, we validate the performance of our online algorithm (*i.e.,* OLSA, IA) with extensive simulations driven by the real-world trace. The simulation setup and algorithm benchmarks are elaborated below.

A. Simulation Setup

1) Data Set: We take advantage of the Shanghai taxi trajectory data set [24] to model users, access point and edge clouds. This data set is obtained from Wireless and Sensor networks Lab (WnSN), Shanghai Jiao Tong University. The reasons for selecting this trace are as follows. Firstly, taxi trace are widely used to predict human mobility from discovering patterns of an urban taxi transportation system [25]. Note we call a region which has a lot of taxi visiting records as Point of Interest (PoIs) [25]. The distribution of PoIs actually follows the mobile service crowd area, making them perfect locations for edge cloud deployment in the future. Secondly, from the perspective of AP construction planning, most PoIs (e.g., business center, residential area.) will be planned to build more base stations and WiFi hotspots to provide superior network access service [26]. Based on the above principals, we use the GPS location data of the taxi trace to simulate the user's mobility.

We select the data records in 15 March, 2007 of roughly 200 taxies. We deceive the movement track of each car by used GPS records at each moment. We simulate 20 time slots for our system, and set the length of time frame to 10 minutes.

2) MEC Details: The total simulation area is almost 5×5 km^2. We divide the whole area into 5×5 square cell grids. Each cell grid occupies 1 km^2, endowed with one MEC node to provide mobile services. We consider that the capacity of AP in each MEC should be slightly larger than the maximum requirement for the resource in the system. As for the edge cloud in each MEC, we simply set the total capacity as 2 times of the service amount at all time slots. The capacity will be distributed to all the edge clouds proportionally to the frequency of users being attached to each MEC. The communication delay between two MEC in our model is measured by the geographical distance between any two entities based on their GPS locations.

3) Users Details: We assume that each user has an independent resource demand of access point. Capricious requirements will put different pressure on the queuing delay of network access points. We randomly set the requirement for each user within the AP capacity of each MEC. Considering that the service's switching delay is static over time and varies among different edge clouds, we generate these switching delays by following a Gauss distribution with the negative tail cutted.

4) Runtime Environment: We build our algorithm in Python 3.5 to validate the performance of our online algorithm. All the measurements are conducted on a Linux Server equipped with E5-2620 v4 @ 2.10GHz core and 64GB RAM. We use CVXPY [27] equipped with Gurobi to model and solve the convex optimization in our problem.

B. Performance Benchmark

We carry out experiments with the above setting. To further understand the impact of our online algorithm on different types of delay, we compare the results of our algorithm with two algorithms, which only consider the following static delay:

- The queuing-opt (QO) algorithm minimizes the queuing delay for all users in network access. This algorithm chooses service placement as close to the user as possible. The purpose of this benchmark is to evaluate the improvement of our algorithm for the subproblem of service placement.
- The switching-opt (SO) algorithm minimizes the switching delay and communication delay as objective, while ignoring the queuing delay in the network access. Here we use random method for network selection in SO. This benchmark is adopted to evaluate the improvements that our algorithm has achieved after jointly optimizing some of the network choices.

C. Effectiveness of OLSA

We will compare our algorithm with benchmarks in the following aspects to analyze the effectiveness of our algorithm. First we will measure the total delay that all users experienced in the system, and secondly we analyze the average delay for a single user to acquire service in the system. Meanwhile, in order to obtain the solution efficiently, we evaluate the number of iterations of the algorithm in the operation process. In addition, we study the total delay of our online algorithm in each time slot.

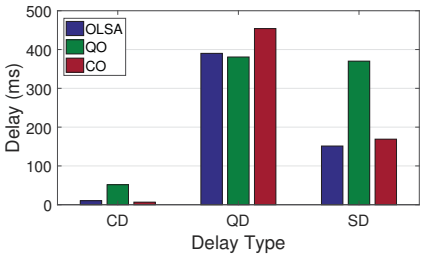

Fig. 4. Total delay of *OLSA* and benchmarks.

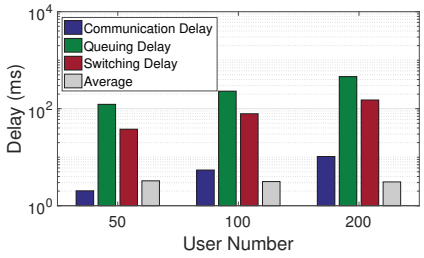

Fig. 5. Delay user-perceived with different service pressure.

1) Total latency analysis: For a MEC, the sum of all types of delays in a period of time for all users is an important indicator for evaluating the performance of the entire system. Here we compare our algorithm to the total delay of the two benchmarks. As shown in Fig. 4, CD, QD, and DS are abbreviated forms of communication delay, queuing delay, and switching delay, respectively. It can be observed that *OLSA* has advantages over the other two benchmarks. In particular, we compare the queuing delay in the *OLSA* algorithm with the QO benchmark and find that our algorithm can obtain the effectiveness of competing with the optimal case. The comparison of communication delay between *OLSA* and CO benchmark also proves the improvement of our algorithm.

2) Delay user-perceived with different service pressure of OLSA: Unlike the total system delay, user average latency is a key metric used to measure the quality of service for each user. Under the pressure of different service demands, the average delay that each user obtain can be used to measure the robustness of the algorithm. In the experiment, we evaluate the average delay of each user under the different pressure of service demands by setting varieties number of people. As shown in Fig. 5, under different people pressure, *OLSA* can obtain a stable delay.

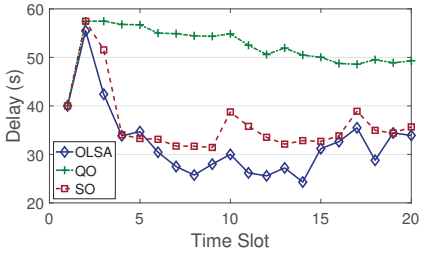

Fig. 6. Total delay in each time slot.

TABLE II
ITERATION TIMES OF *OLSA*.

Iteration times	0	1	2
Time slot num	5	8	7

3) Iterations of OLSA: Since we adapt an iterative algorithm to solve the subproblem, measuring the complexity of our algorithm is the number of iterations that go through a satisfactory solution. Table. II shows the number of iterations for solving subproblems in different time slot. We can observe that the maximum number of iterations is two times, which is about a third of the total time slots, and the number of iterations one times and zero times each accounts for a third. It can be seen that *OLSA* computes the solution that meets the requirements in an extremely limited number of steps. This verifies the high efficiency of our algorithm.

4) Total delay in each time slot: To evaluate the performance of our algorithm at each time period, we analyze by plotting the total delay of the *OLSA* and the two benchmarks in each time period. As we can see from the Fig. 6, the QO benchmark has a large delay at each moment, because it does not optimize service placement, and more service switching delays occur. In addition, we can find that the *OLSA* algorithm can achieve better performance at most times.

VII. CONCLUSION

In this paper, we study the problem of joint optimization on the access network selection and service placement for MEC, towards the goal of improving the QoS by balancing the access delay, communication delay and switching delay. We first demonstrate that, the commonly adopted approach of service placement does not guarantee the QoS of users, because both the access network and edge nodes are vulnerable to congestion. Then we formulate the joint optimization problem as a long-term problem. We design an efficient online framework, which decomposes the long-term optimization problem into a series of one-shot problems. To address the NP-hardness of these one-shot problems, we divide each one-shot problem into two sub-problems, and provide an iteration-based algorithm to get an optimal solution. Both rigorous theoretical analysis on the optimality gap and extensive trace-driven simulations demonstrate the efficacy of our proposed solution.

REFERENCES

[1] N. Alliance, "5G white paper," *Next generation mobile networks, white paper*, pp. 1–125, 2015.

[2] F. Liu, P. Shu, H. Jin, L. Ding, J. Yu, D. Niu, and B. Li, "Gearing resource-poor mobile devices with powerful clouds: architectures, challenges, and applications," *IEEE Wireless communications*, vol. 20, no. 3, pp. 14–22, 2013.

[3] Y. C. Hu, M. Patel, D. Sabella, N. Sprecher, and V. Young, "Mobile edge computinga key technology towards 5G," *ETSI white paper*, vol. 11, no. 11, pp. 1–16, 2015.

[4] Y. Mao, C. You, J. Zhang, K. Huang, and K. B. Letaief, "A survey on mobile edge computing: The communication perspective," *IEEE Communications Surveys & Tutorials*, vol. 19, no. 4, pp. 2322–2358, 2017.

[5] T. Taleb, S. Dutta, A. Ksentini, M. Iqbal, and H. Flinck, "Mobile edge computing potential in making cities smarter," *IEEE Communications Magazine*, vol. 55, no. 3, pp. 38–43, 2017.

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

[7] L. Wang, L. Jiao, T. He, J. Li, and M. Mühlhäuser, "Service entity placement for social virtual reality applications in edge computing," in *Proc. of INFOCOM*, 2018.

[8] 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, 2017.

[9] F. Xu, F. Liu, H. Jin, and A. V. Vasilakos, "Managing performance overhead of virtual machines in cloud computing: A survey, state of the art, and future directions," *Proceedings of the IEEE*, vol. 102, no. 1, pp. 11–31, 2014.

[10] J. T. Piao and J. Yan, "A network-aware virtual machine placement and migration approach in cloud computing," in *Proc. of IEEE Grid and Cooperative Computing*, 2010, pp. 87–92.

[11] F. Liu, P. Shu, and J. C. Lui, "Appatp: An energy conserving adaptive mobile-cloud transmission protocol," *IEEE transactions on computers*, vol. 64, no. 11, pp. 3051–3063, 2015.

[12] A. Ksentini, T. Taleb, and M. Chen, "A Markov decision process-based service migration procedure for follow me cloud," in *Proc. of IEEE ICC*, 2014, pp. 1350–1354.

[13] S. Wang, R. Urgaonkar, M. Zafer, T. He, K. Chan, and K. K. Leung, "Dynamic service migration in mobile edge-clouds," in *Proc. of IFIP/IEEE Networking Conference*, 2015, pp. 1–9.

[14] K. R. Alasmari, R. C. Green, and M. Alam, "Mobile edge offloading using markov decision processes," in *International Conference on Edge Computing*. Springer, 2018, pp. 80–90.

[15] M. Srivatsa, R. Ganti, J. Wang, and V. Kolar, "Map matching: Facts and myths," in *Proc. of ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems*, 2013, pp. 484–487.

[16] A. J. Nicholson, Y. Chawathe, M. Y. Chen, B. D. Noble, and D. Wetherall, "Improved access point selection," in *Proc. of ACM International Conference on Mobile Systems, Applications and Services*, 2006, pp. 233–245.

[17] Y. Sun, S. Zhou, and J. Xu, "Emm: Energy-aware mobility management for mobile edge computing in ultra dense networks," *IEEE Journal on Selected Areas in Communications*, vol. 35, no. 11, pp. 2637–2646, 2017.

[18] R. Roman, J. Lopez, and M. Mambo, "Mobile edge computing, fog et al.: A survey and analysis of security threats and challenges," *Future Generation Computer Systems*, vol. 78, pp. 680–698, 2018.

[19] Y. T. Hou, Y. Shi, and H. D. Sherali, "Optimal base station selection for anycast routing in wireless sensor networks," *IEEE Transactions on Vehicular Technology*, vol. 55, no. 3, pp. 813–821, 2006.

[20] J. Wu, E. W. Wong, Y.-C. Chan, and M. Zukerman, "Energy efficiency-qos tradeoff in cellular networks with base-station sleeping," in *Proc. of IEEE Global Communications Conference*, 2017, pp. 1–7.

[21] L. Zhang, C. Wu, Z. Li, C. Guo, M. Chen, and F. C. Lau, "Moving big data to the cloud: An online cost-minimizing approach," *IEEE Journal on Selected Areas in Communications*, vol. 31, no. 12, pp. 2710–2721, 2013.

[22] G. Inc, "Gurobi optimizer," *URL: http://www. gurobi. com*, 2018.

[23] P. C. Chu and J. E. Beasley, "A genetic algorithm for the generalised assignment problem," *Computers & Operations Research*, vol. 24, no. 1, pp. 17–23, 1997.

[24] Shanghai taxi trajectory traces. [Online]. Available: http://wirelesslab.sjtu.edu.cn/taxi_trace_data.html

[25] X. Li, G. Pan, Z. Wu, G. Qi, S. Li, D. Zhang, W. Zhang, and Z. Wang, "Prediction of urban human mobility using large-scale taxi traces and its applications," *Frontiers of Computer Science*, vol. 6, no. 1, pp. 111–121, 2012.

[26] F. Xu, Y. Li, H. Wang, P. Zhang, and D. Jin, "Understanding mobile traffic patterns of large scale cellular towers in urban environment," *IEEE/ACM transactions on networking*, vol. 25, no. 2, pp. 1147–1161, 2017.

[27] S. Diamond and S. Boyd, "CVXPY: A python-embedded modeling language for convex optimization," *The Journal of Machine Learning Research*, vol. 17, no. 1, pp. 2909–2913, 2016.