An Online Framework for Joint Network Selection and Service Placement in Mobile Edge Computing

Bin Gao, Zhi Zhou, *Member, IEEE,* Fangming Liu*, *Senior Member, IEEE,* Fei Xu, *Member, IEEE,* and Bo Li, *Fellow, IEEE*

Abstract—With the rapid development and deployment of 5G wireless technology, Mobile Edge Computing (MEC) has emerged as a new computing paradigm to facilitate a large variety of infrastructures at the network edge to reduce user-perceived communication delay. One of the fundamental problems in this new paradigm is to preserve satisfactory quality-of-service (QoS) for mobile users in light of densely dispersed wireless communication environment and often capacity-constrained MEC nodes. Such user-perceived QoS, typically in terms of the end-to-end delay, is highly vulnerable to both access network bottleneck and communication delay. Previous works have primarily focused on optimizing the communication delay through dynamic service placement, while ignoring the critical effect of access network selection on the access delay. In this work, we study the problem of jointly optimizing the access network selection and service placement for MEC, with the objective of improving the QoS in a cost-efficient manner by judiciously balancing the access delay, communication delay, and service switching cost. Specifically, we propose an efficient online framework to decompose a long-term time-varying optimization problem into a series of one-shot subproblems. To address the NP-hardness of the one-shot problem, we design a computationally-efficient two-phase algorithm based on matching and game theory, which achieves a near-optimal solution. Both rigorous theoretical analysis on the optimality gap and extensive trace-driven simulations are conducted to validate the efficacy of our proposed solution.

Index Terms—Mobile Edge Computing, Network Selection, Service Placement, Online Algorithm, Stable Matching, Game Theory.

1 Introduction

With the continuous development of wireless communications and the explosive growth of mobile devices over the recent years, our daily life is increasingly exposed to a plethora of mobile applications as exemplified by online social network, mobile game and instant message. The *de-facto* wisdom to host these diverse applications is deploying services on the centralized datacenters [1]. However, due to the inevitable long distance between end-users and service-hosting clouds, the centralized cloud computing paradigm cannot meet the stringent timeliness requirement of the emerging delay-sensitive applications such as virtual/augmented reality (VR/AR) [1], industrial internet-of-things (IIoT), and connected cars [2].

Mobile Edge Computing (MEC) [3] is proposed as a promising technique to fulfill the low-latency requirement of the aforementioned applications. The key idea beneath the MEC technique is to push storage and computation resources from the network core to network edges, which are in closer proximity to users [4]. MEC node is typically built-up in a micro datacenter and collocated with an access point (AP) (*e.g.*, a base stations or a WiFi hotspot) [2]. Users' requests can be served on top of a nearby MEC node and hence the user-perceived delay is drastically reduced due to the greatly shorten network distance [5]. It's highly acknowledged that, MEC is a backbone building block of the morning 5G.

However, restricted by limited resource capacity, single MEC cannot host diverse applications *en masse* and therefore the services should be carefully placed across the MEC nodes. In general, a MEC system divides each user's service application into two parts: a client entry (CE) and a service entry (SE) [6], [7]. The CE runs on the side of client device and communicates with the SE running on the edge cloud to get data or computing services. A natural problem is how to distribute and place SEs to provide users with satisfactory Quality-of-Service (QoS), while achieving the economic efficiency for the system operators.

- *Bin Gao and Fangming Liu are with the National Engineering Research Center for Big Data Technology and System, the Services Computing Technology and System Lab, Cluster and Grid Computing Lab in the School of Computer Science and Technology, Huazhong University of Science and Technology, 1037 Luoyu Road, Wuhan 430074, China. E-mail: {bingao.cs@gmail.com, fmliu@hust.edu.cn}. The corresponding author is Fangming Liu.*
- *Zhi Zhou is with School of Computer Science and Engineering, Sun Yat-sen University, Guangzhou 510006, China. E-mail: zhouzhi9@mail.sysu.edu.cn.*
- *Fei Xu is with School of Computer Science and Technology, Shanghai Key Laboratory of Multidimensional Information Processing, East China Normal University, 3663 N. Zhongshan Road, Shanghai, China. E-mail: kudofaye@gmail.com.*
- *Bo Li is with the Department of Computer Science and Engineering, Hong Kong University of Science and Technology.*

This work was supported in part by NSFC under Grant 61722206 and 61761136014 (and 392046569 of NSFC-DFG) and 61520106005 and 61972158, in part by the National Key Research & Development (R&D) Plan under grant 2017YFB1001703, in part by the Fundamental Research Funds for the Central Universities under Grant 2017KFKJXX009 and 3004210116, in part by the National Program for Support of Top-notch Young Professionals in National Program for Special Support of Eminent Professionals. The research was support in part by RGC RIF grant R6021-20, and RGC GRF grants under the contracts 16207818 and 16209120.

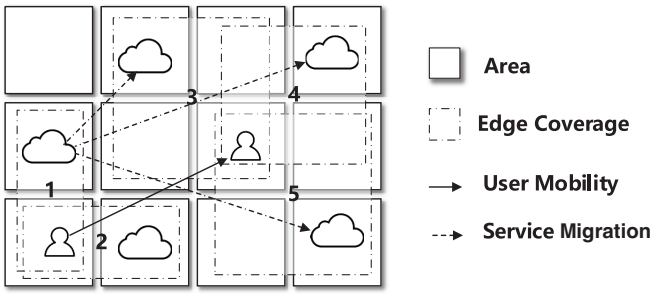

Fig. 1: An illustration of dynamic service migration by following user mobility.

An essential issue complicating the above MEC service placement is that the services should be dynamically migrated across the MEC nodes [8], in order to maintain the QoS perceived by users which often moves across the coverage of various MEC nodes. As an example shown in Fig. 1, we assume that in the first time slot the user is in the lower-left corner area, and in the second time slot the user moves from Area A to G. Before the movement occurs, the user is within the coverage of Edge node 1 and Edge Cloud 2, and the user's SE is running on Edge Cloud 1. When the user moves to Area G, the coverage of his nearby edge cloud is changed to Edge Cloud 3, 4, 5. In the second time slot when the movement occurs, the MEC system operator needs to maintain the perceived QoS of the mobile user: (1) *whether* to migrate the service according to user movement; (2) if adopting service migration, *when* to migrate; (3) if adopting service migration, *where* (*e.g.*, Edge Cloud 3, 4, 5) to migrate.

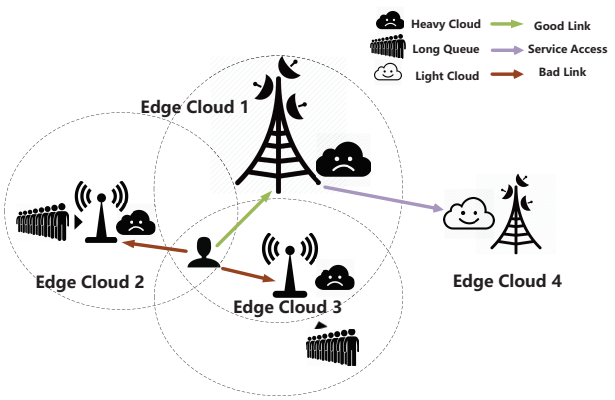

Fig. 2: An illustration of joint network selection and service placement.

However, previous works [9]–[12] in answering the above questions have focused on mobility-aware dynamic service placement, while ignoring the critical effect of the access network congestion on the QoS [13], [14]. To illustrate the importance of access network selection for user-perceived QoS, we consider a typical MEC scenario with four edge clouds as shown in Fig. 2. Each edge cloud is composed of a network access point and a colocated computing cloud. The network access points at Edge Cloud 2 and Edge Cloud 3 have user queues that buffer congested user requests awaiting to be served, which results in long queuing delay for users connecting to these two edge clouds. At this time, if a new user under the coverage of Edge Cloud 1,2 and 3 want to arouse a service entry and select a network point to access the service. Under current situation, the Edge Cloud 1, 2 and 3 covering the new user are faced with an excessive amount of SEs belonging to the other users, resulting in prolonged queuing delay. In this case, a better alternative is to forward the user request from the Edge Cloud 1 to the Edge Cloud 4 with sufficient available computation resources. At this point, we can choose to place the SE of the considered user on Edge Cloud 4 that is out of the user's coverage, but this procedure will suffer additional communication overhead between Edge Cloud 1 and 4. While this approach exacerbates the communication delay incurred by the increased network distance, the access delay can be significantly reduced. Hence, to improve the QoS of a MEC system, it's essential for the MEC system operators to jointly optimize the network selection and service placement strategy for users, so as 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 delay caused by users continuously connecting to poor networks if some users don't move frequently. Although they don't change their locations over a long period of time, the number of users connected to their nearby APs is constantly changing. If the dynamic selection of the access network is not performed, these users who don't move frequently may always connect to the network access point with poor network conditions, and thus they will obtain unacceptable QoS of applications.

Our main contributions are summarized as follows. First, we jointly consider the access network selection and service placement problem in MEC. We take into account the access delay, switching cost and communication delay to improve the QoS for MEC applications. Specifically, the communication delay in this work is incurred when users access services placed in indirectly connected edge clouds. The access delay and switching cost are incurred by the queuing delay for AP and the dynamic migration of services, respectively. As the formulated long-term optimization problem is stochastic by nature, i.e., containing future uncertain information such as user mobility, we focus on designing an efficient online algorithm, which relies on the zero information of the future, in order to accommodate the random and unpredictable mobility of the users.

Compared with our previous version [15], this version makes the following differences. Firstly, we introduce the new configuration setting which is a pair of AP and edge cloud. With the configuration pair, the network selection variables are loosely coupled with the service placement variables, hence avoiding the curse of dimensionality in our original model [15]. After decomposing the long-term problem into a set of one-shot subproblems, secondly, we propose a two-phase approximation algorithm with provable performance, by applying matching and game theory. One of the difficulties of our sub-problem is that the perceived delay of a mobile user at the access point depends on not only its own request but also the overall load of the connected access point, which makes it infeasible to directly apply the stable matching technique. To tackle this challenge, we transform the original problem into a

college admission problem, which is a classical problem in matching theory. We then introduce two roles of objects, which we call configuration pair and user, to make an analogy to the concept of "college" and "student" in the college admission problem respectively. We define the user's preference list over configuration pair based on the worst response time that the configuration pair can achieve, and configuration pair's preference list based on the resource demands required by the users. With these definitions, we obtain a stable matching assignment that guarantees the worst response time for users. Thirdly, we avoid from unbalancing situations which probably incurred by stable matching. We use coalitional game theory to further balance the matching result between our two roles and improve overall response time of the whole MEC system. Finally, through the theoretical analysis of our algorithm, we can strictly derive the competition ratio compared with the offline optimum. Extensive simulations based on the realistic trace and setup further demonstrate the effectiveness of our algorithm.

The rest of this paper is organized as follows. We discuss related work in Sec. 2. In Sec. 3, we introduce our system model and problem formulation. In Sec. 4, we present our online optimization algorithm. In Sec. 5, we analyze our online algorithm to derive a competitive ratio and prove nice properties of our two-phase algorithm. In Sec. 6, we show how our online optimization framework can be plugged into existing MEC architectures. The online framework is evaluated in Sec. 7 and we conclude this paper in Sec. 8.

2 Related Work

The problem of service placement is a common topic, and has been extensively researched in cloud computing. [16]–[18]. Different from it in cloud computing, the problem of service placement in MEC is associated with the characteristic of users' movement. To address the random user mobility, there are roughly three kinds of service placement strategies.

The first is the follow me at the edge (FME) strategy, which dynamically migrates services from current edge servers to the nearby one as the users move. FME strategy typically makes service placement and service migration decisions based on the system geography information or workloads, such as users' spatial locations and resource demands. A basic concept of FME consists of a system operator, who controls a series of geographically distributed edge servers. This operator makes service scheduling decisions for all the mobile users under his control according to service types, QoS demands, task data sizes and so on. The FME policy aims at striking a balance between QoS and switching cost [9], [10].

The second strategy is modeling the service migration problem using Markov decision process (MDP) [10], [11]. MDP takes into consideration the cost and benefit of service migration, and it builds the best policy to decide whether to migrate a service or not. MDP based service migration contains two types of strategies, including one-dimensional and two-dimensional MDP. One-dimensional MDP is first applied in [10] and [19]. One-dimensional MDP assumes that mobile users move following a straight line (e.g., a car driving along the road) [20], [21], which is far from reality and cannot be applied well into practice. Two-dimensional MDP [9], [22] is a more general case, which models users' movements in two-dimensional space, e.g., a plaza. Users have more than two destinations to make a forward movement. However, these works [6], [23] 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. In particular, when an access point is overloaded [24], it will cause network congestion, which will become a major overhead part of the communication with the service.

Another method widely adopted in service migration is time window control mechanism. Time window based service migration defines a look ahead window as a time period that can be predicted in the future. This given time window is used to find the optimal service placement and minimize the average cost [25], [26]. Compared with MDP based service migration, this model can handle complex cost function, heterogeneous network topology and mobile user movement pattern. In addition, time window based service migration doesn't require any probability distribution of the user mobility and can be applied to more realistic scenarios [25]. However, existing time window control mechanisms [25], [26] don't optimize non-service related costs, such as queuing delay incurred when users access network.

Apart from dynamic service migration, there are also some research works optimizing the QoS by the other ways. The work [27] proposes a new human-driven edge computing pattern that helps extend the scope of the traditional MEC. They utilize the computing and communication capabilities of devices on mobile entities as external MEC nodes to improve the service quality of mobile applications. The work [28] takes service placement and request routing into consideration together, and proposes a close-to-optimal algorithm to solve the joint problem by random rounding. However, the work [28] ignores the characteristic of mobility widely existing in mobile edge computing scenario. The work [29] exploits the historical queue information as well as the energy consuming information to reduce the users' experienced delay by leveraging Lyapunov optimization framework. The work [30] considers the effect of energy budget on service placement. The work [30] proposes an online algorithm to improve the quality of service while satisfying the limited energy budget constraints. The work [31] targets to minimize the computation delay and energy consumption of mobile user in mobile edge computing, which neglects the mobility characteristic and communication delay between devices and service entries.

A closely related work to ours is [12]. Without any user mobility as priory knowledge, the work [12] 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 with [12], our work has the following advantages and improvements: (1) Our work considers the nonlinear network access delay, switching cost, and communication delay to minimize overall delay. (2) the work [12] 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

TABLE 1: 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 j 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 cost 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 service resource demand of user k

edge network system. (3) We design an online algorithm to reduce frequent switching cost and balance the access delay and communication delay.

3 System Model and Problem Formulation

3.1 System Model

We consider a MEC system which consists of M access points (e.g., 4G, 5G or WiFi.) and N users. Each AP is equipped with a computing node (e.g., a server or a server cluster) which can be regarded as edge clouds. We denote the set of APs/edge 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 adopts light weight virtualization solutions (e.g., VM or Container.) [32] 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 maximum number of services that can be hosted in MEC node i. At each time slot t, mobile user k is covered by a set of nearby edge clouds via neighboring APs [33], denoted by $\phi_k(t)$. Key parameters and notations are listed in Table 1 for ease of reference.

3.2 Access Point Selection Model

At each time slot t, the system operator makes AP selection decisions for all mobile 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. 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 resource demand $r_k(t)$ for a given user k, each AP cannot exceed the resource capacity limitation:

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

3.3 Service Placement Model

As mentioned above, each user accesses edge clouds via a neighboring AP. For each user k, a service entry is offloaded to one of the edge clouds i to provide service for user k. Specially, there is no necessary relevancy between the access point selection and service placement for a specific user. The service of user k can be placed on any edge cloud $i \in \mathcal{M}$, but a user k only can select AP $j \in \phi_k(t)$ to access edge clouds at time slot t. Since users have limited communication distance in a MEC system, each 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 follows:

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

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

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

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

3.4 QoS Model

3.4.1 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. To analyze the delay performance of users, we model each access point as an M/M/1 queue [34]. 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 and $\sum_{k \in \mathcal{N}} r_k(t) y_{jk}(t)$ is the overall loads connected to AP j. 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 Expression (7) makes sense, we will guarantee that the resource capacity C_j is greater than the demand resource $\sum_{k \in \mathcal{N}} r_k(t) y_{jk}(t)$ at any time.

3.4.2 Communication Delay

In our model, the service placement is independent, and the service doesn't 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 MEC system. 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. Combining the AP selection decision y_{jk} with service placement decision x_{ik}, the overall corresponding communication delay at time slot t in the system can be expressed as:

$$D^c(\vec{x}(t),\vec{y}(t)) = \sum_{k\in\mathcal{N}}\sum_{i\in\mathcal{M}}\sum_{j\in\phi_k(t)} y_{jk}(t)x_{ik}(t)l_{ij}(t). \quad (8)$$

3.4.3 Switching Cost

In the paradigm of MEC, user's mobility is inevitable. When users roam around the mobile edge clouds, keeping the placement of their services invariant would greatly deteriorate the perceived delay of users. It behooves to optimize the user experience via dynamically re-placing or migrating the services. Toggling service across mobile edge clouds incurs the switching cost due to the initialization time of booting new software resources, loading profiles, migrating states, and so on. The total switching cost of all users between time slot $t-1$ and time slot t is given:

$$D^s(\vec{x}(t),\vec{x}(t-1)) = \sum_{k\in\mathcal{N}}\sum_{i\in\mathcal{M}} \sigma[x_{ik}(t)-x_{ik}(t-1)]^+, \quad (9)$$

where $[x_{ik}(t)-x_{ik}(t-1)]^+ = \max\{x_{ik}(t)-x_{ik}(t-1),0\}$. The operator may set the switching cost σ according to the energy price of edge cloud [35], [36], container state migration costs [37].

3.5 Problem Formulation

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

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

3.6 Challenges

In the above subsections, we have formulated the joint optimization problem of network selection and service placement. But to solve this joint problem, we still face several challenges.

First of all, in a long-term period, the problem requires the future system information (i.e., the user mobility pattern), so that MEC system operator can make the global optimal decisions across time slots for users to achieve better performance both in network selection and service placement. Unfortunately, it's difficult to precisely predict the system information from beginning to end beforehand.

Secondly, the partial objective function $D^c(\vec{x}(t),\vec{y}(t))$ is a coupling term that depends on both the network selection variable $\vec{x}$ and the service placement variable $\vec{y}$. This coupling term creates difficulties for the problem and we can't just achieve optimal state by only optimizing network selection or service placement. This coupling term also illustrates the need for joint optimization of network selection and service placement. Only by optimizing the relationship between these two can the objective function be optimized overall.

Thirdly, by degradation, our problem can be proved as a complicated variant of generalized assignment problem [38] which is NP-hard at least. Proof details can be found in appendix. In addition, considering that our problem is generally large-scale existence in reality, the algorithm should be computationally efficient to obtain an outcome.

To overcome the above challenges, we design an online algorithm to provide a long-term optimization without requiring any future information. Then, to solve the subproblems at each time slot in an online manner, we decouple the coupling variables by transforming the problem with newly introduced configuration pair notion. By resorting to the methods of stable matching and cooperative game, we design an efficient algorithm to tackle the subproblem with which is NP-hard difficulty.

4 An Online Optimization Framework

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

4.1 Problem Decomposition via Online Lazy Switching

In a long-term issue, future system information (i.e., the user mobility pattern) is required, 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's difficult to get the system information in the whole time period beforehand. To address this, we can firstly divide the long-term problem (10) into T one-shot optimization problems:

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

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

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 at each time slot, but this comes with the expense of frequent migration cost. 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 migration. This inspires us to carefully consider when to migrate the service

placement and which network access point to choose for the user. The key idea of our online algorithm is to tolerate as much as non-switching delay (e.g., queuing delay and communication delay) as possible until it significantly exceeds the switching cost. Inspired by the work [39], 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 (9) 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:

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

Algorithm 1: The Online Lazy Switching Algorithm (*OLSA*)

```
t = 1;
t̂ = 1; // Time cursor for the last service migration occurred;
Init AP selection decision vector x(1) and service placement decision vector y(1) by Alg. 2 and Alg. 3;
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 Alg. 2 and 3;
        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) isn't derived, compute it by Alg. 2 and 3;
    end if
    t = t + 1;
end while
```

In the algorithm, we initialize the two decision vectors x and y by assigning $T = 1$. At the beginning time slot $T = 1$, we first solve the one-shot optimization problem (11) by Alg. 2 and Alg. 3. After that, we obtain the original switching cost and non-switching delay deceived by above decision variables. Then, at each time slot t, *OLSA* chooses the proper access point and service placement for every user according to the following strategies. Let $\hat{t}$ denote the last service migration moment over the past time. *OLSA* firstly computes overall non-switching 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 non-switching delay is at least β times than the switching cost $D^s(\vec{y}(\hat{t}), \vec{y}(\hat{t}-1))$ or not. If the condition holds, *OLSA* obtains the decision vector $\vec{x}(t)$ and $\vec{y}(t)$ by Alg. 2. 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 at the same place without any migration operation ($x(t) = x(t-1)$).

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

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. In the following subsection, we show how to solve non-switching subproblem (12). We design a two stage algorithm to obtain a near optimal solution. In the first stage, we transform the problem to a stable matching problem which can be efficiently solved by deferred acceptance algorithm [40], [41]. In the second stage, we use the results obtained from the first stage as an input and apply coalitional game theory to further improve the quality of the result.

4.2 A Primer on Configuration Pair

Before we get into the two phase algorithm, we first introduce a conceptual configuration setting, which helps ease the coupled variables in the original subproblem (12).

Considering that at a single time slot, a specific mobile user definitely chooses a nearby network and a MEC cloud to constitute a network-service group, it's a natural way to determine a (network, MEC cloud) group at each decision cycle. Inspired by this point, we subtly orchestrate a configuration pair which is composed of an access point and an MEC cloud. Each configuration pair is assigned with a communication delay d_{ij} which is hinged on the value l_{ij} of formula (8). By importing such conception, the variables which are coupled by each other in the formula (8) can be simply decoupled. Moreover, rather than having to make choices in the three sets $\mathcal{N}$, $\mathcal{M}$, $\phi_k(t)$ before, now the search space is degraded, and we only need to find solutions in the user set and the configuration pair set, which greatly degrades the complexity of the problem.

The configuration pair set, denoted as $\mathcal{C}$, contains $N * N$ elements in total, which is related to the size of the MEC set $\mathcal{N}$. Each element in the set $\mathcal{C}$ represents an available resource group (e.g., access point and MEC cloud). For the sake of simplicity, we use C equal to $N * N$ to indicate the length of the configuration set $\mathcal{C}$.

The matchup between users and configuration pairs at time t is denoted as a binary $M \times C$ matrix $\mathcal{Z}(t)$. Each binary decision $z_{ij}(t) \in \{0, 1\}$ in $\mathcal{Z}(t)$ satisfies:

$$z_{ij}(t) = \begin{cases} 1 & i^{th} \text{ user is connected to } j^{th} \text{ configuration pair;} \\ 0 & \text{otherwise.} \end{cases} \tag{14}$$

4.3 The Stable Matching Phase

Our non-switching subproblem can be considered as a variant of stable marriage problem (SMP) [42], which is known as college admissions problem (CAP). Different from traditional one-to-one matching, the CAP is devoted to many-to-one matching up, that is, one specific college is open to all applicants and can admit a group of more than one student. A student at one time is allowed to show diverse intention of more than one school with a preferred order, but one college can only be accepted as the final sole choice. Applying the analogy to our problem, each configuration pair and mobile user play the role of college and student in the CAP, respectively. In every decision round, each configuration pair provides sufficient resources to multi-users to access the network and deploy service. Each user only can choose one configuration pair to obtain service ability.

In general, there exist two sided sets in the stable matching problem, in which both sides have preferences over the other. In our stable matching phase, let $\mathcal{C}$ and $\mathcal{S}$ denote sets of configuration pairs (acting as colleges) with different capacities to serve network-service group and mobile users (acting as students) with different service demands respectively. Assume that $|\mathcal{C}| = C$ and $|\mathcal{S}| = N$.

Definition 1. A matching μ is a function from the set $\mathcal{C} \cup \mathcal{S}$ to the set $\mathcal{C} \cup \mathcal{S}$ such that each student is assigned to exactly one college and colleges are open to match more than one preferred students. That is, for each $s \in \mathcal{S}$ and $c \in \mathcal{C}$, $\mu(s) = c$ only exists but there are several elements $s1, s2, ..., sn \in \mathcal{S}$ that may compose the matching $\mu(c) = \{s1, s2, ...sn\}$.

According to the core concept of the stable matching [40], we firstly need to subtly define preferences over the set $\mathcal{C}$ and $\mathcal{S}$ and blocking pairs, and then our problem can be seen as a many-to-one stable matching problem. Here we take $a \succ_c b$ to indicate that c prefers a to b. However, our problem cannot apply the method directly. Different from preferences of students and colleges in, the object in our problem, users and configuration pair, have their own special interest. It's necessary to find out the desired point of each role in our problem, and then create the preference list following the objects' desired objective.

First, we take a look at each user's objective. Users in a mobile edge computing system always seek configuration pair to handle their mobile services. From the perspective of a mobile user, he eagerly yearns for a configuration pair that provides low service access delay. The delay in our problem mainly contains two components including queuing delay and communication delay. From the above point, one user i can build his preference list by computing the following definition:

$$\frac{1}{C_j - \theta_j(t)} + d_{ij}(t). \tag{15}$$

where $\theta_j(t)$ is the overall loads of j-th configuration pair and d_{ij} is the communication delay that user i selects j-th configuration pair.

However, there still exist some difficulties when employing the above equation. Most of previous works on many-to-one stable matching depend on the premise that the preferences are static and independent of the others members. However, in our problem, the perceived delay of the user i in expression (15) subjects to not only the requests of user i but also the loads of the network in the configuration pair, which makes it infeasible to directly apply the original stable matching method. To tackle this technical challenge, we define users' preference list according to the worst delay that the selected configuration pair can provide:

$$\nu_{ij}^{max}(t) = \frac{1}{C_j - \beta * C_j} + d_{ij}(t). \tag{16}$$

where β is a factor that indicates the maximum loads the configuration can provide.

After getting the aim of the mobile users, we can define users' preference list as follows.

Definition 2. User's preference list over configuration pair. The preference list of the i^{th} user s_i is $\Gamma(s_i) = \{c_{j*}, ..., \}$ which contains configurations whose capacity is at least equal to i's requested resource. The elements in preferred list $\Gamma(s_i)$ are sorted in the ascending order of the worst delay that a configuration provides according to definition (16).

For configuration pairs, they are more willing to provide services as many as possible. Hence configuration pairs desire the users with smaller resource requirements, who will not cause the overloads of the configuration. Hence, we define the configurations' preferred list according to the resource demand requests from users as below.

Definition 3. Configuration's preference list over users. The preferred list of the j^{th} configuration c_j is $\Gamma(c_j) = \{s_{i*}, ..., \}$, which contains the users whose resource demands don't exceed the configuration's capacity. The elements in the preferred list $\Gamma(c_j)$ are ranked in an ascending order of the resource demand request volume.

Definition 4. Blocking pair. In a matching Θ, a user-configuration pair (s_i, c_j) is a blocking pair if it satisfies any of the following two conditions:

1) $c_j \succ_{s_i} \Theta(s_i)$, and $k_j(t) + r_i(t) \leq E_j$;
2) $k_j(t) + r_i(t) - \sum_{i*} r_{k*}(t) \leq E_j$, where $s_i \succ_{c_j} \Theta(s_{i*})$ and $\Theta(s_{i*}) = c_j$.

According to the definition of blocking pair, we define the stable matching as follows:

Definition 5. Stable Matching. A matching Θ is said to be stable if there doesn't exist any blocking pairs.

Getting all the definitions, the detailed algorithm to obtain a stable matching for our problem is given in Alg. 2.

The procedures of the first stable matching phase is described in Alg. 2. At the beginning, the Algorithm 2 computes the preference lists for each mobile user and configuration pair. Then each mobile user selects its most preferred configuration pair from its preference list as proposals to require a matching. If all of the proposals can be handled without violating the capacity constraint, the configuration pair accepts all the proposals. Otherwise, the configuration pair only holds the most preferred proposals. All above procedures continue until there is no proposal to be proposed.

4.4 The Coalitional Game Phase

Stable matching phase is efficient to obtain a feasible solution for our problem, but it probably results in a unbalanced matching. For instance, three mobile users A, B, C

Algorithm 2: Stable Matching Phase Procedure

Input:
Resource demand of each mobile user in time slot t, $r_i(t)$,$\forall i$;
The capacity of the configuration pair, E_j, $\forall j$;
Output:
Mapping between mobile users and configuration pair, $z_{ij}(t)$,$\forall i, j$;

function StableMatching (resource demand $\forall, i, r_i(t)$, configuration pair capacity $\forall j, E_j$);
Each mobile user and configuration pair computes its own preference list in the Definition (1) and (2) as $\forall i, \Gamma(s_i) = \{c_{j*}, ..., \}$, $\forall j, \Gamma(c_j) = \{s_{i*}, ..., \}$;
while Mobile users still have proposals **do**
  Each mobile user proposes to its most preferred configuration according to its preference list;
  if All of the proposals don't exceed the capacity constraint in Constraint. (3) and (5) **then**
    The configuration pair temporarily holds all the proposals;
  else
    According to configuration pair's preference list, the configuration pair holds the most preferred proposals that will not violate the capacity constraint;
    The configuration pair rejects the other unacceptable proposals;
  end if
end while
Transform the matching Θ to $z_{ij}, \forall i, j$; Unpack z_{ij} to decision x, y;
return x, y.
end function

have request demand of 2,4,6. Access point 1 and 2 are available to the three users with capacity of 20 and 30, respectively. Access point 1 and 2 are all connected to the same Edge Cloud. After stable matching, all the mobile users are connected to the pair of AP 2 and Edge Cloud but leave the pair of AP 1 and Edge Cloud unoccupied. It yields an unbalenced stable matching situation that can be further improved by a transfer that switches user B to AP 1 to reduce response time. Next, we further utilize coalitional game theory [43] to improve this unexpected situation.

Essentially, coalitional game involves a set of players who seek from cooperative groups to strengthen their positions in a given situation. The other basic concept of a coalitional game is the coalition value, which determines the payoffs that all the players receive in a game. In our solution, the first matching phase divides the mobile users set $\mathcal{N}$ into M subsets, each of which is affiliated with one configuration pair. Thus every subset that corresponding to one configuration pair can be viewed as a coalation of mobile users. Thus the output from the prior matching phase is be adopted as an input of the second coalitional game phase. Next we give the definition, including game description and payoff value of our problem.

Definition 5. Game Definition. The coalitional game is defined by the pair $(\mathcal{S}, \Theta)$, where mobile users set $\mathcal{S}$ is the collection of players, and Θ indicates the coalitions, and PV is the payoff value that all the mobile users receive in a game round, which defines in the definition 6.

Definition 6. Transfer Rule. In our coalitional game, a user s has incentive to transfer from the current coalition S_a to the next coalition S_b, formed as $S_{a*} = S_a \setminus s$ and $S_{b*} = S_b \cup s$ if and only if the following two conditions establish:

1) $\kappa_b(t) + r_s(t) \leq E_j$;
2) Payoff value satisfies $PV(s, a, b) = D^q_a(t) + D^c_b(t) - D^q_{a*}(t) - D^c_{b*}(t) > 0$.

The first condition is to ensure that a transfer doesn't go against the capacity constraint of mobile edge cloud in the configuration pair $b*$. The second condition is defined to stimulate players transfer to better coalitions to decrease the overall delay of the whole game. When the transfer process happens to an end, the coalition game reaches to a Nash stable state in which no mobile user can decline the game revenue without hurting the others' benefit.

When the stable matching phase stops with a stable matching Θ, the matching is used as an input for the second coalitional game phase. The matching result is splitted into M partitions, $P = \{S_1, S_2, ...S_M\}$, according to the configuration pair. Then the obtained partition set is applied as an initial partition $P_{initial}$ for the second phase. We iteratively find the transfer pair with minimum transfer value until the algorithm goes to a Nash stable state.

Algorithm 3: Coalitional Game Phase Procedure

Input:
Partition $P = \{P_1, P_2, ..., P_M\}$ obtained from Algorithm 2;
The capacity of the configuration pair, E_j, $\forall j$;
Output:
Mapping between mobile users and configuration pair, $z_{ij}(t)$,$\forall i, j$;

function CoalitionalGame (initial partition P, configuration pair capacity $\forall j, E_j$));
while Partition P hasn't converage to a Nash stable partition **do**
  Each mobile user computes its most preferred transfer rules;
  Init transfer pair (s, a, b) with the Definition 6;
  for all Configuration pair, $\forall j, c_j$ **do**
    Find the minimum transfer (s^*, Θ^*, j) with minimum $TV(s^*, \Theta^*, j)$;
    if $TV(s^*, \Theta^*, j) < TV(s, a, b)$ **then**
      Update the pair $(s, a, b) = (s^*, \Theta^*, j)$;
    end if
  end for
  Agree with the transfer (s, a, b) and update the partition P;
end while
Obtain $z_{ij}, \forall i, j$;
return z_{ij}.
end function

5 THEORETICAL ANALYSIS

In this section, we analyze the theoretical performance of *OLSA* algorithm for MEC. First we discuss the performance relationship between the switching cost and non-switching cost. Then we compare *OLSA* with the offline optimal cost to discern the competitive ratio. We prove that the stable matching in our algorithm is user-side optimal and our coalitional game can converge to a Nash stable status. We also give the complexity analysis of our proposed algorithm in the third subsection. Due to limited space, the detailed proof process can be found in Appendix.

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

5.1 Online Lazy Switching Algorithm Analysis

In this subsection, we derive the performance guarantee of the *OLSA* algorithm, in terms of the optimally gap between the cost incurred by the *OLSA* algorithm and the offline optimal cost. We first deduce the numerical relationships between the non-switching delay, the switching cost, and the optimal value, respectively. Then we determine the competitive ratio of the entire *OLSA* through the derived relationship.

5.1.1 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 cost as follows:*

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

5.1.2 Non-switching Cost 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 non-switching 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 $\epsilon = \max_{t \in [1,T]} \frac{\max D^{ns}(\vec{x}(t), \vec{y}(t))}{\min D^{ns}(\vec{x}(t), \vec{y}(t))}$.

5.1.3 Competitive Analysis

Theorem 2. *The $OLSA$ produces a solution with a competitive ratio of $\epsilon(1 + \frac{1}{\beta})$.*

5.2 Two-Phase Algorithm Analysis

In this subsection, we give analysis of our two-phase algorithm. We first prove that the first stable matching phase always ends to a stable matching assignment and it's user-side optimal. We then demonstrate that the second game phase can converge to Nash stable solution in finite steps.

Theorem 3. *There always exists a stable set of assignments between configuration pairs and users.*

Theorem 4. *Every matching given by the Alg. 2 is as well off as any other stable matching assignments at least.*

Theorem 5. *Given original partition $P_{initial}$ obtained from Alg. 2, Alg. 3 converges to a Nash stable partition P_{final}.*

5.3 Complexity Analysis

For Alg. 2, each mobile user roundly proposes its most preferred configuration pair, the running cycle number of which is related to the length of mobile user's preference list. Since we introduce a virtual configuration pair to handle the unbalanced situation, the length of the mobile user's preference list is $M + 1$. Thus, the Alg. 2 will terminate in $M + 1$ iterations. In each iteration round, the worst case is sorting all of the proposals to determine the preferred mobile user for configuration pair, which needs $O(N \log(N))$ time. Hence, the computation complexity of the Alg. 2 is $O(MN \log(N))$.

For Alg. 3, the computation mainly concentrates on searching for the available transfer pair. The search space of each partition finding round is $N * M$, which causes a $O(MN)$ computation complexity.

6 IMPLEMENTATION CONSIDERATIONS

In this section, we show how our proposed online optimization framework can be plugged into state-of-the-art MEC architecture. We first clarify a recommended MEC architecture from European Telecommunications Standards Institute (ETSI), and then discuss how our method will be implemented into such architecture. At the end of this section, we further discuss how the devices and MEC nodes corporate together to gain performance benefit.

6.1 System Overview

We illustrate the state-of-the-art MEC architecture recommended by ETSI in Fig. 3 [44]. From the view of bottom to the top, the recommended MEC architecture will be abstracted including hardware layer, virtualization layer, application layer, and MEC system orchestrator. The hardware layer is consisted of physical resources like bare servers providing computing capability, and a couple of networks providing network access capability. It has to point out that our mentioned network resources contains not network connecting MEC inner components but outward network access point, for example, 3GPP, directly supplying mobile users with the ability accessing their service entry hosted on MEC clouds.

The recommended MEC architecture contains the MEC hosts and the MEC management entities that's necessary to run MEC service entries. The MEC host includes MEC platform and a virtualization layer that supplies computing, storage, and network resources for MEC applications. In MEC host, MEC platform provides a basic environment that supports the running of the MEC applications. The virtualization layer abstracts the bare metal physician resources to easy-obtained soft resources for MEC applications to apply. Empowered by MEC platform and virtualization layer, MEC applications run on top of the MEC host in virtual machines or containers.

While all the above components of MEC are similar to the traditional centralized cloud, MEC has its own unique characteristics. First, MEC holds on the direct control right of the radio network and this exposes up-to-date radio network information to the applications. Hence by doing this, MEC applications can be aware of the instant network

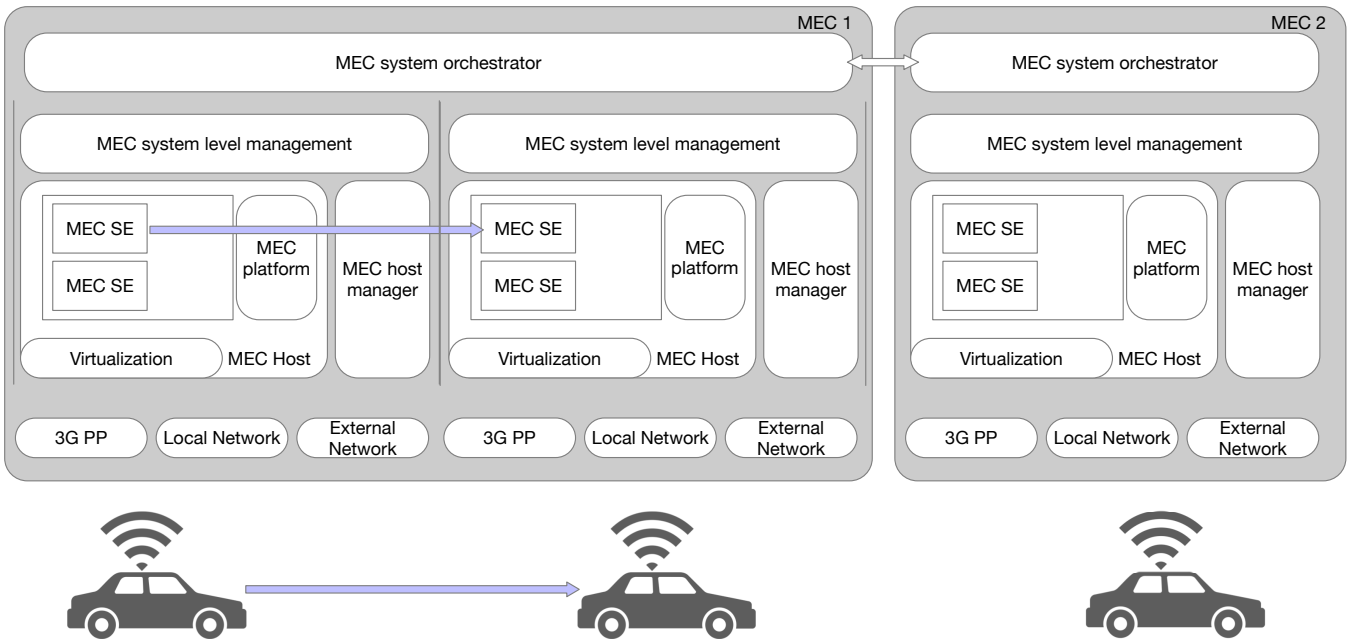

Fig. 3: State-of-the-art MEC architecture recommended by ETSI [44]

information to make further performance optimization. The other particular aspect in MEC is that the MEC platform can easily obtain and utilize the location-related information. The locating position, the service-hosted MEC node, and connected radio network are all known to the MEC applications which can help them improve service performance. These unique characteristics enables that our proposed online optimization framework can be readily integrated into existing MEC architectures.

6.2 System working strategy

Our work is to target the applications serving multiple users with mobility features as well as connecting to multiple different network access points. With the MEC architecture recommended by ETSI, it's reasonable for the mobile applications to get their location and network information, which is crucial to be used in our proposed online optimization framework. While running, the applications collect location and network information from each MEC host, and then hand these information from MEC platform through MEC system level management to the MEC system orchestrator. There will have a central control unit to collect the information from all of the MEC system orchestrator together. After collecting the needed information, the control unit will make the optimization decisions on access point selection and service placement based on the optimization algorithms proposed in Sec. 4.

7 Performance Evaluation

In this section, we conduct both simulated and trace-drive evaluations to validate the performance of our proposed algorithms. We seek to find the answers for the following questions: (1) What is the performance of our two-phase algorithm on solving problem (12)? (2) What is the necessity of two-phase processing? How do the transfers perform in the second phase? (3) What is the performance of our online algorithm without any future system information? (4) What are the influencing factors of service migration? (5) How does the control parameter β affect the service migration?

7.1 Simulated Evaluation

7.1.1 Simulation Setup

Data Set: We take advantage of the ONE simulator [45] to generate the movement traces of mobile users. The ONE simulator is a potent tool that is widely used in generating user movement traces by using different mobile models. We generate the movement trajectory of 1,000 mobile users through this simulator, which belong to 6 groups with different speeds. We use groups of different speeds to tell moving objects, such as cars, or low-speed moving objects, such as pedestrians.

MEC Details: The total simulation area is almost 8×8 km^2. We divide the whole area into 8×8 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 resource requirement in the system. Regarding the edge cloud in each MEC, we simply set the total capacity as $1/10$ of the amount of resource requirement at all time slots. The communication delay between two MEC in our simulations is measured by the geographical distance between any two entities based on their locations, which is in the range of $[0, 16]$ ms.

Users Details: We assume that each user has independent resource demands of access point. Capricious requirements will put different pressure on the queuing delay of network access points. We randomly set the requirements for each user within the AP capacity of each MEC. Considering that the service's switching cost varies among different edge clouds, we generate the switching cost by following a

Gauss distribution with the negative tail cutted, which is distributed in $[0.5, 1]$.

7.1.2 Performance Benchmark

We carry out experiments with the above setting. To further understand the impact of our online algorithm on different types of delay and the efficiency of our two-phase algorithm, we compare the results of our algorithm with the following two baselines:

- **Stable matching (SM)**: at each time slot, the system operator only uses Alg. 2 to select access point and arrange service placement.
- **Frequent switching (FS)**: the system operator utilizes both stable matching and coalitional game to make decisions for network selection and service placement without considering any migration tactics at each time slot. This strategy applies new service placement choice, which causes service migration at each time slot.

7.1.3 Effectiveness of Two-phase Algorithm at Single Time Slot

In this subsection, we evaluate the performance of our two-phase algorithm, named SMCG (stable matching and coalitional game) in the simulations.

The changes of different delay during SMCG iteration. When evaluating response time, we firstly consider the delay changes at different iteration cycles. As shown in Fig. 7, at the beginning of the iteration processing, the queuing delay of the system stays at a high level. With the increasing of iteration, overall queuing delay first declines steeply and then deceases gently. For communication delay, it can be seen a slightly growth with the algorithm iterating. The reasons behind these changes is that in the second game phase, we accept the stable matching result from the prior stable matching phase. In the stable matching, we build users' preference list by assuming that an access point provides the worst delay to all users. Such a mistiness disposing weakens the possibility of optimization queuing delay on network selection but can produce an excellent service placement strategy, which has little reduction space coming into game phase from stable matching. Therefore, in the iteration of game phase, SMCG mainly reduces queuing delay as well as make moderate changes between queuing delay and communication delay, which results in the going up of communication delay. It also can be found in our experiments that our algorithm can converge to an end after hundreds of iterations.

Load distribution. We calculate load distribution before and after adopting our SMCG method. Fig. 4 shows a time slice that user density hotspot is concentrated in area (4,B) and its surroundings. We give load distribution of access point and edge cloud in Fig. 5, and Fig. 6, respectively. As we can see, SMCG significantly reduces the load of the access points in the hotspot area and distributes the load to its surrounding areas, achieving the effect of load balancing. For the edge cloud, their resources are limited, and they will be greatly troubled when they face sudden traffic and congested population. In our experiments, we can find in the Fig. 6 that our solution can make full use of the edge cloud resources of the entire system, offloading service pressure to other edge clouds, improving resource utilization of the entire system, and avoiding fierce competition for resources.

Response time comparison at different time slot without considering service migration. In our experiment, we want to learn how our proposed method does at different time slot. We adopt frequent switching (FS) in this test, that is, we only use SMCG to make decisions of network selection and service placement for users at each time slot. FS strategy will update and migrate service immediately when a new placement decision is computed. We list communication delay, queuing delay and overall delay in Fig. 11, Fig. 12, and Fig. 13, respectively.

- **Communication delay.** According to Fig. 11, the communication delay in our algorithm has fluctuations over time. Both the SM and SMCG method keep stable communication delay, but the communication delay in SMCG is always less than that of SM, which also proves our efficiency on balancing communication delay.
- **Queuing delay.** For queuing delay at different time slot, SMCG method has a level off at around 25, while SM causes a changeable performance, which is two times our SMCG method at least. From this point, our proposed method can provide users with a stable delay even if the system is full of ruleless movement event.
- **Overall delay.** The tendency of overall delay shows the same character as the queuing delay. Due to the stability of communication delay and queuing delay in SMCG, overall delay in system sostenuto owns a smooth performance over time.

7.1.4 Effectiveness of OLSA

Response time in a non-switching phase. Our online algorithm adopts a strategy of delaying the service migration, so how does it affect the response time of the system? We calculate the response time of our algorithm over time, and the results are listed in Fig. 8, Fig. 9 and Fig. 10. The three pictures use different parameter settings, but the main features are similar. We use a star to represent the service migration point chosen by *OLSA*. We can find that during service migration, the overall delay of the entire system will drop to the lowest point. After that, before the next migration occurs, the delay of the entire system will continue to increase. When the delay accumulates to a certain degree, the system will make a decision to migrate the service to reduce the response time of the entire system. According to the statistics of the queuing delay and the communication delay, it can be seen that before the migration event occurs, the queuing delay and the communication delay of the entire system have different trends. The main reason for the increase in the overall system delay is communication delay. Because the system adopts a delayed migration strategy, when users move, their services will be retained at the place where they were placed at the previous point in time. Therefore, users need additional communication overheads to contact with the corresponding service entry, which causes the significant increase of system's communication delay. As for the queuing delay of each access point, it does show a

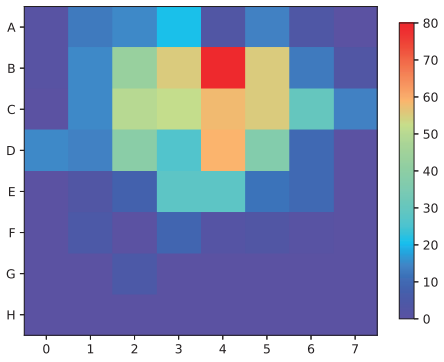

Fig. 4: User density before SMCG.

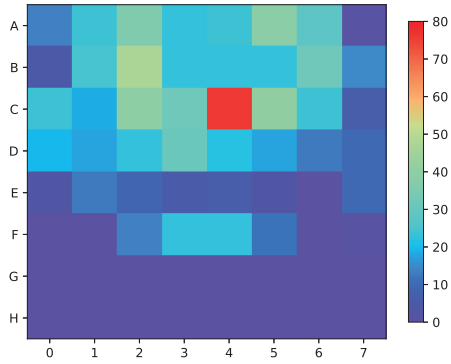

Fig. 5: Access point pressure in SMCG.

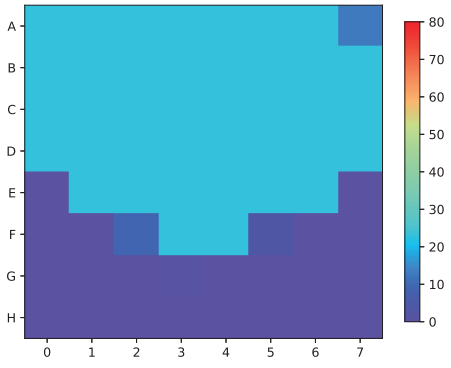

Fig. 6: Service placement distribution in SMCG.

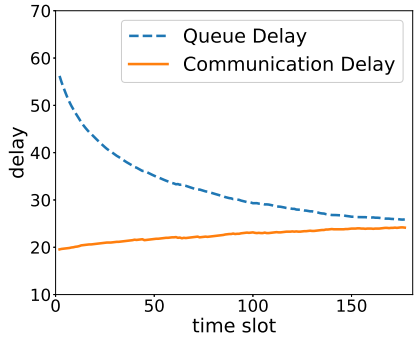

Fig. 7: Queuing delay and communication delay with iteration number in SMCG.

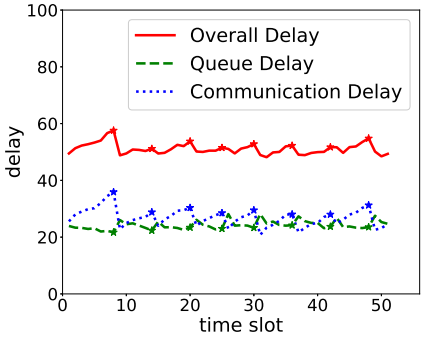

Fig. 8: Different delay of *OLSA* ($\beta = 0.5$).

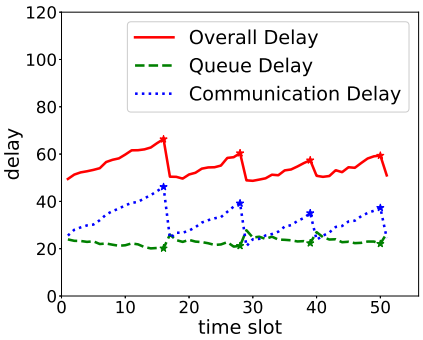

Fig. 9: Different delay of *OLSA* ($\beta = 1$).

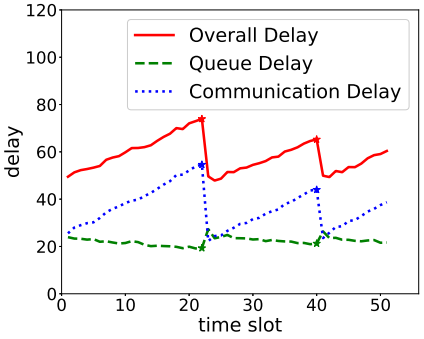

Fig. 10: Different delay of *OLSA* ($\beta = 1.5$).

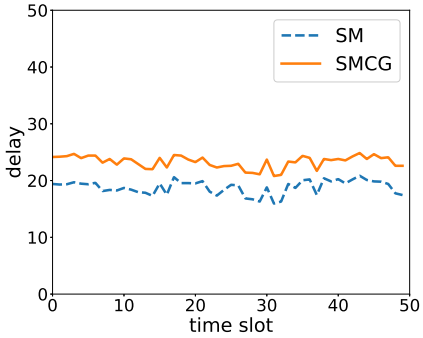

Fig. 11: Communication delay in SM and SMCG under FS setting.

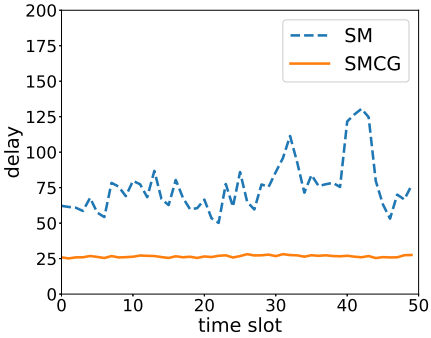

Fig. 12: Queuing delay in SM and SMCG under FS setting.

certain reduction trend. This is because, in the case of non-switching, the service placement location is fixed. The *OLSA* algorithm mainly optimizes network access and has more optimization space to reduce the overall queuing delay of the system.

The impact of control parameter β. In our algorithm, we introduce a control variable β to adjust the frequency of service migration of the system. Therefore, we evaluate the system performance under different β settings. When parameter β is equal to 0.5, 1, and 1.5, the system situation is shown in Fig. 8, Fig. 9 and Fig. 10. Under the parameter β of three different sizes, we find that the system's tolerance for delay grows up with the increase of parameter β. It's found from the experimental results that when β is equal to 0.5, the system migrates services every 5 time slots. When β is equal to 1 and 1.5, the service migration interval of the system is 10 and 20 time slots. Therefore, in the face of applications with different delay requirements, we can dynamically adjust the value of parameter β to control the frequency of system service migration and suppress the increase of the total delay in the system.

Switching cost comparison. Our online method *OLSA* aims to optimize and control the cost of service switching. Therefore, we compare the switching cost of the OLSA

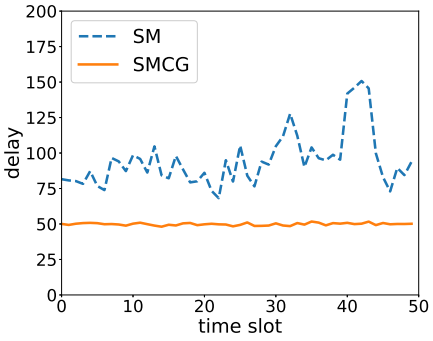

Fig. 13: Overall delay in SM and SMCG under FS setting.

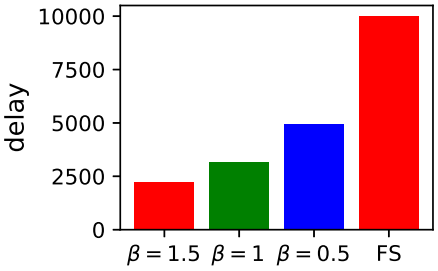

Fig. 14: Switching cost comparison.

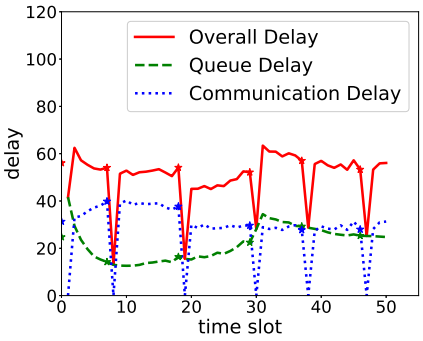

Fig. 15: Different delay of *OLSA* under realistic trace.

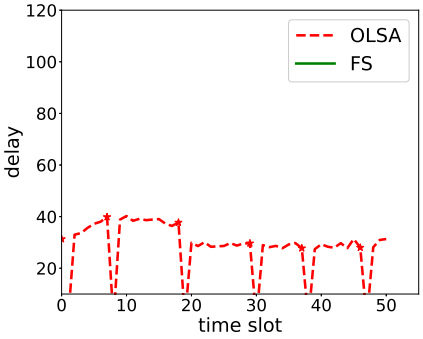

Fig. 16: Communication delay in *OLSA* and FS.

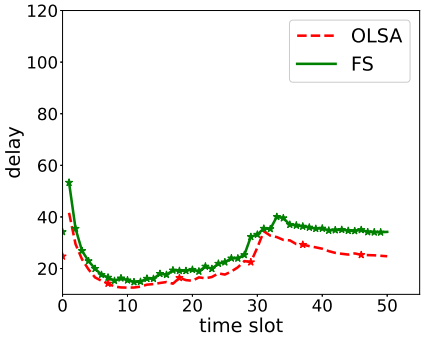

Fig. 17: Queuing delay in *OLSA* and FS.

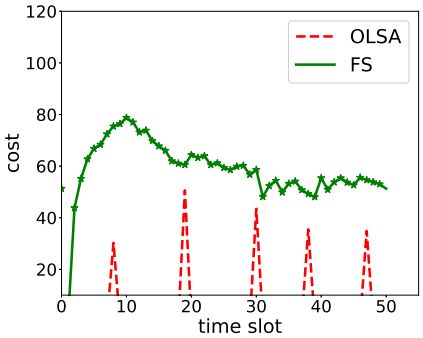

Fig. 18: Switching cost in *OLSA* and FS.

method and the baseline FS under different parameter settings in Fig. 14. As we can see in the Fig. 14, compared with FS, our method can effectively control the switching cost under different parameter settings. For *OLSA* itself, as the parameter β increases, the switching overheads continue to decrease. This is because a larger value of β allows the system to tolerate more communication delay and reduce the frequent service migration of the system. Therefore, the overheads of service migration are smaller.

7.2 Real-trace Evaluation

This subsection conducts evaluations on realistic taxi movement trace from EPFL [46]. This trace includes more than 500 taxis mobility records over 30 days in San Francisco Bay Area.

7.2.1 Evaluation Setup

By removing the abnormal taxies, we finally obtain 535 taxies and test the records of these taxies in 50 time slots in an area of about $10KM^2$. Other setup configurations still follow the configuration in Sec. 7.1.1 and Sec. 7.1.2, including MEC details, users details. We compare *OLSA* with the benchmark FS to illustrate the effectiveness.

7.2.2 Effectiveness of OLSA

Delay in *OLSA*. Fig. 15 shows different delay components when $\beta = 1$. Comparing Fig. 15 and Fig. 9, we can observe that in realistic trace, the communication delay is not increased step by step like simulated data. The queue delay in realistic data trace, however, varies more than simulated data set. This is because the realistic data set contains more natural characteristics with random factors, while the simulated data set is created by following some settled policies.

Comparison between *OLSA* and FS. We also perform comparison between *OLSA* and the baseline FS, in terms of the total cost, communication delay, queuing delay, and switching cost, the results are illustrated in Fig. 15 to Fig. 18. The performance of *OLSA* is always better than FS's because FS suffers from frequent service switching which will incur high switch cost like Fig. 18 shows. Fig. 16 shows the communication delay in *OLSA* and FS setting. Because FS frequently switches its services at each time slot, so its communication delay is always zero. Although at most time slots, the system should endure communication delay, the overall delay is better in *OLSA* because it avoids unnecessary service switching cost. Regarding queuing delay in Fig. 17, the trends of *OLSA* and FS is similar, but our *OLSA* is slightly better than FS, because *OLSA* comprehensively considers the switching cost, the queuing delay and the communication delay to make service decisions and network selection decisions, which can gain global optimizations to make trade-off or achieve good performance.

8 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 queuing delay, communication delay and service switching cost. We first argue that, the commonly adopted approach of unitary service placement doesn't necessarily improve the user-perceived QoS, since the later is vulnerable to not only the communication delay, but also the network access delay. Then we formulate the joint optimization on access network selection and service placement as a long-term cost-minimization problem. We design an efficient online framework, which decomposes this long-term optimization problem into a series of one-shot problems. To address the NP-hardness of these one-shot problems, we provide a two-phase algorithm to get an efficient near-optimal solution. Both rigorous theoretical analysis on the optimality gap and extensive trace-driven simulations demonstrate the efficacy of our proposed solution.

References

[1] 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.

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

[3] 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.

[4] 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.

[5] 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.

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

[7] 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.

[8] S. Wang, J. Xu, N. Zhang, and Y. Liu, "A survey on service migration in mobile edge computing," *IEEE Access*, vol. 6, pp. 23 511–23 528, 2018.

[9] T. Taleb and A. Ksentini, "An analytical model for follow me cloud," in *Proceedings of IEEE GLOBECOM*, 2013.

[10] A. Ksentini, T. Taleb, and M. Chen, "A Markov decision process-based service migration procedure for follow me cloud," in *Proceedings of IEEE ICC*, 2014.

[11] S. Wang, R. Urgaonkar, M. Zafer, T. He, K. Chan, and K. K. Leung, "Dynamic service migration in mobile edge-clouds," in *Proceedings of IFIP/IEEE Networking*, 2015.

[12] 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.

[13] 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.

[14] B. Toghi, M. Saifuddin, H. N. Mahjoub, M. Mughal, Y. P. Fallah, J. Rao, and S. Das, "Multiple access in cellular v2x: Performance analysis in highly congested vehicular networks," in *Proceedings of IEEE VNC*, 2018.

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

[16] M. Mishra, A. Das, P. Kulkarni, and A. Sahoo, "Dynamic resource management using virtual machine migrations," *IEEE Communications Magazine*, vol. 50, no. 9, pp. 34–40, 2012.

[17] C. Clark, K. Fraser, S. Hand, J. G. Hansen, E. Jul, C. Limpach, I. Pratt, and A. Warfield, "Live migration of virtual machines," in *Proceedings of USENIX NSDI*, 2005.

[18] J. Shetty, M. Anala, and G. Shobha, "A survey on techniques of secure live migration of virtual machine," *International Journal of Computer Applications*, vol. 39, no. 12, pp. 34–39, 2012.

[19] S. Wang, R. Urgaonkar, T. He, M. Zafer, K. Chan, and K. K. Leung, "Mobility-induced service migration in mobile micro-clouds," in *Proceedings of IEEE MILCOM*, 2014.

[20] K. LaCurts, J. C. Mogul, H. Balakrishnan, and Y. Turner, "Cicada: Introducing predictive guarantees for cloud networks," in *Proceedings of HotCloud*, 2014.

[21] M. L. Puterman, *Markov Decision Processes: Discrete Stochastic Dynamic Programming*. John Wiley & Sons, 2014.

[22] R. Langar, N. Bouabdallah, and R. Boutaba, "A comprehensive analysis of mobility management in mpls-based wireless access networks," *IEEE/ACM Transactions on Networking*, vol. 16, no. 4, pp. 918–931, 2008.

[23] M. Srivatsa, R. Ganti, J. Wang, and V. Kolar, "Map matching: Facts and myths," in *Proceedings of ACM SIGSPATIAL*, 2013.

[24] A. J. Nicholson, Y. Chawathe, M. Y. Chen, B. D. Noble, and D. Wetherall, "Improved access point selection," in *Proceedings of ACM MobiSys*, 2006.

[25] S. Wang, R. Urgaonkar, T. He, K. Chan, M. Zafer, and K. K. Leung, "Dynamic service placement for mobile micro-clouds with predicted future costs," *IEEE Transactions on Parallel and Distributed Systems*, vol. 28, no. 4, pp. 1002–1016, 2016.

[26] E. Cho, S. A. Myers, and J. Leskovec, "Friendship and mobility: user movement in location-based social networks," in *Proceedings of ACM SIGKDD*, 2011.

[27] P. Bellavista, S. Chessa, L. Foschini, L. Gioia, and M. Girolami, "Human-enabled edge computing: Exploiting the crowd as a dynamic extension of mobile edge computing," *IEEE Communications Magazine*, vol. 56, no. 1, pp. 145–155, 2018.

[28] 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," in *Proceedings of IEEE INFOCOM*, 2019.

[29] H. Ma, Z. Zhou, and X. Chen, "Leveraging the power of prediction: Predictive service placement for latency-sensitive mobile edge computing," *IEEE Transactions on Wireless Communications*, 2020.

[30] H. Badri, T. Bahreini, D. Grosu, and K. Yang, "Energy-aware application placement in mobile edge computing: a stochastic optimization approach," *IEEE Transactions on Parallel and Distributed Systems*, 2019.

[31] S. Bi, L. Huang, and Y.-J. A. Zhang, "Joint optimization of service caching placement and computation offloading in mobile edge computing systems," *IEEE Transactions on Wireless Communications*, 2020.

[32] 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.

[33] 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.

[34] J. Wu, E. W. Wong, Y.-C. Chan, and M. Zukerman, "Energy efficiency-qos tradeoff in cellular networks with base-station sleeping," in *Proceedings of IEEE GLOBECOM*, 2017.

[35] S. Chen, L. Jiao, L. Wang, and F. Liu, "An online market mechanism for edge emergency demand response via cloudlet control," in *Proceedings of IEEE INFOCOM*, 2019.

[36] Z. Zhou, F. Liu, Y. Xu, R. Zou, H. Xu, J. C. Lui, and H. Jin, "Carbon-aware load balancing for geo-distributed cloud services," in *Proceedings of IEEE MASCOTS*, 2013.

[37] Y. Niu, F. Liu, and Z. Li, "Load balancing across microservices," in *Proceedings of IEEE INFOCOM*, 2018.

[38] D. B. Shmoys and É. Tardos, "An approximation algorithm for the generalized assignment problem," *Mathematical programming*, vol. 62, no. 1-3, pp. 461–474, 1993.

[39] 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.
[40] 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, 2008.
[41] D. Gale and L. S. Shapley, "College admissions and the stability of marriage," *The American Mathematical Monthly*, vol. 69, no. 1, pp. 9–15, 1962.
[42] T. Wang, F. Liu, J. Guo, and H. Xu, "Dynamic sdn controller assignment in data center networks: Stable matching with transfers," in *Proceedings of IEEE INFOCOM*, 2016.
[43] T. Wang, F. Liu, and H. Xu, "An efficient online algorithm for dynamic sdn controller assignment in data center networks," *IEEE/ACM Transactions on Networking*, vol. 25, no. 5, pp. 2788–2801, 2017.
[44] ETSI, "MEC White Papers," https://portal.etsi.org/TB-SiteMap/MEC/MEC-White-Papers.
[45] A. Keränen, J. Ott, and T. Kärkkäinen, "The ONE Simulator for DTN Protocol Evaluation," in *Proceedings of the 2nd International Conference on Simulation Tools and Techniques*, 2009.
[46] M. Piorkowski, N. Sarafijanovic-Djukic, and M. Grossglauser, "Crawdad data set epfl/mobility (v. 2009-02-24)," 2009.
[47] 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.

Fangming Liu (S'08, M'11, SM'16) received the B.Eng. degree from the Tsinghua University, Beijing, and the Ph.D. degree from the Hong Kong University of Science and Technology, Hong Kong. He is currently a Full Professor with the Huazhong University of Science and Technology, Wuhan, China. His research interests include cloud computing and edge computing, datacenter and green computing, SDN/NFV/5G and applied ML/AI. He received the National Natural Science Fund (NSFC) for Excellent Young Scholars, and the National Program Special Support for Top-Notch Young Professionals. He is a recipient of the Best Paper Award of IEEE/ACM IWQoS 2019, ACM e-Energy 2018 and IEEE GLOBECOM 2011, as well as the First Class Prize of Natural Science of Ministry of Education in China.

Fei Xu received the PhD degree in computer science and engineering from the Huazhong University of Science and Technology, Wuhan, China, in 2014. He received Outstanding Doctoral Dissertation Award in Hubei province, China, and ACM Wuhan & Hubei Computer Society Doctoral Dissertation Award in 2015. He is currently an associate professor with the School of Computer Science and Technology, East China Normal University, Shanghai, China. His research interests include cloud computing, virtualization technology, and distributed systems.

Bin Gao received the B.Eng. degree from the Huazhong University of Science and Technology (HUST), Wuhan, China, in 2017. He is currently pursuing his master degree in HUST. His research interests include mobile edge computing, cloud computing, geo-distributed data analytics.

Bo Li (S'89-M'92-SM'99-F'11) is a Chair Professor in the Department of Computer Science and Engineering, Hong Kong University of Science and Technology, which he has been affiliated with since 1996. He held the Cheung Kong Chair Professor in Shanghai Jiao Tong University between 2010 and 2015. His research interests cover broad areas in networking and distributed systems, with recent focuses on big data and machine learning systems, cloud and edge computing. He was a co-recipient of six Best Paper Awards from IEEE including the Test-of-Time Paper Award from IEEE INFOCOM (2015). He received his PhD in the Electrical and Computer Engineering from University of Massachusetts at Amherst, and his B. Eng. (summa cum laude) in the Computer Science from Tsinghua University, Beijing, China.

Zhi Zhou received the B.S., M.E., and Ph.D. degrees in 2012, 2014, and 2017, respectively, all from the School of Computer Science and Technology at Huazhong University of Science and Technology (HUST), Wuhan, China. He is currently an associate professor in the School of Computer Science and Engineering at Sun Yat-sen University, Guangzhou, China. In 2016, he was a visiting scholar at University of Göttingen. He was nominated for the 2019 CCF Outstanding Doctoral Dissertation Award, the sole recipient of the 2018 ACM Wuhan & Hubei Computer Society Doctoral Dissertation Award, and a recipient of the Best Paper Award of IEEE UIC 2018. His research interests include edge computing, cloud computing, and distributed systems.

APPENDIX

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

Proof. We construct a polynomial-time reduction to problem (11) from the generalized assignment problem (GAP), a classic combinatorial optimization problem which is known to be NP-hard [47]:

$$\begin{aligned} \min \quad & \sum_{i=1}^{m}\sum_{j=1}^{n} c_{ij}x_{ij} \\ \text{s.t.} \quad & \sum_{i=1}^{m} x_{ij} = 1, \forall j = 1, ..., n, \\ & \sum_{j=1}^{n} w_{ij}x_{ij} \leq t_i, \forall i = 1, ..., m, \\ \text{Var} \quad & x_{ij} \in \{0, 1\}, \forall i = 1, ..., m, j = 1, ..., n. \end{aligned} \tag{17}$$

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 (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 problem (11). Given the NP-hardness of GAP, the one-shot optimization problem (11) must be NP-hard as well. □

Lemma 1. *In a time slice $[1, T]$, given any control parameters β, the $OLSA$ algorithm provides a determined bound for overall switching cost 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 migration occurring. Before $(i+1)$-time migration comes, according to judgement condition of the online algorithm, we have the bound that the non-switching 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:

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

□

Lemma 2. *Let $(\vec{x}^*(t), \vec{y}^*(t))$ denote the optimal solution to problem (10). In a time slice $[1, T]$, the overall non-switching 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 non-switching delay to the minimum switching cost during the time period $[1, T]$. On this ground, the non-switching delay incurred at each time slot is ϵ times the optimal non-switching 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:

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

□

Theorem 2. *The $OLSA$ produces a solution with a competitive ratio of $\epsilon(1 + \frac{1}{\beta})$.*

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

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

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

Theorem 3. *There always exists a stable set of assignments between configuration pairs and users.*

Proof. We will prove the existence of stable matching by analyzing the iterative procedures. We assume that if a configuration pair is not willing to accept a user under any situations, then this user will not even be permitted to propose to these pairs. In the procedures, all users firstly raise their first choice of configuration pair as proposals. A configuration pair with capacity of E_j then places on its waiting list the top E_j who ranks the topest, or all of the users' proposals if they're fewer than E_j. All of the rest will be rejected. Rejected users then again ask their second choice and each configuration pair select the top E_j from among the new proposers and those on its waiting list, makes these new constitute its new waiting list, and reject the rest. The procedure will continue to find the matching until every user is put on a waiting list or has been rejected by every configuration that he desires. At this point, each configuration pair accepts all the users on its waiting list and then constructs a matching assignment Θ. Here, we consider two kinds of user-configuration (s, c) that is not in Θ.

Case 1: User s never proposes to configuration c. This happens when s prefers another configuration pair c' to c. In this situation, pair (s, c) is not stable.

Case 2: User s proposes to configuration c. But c rejects s. c prefers the better choice s' to s. Hence, pair (s, c) is not stable, either.

Hence, the matching in the Θ keeps stable. □

Theorem 4. *Every matching given by the Alg. 2 is as well off as any other stable matching assignments at least.*

Proof. We will prove this theorem by induction. Here we call a configuration pair "possible" for a specific user if there is a stable assignment between them. Assume that at a given point in the processing procedure no user has yet been turned away from a configuration pair that is probably suitable for it. There is a time point when configuration A is receiving a full set of better users $s_1, s_2, ...s_q$, and thus A rejects the worse option s_α. We must show that A is impossible for s_α. We understand that each s_i prefers configuration pair A to all the other users, except for the pairs that rejected him. Similarly, those pairs also are impossible for him. Consider a hypothesis that sends s_α to A and everyone else to configuration pairs that are possible for them. At least one of the member in s_i will have to connect to a less desirable place than A. However, this matching is unstable because not s_i but A will agitate it to the better benefit of both. For this reason, the hypothetical matching is unstable and A is impossible for s_α. Our algorithm only rejects users from configuration pairs which they couldn't be accepted to any other stable matching. Therefore, the final assignment in our algorithm is optimal. □

Theorem 5. *Given original partition $P_{initial}$ obtained from Alg. 2, Alg. 3 converges to a Nash stable partition P_{final}.*

Proof. Denote the partition that's after k iterations as P_k. Alg. 3 can be seen as an order of partitions:

$$P_0 = P_{initial} \rightarrow P_1 \rightarrow P_2 ...$$

Because a transfer pair (s, a, b) only affects the partition set S_a and S_b (forming the new coalitions $S_{a*} = S_a \backslash \{s\}$ and $S_{b*} = S_b \backslash \{s\}$ each close transfer, e.g., $P_l \rightarrow P_{l+1}$). Every neighboring transfer follows an order:

$$D^q_{a*}(t) - D^c_{b*}(t) < D^q_a(t) + D^c_b(t),$$

which is transitive and irreflexive. Because the partition number of set S is finite, Alg. 3 will finally converge to an end partition P_{final}. In addition, we assume that P_{final} is not Nash stable, and there must exist a transfer pair (s, a, b) that can lead to better social welfare. Alg. 3 has to continue, and P does not converge. However, this disagrees the previous demonstration of Alg. 3's convergence. Therefore, Alg. 3 will converge to a Nash stable partition. □