Follow Me at the Edge: Mobility-Aware Dynamic Service Placement for Mobile Edge Computing

Tao Ouyang, Zhi Zhou, Xu Chen
School of Data and Computer Science, Sun Yat-sen University, Guangzhou, China
ouyt9@mail2.sysu.edu.cn, {zhouzhi9, chenxu35}@mail.sysu.edu.cn

***Abstract*—Mobile edge computing is a new computing paradigm in which cloud computing capabilities are pushed from the network core to the network edge to serve the end-user in proximity. However, with the sinking of computing capabilities, the new challenge incurred by user mobility arises: since end-users typically move erratically, the services should be dynamically migrated among multiple edges to maintain the service performance, i.e., user-perceived latency. Tackling this problem is non-trivial since frequent service migration would greatly increase the operational cost. To address this challenge in terms performance-cost trade-off, in this paper we study the mobile edge service performance optimization problem under long-term cost budget constraint. To address user mobility which is typically unpredictable, we first apply Lyapunov optimization to decompose the long-term optimization problem into a series of real-time optimization problems which do not require a priori knowledge such as user mobility. As the decomposed problem is NP-hard, we further propose an efficient heuristic based on the Markov approximation technique. Rigorous theoretical analysis and extensive evaluations demonstrate the efficacy of the proposed solution.**

I. Introduction

With the explosive growth of mobile devices, the recent years have witnessed an unprecedented shift of user preferences from traditional desktops and laptops to smartphones and other connected-devices. Proliferation of the powerful and reliable cloud computing, together with widespread fourth/fifth generation (4G/5G) Long Term Evolution (LTE) networks and WiFi access, have brought rich cloud-hosted mobile services to end users [1] [2]. It is readily acknowledged, however, for some emerging mobile applications as exemplified by augmented reality and interactive gaming [3], the long communication latency to the centralized cloud data center (typically hundreds of milliseconds) can far exceed the stringent timeliness requirement (typically tens of milliseconds) of these mission-critical mobile applications.

To satisfy these mission-critical mobile applications that require ultra-low latency, the concept of mobile edge computing (MEC) [4] [5] has been recently proposed as an enhancement of centralized cloud computing. With MEC, the cloud computing and storage capabilities are pushed from the network core to network edges *in close proximity to mobile devices and users*. Here an edge is typically a micro-data center that is resource-rich, attached to a base station (BS) [6] or an access point [7], and available for use by nearby devices. In the paradigm of MEC, as user workload is served by a nearby edge node rather than the remote cloud, the end-to-end latency is significantly reduced.

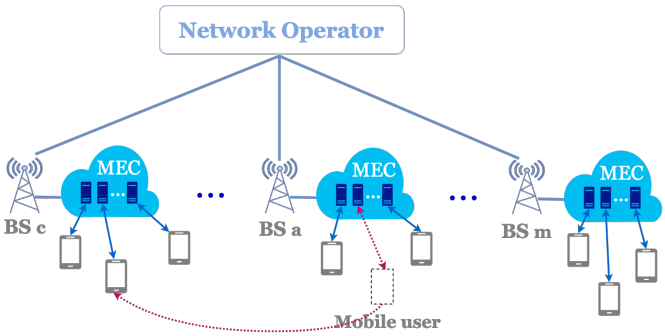


Fig. 1: An example of dynamic service placement when a user roams throughout the network in MEC

However, with the presence of user mobility [8], enhancing low-latency and smoothing user experience are far more than simply pushing the cloud capabilities to the network edge. Let us consider a practical scenario as shown in Fig. 1, when a mobile user is within the geographical coverage of a MEC node a, it is clear that if we want to minimize the user-perceived latency, the user should be served by the nearest edge, i.e., MEC node a. Considering the user mobility and assuming that after a while, the aforementioned mobile user moves to the coverage of MEC node c. Then, if keeping the service profile of this user placed at MEC node A to serve this user, his perceived latency would greatly deteriorate due to the extended network distance. This example demonstrates that, to optimize the user experience of MEC, the service profiles of mobile users should be dynamically re-placed among edges to *follow the mobility* of users.

Unfortunately, optimizing the user experience via dynamic service profile placement is non-trivial, due to the following two salient features of MEC: *First*, the user-perceived latency is jointly determined by the communication delay and computing delay [9]. Therefore, if aggressively placing the service profile of each user at the nearest MEC node, then some MEC nodes could be overloaded, leading to increased computing

[1]This work was supported in part by the National Key Research and Development Program of China under grant No.2017YFB1001703; the National Science Foundation of China under Grant No.U1711265; the Fundamental Research Funds for the Central Universities under grant No.17lgjc40, and the Innovative R&D Team Introduction Program of Guangdong Province. (Corresponding author: Xu Chen and Zhi Zhou)

978-1-5386-2542-2/18/$31.00

latency. *Second*, following the user mobility requires frequent service migration among multiple MEC nodes. In return, such frequent service migration incurs additional operation cost such as usage of the expensive wide-area-network (WAN) bandwidth and system energy consumption [10]. As a result, an effective dynamic service placement strategy should carefully (1) cooperate the communication delay and computing delay to minimize the user-perceived latency, and (2) navigate the *performance-cost trade-off* in a cost-efficient manner.

Following the above two guidances on dynamic service placement for MEC, in this paper, we propose a *mobility-aware dynamic service placement framework for cost-efficient MEC*. In particular, to strike a nice balance between the service performance and the operational cost incurred by cross-edge service migration, we propose to minimize the user-perceived latency over the long run, under the constraint of a long-term migration cost budget which is pre-defined monthly or yearly by the network operator in practice. By applying Lyapunov optimization technique to the formulated stochastic optimization problem, our framework can effectively incorporate the long-term migration cost budget into real-time optimizations, and make online decisions on dynamic service placement, *without requiring any a priori future information (e.g., user mobility)*. To address the challenge of the NP-hardness of the resulted real-time optimizations, we further develop an efficient heuristic based on the Markov approximation technique to seek a near-optimal solution. Both rigorous theoretical analysis and extensive evaluations demonstrate the cost-efficiency of the proposed mobility-aware online service placement framework.

The rest of this paper is organized as follows. Section II reviews related work. The system model and problem formulation are presented in Section III. Section IV proposes an online service placement algorithm to seek a near-optimal strategy. Section V presents the theoretical analysis of the proposed framework. Performance evaluation is carried out in Section VI. Section VII concludes this paper.

II. Related Work

Service placement is not a new topic, as it has been extensively studied in the paradigm of cloud computing. Specifically, the goal of service placement in cloud computing can be categorized into: (1) consolidating the services to a smaller set of physical servers to improve the resource utilization and reduce the operational cost [11], (2) placing the services to a set of heterogeneous nodes to leverage the heterogeneities on energy efficiency or cost efficiency [12], and (3) placing the services to different nodes to perform network load balancing [13]. However, as we have discussed in Section I, the goal of service placement in MEC is to follow the user mobility and thus to reduce the user-perceived latency adaptively.

A key challenge towards efficient service placement in MEC is to follow the mobility of users and devices. In addressing this challenge, some work is based on the assumption of perfect predictability on future information. For example, the authors in [14] tackle the trade-off between the execution overhead and latency, with a mobility-based prediction scheme which estimates the data transfer throughput, handoff time and VM migration management in advance. Moreover, the recent work [15] further studies how to place service by predicting the future cost incurred by data transmission, processing and service migration. Unfortunately, the future information such as user mobility is extremely challenging to accurately predict in realistic environments.

In response to the challenge that user mobility may not be readily predictable in practice, another stream of recent work resorts to a milder assumption that the user mobility follows a Markovian process, and then applies the technique of Markov Decision Process (MDP). Specifically, a preliminary research in [16] explores how service migration impacts the perceived latency of mobile users, via utilizing Markov chains to analyze whether to migrate services or not. Both [17] and [18] try to determine an optimal threshold decision policy on service migration based on MDP. In [19], the optimal service migration strategy is devised by formulating the service placement problem as a sequential decision-making problem. While the Markovian assumption is not suitable when the user has a desired destination to reach in [20]. In comparison, our online service placement strategy does not make any assumption on the user mobility, yet can achieve a performance that can be arbitrarily close to the offline optimum. Moreover, all these works do not consider the practical operational cost constraint for dynamic service placement.

A closely related work to our proposed mobility-aware online service placement framework is [21]. Without requiring the future user mobility as a priori knowledge, an energy-aware mobility management scheme is proposed in [21] to minimize the total delay (including both communication and computation delay) under the long-term energy consumption constraint. However, it is worth noting that our work substantially differs from and complements to [21] in at least the following four aspects: (1) the study in [21] only considers a single-user service placement scenario, while we consider a more practice-relevant multi-user case. (2) We consider the efficient allocation of the limited edge resource to multiple users, and thus to coordinate computation and communication delay to minimize the total latency. (3) To avoid excessive operational cost incurred by frequent service migration, we navigate the performance-cost tradeoff in a cost-efficient manner. (4) On efficient algorithmic design, the work [21] is based on multi-arm bandit optimization, while we blend the advantages of Lyapunov optimization and Markov approximation techniques.

III. System Model and Problem Formulation

As shown in Fig.1, we consider a network operator running a set $\mathcal{M} = \{1, 2, ..., M\}$ of MEC nodes to serve a set $\mathcal{N} = \{1, 2, ..., N\}$ of mobile users. Each MEC node is attached to a local base station or wireless access point, via high speed local-area network (LAN). Inspired by the recent work [22] [23] on resource allocation for MEC, in this paper we adopt a device-oriented service model for MEC, rather than the traditional application-oriented service model for cloud computing [24]. Specifically, the service profile and environment

TABLE I: Key notations in our model

Notation	Definition
$c_i^k(t)$	Whether the service profile of user k is placed at MEC node i (=1) or not (=0)
$R^k(t)$	The amount of computation capacity required by user k
$N_i(t)$	The number of services served by MEC node i
F_i	The maximum computing capacity of MEC node i
$D^k(t)$	The computing delay for user k
$H_i^k(t)$	The communication delay when the service profile of user k is placed on MEC node i
$L^k(t)$	The communication delay for user k
$T^k(t)$	The total latency for user k
$E_{ji}^k(t)$	The cost of migrating service of user k from source MEC node j to destination MEC node i
$E(t)$	The total migration cost for all users
E_{avg}	The long-term time-averaged cost budget
V	Lyapunov control parameter
β	Markov approximation ratio

for the applications ran on each mobile device (rather than each application) is assigned to a dedicated virtual machine [25] or container [26]. To better capture the user mobility, the system is assumed to operate in a slotted structure and its timeline is discretized into time frames $t \in \mathcal{T} = \{0, 1, 2..., T\}$. At each time slot t, each mobile user sends a service request to the network operator, which then determines the optimal MEC node to serve this user, by carefully choosing the best MEC node to process the service profile of the user. Table I summarizes the key parameter notations in our paper.

A. Service Placement Model

To maintain satisfactory Quality-of-Service (QoS), i.e., low service latency for mobile users which typically move erratically, the service profile of each user should be dynamically migrated across multiple edges to follow the user mobility. Here we take a binary indicator $c_i^k(t)$ to denote the dynamic service placement decision variable, let $c_i^k(t) = 1$ if the service profile of user $k \in \mathcal{N}$ is placed at the MEC node $i \in \mathcal{M}$ at time slot t, and $c_i^k(t) = 0$ else. Note that at a given time slot, since each user is served by one and only one MEC node, we have the following constraints for $c_i^k(t)$:

$$\sum_{i=1}^{M} c_i^k(t) = 1, \quad \forall k, t. \tag{1}$$

$$c_i^k(t) \in \{0, 1\}, \quad \forall i, k, t. \tag{2}$$

Based on the above defined service placement decision, we are now readily to formulate the user perceived latency which is determined by the service placement.

B. QoS Model

In the paradigm of MEC, the QoS, i.e., user-perceived latency is jointly determined by the computing delay and communication delay.

Computing delay: at each MEC node, multiple mobile users will simultaneously share the computing resource to process their service requests. Nevertheless, due to the limited computing capacity of the MEC node, dynamic service placement can utilize edge resources efficiently. In this paper, we use $R^k(t)$ to denote the amount of computing capacity (in terms of CPU cycles) required by the service request of user k at time slot t. Taking video stream analytics as an instance [21], the amount of required computation capacity is determined by to the input data size of the video and the corresponding computation intensity of the analytic task (i.e., the amount of computation intensity required for each unit of input data). For ease of exposition, we assume that the computing resource at each MEC node is equally allocated to the users served by this node[1]. Therefore, the computation delay for mobile user k at time slot t is given by

$$D^k(t) = \sum_{i=1}^{M} c_i^k(t) R^k(t) N_i(t) / F_i,$$

where $N_i(t)$ is the number of users served by MEC node i during the time slot t, which follows

$$N_i(t) = \sum_{k=1}^{N} c_i^k(t).$$

Moreover, F_i represents the maximum computing capacity (in CPU cycles per second) of MEC node i.

Communication delay: in MEC, the communication delay between a mobile device and the MEC node generally contains the network propagation delay and the data transmission delay. In particular, the network propagation delay is determined by the network distance, while the data transmission delay is jointly determined by the amount of data transferred and the link bandwidth. Given the service request information as well as the current location of user k, the communication delay to MEC node i can be characterized by a general model $H_i^k(t)$. When considering the service placement decision $c_i^k(t)$, the communication latency experienced by user k can be further expressed as

$$L^k(t) = \sum_{i=1}^{M} c_i^k(t) H_i^k(t).$$

By combining the computing delay $D^k(t)$ and communication delay $L^k(t)$, we denote the total latency experienced by user k at time t as

$$T^k(t) = D^k(t) + L^k(t). \tag{3}$$

C. Migration Cost Model

While dynamic service placement empowers satisfactory QoS by migrating service profiles among edges to follow the user mobility, it is worth noting that cross-edge service migration would incur additional operational cost. Specifically,

[1]Our algorithm can be extended to other resource allocation models, since we do not impose structural assumption on the service computing model in our solution.

when transferring the service profile of each user across edges, enormous usage of the scarce and expensive wide-area-network (WAN) bandwidth would be caused. In addition, cross-edge transferring also increases the energy consumption of network devices such as routers and switches. To model the operational cost incurred by cross-edge service migration, we use $E_{ji}^k(t)$ to denote the cost of migrating the service profile of user k from source MEC node j to destination MEC node i. Without loss of generality, we assume that $E_{ji}^k(t) = 0, \forall j = i$. Then, given the service placement decision $c_i^k(t-1)$ at time slot $t-1$, and $c_i^k(t)$ at time slot t. The service migration cost of user k at time slot t can be computed by $\sum_{i=1}^{M}\sum_{j=1}^{M} c_j^k(t-1)c_i^k(t)E_{ji}^k(t)$. Considering all the N users, the total service migration cost at time slot t can be further denoted as

$$E(t) = \sum_{k=1}^{N}\sum_{i=1}^{M}\sum_{j=1}^{M} c_j^k(t-1)c_i^k(t)E_{ji}^k(t).$$

With the presence of user mobility, it is intuitive that to ensure a desirable level of QoS, the service profile should be actively migrated to follow the user mobility. However, frequent migration would incur excessive operational cost in return. Then, a natural question is how to navigate such a performance-cost trade-off in a cost-efficient manner.

D. Navigating the Performance-Cost Trade-off

To optimize multiple conflicting objectives in a balanced manner, the most commonly adopted approach is to assign different weights to those conflicting objectives and then optimize the weighted sum of them. Unfortunately, in our problem, how to properly defining the weights of performance and cost in realistic environments is not straightforward. In response, considering the fact that network providers generally operate within a long-term (e.g., yearly) cost budget, we propose to optimize the long-term performance under the predefined long-term cost budget. Specifically, we introduce E_{avg} to denote the long-term time-averaged cost budget over a time span of T time slots, which satisfies:

$$\lim_{T\to\infty}\frac{1}{T}\sum_{t=1}^{T}E(t) \leq E_{avg}. \tag{4}$$

Then, our problem of minimizing the long-term time-average service latency under the constraints of long-term cost budget can be formulated as the following stochastic optimization:

$$\begin{aligned}\mathcal{P}1: \quad &\min_{c(t)} \quad \frac{1}{T}\lim_{T\to\infty}\sum_{t=1}^{T}\sum_{k=1}^{N}T^k(t) \\ &\text{s.t.} \quad (1)-(4).\end{aligned} \tag{5}$$

In general, a prime challenge impedes the derivation of the optimal long-term policy $\mathcal{P}1$ is that it requires the future system information (i.e. the user mobility pattern, generated service description), so that network operator can make the global optimal migration decision for request services to achieve a desirable trade-off between average latency and migration cost. Unfortunately, the mobility pattern and request arrival processes are extremely difficult to gain in advance. Moreover, even though the long-term service placement optimization has been decomposed into real-time decoupling problems, preventing frequent service migration with the long-term migration cost constraint is non-trivial. In the current literature, some approaches have been proposed to handle this problem. For example, in [15], by finding an optimal look-ahead window size, the long-term optimization problem can be approximately discretized into a series of equivalent short-path problems. However, the near-future information cannot be predicted accurately for dynamic mobile wireless network. Fortunately, in the queuing theory [27], the long-term migration budget constraint (4) in this optimization problem can be regarded as the queue stability control, i.e., the time-averaged migration $\lim_{T\to\infty}\frac{1}{T}\sum_{t=1}^{T}E(t)$ is beneath the long-term budget E_{avg}. Moreover, Lyapunov optimization technique provides an efficient approach to decouple the long-term problem, which does not require any a priori system information while maintaining the queue stability in an online way. Hence, we propose an online algorithm that transforms the original problem to a series of real-time minimization problems.

IV. Online Service Placement Algorithm

In this section, we describe a novel framework that makes online service placement decisions. To solve the $\boldsymbol{\mathcal{P}1}$, we first convert the original problem to a queue stability control problem based on Lyapunov optimization.

A. Problem Transformation via Lyapunov Optimization

1) The construction of virtual queue for long-term service placement migration cost: Due to the dynamic and stochastic property of the system (e.g., time-varying and uncertainly user mobility and request arrival process), a prime challenge of $\boldsymbol{\mathcal{P}1}$ is to navigate the performance-cost trade-off in a cost-efficient manner without global information over the long run. A key idea of Lyapunov optimization is to strike a desirable balance between current perceived latency and migration cost while maintaining the cost queue stable by introducing virtual queue for the long-term budget. First, we define a virtual queue as a historical measurement of the exceeded migration cost and assume that initial queue backlog is 0 (i.e., $Q(0) = 0$).

$$Q(t+1) = max[Q(t) + E(t) - E_{avg}, 0], \tag{6}$$

where $Q(t)$ is the queue length at time slot t, which represents the over cost on executed service migration by the end of time slot t.

Intuitively, the value of $Q(t)$ can be regarded as an evaluation criteria to assess the migration cost condition. A large value of $Q(t)$ implies the cost has exceeded the long-term budget E_{avg} since carrying out online service placement algorithm. In order to guarantee that the time-averaged service migration cost is lower than budget E_{avg}, i.e., inequality (4) holds, the virtual queue $Q(t)$ must be stable, i.e., $\lim_{T\to\infty}\mathbb{E}\{Q(T)\}/T = 0$. Furthermore, by total summing

the inequality $Q(t+1) \geq Q(t) + E(t) - E_{avg}$ derived from equation (6) and rearranging it, we can gain:

$$\frac{Q(T) - Q(0)}{T} + E_{avg} \geq \frac{1}{T} \sum_{t=0}^{T-1} E(t).$$

For $Q(0) = 0$, we can take expectations of the above inequality and have

$$\lim_{T \to \infty} \frac{\mathbb{E}\{Q(T)\}}{T} + E_{avg} \geq \frac{1}{T} \sum_{t=0}^{T-1} \mathbb{E}\{E(t)\}.$$

Hence, the stability of the virtual queue can ensure that the time-averaged migration cost does not exceed the budget.

2) Queue stability: To stabilize the virtual queue, we first define a quadratic Lyapunov function and Lyapunov drift function respectively as following:

$$L(\Theta(t)) \triangleq \frac{1}{2} Q(t)^2. \tag{7}$$

This represents a scalar measure of cost queue congestion. For instance, a small value of $L(\Theta(t))$ implies the queue backlog is small. Thus, if a policy consistently pushes the quadratic Lyapunov function towards a bounded level, it implies that the virtual queue is stable.

To remain the virtual queue stable, we introduce the *one-step conditional Lyapunov drift* to push the quadratic Lyapunov function towards a lower congestion region:

$$\Delta(\Theta(t)) \triangleq \mathbb{E}\Big[L(\Theta(t+1)) - L(\Theta(t)) | \Theta(t)\Big]. \tag{8}$$

The drift $\Delta(\Theta(t))$ denotes the migration cost queue change in the Lyapunov function over one time slot, which depends on the service placement policy.

3) Joint Lyapunov drift and user-perceived latency minimization: After constructing the virtual cost queue, the original problem has been decomposed into a series of real-time optimization problems. Our goal is to find a current placement policy to coordinate the perceived latency and migration cost. By incorporating queue stability into delay performance, we define a *Lyapunov drift-plus-penalty* function to solve the real-time problem.

$$\Delta(\Theta(t)) + V \sum_{k=1}^{N} T^k(t), \tag{9}$$

where V is a non-negative control parameter that adjust the trade-off between delay performance and migration cost queue backlogs. It shows the attention on the delay performance compared to migration cost budget. Moreover, the following lemma provides the performance guarantee of the drift-plus-penalty function.

Lemma 1: For all possible values of $\Theta(t)$ by using any placement schedule over all time slots, the following statement holds:

$$\begin{aligned} \Delta(\Theta(t)) + V \sum_{k=1}^{N} T^k(t) \leq B + \sum_{k=1}^{N} V\mathbb{E}\Big[T^k(t) | \Theta(t)\Big] \\ + Q(t)\mathbb{E}\Big[E(t) - E_{avg} | \Theta(t)\Big], \end{aligned} \tag{10}$$

where $B = \frac{1}{2}(E_{avg}^2 + E_{max}^2)$ is a constant value for all time slots, and $E_{max} = \max_t E(t)$.

The detailed proof is given in the technical [28]. Based on the *Lemma 1*, the *drift-plus-penalty* function has a supremum bound at every time slot t.

B. Online Service Placement Algorithm

In this section, we convert the problem $\boldsymbol{\mathcal{P}1}$ to a series of real-time drift-plus-penalty supremum bound minimizations. While the *drift-plus-penalty* expression involves the $max[*]$ term in equation (6), which complicates reaching solution to placement issue. Following the lemma 1, we observe that minimizing the right side of inequality (10) can approximate the supremum bound closely, which is equivalent to minimizing the *drift-plus-penalty*. Therefore, based on the aforementioned parameters definition, we rearrange it for a concise form, and obtain an optimal service placement policy $c^*(t)$.

$$\begin{aligned} &\sum_{k=1}^{N} V\mathbb{E}\Big[T^k(t) | \Theta(t)\Big] + Q(t)\mathbb{E}\Big[E(t) - E_{avg} | \Theta(t)\Big] \\ &\leq \sum_{k=1}^{N} \sum_{i=1}^{M} c_i^k(t) \Big(\frac{V R^k(t) \sum_{k=1}^{n} c_i^k(t)}{F_i} + V H_i^k(t) + \rho_i^k(t) \Big), \end{aligned} \tag{11}$$

where $\rho_i^k(t) = \sum_{j=1}^{M} c_j^k(t-1) Q(t) E_{ji}^k$ is a constant at every time slot t, which does not affect the decision making. Thus, the major part of our online service placement algorithm is to solving following $\boldsymbol{\mathcal{P}2}$ to minimize the real-time supremum bound for the *drift-plus-penalty* function.

$$\begin{aligned} \boldsymbol{\mathcal{P}2}: \min_{c(t)} & \sum_{k=1}^{N} \sum_{i=1}^{M} c_i^k(t) \Big(\frac{V R^k(t) \sum_{k=1}^{N} c_i^k(t)}{F_i} + V H_i^k(t) + \rho_i^k(t) \Big) \\ \text{s.t.} \quad & (1) - (4). \end{aligned} \tag{12}$$

For simplify the formulation, we use $U(\boldsymbol{c},\boldsymbol{t})$ to replace the objective function of problem $\boldsymbol{\mathcal{P}2}$, where $\boldsymbol{c}$ is feasible service placement policy. In Algorithm 1, we describe the implementation of the online service placement algorithm. In each time slot t, a close-to-optimal service placement schedule can be obtained when solving $\boldsymbol{\mathcal{P}2}$, and migration cost virtual queue will be updated subsequently for next time slot calculation.

Algorithm 1 Online Service Placement Algorithm

1: **Initialization**: We set the cost queue backlog $Q(0) = 0$ at beginning.
2: **End initialization**

3: **for** each time slot $t = 1, 2, ..., \infty$ **do**
4: **Solve** the problem $\boldsymbol{\mathcal{P}2}$: $c^*(t) = arg\min(12)$.
5: **Update** the virtual queue: run (6) based on $c^*(t)$.
6: **end for**

Unfortunately, this real-time optimization problem is NP-hard in general, due to its combinatorial nature. To address this challenge, we apply Markov approximation [29] to obtain a near-optimal solution for this real-time problem.

C. Approximation Method

Since no computational-efficient approaches to solve problem $\boldsymbol{\mathcal{P}2}$, we design a global service placement optimization that can obtain the minimum solution approximatively. The problem $\boldsymbol{\mathcal{P}2}$ is a combinatorial optimization of finding the optimal service placement policy, we leverage the idea of Markov approximation in [29] to optimize the policy. To proceed, then we can convert the problem $\boldsymbol{\mathcal{P}2}$ to the following equivalent problem:

$$\begin{aligned} \min \quad & \sum_{\mathbf{c}\in c(t)} q_{\mathbf{c}} U(\boldsymbol{c},\boldsymbol{t}) \\ s.t. \quad & \sum_{\mathbf{c}\in c(t)} q_{\mathbf{c}} = 1, \end{aligned} \tag{13}$$

where $q_{\mathbf{c}}$ is a decision variable, which means the probability of the placement policy $\mathbf{c}$ is adopted at current time slot t; $c(t)$ is the collection of all feasible placement policies. Obviously, the optimal solution to problem (13) is to choose the minimum cost placement policy with probability one. The problem can be approximately treated as the following *convex log-sum-exp* problem [29].

$$\begin{aligned} \min \quad & \sum_{\mathbf{c}\in c(t)} q_{\mathbf{c}} U(\boldsymbol{c},\boldsymbol{t}) + \frac{1}{\beta} \sum_{\mathbf{c}\in c(t)} q_{\mathbf{c}} \log q_{\mathbf{c}} \\ s.t. \quad & \sum_{\mathbf{c}\in c(t)} q_{\mathbf{c}} = 1, \end{aligned} \tag{14}$$

where β is a positive constant that charges the approximation ratio of the entropy term. When $\beta \to \infty$, the problem (14) becomes the original problem (13). Besides, it can be implemented in a distributed manner (e.g., only one service can be altered its serving BS when updating the placement policy in each iteration), which is more robust to the dynamic system. If handling the problem with the Karush-Kuhn-Tucker (KKT) [30], we can obtain the optimal solution to problem (14)

$$q_{\mathbf{c}}^{*} = \frac{exp\big(-\beta U(\boldsymbol{c},\boldsymbol{t})\big)}{\sum_{\mathbf{c'}\in c(t)} exp\big(-\beta U(\boldsymbol{c'},\boldsymbol{t})\big)}, \forall \mathbf{c} \in c(t). \tag{15}$$

According to the probability $q_{\mathbf{c}}^{*}$, we can gain the optimal policy. Then, we design a service placement algorithm that constantly updates the placement policy c to form a discrete-time Markov chain [29]. Once the Markov chain achieves to the stationary distribution as shown in (16), the optimal placement profile which minimizes the real-time supremum bound in (13) can be derived by setting parameter β as large as possible. In this algorithm, the Markov chain is irreducible, which traverses all feasible states under different placement policies. Besides, designing a desired time-reversible Markov chains needs to hold the following balance equation:

$$q_{\mathbf{c}}^{*} q_{\mathbf{c},\mathbf{c}'} = q_{\mathbf{c}'}^{*} q_{\mathbf{c}',\mathbf{c}}, \forall \mathbf{c}, \mathbf{c}' \in c(t), \tag{16}$$

where $q_{\mathbf{c},\mathbf{c}'}$ is the probability of the placement policy update from $\mathbf{c}$ to $\mathbf{c}'$.

Algorithm 2 Markov Approximation based Placement Policy Search

1: **Initialization:** Initialize the service placement policy $\mathbf{c}$ as randomly assigning a MEC node for each service.
2: **End initialization**
3: **loop** for each service placement update iteration
4: **Choose** a service k randomly and carry out the following operations:
5: **Calculate** the bound $U(\boldsymbol{c'},\boldsymbol{t})$ for any other feasible service placement policy.
6: **Select** a placement policy acc. to (17) probabilistically.
7: **Update** the service placement policy by placing the service to the new MEC node.
8: **Record** the placement policy $\mathbf{c}$* with the smallest $U(\boldsymbol{c}^{*},\boldsymbol{t})$, found up to now.
9: **end loop**

The Markov approximation based service placement policy algorithm is described in Algorithm 2, which can be implemented in the network operator that can gather sufficient network information and computing capability for real-time decision making. In the algorithm, a random service will be picked to update its placement policy for each update iteration. In this situation, a state transition of services from $\mathbf{c}$ to $\mathbf{c}'$ only occurs if only one user service is migrated. Since knowing the targeting migration policy performance (i.e., supposing we migrate service from MEC node a to MEC node c, the new joint cost of perceived latency and migration in (12) can be calculated easily), the probability of each feasible migration adjustment is directly proportional to the difference of the total cost under two placement policies $\mathbf{c}$ and $\mathbf{c}'$, denoted as follows:

$$q_{\mathbf{c},\mathbf{c}'} = \alpha \exp\Big(-\frac{1}{2}\beta\big(U(\boldsymbol{c'},\boldsymbol{t}) - U(\boldsymbol{c},\boldsymbol{t})\big)\Big). \tag{17}$$

Note that during each policy iteration, the network operator will record the best policy found up to now. As shown in [29], by proper parameter tuning Markov approximation algorithm can converge in a linear rate, and hence we can find the near-optimal placement solution in a fast manner. Next we analyze the complexity of the Markov approximation algorithm. For each update iteration, system chooses arbitrarily a mobile device to update its service placement. During the process of calculating the total cost $U(\boldsymbol{c'},\boldsymbol{t})$ for all feasible placement policies, the possible placement configurations enumerates at most MN. Assuming that this algorithm needs to be executed I iterations to achieve the convergence, then the total time complexity of Algorithm 2 is $O(IMN)$.

V. Theoretical Analysis

In this section, we analyze theoretically the performance of *mobility-aware dynamic service placement algorithm for MEC*, which integrates both Lyapunov optimization and Markov approximation. First, we discuss the optimality gap between the Markov approximation and optimal solution

alone. Then, we compare the performance of our online algorithm (i.e., Markov approximation in the Lyapunov framework) with the offline optimum.

A. Markov Approximation

With above description of Markov approximation algorithm, the probability of a service placement state switch from $\mathbf{c}$ to $\mathbf{c}'$ in the Markov chain is denoted in (17). It is obvious that our algorithm can be converged to a distinctive stationary distribution for its time reversibility.

***Theorem** 1:* There exists a distinctive stationary distribution for the service placement algorithm as stated in equation (16).

The detailed proof is given in the technical report [28]. As shown in Theorem 1, we can obtain the minimal supremum bound for the *drift-plus-penalty* function as the parameter β increasing to a large enough value in our service placement algorithm. We denote the minimal supremum bound and expected supremum bound by proposed algorithm as $S^* = \min \sum_{\mathbf{c}\in c(t)} U(\boldsymbol{c},\boldsymbol{t})$ and $\widetilde{S} = \sum_{\mathbf{c}\in c(t)} q^*_{\mathbf{c}} U(\boldsymbol{c},\boldsymbol{t})$ respectively.

***Theorem** 2:* For the algorithm, the optimality gap is given as following:

$$0 \leq \widetilde{S} - S^* \leq \frac{1}{\beta} \ln |\delta|, \tag{18}$$

where $|\delta|$ is the amount of feasible service placement policies of all mobile users at time slot t.

The detailed proof is given in the technical report [28]. By Theorem 2, the error of worse-case solution in our algorithm is no more than $\frac{1}{\beta} \ln |\delta|$. Thus, if setting the of value parameter β as large as possible, we can approach an almost equivalent solution to the minimal supremum bound. Fortunately, the value of β is usually large enough in an acceptable scope, the performance deviation of the optimum is quite small [29].

B. Optimality Analysis

As we have mentioned, the transformed problem $\boldsymbol{\mathcal{P}2}$ is NP-hard. Fortunately, the minimization error of the $\boldsymbol{\mathcal{P}2}$ is acceptable under the control of our online algorithm. We use $\widetilde{T}^k(t)$ and T^{opt} to respectively denote the delay performance in time slot t by the proposed algorithm and the infimum time average performance delay with the overall information. Then the following theorems will give a supremum bound of the time-averaged delay performance and the migration cost queue backlogs.

***Theorem** 3:* Assume the migration cost $E(t)$ is i.i.d. over time slots, for any non-negative control parameter V, the long-term delay performance implemented by online algorithm satisfies that

$$\lim_{T\to\infty} \frac{1}{T} \sum_{t=0}^{T-1} \sum_{k=1}^{N} \mathbb{E}\{\widetilde{T}^k(t)\} \leq T^{opt} + \frac{B}{V} + \frac{1}{\beta V} \ln |\delta|. \tag{19}$$

***Theorem** 4:* Assuming that $E_{avg} > 0$ and initializing the migration cost queue backlog is 0, thus for all time slots we having the following bound:

$$\lim_{T\to\infty} \frac{1}{T} \sum_{t=0}^{T-1} \mathbb{E}\{Q(t)\} \leq \frac{B + VT^{opt}}{\varepsilon} + \frac{1}{\beta\varepsilon} \ln |\delta|. \tag{20}$$

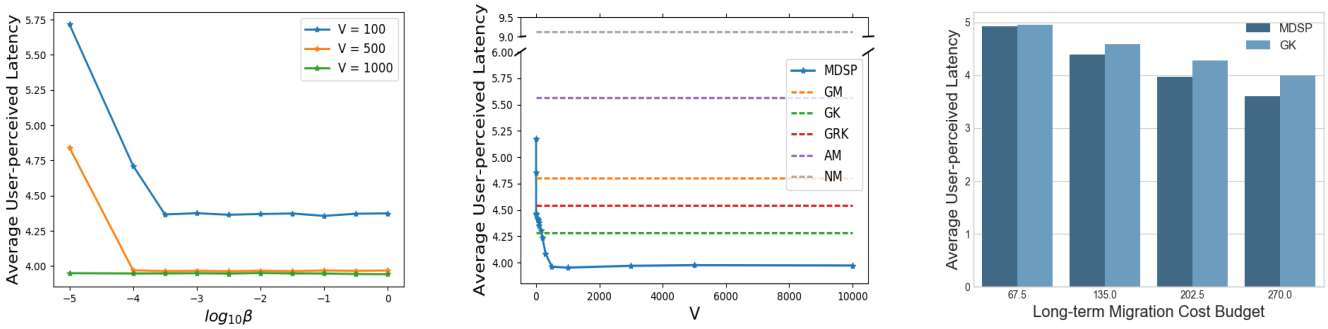
The detailed proof is given in the technical report [28]. Where $\varepsilon > 0$ is a finite constant that represents the distance between the time-averaged migration cost by some control policy and long-term cost budget. From Theorem 1, it is known that the delay performance of the online algorithm can be approached closely to the offline optimum with the adjustable control parameter increasing V. Besides, the bound of migration cost queue backlog is also determined by the parameter V. In short, a performance-cost trade-off of $[O(1/V), O(V)]$ exists in our online algorithm, where we can set the parameter V to a desirable value to achieve the balance of the long-term delay performance and migration cost.

VI. Performance Evaluation

In this section, we conduct numerical studies to evaluate the time-averaged perceived latency performance under the long-term migration cost constraint of the proposed algorithms and to verify the derived theoretical results.

A. Simulation Setup

We adopt the ONE simulator [31] to conduct system simulation, where mobile devices move along the roads or streets based on Shortest Path Map-Based Movement Model [31] in a downtown Helsinki, Finland. The size of the simulation area is 4500×3400 m^2. For simplicity, we divide the whole area into 63 square parts. Each part occupies 500×500 m^2, endowed with one MEC node to provide mobile services. Besides, we set two kinds of mobile users: about 85.7% $(\frac{6}{7})$ of the mobile users is pedestrians with speed uniformly distributed in $[0.5, 1.5]$ m/s, the remaining users is drivers with speed uniformly distributed in $[2.7, 11.1]$ m/s. The hop distance between two MEC nodes is calculated by Manhattan distance. We simulate 2000 time slots for our system, and the interval of a time slot is 5 minutes. During each time slot t, we assume that the placement for service profile of users and wireless connections between user and edge are unchanged. The request arrival process $R^k(t)$ for each user k is uniformly distributed within [0.3168, 0.528] of the maximum MEC computing capacity F_i, which is available for all mobile service executions. To simplify the problem, we assume that the maximum computing capacity of all MEC nodes is the same and the current communication delay follows uniform distribution within [1, 1.35] of the optimal delay, which is 0.6 min per hop for every service. It is the same as migration cost, perturbed by timing a random parameter in [1, 1.35]. The difference is that one hop migration takes 1 unit cost, and plus 0.5 unit cost in the end, which is allied to the request arrival process $R^k(t)$.



(a) Averge time cost with different values of control parameter V and β

(b) Averge time cost with different values of control parameter V under different migration policies

(c) Averge time cost with different long-term budgets E_{avg} under distributed algorithm

Fig. 2: Optimality analysis

B. Performance Benchmark

We consider two representative situations and three greedy approaches as a benchmark to evaluate the performance of *mobility-aware dynamic service placement algorithm for MEC* (MDSP). One situation is no matter what the distribution of mobile user is, the service VM is always migrated to execute on its nearest MEC node, i.e., "Always migration" (AM) strategy. On the contrary, another is always to keep the initial assignment policy unchanged, i.e., "No migration" (NM) strategy. Furthermore, the three greedy algorithms are described as follow:

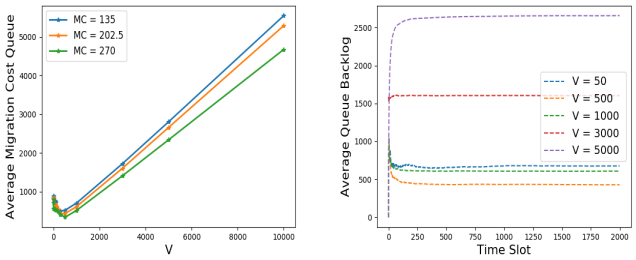
1) Greedy for migrating services to the current location positions (*GM*): this algorithm migrates the request services to the nearest MEC node at every opportunity over a long period of time.
2) Greedy for randomly migrating different K services to the positions where the time cost of each service is minimum (*GRK*): this algorithm randomly picks up different K services and migrates them to the corresponding MEC nodes where at every opportunity over a long run.
3) Greedy for migrating K services in descending order by time cost to the positions where the time cost of each service is minimum (*GK*): this algorithm migrates K services in sequence to the corresponding MEC nodes where at every opportunity over a long run.

C. Latency Cost Trade-off

Obviously, the optimization of the user-perceived latency and migration trade-off in a cost-efficient manner is the key to the long-term service placement problem, which guides the following analysis for our MDSP algorithm.

Average user-perceived latency optimality. To analyze key elements to influence the user-perceived latency, we formulate a standard of comparison, where 315 mobile users move in the city and the long-term time-averaged migration cost budget for the network operator is set as 202.5 cost units, approximation control parameter β is set as 0.1. Fig. 2(a) depicts the average user-perceived latency under different values of control parameter β and V. We find that the average latency decreases in inverse proportion and finally converges to the minimum value as β increases with $V = 100$ and $V = 500$. And when the value of V is 1000, the average latency does not change dramatically, which means the large V weakens the influence of parameter β. Fig. 2(b) shows the average latency with different values of control parameter V under various online algorithms. We can observe that the average latency decreases with V increasing, and gradually approaches a minimum value in our MDSP algorithm. This confirms Theorem 3 we have mentioned in theoretical analysis that the time-averaged latency performance is proportional to the $1/V$. Besides, compared with the benchmark, MDSP algorithm does have remarkable improvement in average latency performance, especially in the AM and NM strategy. For the AM strategy, the major reason for the poor performance is the low utilization of edge resources. In general, only almost two-thirds of the MEC nodes provide all user services during every time slot t. Even though GRK and GK make up this deficiency of the inefficient utilization, the unreasonable migration policy still exists since every migration selection update is a local optimization. We find the particular migration sequence, such as descending order, can alleviate the performance gap to some extent. Fig. 2(c) shows the time-averaged latency performance with different 4 long-term budget under MDSP and GK algorithms. Intuitively, the larger migration cost budget can provide more opportunities for further enhancements on the local optimization (i.e., GRK, GK). When the long-term migration cost is small, the difference of average latency performance is tiny between MDSP and GK algorithms. While as the migration cost budget increases, the more notable improvement appears in our MDSP algorithm.

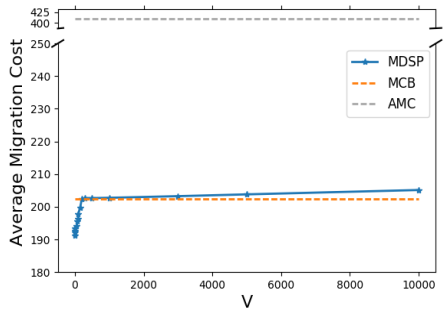
Queue stability. In Fig. 3 (a) plots the time-averaged migration cost queue with different values of control parameter V under three different long-term time-averaged migration cost budget. Broadly, as V increases, the time-averaged backlog queue increases in a linear fashion, which is matched in Theorem 4. Along with Fig. 2(a), the performance of time-averaged latency and migration cost follows the $[O(1/V), O(V)]$ trade-off. As shown in Fig. 3(b), the varying curve of average



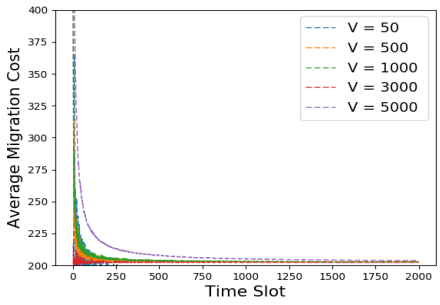
(a) Averge migration cost queue with different values of control parameter V and migration cost budget

(b) Averge migration cost queue with different values of control parameter V at each time slot

Fig. 3: Queue stability



(a) Averge migration cost with different values of control parameter V

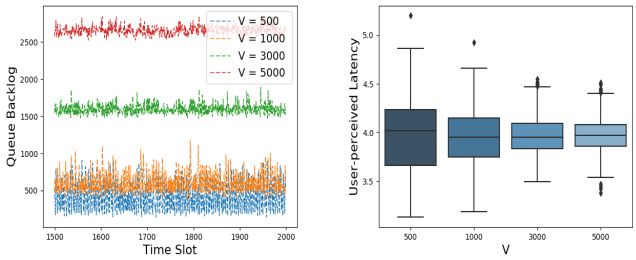


(b) Averge migration cost with different values of control parameter V at each time slot

Fig. 4: Convergence of average migration cost

migration backlog queue gradually become stable in our algorithm.

Convergence of average migration cost. Fig. 4(a) plots the average migration cost with different values of V under MDSP algorithm. Note that the migration cost budget (MCB) is almost a half of the all services migration cost (AMC). In this situation, a large value of V makes system care more about user-perceived latency, which may violate the long-term cost budget in finite time slots, such as $V = 5000$. While in Fig. 4(b), as time slot increases, the average migration cost decreases remarkably and gradually converges to the migration cost budget under different values of control parameter V. The reason for this problem is insufficient time slots. As we



(a) Dynamic adjustment for migration cost queue at each time slot

(b) Distribution of user-perceived latency with different values of control parameter V

Fig. 5: Adaptability to system dynamics

have discussed, if the migration cost queue is stable, i.e., $\lim_{T\to\infty} \mathbb{E}\{Q(T)\}/T = 0$, the actual migration cost would not violate the budget. In Fig. 2(e), we know that all migration backlog queues gradually converge to some certain finite value. Thus, if increasing time slots, the long-term constraint can be satisfied.

Adaptability to system dynamics. To further explore the dynamic adjustment of our algorithm to minimize the average latency performance under the control of migration cost budget, we depict a part of real-time fluctuation of the migration backlog queue and the distribution of latency performance with different values of V. As shown in Fig. 5(a), for the clear fluctuation exposition, we select a partial time snippet from the long run. It can be observed that the real-time migration backlog queue fluctuates frequently, which means the system will adjust migration policy frequently. When the current migration backlog queue is large and thus the remaining available migration cost is relatively scarce, the MDSP algorithm will endeavor to reduce the queue backlog to prevent the over-budget. Contrarily, when the current migration backlog queue is small, minimizing the time cost is the prime goal since the remaining available migration cost is abundant. Surprisingly, the large value of V reduces the fluctuation range of the migration backlog queue, which makes system real-time latency performance more stable. In fig. 5(b), the distribution of latency performance is more centralized to median the with the increases of V, which is consistent with the dynamic adjustment of migration backlog queue.

D. Efficient in Different Dense Networks

Fig. 6 suggests that our MDSP algorithm still works efficiently in different dense networks (i.e., the percentage ratio of user amount to MEC node amount). Higher user dense network will lead to the rapid growth of computation delay, which is the main factor of perceived latency increasing. Our algorithm can balance the edge load by migrating services to slow the total perceived latency growth. Nevertheless, the computation delay has been growing significantly faster than total perceived latency, while the network propagation delay is not affected by user amount. To maintain the original quality of service, network operator should improve the computation capacity of edges accordingly.

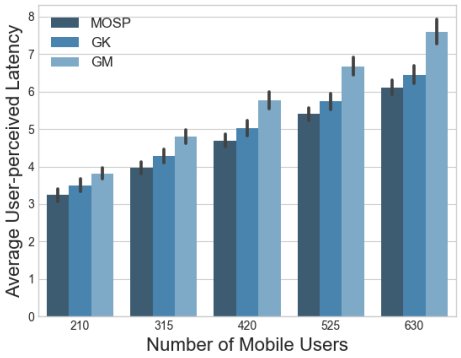


Fig. 6: Average user-perceived latency with different dense networks.

VII. Conclusion

In this paper, we study the mobile edge service performance optimization problem with long-term time-averaged migration cost budget. We design a novel mobility-aware online service placement framework to achieve a desirable balance between time-averaged user-perceived latency and migration cost. To tackle the unavailable future system information, which involves mobility pattern and request arrival processes, we utilize Lyapunov optimization technique to incorporate the long-term budget into a series of real-time optimization problems. Since the decomposed optimization is an NP-hard problem, we develop an efficient heuristic based on the Markov approximation technique to approach a near-optimal solution. Furthermore, we provide the theoretical proof of our proposed algorithm performance. Besides, extensive simulation demonstrates the effectiveness of our online algorithm while maintaining the long-term migration cost constraint.

References

[1] "Cisco Global Cloud Index: Forecast and Methodology, 2014–2019," *White Paper, Cisco*, 2013.

[2] J. G. Andrews, S. Buzzi, W. Choi, S. V. Hanly, A. Lozano, A. C. K. Soong, and J. C. Zhang, "What Will 5G Be?" *IEEE Journal on Selected Areas in Communications*, vol. 32, no. 6, pp. 1065–1082, 2014.

[3] M. Patel, B. Naughton, C. Chan, N. Sprecher, S. Abeta, A. Neal *et al.*, "Mobile-Edge Computing Introductory Technical White Paper," *White Paper, Mobile-Edge Computing (MEC) Industry Initiative*, 2014.

[4] A. Ahmed and E. Ahmed, "A Survey on Mobile Edge Computing," in *Proc. of International Conference on Intelligent Systems and Control (ISCO)*, 2016.

[5] Y. C. Hu, M. Patel, D. Sabella, N. Sprecher, and V. Young, "Mobile Edge Computing - A Key Technology towards 5G," *White Paper, ETSI*, 2015.

[6] X. Chen, L. Jiao, W. Li, and X. Fu, "Efficient Multi-User Computation Offloading for Mobile-Edge Cloud Computing," *IEEE/ACM Transactions on Networking*, vol. 24, no. 5, pp. 2795–2808, 2016.

[7] X. Chen, "Decentralized Computation Offloading Game for Mobile Cloud Computing," *IEEE Transactions on Parallel and Distributed Systems*, vol. 26, no. 4, pp. 974–983, 2015.

[8] D. Xenakis, N. Passas, L. Merakos, and C. Verikoukis, "Mobility Management for Femtocells in LTE-Advanced: Key Aspects and Survey of Handover Decision Algorithms," *IEEE Communications Surveys & Tutorials*, vol. 16, no. 1, pp. 64–91, 2014.

[9] X. Chen, L. Pu, L. Gao, W. Wu, and D. Wu, "Exploiting Massive D2D Collaboration for Energy-efficient Mobile Edge Computing," *IEEE Wireless Communications*, vol. 24, no. 4, pp. 64–71, 2017.

[10] C. Shen, C. Tekin, and M. van der Schaar, "A Non-Stochastic Learning Approach to Energy Efficient Mobility Management," *IEEE Journal on Selected Areas in Communications*, vol. 34, no. 12, pp. 3854–3868, 2016.

[11] J. W. Jiang, T. Lan, S. Ha, M. Chen, and M. Chiang, "Joint VM Placement and Routing for Data Center Traffic Engineering," in *Proc. of IEEE INFOCOM*, 2012.

[12] 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.

[13] A. Fischer, J. F. Botero, and M. T. Beck, "Virtual Network Embedding: A Survey," *IEEE Communications Surveys & Tutorials*, vol. 15, no. 4, pp. 1888–1906, 2013.

[14] A. Nadembega, A. S. Hafid, and R. Brisebois, "Mobility Prediction Model-based Service Migration Procedure for Follow Me Cloud to Support QoS and QoE," in *Proc. of International Conference on Communications Communications (ICC)*, 2016.

[15] 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, 2017.

[16] T. Taleb and A. Ksentini, "An Analytical Model for Follow Me Cloud," in *Proc. of IEEE Global Communications Conference (GLOBECOM)*, 2013.

[17] A. Ksentini, T. Taleb, and M. Zafer, "A Markov Decision Process-based Service Migration Procedure for Follow Me Cloud," in *Proc. of International Conference on Communications Communications (ICC)*, 2014.

[18] S. Wang, R. Urgaonkar, and T. He, "Mobility-Induced Service Migration in Mobile Micro-clouds," in *Proc. of IEEE Military Communications (MILCOM)*, 2014.

[19] S. Wang, R. Urgaonkar, and M. Chen, "Dynamic Service Migration in Mobile Edge-Clouds," in *Proc. of IFIP Networking Conference (IFIP Networking)*, 2015.

[20] M. Srivatsa, R. Ganti, J. Wang, and V. Kolar, "Map Matching: Facts and Myths," in *Proc. of the 21st ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems*, 2013.

[21] 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.

[22] R. Urgaonkar, S. Wang, T. He, M. Zafer, K. Chan, and K. K. Leung, "Dynamic Service Migration and Workload Scheduling in Edge-Clouds," *Performance Evaluation*, vol. 91, pp. 205–228, 2015.

[23] P. Mach and Z. Becvar, "Mobile Edge Computing: A Survey on Architecture and Computation Offloading," *IEEE Communications Surveys & Tutorials*, vol. 19, no. 3, pp. 1628–1656, 2017.

[24] Z. Zhou, F. Liu, H. Jin, B. Li, B. Li, and H. Jiang, "On Arbitrating the Power-Performance Tradeoff in SaaS Clouds," in *Proc. of IEEE INFOCOM*, 2013.

[25] L. Chaufournier, P. Sharma, F. Le, E. Nahum, P. Shenoy, and D. Towsley, "Fast Transparent Virtual Machine Migration in Distributed Edge Clouds," in *Proc. of the Second ACM/IEEE Symposium on Edge Computing*, 2017.

[26] L. Ma, S. Yi, and Q. Li, "Efficient Service Handoff Across Edge Servers via Docker Container Migration," in *Proc. of the Second ACM/IEEE Symposium on Edge Computing*, 2017.

[27] M. J. Neely, "Stochastic Network Optimization with Application to Communication and Queuing Systems," *Synthesis Lectures on Communication Networks*, vol. 3, no. 1, pp. 1–211, 2010.

[28] T. Ouyang, Z. Zhou, and X. Chen, "Follow Me at the Edge: Mobility-Aware Dynamic Service Placement for Mobile Edge Computing," Tech. Rep., 2018. [Online]. Available: https://mega.nz/#!XmhWBB4a!R7XTImpNvxrZTCaDYnhtHIZGOGydbGcbN3OpAkjssIY

[29] M. Chen, S. C. Liew, Z. Shao, and C. Kai, "Markov Approximation for Combinatorial Network Optimization," in *Proc. of the IEEE INFOCOM*, 2010.

[30] S. Boyd and L. Vandenberghe, *Convex Optimization*. Cambridge University press, 2004.

[31] A. Keränen, J. Ott, and T. Kärkkäinen, "The ONE Simulator for DTN Protocol Evaluation," in *Proc. of the 2nd International Conference on Simulation Tools and Techniques*, 2009.