QoS-aware Online Service Provisioning and Updating in Cost-efficient Multi-tenant Mobile Edge Computing

Shuaibing Lu, *Member, IEEE,* Jie Wu, *Fellow, IEEE,* Pengfan Lu, Ning Wang, Haiming Liu, and Juan Fang, *Member, IEEE*

Abstract—The vigorous development of IoT technology has spawned a series of applications that are delay-sensitive or resource-intensive. Mobile edge computing is an emerging paradigm that provides services between end devices and traditional cloud data centers to users. However, with the continuously increasing investment of demands, it is nontrivial to maintain a higher quality-of-service (QoS) under the erratic activities of mobile users. In this paper, we investigate the service provisioning and updating problem under the multiple-users scenario by improving the performance of services with long-term cost constraints. We first decouple the original long-term optimization problem into a per-slot deterministic one by using Lyapunov optimization. Then, we propose two service updating decision strategies by considering the trajectory prediction conditions of users. Based on that, we design an online strategy by utilizing the committed horizon control method looking forward to multiple slots predictions. We prove the performance bound of our online strategy theoretically in terms of the trade-off between delay and cost. Extensive experiments demonstrate the superior performance of the proposed algorithm.

Index Terms—mobile edge computing, online service provisioning, cost-efficient, quality-of-service (QoS).

1 Introduction

The vigorous development of Internet of things (IoT) technology has led to the explosive growth of mobile terminal equipment and data volume. At the same time, a series of resource-intensive and delay-sensitive applications, such as augmented reality (AR)/virtual reality (VR), intelligent driving, and dynamic content delivery, have emerged and been widely used [1], [2], [4], [6]. It is difficult for the traditional cloud data center to meet the performance requirements due to the long distance from massive terminals. Mobile Edge Computing (MEC) is a promising framework for solving this problem by deploying edge servers at base stations to supply computation, storage, and networking resources for multiple users [3]. Ensuring the quality of service (QoS) is a key challenge with significant real-world implications, as the vigorous development of IoT technology has generated numerous delay-sensitive or resource-intensive applications. However, the finite capabilities of edge servers and the erratic activities of multiple end-users pose challenges in guaranteeing the QoS. Therefore, there are two key problems: (i) How to guarantee the QoS to avoid service interruption with unknown trajectories when users are away from the original edge servers? (ii) How to realize service provisioning, and updating the services that can efficiently utilize the limited resources without overwhelming the cost constraint? In this paper, we investigate the service provisioning and updating problem under the multiple-users scenario by improving the performance of services with a long-term cost constraint.

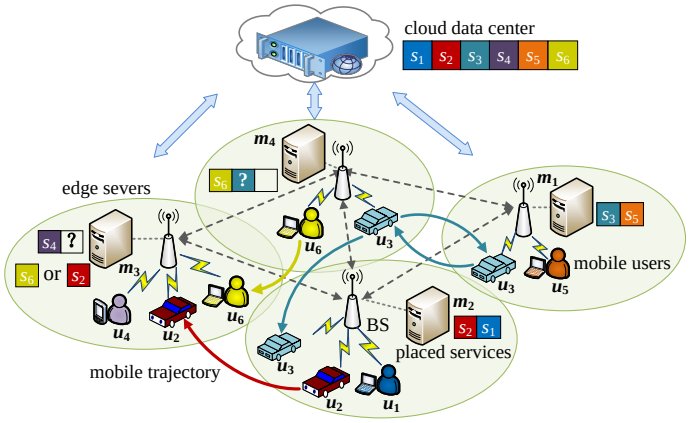


Fig. 1. An illustrating example.

1.1 Motivation and Challenges

Service provisioning and updating refers to the decision-making process of the locations of services within a specific edge computing network to balance the interests of service providers and consumers to achieve the greatest efficiency possible [7]. Numerous mobile devices and sensors in edge networks need to interface with services and exchange data in real-time, which requires efficient service provisioning

- *Shuaibing Lu, Pengfan Lu, and Juan Fang are with the Faculty of Information Technology, Beijing University of Technology, Beijing, China, 100124.*
  E-mail: lushuaibing@bjut.edu.cn.
- *Jie Wu is with the Department of Computer and Information Sciences, Temple University, 1805 N. Broad .Street, Philadelphia, PA 19122.*
  E-mail: jiewu@temple.edu.
- *Haiming Liu is with the School of Software Engineering, Beijing Jiaotong University, Beijing, China, 100091.*
  E-mail: liuhaiming@bjtu.edu.cn.

and updating strategy to enhance the capacity to accommodate real-time data processing. We illustrate the motivation and challenges of the online service provisioning and updating problem by using an example in Figure 1. The squares with six different colors represent the services s_1 to s_6, which are initially provisioned in the cloud data center. We assume that the services required by the users have been deployed on the edge servers, and each service only serves one user. For mobile users, the QoS can be guaranteed through provisioning a replication or migration among edge servers. (i) The trajectories of multiple users are diverse and erratic, hence it is non-trivial to find an efficient strategy that can improve the QoS of mobile users by considering the cost constraint. Taking service s_3 as an example, we suppose that end user u_3 moves from an area in m_1 to m_4 at time slot t and goes back to m_1 after several slots. One extreme solution is to migrate or provision a replication of s_3 on edge server m_4 which may bring a lower delay for user u_3. However, the total cost will be the maximum one among all feasible assignments if the replication or migration costs of services are extremely high. Another extreme assignment is to retain service s_3 within m_1, which minimizes the extra (replication or migration) cost. When u_3 moves to m_4, the service can only be enjoyed through communication with m_1, which will make the quality of service decrease. Therefore, when and where to migrate or replicate services is crucial for balancing the trade-off between long-term cost and users' total delay. (ii). Since the capabilities of edge servers are limited, determining which services are chosen to be placed in order to obtain a better performance when multiple users make the same decision at the same time is non-trivial. Taking services s_2, s_3, and s_6 as examples, we suppose that users u_2 and u_6 move from m_1 towards m_4 during the same time slot. If both services want to migrate or replicate to m_3, there will be a conflict due to the fact that the remaining capacity can only receive one service. Let's consider a more serious scenario such that if m_3 finally agrees to allow service s_6 to migrate or replicate on itself, which means that service s_2 is still locating on m_2, while during that time slot, if user u_3 moves from server m_4 to m_2, there will be no choice for s_3 due to the limitation of capacity on m_2. Therefore, the problem of how to make a better choice by jointly considering the resource efficiency and users' performance is a challenge.

1.2 Contributions and Paper Organization

In this paper, we investigate the online service provisioning and updating problem under the multiple-users scenario by improving the performance of services with long-term cost constraints in mobile edge computing. Our contributions can be summarized as follows:

- We investigate the online service provisioning and updating problem by formulating to minimize the average long-term delay of multiple mobile users, and we decouple the original long-term optimization problem into a per-slot deterministic one by using Lyapunov optimization.
- We propose two service updating decision strategies by considering the trajectory prediction conditions of users. For one scenario, namely the service updating without available predictive information, we propose a novel strategy by introducing the conflict resolution factor. For the other scenario, which is the service updating with multi-step prediction, we optimize the total delay of users per-slot by converting a weighted graph under the constructed activity set.
- Based on that, we design an online strategy by utilizing the committed horizon control method looking forward to multiple slots predictions. We prove the performance bound of our online service provisioning strategy theoretically in terms of the trade-off between delay and cost.
- We conduct extensive experiments to compare our strategy with several baselines based on the Microsoft GPS trajectory dataset which was reconstructed by 40 users. The results are shown from different perspectives to provide conclusions. Extensive experiments demonstrate the superior performance of the proposed algorithm.

TABLE 1
Symbols and Definitions

Symbols	Definitions
$\mathbf{M}$	Set of mobile edge servers, where $\mathbf{M} = \{m_j\}$.
$\mathbf{S}$	Set of services on cloud data center, where $\mathbf{S} = \{s_h\}$.
$\mathbf{S}_{m_i}(t)$	Set of services provisioning on m_i at t.
$\mathbf{U}$	Set of mobile users, where $\mathbf{U} = \{u_i\}$.
$\hat{\mathbf{U}}(t)$	The activity set of users at time slot t.
$\mathbb{D}_{u_i}(t)$	Total delay of user u_i at time slot t.
$D^c_{u_i}(t)$	Computing delay of user u_i at time slot t.
$D^l_{u_i}(t)$	Communication delay of user u_i at time slot t.
$D^u_{u_i}(t)$	Updating delay of user u_i at time slot t.
$\mathbb{C}_{s_h}(t)$	Total cost of s_h at time slot t.
$C^m_{s_h}(t)$	Migration cost of s_h at time slot t.
$C^r_{s_h}(t)$	Replication cost of s_h at time slot t.
η_h	The conflict resolution factor of service s_h.
$L_{u_i}(t)$	The location of u_i at time slot t.
$\widehat{\mathbf{L}}_{u_i\|[\tau,\tau+\omega]}$	The trajectory of u_i in a ω steps prediction window.

The remainder of this paper is organized as follows. Section 2 surveys related works. Section 3 describes the model and then formulates the problem. Section 4 investigates the service provisioning and updating problem based on Lyapunov optimization. Section 5 investigates the online optimization provisioning strategy by considering multi-step prediction. Section 6 includes the experiments. Finally, Section 7 concludes the paper.

2 Related Work

As an emerging paradigm, edge computing extends services closer to end-users. However, the finite capabilities of edge servers and the erratic activities of users pose new challenges [5]. One of the main open branches is the service provisioning problem, which is well-investigated in edge computing under mobility scenarios [6]. Various works have been studied from different aspects of this problem. Elgazzar et al. [8] examined the reliability of mobile devices as data service providers and proposed a cloud-assisted framework that leverages task offloading to improve service performance. Qiu et al. [9] addressed the deployment

optimization of VNFs and backup VNFs in a fault-prone MEC environment with dynamically changing fault probabilities. Li et al. [10] concentrated on the reliability of Virtual Network Functions (VNFs) in the MEC network and suggested deploying primary and backup VNF instances to meet user reliability requirements. Yu et al. [11] investigated the service provisioning problem in mobile edge computing, which aimed to minimize the traffic load caused by service request forwarding, and proposed an efficient decentralized algorithm based on matching theory. Mao et al. [12] proposed an approximate algorithm to deploy service function chains at the edge and the cloud, and they used the next fit strategy and double spanning tree to effectively avoid redundant data traffic and reduce the latency. The resource cost at the edge and on the cloud, and the optimal point of all corresponding communication delays were optimized. Gu et al. [13] proposed the JMDLS-RR algorithm, which cooperatively deploys microservices by combining intra-server and inter-server layer sharing to maximize the service capacity of the edge cloud. Nezami et al. [14] formulated a decentralized load-balancing problem for IoT service provisioning, and they introduced a decentralized multi-agent system that utilized edge servers to balance the workloads and minimized the costs involved in service execution. Zhang et al. [15] solved the computation and delay costs minimization problem by proposing an efficiently approximate algorithm based on semi-definite relaxation. The above works optimized the cost and delay of services from the offline scenario.

In the online scenario, Chen et al. [16] studied the service collaboration with master-slave dependency among service chains of mobile users and jointly optimized the cost and delay by introducing a distributed algorithm based on Markov approximation. Xu et al. [17] proposed an efficient online algorithm based on Gibbs sampling which can achieve provable close-to-optimal performance. Ren et al. [18] proposed a novel framework called EdgeMatrix and designed a multi-task mechanism to solve the problem of joint service orchestration and request scheduling between edge cloud clusters. They used a dual timescale framework which coordinated resources and services on a large timescale and scheduled requests on a small one, which can significantly shorten the running time. Shang et al. [19] designed an online service provisioning and throughput adjustment algorithm to coordinate the migration of virtual services, as well as adjusted its data throughput according to real-time bandwidth fluctuations to reduce latency and improve the Quality of Experience (QoE). They solved the support challenge of interactive virtual service QoE in mobile edge computing caused by high user mobility and unstable network conditions. Wang et al. [20] described dynamic task placement as an online multi-user dobby slot machine process and proposed a decentralized algorithm to optimize users' rewards affected by network delays. Han et al. [21] transformed the online multi-component service placement into an ant colony optimization problem, and they proposed a level traversal component ranking method to achieve faster convergence. In the online optimization problem of edge computing, some existing works utilize the Lyapunov optimization method. Li et al. [22] proposed a two-timescale algorithm based on Lyapunov optimization, which achieves efficient performance close to offline optimal results by purchasing computing resources and making task offloading decisions at different timescales. Liu et al. [23] discussed the challenge of obtaining feasible computation offloading strategies in an online environment due to limited resources of unmanned aerial vehicles (UAVs) and dynamic changes in applications and environments, which achieved long-term efficient and stable performance. In cloud computing system optimization, the application of Lyapunov optimization techniques is also a promising approach. Zhou et al. [24] proposed an analytical framework for optimizing the trade-off between power consumption and performance in SaaS cloud platforms, and they used Lyapunov optimization techniques to design an optimal control framework for online decision-making on request admission control. Fang et al. [25] designed a stochastic control algorithm using Lyapunov optimization and weight perturbation techniques, which achieved the maximization of profit for the management platform through online decision-making. Qi et al. [26] proposed an online scheduling algorithm based on Lyapunov optimization to optimize the operation of service systems in a cloud environment, utilizing queue stability to ensure Quality of Service (QoS). These works focus on optimizing the cost and delay of the service provisioning problem; however, they ignore the erratic movements of users.

In order to tackle the challenge of users' mobility, some existing works were proposed based on service migration. Ning et al. [27] studied the service provisioning problem by constructing a stochastic mobility system, and they introduced a distributed Markov approximation algorithm which is linear to the number of users in order to determine the configurations of services provisioning. T. Kim et al. [28] proposed a system called MoDEMS to optimize the service provisioning based on user mobility in edge computing, and they developed a linear integer programming problem and a Seq-Creedy heuristic method to generate a migration plan to minimize system costs and user delays. Zeng et al. [29] formulated an optimization problem to jointly decide the service provisioning policy and the routing decision, and they developed an online distributed algorithm with provable performance guarantees in terms of convergence and competitive ratio. Li et al. [30] focused on the service migration problem for mobile users through modeling a Markov Decision Process (MDP) model, and they solved it by using deep reinforcement learning. In addition, some works consider using the information of the prediction. Liu et al. [31]. introduced a prediction-based dynamic task assignment algorithm that assigned the workloads to edge servers based on the prediction of capacities and costs in each time slot. Jin et al. [32] designed a set of novel polynomial-time algorithms to make adaptive decisions by solving continuous solutions. These continuous solutions are based on the predicted inputs about the dynamic and uncertain cloud-edge environments via online learning. Ma et al. [33] propose a multiple-slots predictive service placement algorithm to incorporate the prediction of user mobility based on a frame-based design. However, these works do not take into account the impact of additional prediction error on the service provisioning. In this paper, we study the online service provisioning and updating problem in mobile

edge computing. Our objective is to improve the QoS by minimizing the total delay while considering maintaining the long-term cost under the constraint.

3 Model and Problem Formulation

In this paper, we focus on the QoS-aware online service provisioning and updating problem in mobile edge computing for multiple users with cost efficiency. We build a system model that describes the physical edge nodes and multiple users, and then we formalize our problem to minimize the long-term average delay under the constraints within this model.

3.1 System Model

As shown in Figure 1, we consider a three layer network architecture that includes the cloud data center, edge servers, and the mobile end-users. We suppose that the services required by users are initially provisioning in the cloud data center, which is denoted as set $\mathbf{S} = \{s_h\}$. Let $\mathbf{M} = \{m_j\}$ denote a substrate set of edge servers that are supported by the operators. Let $\mathbf{U} = \{u_i\}$ denote the set of mobile users, and these users subscribe to the services one-to-one. In order to capture the mobility of users, we assume that the system in a slotted structure and its timeline is discretized into time frame $t \in \{0, 1, 2, ...T-1\}$ [33], [34], [37]. In this paper, we suppose that users move erratically and frequently among several edge servers. At each time slot, the operators determine whether provisioning replications or migration follow with users according to navigating the trade-off between delay and cost. We list the main notations throughout this paper in Table 1 for ease of reference.

3.1.1 QoS model

In our study, the QoS of users is determined by computing delay, communication delay, and updating delay. We use $\mathbb{D}(t) = \sum_{i=1}^{|\mathbf{U}|} \mathbb{D}_{u_i}(t)$ to denote the total delay at time slot t, where $\mathbb{D}_{u_i}(t)$ is the delay of u_i. The computing delay is defined as $D^c_{u_i}(t) = \sum_{m_j \in \mathbf{M}} \frac{r_{u_i}(t)}{z^c_{m_j}}$, where $r_{u_i}(t)$ is the service request of user u_i at time slot t, and $z^c_{m_j}$ is the computing capacity of m_j measured by the number of CPU cycles [35], [36]. We use $D^l_{u_i}(t)$ to represent the communication delay that occurs when users are far away from the location of the service. Let t_{u_i,m_j} denote the maximum transmission rate between user u_i and edge server m_j, where $t_{u_i,m_j}(t) = b_{u_i,m_j}(t) \cdot \log_2(1 + \frac{\beta \cdot g(u_i,m_j)}{N})$ [37], [38], [44], [45]. We set $b_{u_i,m_j}(t)$ as a binary indicator variable indicating whether user u_i is connected to server m_j. Here, by β we denote the transmission power of the local mobile device of u_i. Let $g(u_i, m_j)$ be the channel gain between the user u_i and the edge server m_j, where $g(u_i, m_j) = 127 + 30 \cdot \log p(u_i, m_j)$ [39]. $p(u_i, m_j)$ represents the distance between u_i and m_j that determine the network propagation. We use N to represent the noise power. The communication delay is defined as $D^l_{u_i}(t) = \sum_{m_j \in \mathbf{M}} \frac{d_{u_i}(t)}{t_{u_i,m_j}(t)}$, where $d_{u_i}(t)$ denotes the data size of the request [37], [38]. We use $D^u_{u_i}(t)$ to represent the updating delay, which occurs when the location of service s_i that is serving u_i changes. Here, we consider two scenarios. One is that the operator can place a replication on the edge server to which u_i is currently connected. The other is that operator can migrate service s_i to the edge server to which user u_i goes forward. The costs of both scenarios are discussed in the next subsection. The updating delay is defined as $D^u_{u_i}(t) = \Upsilon(v_i) + \Psi(s_i)$, where $\Upsilon(s_i)$ is the delay of rebooting software resources, and $\Psi(s_i)$ is the delay of transmitting service profiles [39].

3.1.2 Cost Model

We use $\mathbb{C}(t)$ to denote the total cost of users in set $\mathbf{U}$ at time slot t, where $\mathbb{C}(t) = \sum_{h=1}^{|\mathbf{S}|} \mathbb{C}_{s_h}(t)$. Let $\mathbb{C}_{s_h}(t)$ denote the cost of service s_h, where $\mathbb{C}_{s_h}(t) = C^m_{s_h}(t) + C^r_{s_h}(t)$. We use $C^m_{s_h}(t)$ and $C^r_{s_h}(t)$ to represent the migration cost and replication cost, respectively. Let $x_{s_h}(t)$ denote the decision of s_h, when s_h decides to stay at the edge server with the same location in the previous step, $x_{s_h}(t) = 0$, otherwise, $x_{s_h}(t) = 1$.

3.2 Problem Formulation

On the basis of the models above, our problem is formulated to minimize the long-term average delay subject to the resource and cost constraints, which is presented as follows:

$$\mathbf{P}_1: \text{minimize} \lim_{T\to\infty} \frac{1}{T} \sum_{t=0}^{T-1} \sum_{i=1}^{|U|} \mathbb{D}_{u_i}(t) \tag{1}$$

$$\text{s.t.} \quad \mathbb{D}_{u_i}(t) = D^c_{u_i}(t) + D^l_{u_i}(t) + x_{s_h}(t) \cdot D^u_{u_i}(t), \tag{2}$$

$$\lim_{T\to\infty} \frac{1}{T} \sum_{t=0}^{T} \sum_{h=1}^{|\mathbf{S}|} \mathbb{C}_{s_h}(t) \le \overline{\Gamma}, \mathbb{D}_{u_i}(t) \le \overline{D}, \forall u_i \in \mathbf{U}, \tag{3}$$

$$\sum_{\mathbf{S}_{m_i} \in \mathbf{S}} W(\mathbf{S}_{m_i}(t)) \le R^s_{m_i}, \forall m_i \in \mathbf{M}, \tag{4}$$

$$x_{s_h}(t) \in \{0, 1\}, \forall s_h \in \mathbf{S}. \tag{5}$$

$\mathbf{P}_1$ is the objective function, and equations (2) to (5) are the constraints. Equation (2) is the total delay of each user, which needs to be lower than $\overline{D}$ to ensure the QoS. Equation (3) states that the long-term average cost cannot exceed the threshold $\overline{\Gamma}$ determined by the operators. Equation (4) states the constraint on the resource, which means the services placed on m_i should be under the limitation $R^s_{m_i}$. Here, we use $W(\mathbf{S}_{m_i}(t))$ to denote the amount of storage resources occupied for provisioned services on edge server m_i. Equation (5) states the decision of s_h which provides service for u_i at time slot t. As shown in the above equations, in order to obtain the optimal solution of $\mathbf{P}_1$, complete offline information is required, i.e., the distribution of users' trajectories over all time slots, which is difficult to predict in advance. Thus, the main challenge that complicates the derivation of the optimal solution to the above problem is the lack of future information. In addition, the constraints that $\mathbf{P}_1$ must satisfy during the optimization process that make it very difficult to solve even if the future information is known a priori. Therefore, these challenges require an online approach that enables service provisioning and updating decisions to be made efficiently.

4 Service Updating Decision Strategy based on Lyapunov Optimization

In this section, we first introduce two service updating decision strategies based on Lyapunov optimization by

considering the trajectory prediction condition of multiple mobile users.

4.1 Decoupling based on Lyapunov Optimization

Since the major challenge of directly solving $\mathbf{P}_1$ is that the long-term cost constraint of providers couples the service provisioning and updating decisions across different time slots, we first decouple the original problem into per-frame deterministic problems by applying Lyapunov optimization in this subsection. In order to deal with the constraint on average cost $\overline{\Gamma}$ in Equation (3), we introduce a virtual queue $Q(t)$ which denotes the historical measurement of the extra cost of services at time slot t. The queue updates according to the following equation

$$Q(t+1) = \max\{Q(t) + \mathbb{C}(t) - \overline{\Gamma}, 0\} \tag{6}$$

Intuitively, the condition of the total extra cost $\mathbb{C}(t)$ that is produced by the replication or migration of services can be evaluated by $Q(t)$. When the value of $Q(t)$ is large, it represents that the cost has exceeded the long-term cost $\overline{\Gamma}$. Specifically, Equation (6) implies $Q(t+1) \geq Q(t) + \mathbb{C}(t) - \overline{\Gamma}$, and then we have $\mathbb{C}(t) - \overline{\Gamma} \leq Q(t+1) - Q(t)$. By summing this inequality during all time slots, we have $\sum_{t=0}^{T-1}(\mathbb{C}(t) - \overline{\Gamma}) \leq Q(T) - Q(0)$. Initialize $Q(0) = 0$ and divide by t time slots. One can take expectations and derive that the expected backlog over time slot in $[0, T-1]$ is less than the threshold.

$$\frac{1}{T}\sum_{t=0}^{T-1}\mathbb{E}[\mathbb{C}(t)] \leq \lim_{T\to\infty}\frac{1}{T}\mathbb{E}[Q(T)] + \overline{\Gamma} \tag{7}$$

As shown in Equation 7, we have that the constraint on the cost can be guaranteed by stabilizing the virtual queue $Q(t)$. Therefore, a quadratic Lyapunov function for each slot t is defined as $L(Q(t)) \triangleq \frac{1}{2}Q(t)^2$ [27], [33], [40], where $Q(t)$ is a vector that evolves over slots in $[0, T-1]$. Here, the quadratic Lyapunov function can be considered as a scalar measure of queue deviation which is similar to $Q(t)$. In order to keep the queue stable, which means enforcing the extra cost constraint by promoting the Lyapunov function to lower values continuously, we introduce the one-step conditional Lyapunov drift as follows.

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

Lemma 1. Given the updating decisions of services in set **S** according to multiple mobile users **U** in each time slot t, the statement holds:

$$\Delta(Q(t)) \leq ß + Q(t)\mathbb{E}[(\mathbb{C}(t) - \overline{\Gamma})|Q(t)] \tag{9}$$

where $ß \triangleq \frac{1}{2}(\tilde{\mathbb{C}}(t)^2 + \overline{\Gamma}^2)$.

Proof: We rearrange Equation (8) for a concise form, where $\Delta(Q(t)) \triangleq \mathbb{E}[L(Q(t+1)) - L(Q(t))|Q(t)] = \frac{1}{2}\mathbb{E}[(\mathbb{C}(t) - \overline{\Gamma})^2|Q(t)] + Q(t)\mathbb{E}[(\mathbb{C}(t) - \overline{\Gamma})|Q(t)]$. For each service, we use $\tilde{\mathbb{C}}_{s_h}(t)$ to denote the cost of updating the decision of s_h in set **S** by choosing the minimum delay of user $u_h \in \mathbf{U}$ at time slot t. Based on that, the total cost of all services will be $\tilde{\mathbb{C}}(t) = \sum_{s_h\in\mathbf{S}}\tilde{\mathbb{C}}_{s_h}(t)$. Because of the division of the time space taking into account the user's mobility on the boundary, the service provider will not change in one time slot. Thus, we have $\tilde{\mathbb{C}}(t) \geq \mathbb{C}(t)$. Then, we have $\Delta(Q(t)) \leq \frac{1}{2}(\tilde{\mathbb{C}}(t) - \overline{\Gamma})^2 + Q(t)\mathbb{E}[(\mathbb{C}(t) - \overline{\Gamma})|Q(t)] \leq \frac{1}{2}(\tilde{\mathbb{C}}(t)^2 + \overline{\Gamma}^2) + Q(t)\mathbb{E}[(\mathbb{C}(t) - \overline{\Gamma})|Q(t)]$. Therefore, we can obtain that the one-step conditional Lyapunov drift holds $\Delta(Q(t)) \leq ß + Q(t)\mathbb{E}[(\mathbb{C}(t) - \overline{\Gamma})|Q(t)]$ at each time slot t, where $ß \triangleq \frac{1}{2}(\tilde{\mathbb{C}}(t)^2 + \overline{\Gamma}^2)$. Therefore, the proof of Lemma 1 is complete. ■

Algorithm 1 Updating Strategy with No Prediction (USNP)

Input: Sets of edge servers **M**, users **U**, and services **S**;
Output: Service updating decision $\mathbf{X}(t)$ of **U** at time slot t;

```
for users k = 1 to k = |U| in U do
    Choose the updating decision by optimizing P2;
    for edge servers i = 0 to i = |M| in M do
        if Σ_{s_{m_i}∈S} W(S_{m_i}(t)) ≥ R^s_{m_i} then
            Choose service by i = arg min{η_h};
        end if
    end for
end for
return Service updating decision X(t) of S;
```

According to the Lyapunov optimization framework, we obtain the upper bound of the Lyapunov drift function by introducing a Lyapunov drift-plus-penalty function in each time slot t.

$$P(t) \triangleq \Delta(Q(t)) + V\mathbb{E}[\mathbb{D}(t)|Q(t)] \tag{10}$$

Here, we define V as a non-negative parameter for adjusting the trade-off between the extra cost queue and the delay. In each time slot, the performance of the service provisioning strategy is guaranteed by minimizing an upper bound of the following function.

$$P(t) \leq ß + Q(t)\mathbb{E}[(\mathbb{C}(t) - \overline{\Gamma})|Q(t)] + V\mathbb{E}[\mathbb{D}(t)|Q(t)] \tag{11}$$

Based on that, the service provisioning and updating problem is formulated by minimizing the right side of Equation (11) at each time slot, which is formulated as follows.

$$\mathbf{P}_2: \text{minimize} \quad ß + Q(t)(\mathbb{C}(t) - \overline{\Gamma}) + V\mathbb{D}(t) \tag{12}$$

$$\text{s.t.}(2) - (5). \tag{13}$$

4.2 Optimal Services Updating Decision Strategy

In this subsection, we propose a service updating decision strategy by optimizing $\mathbf{P}_2$ under the constraints in each time slot. We start with a definition as follows.

Definition 1 (Optimal Service Updating (OSU) Problem). Given the distribution of users **U**, the topology of edge network **G**, and the function $\Theta(t)$, an OSU problem is how to find a decision for services in **S** to minimize $\mathbf{P}_2$ under the constraints at time slot t.

On the basis of Definition 1, we discuss two scenarios. One is the services updating without prediction, and the other is the service updating with prediction.

4.2.1 OSU with no prediction

The first scenario we considered is the OSU problem without available information caused either by the inaccurate prediction results, or by it being the initial or training stages of mobile users in per-slot. The specific steps are shown in Algorithm 1. We use the sets of edge servers $\mathbf{M}$, users $\mathbf{U}$, and services $\mathbf{S}$ as the input. The output is the service updating decision $\mathbf{X}(t)$ at time slot t. For each user in set $\mathbf{U}$, we choose the updating decision by optimizing $\mathbf{P}_2$ in lines 1 to 2. Then, we check the feasibility of services on edge servers by checking whether $\sum_{\mathbf{S}_{m_i}\in\mathbf{S}} W(\mathbf{S}_{m_i}(t)) \geq R^s_{m_i}$. Here, we use $\sum_{\mathbf{S}_{m_i}\in\mathbf{S}} W(\mathbf{S}_{m_i}(t))$ to denote the total number of services provisioning on m_i. In order to avoid conflicts caused by aggregation requests of multiple users, we introduce a definition of the conflict resolution factor for the service, and the specific definition is as follows.

Definition 2 (conflict resolution factor). Let η_h indicate the conflict resolution factor of service s_h and $\eta_h = \mathbb{C}_{s_h}(t)/\overline{\mathbb{D}^l_{u_h}(t)}$, where $\overline{\mathbb{D}^l_{u_h}(t)} = \mathbb{D}^l_{u_h}(t)|_{s_h\notin\mathbf{S}_{m_i}(t)}$.

Here, we use $\mathbb{C}_{s_h}(t)$ to denote the total extra cost of service s_h when it migrates or replicates on edge server m_i at time slot t, where $s_h \in \mathbf{S}_{m_i}(t)$. In line 4, we choose a service by an increasing order $i = \arg\min\{\eta_h\}$. Finally, the service updating decision $\mathbf{X}(t)$ is returned in line 6.

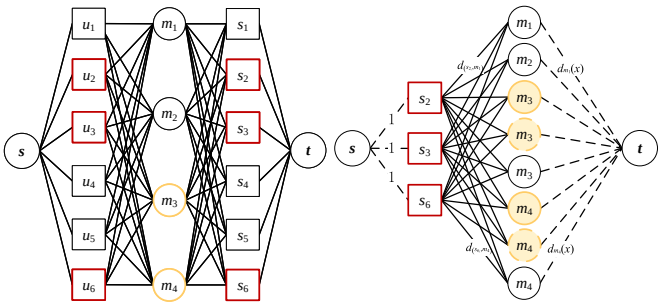
4.2.2 OSU with prediction

In this subsection, we explore a more realistic and complicated scenario in which we consider the trajectory prediction for the service provisioning and updating decision strategy. Here, we introduce an online strategy in the view of the committed horizon control method, where the predictions are looking forward to several slots for multiple slots by utilizing the existing well-performance model.

Lemma 2. The decision of OSU problem can be solved by minimizing $\Theta(t)$, where $\Theta(t) = Q(t)\mathbb{C}(t) + V\mathbb{D}(t)$.

Proof: We first rearrange $\mathbf{P}_2$ by introducing an intermediate variable $\mathbb{P}$, where $\mathbb{P}(t) = ß + Q(t)(\mathbb{C}(t) - \overline{\Gamma}) + V\mathbb{D}(t) = ß + Q(t)\mathbb{C}(t) - Q(t)\overline{\Gamma} + V\mathbb{D}(t)$. The value of ß is related to the distribution of users in set $\mathbf{U}$, which is a constant value. Meanwhile, the value of $Q(t)$ depends on the decision of services in the previous time interval $[0, t-1]$, which means that the decision at time slot t has no effect on the value of $Q(t)$. We reconstruct $\mathbb{P}(t)$ as $\mathbb{P}(t) = W + \Theta(t)$, where $W = ß + Q(t) - \overline{\Gamma}$ and $\Theta(t) = Q(t)\mathbb{C}(t) + V\mathbb{D}(t)$. Therefore, we can obtain that the network determines the service updating strategies by solving the optimization of $\Theta(t)$ in each time slot. ■

Based on the conversion above, we rearrange $\Theta(t)$ by considering the combinational decision-making where $\Theta(t) = Q(t)\sum_{h=1}^{|\mathbf{S}|}\mathbb{C}_{s_h}(t) + V\sum_{i=1}^{|U|}\mathbb{D}_{u_i}(t)$. The value of the total extra cost of service s_h depends on the decision choosing to migrate or place replications, i.e., $\mathbb{C}_{s_h}(t) = C^m_{s_h}(t) + C^r_{s_h}(t)$, which will affect the result of the delay. Taking the decision of s_h as an example, if service s_h decides to migrate or place replications on other edge servers, it will produce a migration cost $C^m_{s_h}(t)$ or replication cost $C^r_{s_h}(t)$. Meanwhile, the communication part of $D^l_{u_i}(t)$ will decrease while the updating part of $D^u_{u_i}(t)$ will increase for $\mathbb{D}_{u_i}(t)$. We reconstruct $\Theta(t)$ on the basis of the interaction based on the relationship between services and users, where $\Theta(t) = \sum_{h=1}^{|\mathbf{S}|}\Theta_h(t)$. For each service, we have $\Theta_h(t) = Q(t)\mathbb{C}_{s_h}(t) + (D^l_{u_h} + D^u_{u_h}) + D^c_{u_h}$.



(a) original connectivity graph. (b) extracted connectivity graph.

Fig. 2. The connectivity graphs of Fig 1.

Based on that, we use $d_{(s_h,m_i)}$ to represent the weight between service s_h and edge server m_i at time slot t, where $d_{(s_h,m_i)}(t) = Q(t)\mathbb{C}_{s_h}(t)$. We suppose that $d_{m_i}(x)$ is the delay function. We replace each edge in $\mathbf{G}^\circ$ with $|\hat{\mathbf{U}}(t)|$ parallel edges between the same server m_i and the destination t, and each with weight $d_{m_i}(x)|_{u_x\in\hat{\mathbf{U}}(t)}$. Then, the weight between edge server m_i and the destination t that is connected to it is $d_{m_i}(x) = (D^l_{u_x} + D^u_{u_x}) + D^c_{u_x}$. Therefore, we have $\Theta_h(t) = d_{(s_h,m_i)}(t) + d_{m_i}(x)(t)$.

On the basis of the interaction, we propose a novel Updating Strategy with No Prediction (USNP) to optimize the provisioning strategy at each time slot, which is shown in Algorithm 2. We first construct an original connectivity graph by considering the information and connection between services and edge servers. We add two virtual nodes which are source ***s*** and destination ***t***, and the middle two layers are services and storage resources of edge servers. The original connectivity graph is shown in Figure 2(a), where users are represented by the left squares in this diagram, edge servers by the middle circles, and services by the right squares. We assume that edge nodes are bidirectionally reachable, which means that users can access all edge nodes with different costs and delays. In addition, the service is able to choose any edge server for provisioning when the capacity of the edge server allows, however, different positions will result in different delays. Thus, the users and services are fully connected to the edge servers. In each time slot, the activities of users are dynamic and independent. This means that some users may be remaining in their original locations, while others may be far away from the connected edge servers. We use thick red lines to mark services where their users are far away from the original locations, and thick yellow lines to mark edge servers with remaining resources. For the users whose locations are not changing, the corresponding service will not be migrated or placed by an additional replica, so there is no extra cost or delay produced. Therefore, we consider optimizing the provisioning of services by constructing an activity set $\hat{\mathbf{U}}(t)$ to reduce the dimensional space. The formal definition is given as follows.

Definition 3 (Activity Set). Let $\hat{\mathbf{U}}(t)$ indicate the activity set

Algorithm 2 Updating Strategy with Prediction (USP)

Input: Sets of edge servers **M**, users **U**, and services **S**;
Output: Service updating decision $\mathbf{X}(t)$ of **S** at time slot t;

1: Construct the original connectivity graph **g** based on the provisioning of **S**, the connections of **G**, and **U**;
2: **for** users $i = 1$ to $i = |\mathbf{U}|$ in **U do**
3: Calculate $\varsigma_{u_i}(t) = (L_{u_i}(t-1), L_{u_i}(t))$;
4: **if** $\varsigma_{u_i}(t)$==1 **then**
5: Construct the activity set with $\hat{\mathbf{U}}(t) \leftarrow u_i$;
6: Update user set at time slot t with $\mathbf{U}(t) = \mathbf{U}(t)/u_i$;
7: **else**
8: Update $\mathbf{U}(t) \leftarrow u_i$;
9: **end if**
10: **end for**
11: Construct the extracted connectivity graph $\mathbf{G}^\circ$ based on the activity set $\hat{\mathbf{U}}(t)$;
12: Replace the link with $|\hat{\mathbf{U}}(t)|$ parallel ones with weight $d_{m_i}(x)|_{u_x \in \hat{\mathbf{U}}(t)}$;
13: Find a feasible service updating decision with min-cost flow of $\hat{\mathbf{U}}(t)$;
14: **return** Service updating decision $\mathbf{X}(t)$ of services **S**;

of users at time slot t, where $u_i \in \hat{\mathbf{U}}(t)$ is the user whose current location $L_{u_i}(t)$ is going far away from the edge server for initial connection $L_{u_i}(t-1)$.

Here, we use $L_{u_i}(t)$ to denote the edge server that user u_i becomes connected to at time slot t. Since one user can only be served by one service, the number of users and services are equal. Based on that, we do an extraction by considering the current status of users and the topology of the edge network. The extracted connectivity graph is shown in Figure 2(b). We use the white circle to indicate that a container on the edge server has been occupied. Due to the fact that each server provisioning service has two alternatives, replication and migration, these options are accessible. The yellow dashed circle indicates migration, and the edge node, where a copy is placed, is represented by a solid yellow circle. Based on that, we assign values to network edges. The virtual source node which is connected to all user services with the weight of 1. This means that only one service provisioning decision can be made for each service, i.e., only one server can be selected for one service. Here, the weight of a service and each edge node decision edge for migration is the sum of migration delay and cost, while the weight of replication decisions is the sum of replication cost and delay. Each edge server involves two service provisioning decisions, each of which is connected to a virtual destination node. Here, the weight of the migration decision between the edge server and the virtual destination is the migration delay, while the replication decision is the replication delay.

The specific steps are shown in Algorithm 2. We use the sets of **M**, **U**, and **S** as the inputs. The service updating decision $\mathbf{X}(t)$ of **S** at time slot t is used as the output. We construct the original connectivity graph **g** based on the provisioning of **S**, the connections of **G**, and **U** in line 1. In line 2, we start to construct the activity set $\hat{\mathbf{U}}(t)$. We first check the locations of users in set **U**, where $\varsigma_{u_i}(t) = (L_{u_i}(t-1), L_{u_i}(t))$ in line 3. If $\varsigma_{u_i}(t) = 1$, this denotes that u_i has gone away from the edge server at time slot $t-1$. Then, we construct the activity set by adding u_i into set $\hat{\mathbf{U}}(t)$, where $\hat{\mathbf{U}}(t) \leftarrow u_i$; Otherwise, it denotes that u_i always stays near the edge server from $t-1$ to t, and we update $\mathbf{U}(t) \leftarrow u_i$. Based on this, we start to construct the extracted connectivity graph $\mathbf{G}^\circ$ based on the activity set $\hat{\mathbf{U}}(t)$ in line 9. In line 10, we replace the link with $|\hat{\mathbf{U}}(t)|$ parallel ones with weight $d_{m_i}(x)|_{u_x \in \hat{\mathbf{U}}(t)}$ between edge servers and destination t. Then, we find a feasible service updating decision with min-cost flow of $\hat{\mathbf{U}}(t)$, and we return the updating decision $\mathbf{X}(t)$ of services **S** in line 12.

5 Online Optimization of Service Provisioning Strategy

In this section, we design an Online Optimization of Service Provisioning Strategy (O-OSP$_\omega$) by utilizing the committed horizon control method with ω steps prediction. The main ideas of O-OSP$_\omega$ are to leverage the prediction model to look forward the trajectories of users in multiple steps and to use the information to realize the optimization of service provisioning.

The specific steps are shown in Algorithm 3. We illustrate the whole process through Figure 3. We suppose that the chosen prediction model exists a small part of adjustment stage in the initial τ time slots (orange covered area in Figure 3), which means that the information in first τ steps is unavailable. Thus, we get service updating decision $\mathbf{X}(t)$ using Algorithm 1 in line 2. After that, we obtain the service updating decision $\mathbf{X}(t)$ using Algorithm 2 based on $\widehat{\mathbf{L}}_{\mathbf{U}|[t,t+\omega]}$ in line 4. Here, $\widehat{\mathbf{L}}_{u_i|[\tau,\tau+\omega]}$ is the trajectory of user u_i in a ω time steps prediction window starting at time τ, where $\widehat{\mathbf{L}}_{u_i|[\tau,\tau+\omega]} = \{\hat{L}_{u_i}(\tau), \hat{L}_{u_i}(\tau+1), ..., \hat{L}_{u_i}(\tau+\omega)\}$ (blue covered area in Figure 3). In line 5, we set $\tilde{t} = (t-\tau) \mod \omega$, and we check whether the prediction steps are less than ω. In lines 9 to 13, we update the service provisioning for services by introducing a novel factor feasible decision frequency. We use $a_{s_h}(t) = x_{s_h}(t) \cdot y_{s_h}(t)$ to represent the decision value of s_h, where $y_{s_h}(t) \in \{-1, 1\}$ and $x_{s_h}(t) \in \{0, 1\}$ as shown in equation (5). Since $x_{s_h}(t) = 0$ when service s_h decides to stay at the original location, the decision value will be $a_{s_h}(t) = 0$. On the contrary, when service s_h decides on migration, $y_{s_h}(t) = -1$ and $x_{s_h}(t) = 1$, and then the value of $a_{s_h}(t) = -1$. Similarly, when service s_h makes a decision on replication, $y_{s_h}(t) = 1$ and $x_{s_h}(t) = 1$, and then the value of $a_{s_h}(t) = 1$. Based on that, we use a queue $A_{s_h}^{(x)}$ to record the decision values of service s_h in x time steps, i.e., $A_{s_h}^{(\omega)} = \{a_{s_h}(t+1), a_{s_h}(t+2), ..., a_{s_h}(t+\omega)\}$.

Definition 4 (feasible decision frequency). Let $\varrho_{s_h|\omega}^{a^\circ}(t)$ indicate the feasible decision frequency of s_h under the value a°, where $\varrho_{s_h|\omega}^{a^\circ}(t) = \frac{1}{\omega}\sum_{x=0}^{x=\omega-1} f(A_{s_h}^{(x)}, a^\circ)$.

Here, $f(A_{s_h}^{(x)}, a^\circ)$ is a function to indicate whether the result in queue $A_{s_h}^{(x)}$ is equal to a°, i.e., $a_{s_h} = a^\circ$. Then, we choose the updating decision of s_h by setting $X_{s_h}(\tilde{t}) = \arg\max_{a^\circ \in A_{s_h}^{(\omega)}} \{\varrho_{s_h|\omega}^{a^\circ}\}$.

Theorem 1. By applying O-OSP$_\omega$, the time-average system delay satisfies:
$\frac{1}{T}\sum_{t=0}^{t=T-1} \mathbb{D}(t) \leq \frac{1}{2}(OPT + ß + V|\mathbf{U}|\overline{D}) + \epsilon + \frac{1}{\omega} W \cdot \alpha \cdot T.$

Algorithm 3 Online Optimization of Service Provisioning strategy (O-OSP$_\omega$)

Input: Sets of edge servers $\mathbf{M}$, users $\mathbf{U}$, and services $\mathbf{S}$;
Output: Service updating decision $\mathbf{X}$ of $\mathbf{S}$ in each time slot;

for $t = 0$ to $t = \tau$ **do**
  Get service updating decision $\mathbf{X}(t)$ using Algorithm 1;
end for
for $t = \tau$ to $t = T - 1$ **do**
  Get service updating decision $\mathbf{X}(t)$ using Algorithm 2 based on $\widehat{\mathbf{L}}_{\mathbf{U}|[t,t+\omega]}$;
  Set $\tilde{t} = (t - \tau) \mod \omega$;
  if $\tilde{t} = t - \tau$ **then**
    Set $\mathbf{X}(t) = \mathbf{X}(\tilde{t})$;
  else
    for service $h = 1$ to $h = |\mathbf{S}|$ **do**
      Update the decision value of s_h into $A_{s_h}^{(\omega)}$;
      Calculate the decision policy frequencies;
      Set $X_{s_h}(\tilde{t}) = \arg\max_{a^\circ \in A_{s_h}^{(\omega)}} \{\varrho_{s_h|\omega}^{a^\circ}\}$;
    end for
    Set $\mathbf{X}(t) = \{X_{s_h}(\tilde{t})|_{s_h \in \mathbf{S}}\}$;
  end if
end for
return Service updating decision $\mathbf{X}(t)$ of $\mathbf{S}(t)$;

Proof: We conduct the proof via introducing $\mathbb{P}_{OSP}(t)$, where $\mathbb{P}_{OSP}(t) = \mathfrak{B} + Q(t)(\mathbb{C}(t) - \overline{\Gamma}) + V\mathbb{D}(t)$ under the O-OSP$_\omega$ strategy. For each time slot, we use $\mathbb{P}(t)$ to represent the decision policy with random frequency. We use $\delta(t)$ to denote the prediction error at time slot t. Then, we have the average value $\frac{1}{\omega}\sum_{t+1}^{t+\omega} b(t) \leq \frac{1}{\omega} \cdot \omega \cdot \epsilon = \epsilon$. Thus, we have

$$\mathbb{P}_{OSP}(t) \leq \frac{1}{\omega}\sum_{t+1}^{t+\omega} \mathbb{P}(t) \leq OPT + 2\epsilon + \frac{2}{\omega} W \cdot \alpha \cdot T, \quad (14)$$

which can be obtained in [41]. Here, $W = \max_{m_i \in \mathbf{M}}\{W(\mathbf{S}_{m_i}(t))\}$, which denotes the maximum available storage resource of edge servers. Since each server needs to make decisions for the services that are placed on it, there exists a stationary and randomized policy π for $\mathbf{P}_2$ which satisfy $\mathbb{E}[\mathbb{C} - \overline{\Gamma}] \leq \delta$. Thus, we have $\mathbb{P}_{OSP}(t) \leq \mathfrak{B} + Q(t) \cdot \delta + V\mathbb{D}(t)$. By letting δ go to zero, we have $\mathbb{P}_{OSP}(t) \leq \mathfrak{B} + V\mathbb{D}(t)$.

$$\mathbb{P}_{OSP}(t) \leq \mathfrak{B} + V\mathbb{D}(t) \leq \mathfrak{B} + V|\mathbf{U}|\overline{D}. \quad (15)$$

We sum the inequalities labeled as Equations (14) and (15), and then we have

$$\mathbb{P}_{OSP}(t) \leq \frac{1}{2}(OPT + \mathfrak{B} + V|\mathbf{U}|\overline{D}) + \epsilon + \frac{1}{\omega} W \cdot \alpha \cdot T. \quad (16)$$

Therefore, the proof of Theorem 1 is complete. ■

In this paper, the O-OSP$_\omega$ algorithm uses ω steps trajectory prediction and min-cost flow; the time complexity will be determined by the maximum value of these two parts. Since the trajectories of users are predicted through social-LSTM, and the provisioning process is based on the predicted results. The time complexity is $O(\mathbf{TBH}^2)$, where $\mathbf{T}$ is the length of the sequence determined by the steps ω, i.e., $\mathbf{T} = 2 + 2\omega$, $\mathbf{B}$ is the batch size, and $\mathbf{H}$ is the scale of the network hidden layer. In addition, the complexity of min-cost flow is $O(\mathbb{G}\mathbb{E}\mathcal{T})$, where $\mathbb{G}$ is the total number of nodes in the extracted connectivity graph, i.e., $\mathbb{G} = 2 + |\hat{\mathbf{U}}(t)| + |\bar{\mathbf{M}}| + \sum_{m_i \in \mathbf{M}/\bar{\mathbf{M}}} R_{m_i}^s$. Here, $|\bar{\mathbf{M}}|$ is the total number of edge servers which are fully occupied. $\mathbb{E}$ is the total number of edges in the extracted connectivity graph, i.e., $\mathbb{E} = |\hat{\mathbf{U}}(t)| + |\hat{\mathbf{U}}(t)|(|\bar{\mathbf{M}}| + \sum_{m_i \in \mathbf{M}/\bar{\mathbf{M}}} R_{m_i}^s) + |\bar{\mathbf{M}}| + \sum_{m_i \in \mathbf{M}/\bar{\mathbf{M}}} R_{m_i}^s = (|\hat{\mathbf{U}}(t)| + 1)(|\bar{\mathbf{M}}| + \sum_{m_i \in \mathbf{M}/\bar{\mathbf{M}}} R_{m_i}^s + 1) - 1$. $\mathcal{T}$ is the number of times the cumulative augmented path reaches the maximum flow. Therefore, the complexity of O-OSP$_\omega$ is $O(\max\{\mathbf{TBH}^2, \mathbb{G}\mathbb{E}\mathcal{T}\})$, which is the maximum value of both trajectory prediction and min-cost flow.

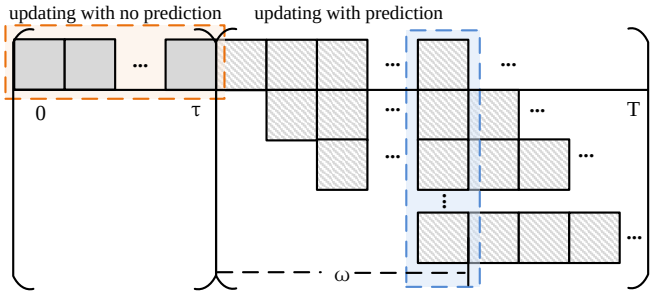


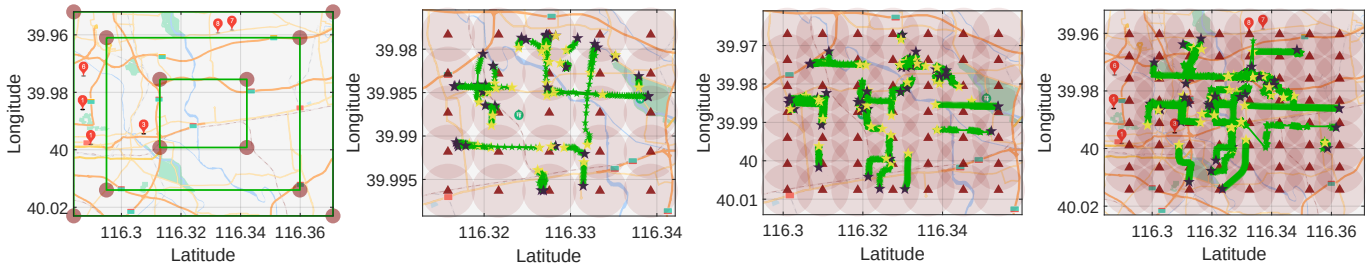
Fig. 3. An illustrating example of Algorithm 3.

6 Experiments

In this section, we conduct the experiments based on the Microsoft GPS trajectory dataset [42], [43] to study the service provisioning problem for multiple mobile users in edge computing networks.

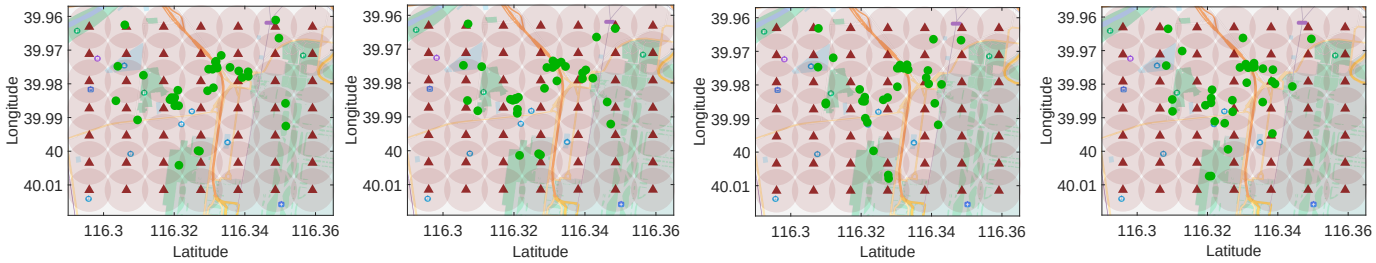
6.1 Basic Setting

We build our prototype on a workstation that runs a Linux operating system with E5-2620 CPU, NVIDIA RTX5000 GPU, 128Gb memory, and a 2Tb hard disk. We choose the Social-LSTM model to predict the future trajectories of users which can achieve an average accuracy of over 70%. We used the published Microsoft GPS trajectory dataset which has been collected in the Geolife project [42], [43]. The Microsoft GPS trajectory dataset is a GPS trajectory dataset collected from 182 users over a period of more than three years (from April 2007 to August 2012) as part of the (Microsoft Research Asia) Geolife project. These trajectories were recorded by various GPS loggers and GPS phones, and they have a variety of sampling rates. This dataset contains 17,621 trajectories with a total distance of about 1.2 million kilometers and a total duration of over 48,000 hours [42], [43]. Since this dataset recorded 182 users' outdoor trajectories in a broad range, we process it according to the features of users' activities. We first observed the activity tracks of 182 users and marked the longitude and latitude of the origin center coordinates $[116.327544, 39.987317]$. Then, we take this location as the central point and divide the area into three different scopes by setting the radius to $r = 1.0$, $r = 2.5$, and $r = 3.0$ kilometers. The division of the activities' scopes from different groups is shown in Figure 4 (a). Based on that, we construct three datasets with different characteristics by comprehensively considering the



(a) activities' scopes of users. (b) users activities under $r = 1.0$ with 20-steps. (c) users activities under $r = 2.5$ with 60-steps. (d) users activities under $r = 3.0$ with 120-steps.

Fig. 4. Users' activities under different scopes of users with scaling steps.



(a) time slot 0. (b) time slot 20. (c) time slot 40. (d) time slot 59.

Fig. 5. The distribution of users at different time slots.

time, scopes and trajectories of users' activities. For each group of datasets, we select 40 users to construct our dataset **U** and traverse their trajectories to find the ones within the areas under a scaled time series. We continuously collect users' data during 20 ($r = 1.0km$), 60 ($r = 2.5km$), and 120 ($r = 3.0km$) consecutive time slots for each group, respectively. The trajectories of **U** in different scopes are shown in Figures 4 (b) to (d). We can see that in the first group of users, the overlap of user activity trajectories is not obvious due to the small geographic location and time range. Then we expanded the activities' scopes of users while increasing the tracked time slots into 60 and 120. We found that the probability of trajectory overlap in the time slot increases. Specifically, we take the group in $r = 2.5$ during 60 consecutive time slots as an example to show the distribution of users in different slots in Figure 5, which includes the initial locations in time slots 0, 20, 40, and 59 in Figures 5(a), (b), (c), and (d). We found that the locations of users vary in different time slots, however, the number of connected users will remain at a high level for edge servers with a high frequency of utility. Based on that, we simulate the edge computing network based on **U**, and we set up 49 edge servers with the service range of 450 meters. We set the computing capacity of each server to range from 2GHz to 5GHz, and the data size of each service is 1GB. The storage of each edge server ranges from 5GB to 10GB, which also denotes the number of services that can be placed on edge servers. Compared to the proposed online service provisioning strategy, three baselines are used.

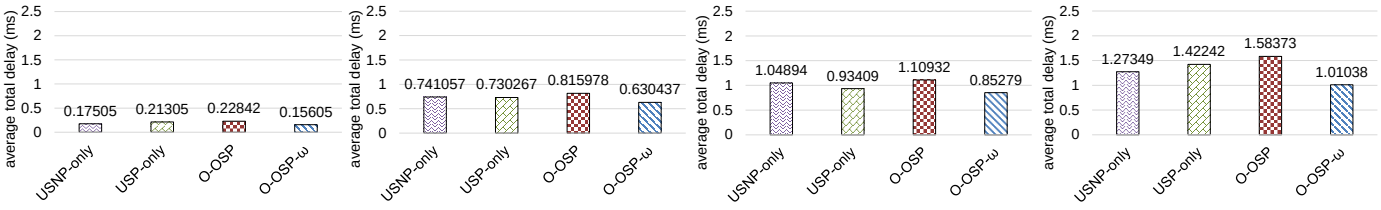
- **USNP-only**: Services provisioning and updating without using the prediction information, and the decisions are only made by USNP.
- **USP-only**: Services provisioning and updating by using the prediction information, and the decisions are only made by USP.
- **O-OSP**: Online services provisioning and updating based on O-OSP$_\omega$ without considering ω steps prediction.

6.2 Experiment Results

6.2.1 *Average total delay under different strategies*

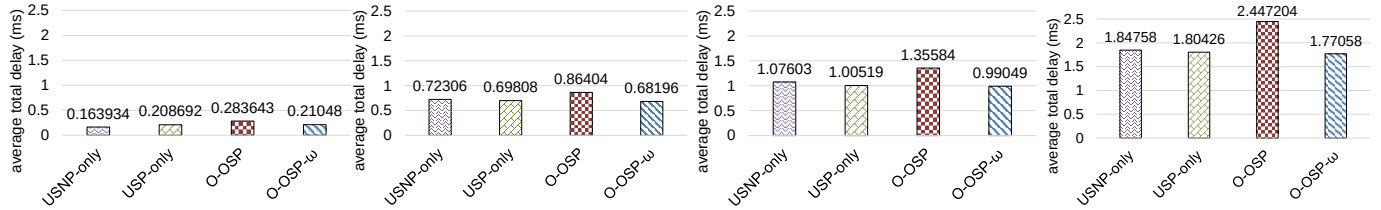
We investigate the average total delays under these four strategies with four groups of users ($|\mathbf{U}| = 10$, $|\mathbf{U}| = 20$, $|\mathbf{U}| = 30$, and $|\mathbf{U}| = 40$) in different time scales. The results are shown in Figures 7 to 8. Combined with the distribution characteristics of user activities in consecutive time slots for each group under different activity scopes 20 ($r = 1.0$), 60 ($r = 2.5$), and 120 ($r = 3$), respectively, we have the following observations.

(i) For each group, the numbers of users in set **U** affect the results of strategies. We analyze the average total delay for different groups of users under the same trajectory. Viewing the results in Figures 6 to 8 as a whole, the tendencies of the average total delays in groups of 20-step, 60-step, and 120-step climb with the increasing number of users. The main reason for this situation is the resource competition problem caused by the increase of users in the same active area. For the group with the shortest trajectory steps (20-step), algorithm USNP-only has the largest fluctuation on the total delay, which reaches about 1.3 multiples on average. By comparison, the fluctuation of the average total delay under algorithms O-OSP and O-OSP$_\omega$ are relatively stable and slow. However, for different numbers of users ($|\mathbf{U}| = 10$, $|\mathbf{U}| = 20$, $|\mathbf{U}| = 30$, and $|\mathbf{U}| = 40$), the average total of algorithm O-OSP is the highest, which means that



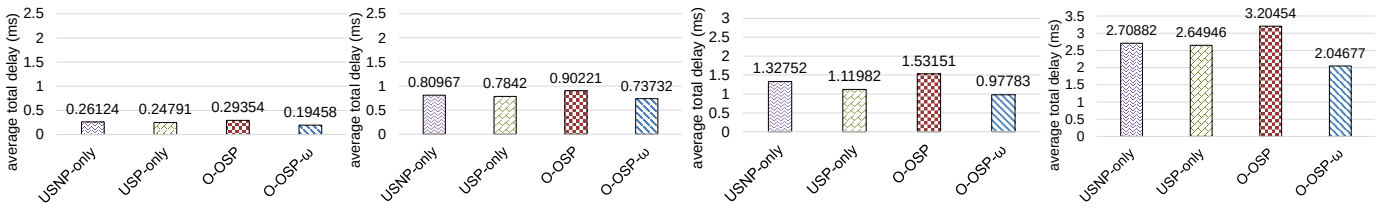
(a) # of users (|**U**|=10). (b) # of users (|**U**|=20). (c) # of users (|**U**|=30). (d) # of users (|**U**|=40).

Fig. 6. Average total delay under different strategies of users with 20-step trajectory ($r = 1.0km$).



(a) # of users (|**U**|=10). (b) # of users (|**U**|=20). (c) # of users (|**U**|=30). (d) # of users (|**U**|=40).

Fig. 7. Average total delay under different strategies of users with 60-step trajectory ($r = 2.5km$).



(a) # of users (|**U**|=10). (b) # of users (|**U**|=20). (c) # of users (|**U**|=30). (d) # of users (|**U**|=40).

Fig. 8. Average total delay under different strategies of users with 120-step trajectory ($r = 3.0km$).

if you only focus on the direction of the users in the next step, there may be extra overhead by the prediction error or the repeated operations of users' trajectories. As shown in Figure 6, O-OSP$_\omega$ can obtain the minimum delay for different groups of users, which means that considering the multi-step prediction can effectively help to avoid extra overhead being produced by the complex environment when users increase. The group with longest trajectory steps (120-step) showed similar trends to the one with 20 steps. The group with 60 steps has a slight difference. As shown in Figures 7(b), (c), and (d), O-OSP$_\omega$ has the lowest average total delays under the groups of 20, 30, and 40. Meanwhile, the average total delay decreases notably with an increasing number of users. However, as shown in Figure 7(a), USNP-only obtains the lowest delay in the group with 10 users. On one hand, there are abundant resources when there are fewer users, which leads to deviation in the decision-making under the prediction trajectory. On the other hand, we found that the trajectories for the selected 10 users in group one hardly changed, which results in errors in the algorithms using the prediction information.

(ii) For different groups, the activities' scopes and trajectories of users affect the results of strategies. Comparing Figures 6 to 8, expanding the activities' scopes has no significant impact on the average total delay of users, which is reflected in that the tendency has not changed significantly on the whole. For each group, the total average latency of algorithm O-OSP is the largest. Compare to the other three strategies, the result of O-OSP$_\omega$ is the minimum. The delay of USNP-only is higher than that USP-only in different trajectory ranges when the numbers of users are 10, 20, and 30. The tendency is slightly different for the group with 40 users, which is presented in the results under these four strategies most obviously. Like the group with a small range of scale shown in Figures 6 (d), USP-only is better than USNP-only. This fluctuation is mainly due to the increased number of users moving in a small range, which will bring resource constraints. Once a service provisioning is improperly, this will result in a relatively strong impact. However, the average total delay using algorithm USP-only is indeed lower than USNP-only for users $|\mathbf{U}| = 40$ with a wide range of activities, 120-step trajectory. When the range of activities stays in the middle, the effects of these two algorithms (USNP-only and USP-only) are very close to the O-OSP$_\omega$, which means that they obtain better results in the current active range.

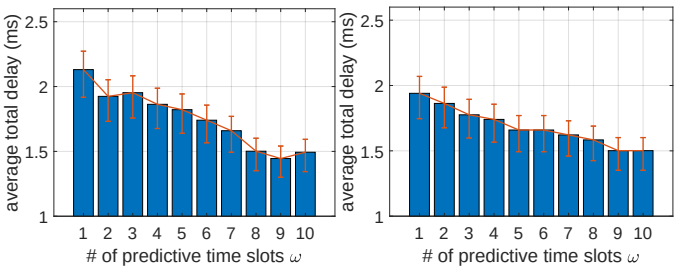
(iii) Prediction with ω slots in O-OSP$_\omega$ can effectively reduce the problem of service quality degradation caused by erratic activities of mobile users. As shown in Figure 7(d), the average total delay of O-OSP becomes significantly higher than that of the other algorithms. In this case, besides the lowest average total delay of O-OSP$_\omega$, USNP-only and UPS-only can also achieve better performances. The reason is that the increase in delay is due to the scaling of users

under limited resources. Especially in the case of the trajectories of users changing frequently, it may be inappropriate to determine the location of the service only by one step, which will affect the delay of other users. The simulation results show that our algorithm can reduce the average total delay of 28.7% (20-step trajectory, $r = 1.0km$), 17.8% (60-step trajectory, $r = 2.5km$), and 17.8% (120-step trajectory, $r = 3.0km$) when comparing with baselines, respectively.

6.2.2 Average total delay with different ω time slots

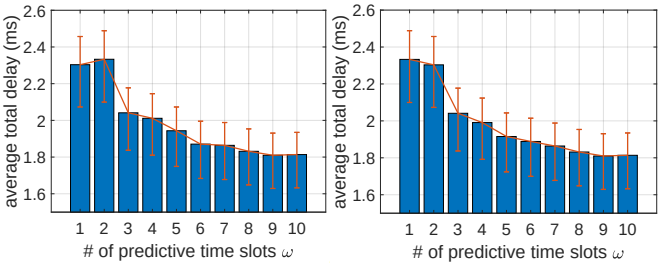
Based on the compared results above, we study the average total delay of O-OSP$_\omega$ with different predictive ω slots. We predict the trajectories of 40 users ($|\mathbf{U}|$) using the Social-LSTM model in multiple groups (20-step, 60-step, and 120-step), and we choose two sub-groups for each one with different accuracies. The results are shown in Figures 9 to 11. Additionally, we have the following observations.

(i) The value of ω can influence the efficiency of O-OSP$_\omega$, and there are existing differences with the increase of ω in the tendency of the average total delay even if the prediction accuracies are similar. As shown in Figures 9(a) to 11(a), the accuracies of these groups are 81%, 77%, and 72% which means that the differences between any two groups' fluctuations within 10%. However, it is clear that the fluctuations between the average total delays under different prediction steps ω vary greatly. For the first group of 20-step, the average accuracy of prediction under different steps (ω) is the highest (81%). The initial change of ω, where increases from $\omega = 1$ to $\omega = 2$ occurred, resulted in a very significant drop in the average total delay. But when the steps increase to $\omega = 3$, the delay increased slightly. Then, there is an approximately linear decline from $\omega = 4$ to $\omega = 9$. When the number of the prediction step becomes too large where $\omega = 10$, the average total delay of users in the group with 20-step trajectory decreases. For the second group of 60-step, which are shown in Figures 10(a) and 10(b), we have 72% and 57% percent accuracies for the comparative experiments. When the ω steps range from 1 to 9, the average total delay of users keeps decreasing. For each group, we can see that there is an obvious change between $\omega = 2$ and $\omega = 3$ which means that the initial change of ω has no effect on the delay, and the inflection point appears when $\omega = 3$. Then, there is an approximately linear decline from $\omega = 3$ to $\omega = 9$. When the slots scale into $\omega = 9$ and $\omega = 10$ ($\omega \geq 9$), the average total delay does not change notably. The reason for this is that the prediction of users' trajectories too far ahead of their movements may cause inaccurate results which may lead to invalid decisions. For group three with the widest range of activity scope (120-step), which are shown in Figures 11(a) and 11(b), the average total delay decreases smoothly for the group with 77% accuracy. When the value of ω increases to 8, the subsequent increase of ω has little effect on the delay. From the analysis by comparing Figure 9(a) to Figure 11(a), we have that when the accuracy rates are higher and close, the limitation on the activity scope of users causes local anti-correlation between the number of prediction steps ω and the average total delay. Therefore, the total average delay under the O-OSP$_\omega$ strategy decreases in a range with the increasing value of ω, and the setting of ω is related to the characteristics of users and the prediction model.



(a) group with 81% accuracy. (b) group with 54% accuracy.

Fig. 9. Average total delay of users with 20-step trajectory ($\omega \in [1, 10]$).

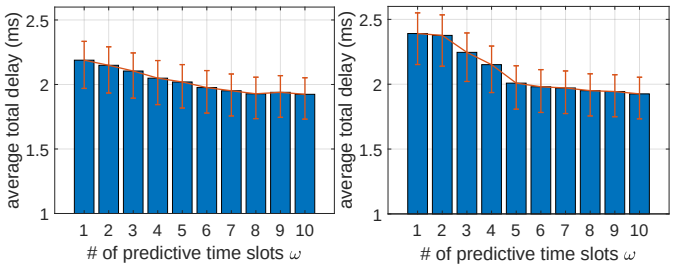


(a) group with 72% accuracy. (b) group with 57% accuracy.

Fig. 10. Average total delay of users with 60-step trajectory ($\omega \in [1, 10]$).

Comparing the three groups comprehensively, the number of predicted steps about $\omega = 9$ can obtain good results.

(ii) The accuracy of the chosen prediction model has little effect on the results of O-OSP$_\omega$. As shown in Figures 9 to 11, we selected two sub-groups with large differences which can reach about 20% on average in the prediction accuracy under different scope of activities for analysis. When the number of prediction steps is extremely small, i.e., $\omega = 1$, the lower accuracy has little effect on the average total delay. In some groups, the delay of users decreases with accuracy. As per users with 60-step trajectories in Figure 10, when the accuracy decreases to 57%, the average total delay under $\omega = 1$ is barely growing. For the users with 120-step trajectories in Figure 11, compared the group with 61% accuracy to the 72% one, there is an obvious growth on the average total delay under $\omega = 1$ which is caused by the scaling activity scope. At the same time, the result of the decreased delay may also occur in the case of lower accuracy under a small prediction step, such as for users with 20-step trajectories in Figure 9 when $\omega = 1$. When the number of prediction steps gradually change, the delays for these three groups decrease and the gap between sub-groups become narrower with the increase of ω. For the group with a wide range of activity which leads to higher uncertainty on the trajectories of users (120-step trajectory, $r = 3.0km$), the total average delay with smaller prediction steps shows increasing tendencies, and the results are getting closer when the steps are over $\omega \geq 6$. For the other two groups of users (20-step trajectory, $r = 1.0km$, and 120-step trajectory, $r = 3.0km$) which are shown in Figures 9 and 10, the average total delay under $\omega = 6$ in these two groups with 54% and 57% accuracies are also basically the same. Therefore, we have that even if the accuracy of the prediction model is imprecise, O-OSP$_\omega$ can still obtain a better result.



(a) group with 77% accuracy. (b) group with 61% accuracy.

Fig. 11. Average total delay of users with 120-step trajectory ($\omega \in [1, 10]$).

7 Conclusion

In this paper, we investigate the service provisioning and updating problem in a multi-user scenario by improving the performance of services within the long-term cost constraint. First, we decouple the original long-term optimization problem into a deterministic per-slot problem using Lyapunov optimization. Based on that, we propose two service updating decision strategies by considering the trajectory prediction conditions of users. Based on this, we design an online strategy by utilizing the committed horizon control method while looking ahead to ω slots predictions. We prove the performance bound of our online strategy theoretically in terms of the trade-off between delay and cost. Finally, we conduct extensive experiments based on the Microsoft GPS trajectory dataset, and we demonstrate the superior performance of the proposed algorithm.

In our future work, we intend to investigate the applicability of our algorithm to edge service markets, focusing on the study of backup strategies and service reliability in dynamic environments. In addition, we plan to find feasible and effective solutions considering fairness.

Acknowledgments

This work was supported by the Fundamental Research Funds for the Central Universities (2021RC258), the China Postdoctoral Science Foundation (Grant No. 2021M700366), and the National Natural Science Foundation under grant (Grant No. 92267107).

References

[1] Tu, S., Waqas, M., Rehman, S. U., Mir, T., Halim, Z., & Ahmad, I. (2021). "Social phenomena and fog computing networks: A novel perspective for future networks," IEEE Transactions on Computational Social Systems, 9(1), 32-44.

[2] Waqas, M., Tu, S., Halim, Z., Rehman, S. U., Abbas, G., & Abbas, Z. H. (2022). "The role of artificial intelligence and machine learning in wireless networks security: principle, practice and challenges. Artificial Intelligence Review," 1-47.

[3] Dang, T. K., Mohan, N., Corneo, L., Zavodovski, A., Ott, J., & Kangasharju, J. (2021). "Cloudy with a chance of short RTTs: analyzing cloud connectivity in the internet." In Proceedings of the 21st ACM Internet Measurement Conference, pp. 62-79.

[4] Chen, Y., Wu, J., & Ji, B. (2018, September). "Virtual network function deployment in tree-structured networks," In 2018 IEEE 26th International Conference on Network Protocols (ICNP) (pp. 132-142). IEEE.

[5] Siriwardhana, Y., Porambage, P., Liyanage, M., & Ylianttila, M. (2021). "A survey on mobile augmented reality with 5G mobile edge computing: architectures, applications, and technical aspects," IEEE Communications Surveys & Tutorials, 23(2), 1160-1192.

[6] Salaht, F. A., Desprez, F., & Lebre, A. (2020). "An overview of service placement problem in fog and edge computing. ACM Computing Surveys (CSUR)," 53(3), 1-35.

[7] Wang, L., Jiao, L., He, T., Li, J., & Bal, H. (2020). Service placement for collaborative edge applications. IEEE/ACM Transactions on Networking, 29(1), 34-47.

[8] Elgazzar, K., Martin, P., & Hassanein, H. S. (2013, December). Empowering mobile service provisioning through cloud assistance. In 2013 IEEE/ACM 6th International Conference on Utility and Cloud Computing (pp. 9-16). IEEE.

[9] Qiu, Y., Liang, J., Leung, V. C., Wu, X., & Deng, X. (2022). Online Reliability-Enhanced Virtual Network Services Provisioning in Fault-Prone Mobile Edge Cloud. IEEE Transactions on Wireless Communications, 21(9), 7299-7313.

[10] Li, J., Liang, W., Huang, M., & Jia, X. (2020). Reliability-aware network service provisioning in mobile edge-cloud networks. IEEE Transactions on Parallel and Distributed Systems, 31(7), 1545-1558.

[11] Yu, N., Xie, Q., Wang, Q., Du, H., Huang, H., & Jia, X. (2018, December). "Collaborative service placement for mobile edge computing applications," In 2018 IEEE Global Communications Conference (GLOBECOM) (pp. 1-6). IEEE.

[12] Mao, Y., Shang, X., & Yang, Y. (2022, May). "Joint Resource Management and Flow Scheduling for SFC Deployment in Hybrid Edge-and-Cloud Network," In IEEE INFOCOM 2022-IEEE Conference on Computer Communications (pp. 170-179). IEEE.

[13] Gu, L., Chen, Z., Xu, H., Zeng, D., Li, B., & Jin, H. (2022, May). "Layer-aware Collaborative Microservice Deployment toward Maximal Edge Throughput," In IEEE INFOCOM 2022-IEEE Conference on Computer Communications (pp. 71-79). IEEE.

[14] Nezami, Z., Zamanifar, K., Djemame, K., & Pournaras, E. (2021). "Decentralized edge-to-cloud load balancing: Service placement for the Internet of Things," IEEE Access, 9, 64983-65000.

[15] Zhang, G., Zhang, S., Zhang, W., Shen, Z., & Wang, L. (2021). "Joint service caching, computation offloading and resource allocation in mobile edge computing systems," IEEE Transactions on Wireless Communications, 20(8), 5288-5300.

[16] Chen, H., Deng, S., Zhu, H., Zhao, H., Jiang, R., Dustdar, S., & Zomaya, A. Y. (2022). "Mobility-Aware Offloading and Resource Allocation for Distributed Services Collaboration," IEEE Transactions on Parallel and Distributed Systems, 33(10), 2428-2443.

[17] Xu, J., Chen, L., & Zhou, P. (2018, April). "Joint service caching and task offloading for mobile edge computing in dense networks," In IEEE INFOCOM 2018-IEEE Conference on Computer Communications (pp. 207-215). IEEE.

[18] Ren, Y., Shen, S., Ju, Y., Wang, X., Wang, W., & Leung, V. C. (2022, May). "EdgeMatrix: A Resources Redefined Edge-Cloud System for Prioritized Services," In IEEE INFOCOM 2022-IEEE Conference on Computer Communications (pp. 610-619). IEEE.

[19] Shang, X., Huang, Y., Mao, Y., Liu, Z., & Yang, Y. (2022, May). "Enabling QoE Support for Interactive Applications over Mobile Edge with High User Mobility," In IEEE INFOCOM 2022-IEEE Conference on Computer Communications (pp. 1289-1298). IEEE.

[20] Wang, X., Ye, J., & Lui, J. C. (2022, May). "Decentralized task offloading in edge computing: a multi-user multi-armed bandit approach," In IEEE INFOCOM 2022-IEEE Conference on Computer Communications (pp. 1199-1208). IEEE.

[21] Han, P., Liu, Y., & Guo, L. (2021). "Interference-aware online multicomponent service placement in edge cloud networks and its ai application," IEEE Internet of Things Journal, 8(13), 10557-10572.

[22] Li, R., Zhou, Z., Chen, X., & Ling, Q. (2019). Resource price-aware offloading for edge-cloud collaboration: A two-timescale online control approach. IEEE Transactions on Cloud Computing, 10(1), 648-661.

[23] Liu, B., Zhang, W., Chen, W., Huang, H., & Guo, S. (2020). Online computation offloading and traffic routing for UAV swarms in edge-cloud computing. IEEE Transactions on Vehicular Technology, 69(8), 8777-8791.

[24] Liu, F., Zhou, Z., Jin, H., Li, B., Li, B., & Jiang, H. (2013). On arbitrating the power-performance tradeoff in SaaS clouds. IEEE Transactions on Parallel and Distributed Systems, 25(10), 2648-2658.

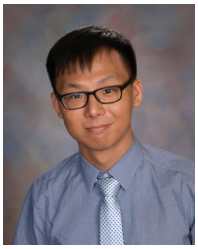
[25] Fang, W., Yao, X., Zhao, X., Yin, J., & Xiong, N. (2016). A stochastic control approach to maximize profit on service provisioning for

mobile cloudlet platforms. IEEE Transactions on Systems, Man, and Cybernetics: Systems, 48(4), 522-534.
[26] Qi, Y., Pan, L., & Liu, S. (2022). A Lyapunov optimization-based online scheduling algorithm for service provisioning in cloud computing. Future Generation Computer Systems, 134, 40-52.
[27] Ning, Z., Dong, P., Wang, X., Wang, S., Hu, X., Guo, S., ... & Kwok, R. Y. (2020). "Distributed and dynamic service placement in pervasive edge computing networks," IEEE Transactions on Parallel and Distributed Systems, 32(6), 1277-1292.
[28] Kim, T., Sathyanarayana, S. D., Chen, S., Im, Y., Zhang, X., Ha, S., & Joe-Wong, C. (2022, May). "Modems: Optimizing edge computing migrations for user mobility," In IEEE INFOCOM 2022-IEEE Conference on Computer Communications (pp. 1159-1168). IEEE.
[29] Zeng, Y., Huang, Y., Liu, Z., & Yang, Y. (2020, June). "Online Distributed Edge Caching for Mobile Data Offloading in 5G Networks," In 2020 IEEE/ACM 28th International Symposium on Quality of Service (IWQoS) (pp. 1-10). IEEE.
[30] Li, Z., Jiang, C., & Lu, J. (2021, December). "Distributed Service Migration in Satellite Mobile Edge Computing," In 2021 IEEE Global Communications Conference (GLOBECOM) (pp. 1-6). IEEE.
[31] Liu, E., Deng, X., Cao, Z., & Zhang, H. (2018, December). "Design and evaluation of a prediction-based dynamic edge computing system," In 2018 IEEE Global Communications Conference (GLOBECOM) (pp. 1-6). IEEE.
[32] Jin, Y., Jiao, L., Qian, Z., Zhang, S., & Lu, S. (2021, May). "Learning for learning: predictive online control of federated learning with edge provisioning," In IEEE INFOCOM 2021-IEEE Conference on Computer Communications (pp. 1-10). IEEE.
[33] Ma, H., Zhou, Z., & Chen, X. (2020). "Leveraging the power of prediction: Predictive service placement for latency-sensitive mobile edge computing," IEEE Transactions on Wireless Communications, 19(10), 6454-6468.
[34] Ouyang, T., Zhou, Z., & Chen, X. (2018). "Follow me at the edge: Mobility-aware dynamic service placement for mobile edge computing," IEEE Journal on Selected Areas in Communications, 36(10), 2333-2345.
[35] Ale, L., Zhang, N., Fang, X., Chen, X., Wu, S., & Li, L. (2021). Delay-aware and energy-efficient computation offloading in mobile-edge computing using deep reinforcement learning. IEEE Transactions on Cognitive Communications and Networking, 7(3), 881-892.
[36] Xiao, Z., Shu, J., Jiang, H., Lui, J. C., Min, G., Liu, J., & Dustdar, S. (2022). Multi-objective parallel task offloading and content caching in D2D-aided MEC networks. IEEE Transactions on Mobile Computing.
[37] Lu, S., Wu, J., Shi, J., Lu, P., Fang, J., & Liu, H. (2022). "A Dynamic Service Placement Based on Deep Reinforcement Learning in Mobile Edge Computing," Network, 2(1), 106-122.
[38] Taleb, T., Ksentini, A., & Frangoudis, P. A. (2016). "Follow-me cloud: When cloud services follow mobile users," IEEE Transactions on Cloud Computing, 7(2), 369-382.
[39] Gao, B., Zhou, Z., Liu, F., & Xu, F. (2019, April). "Winning at the starting line: Joint network selection and service placement for mobile edge computing," In IEEE INFOCOM 2019-IEEE conference on computer communications (pp. 1459-1467). IEEE.
[40] Neely, M. J. (2010). "Stochastic network optimization with application to communication and queueing systems," Synthesis Lectures on Communication Networks, 3(1), 1-211.
[41] Comden, J., Yao, S., Chen, N., Xing, H., & Liu, Z. (2019). "Online optimization in cloud resource provisioning: Predictions, regrets, and algorithms," Proceedings of the ACM on Measurement and Analysis of Computing Systems, 3(1), 1-30.
[42] Zheng, Y., Li, Q., Chen, Y., Xie, X., & Ma, W. Y. (2008, September). "Understanding mobility based on GPS data," In Proceedings of the 10th international conference on Ubiquitous computing (pp. 312-321).
[43] Zheng, Y., Zhang, L., Xie, X., & Ma, W. Y. (2009, April). "Mining interesting locations and travel sequences from GPS trajectories," In Proceedings of the 18th international conference on World wide web (pp. 791-800).
[44] Qiao, G., Leng, S., Maharjan, S., Zhang, Y., & Ansari, N. (2019). Deep reinforcement learning for cooperative content caching in vehicular edge computing and networks. IEEE Internet of Things Journal, 7(1), 247-257.
[45] Wang, S., Guo, Y., Zhang, N., Yang, P., Zhou, A., & Shen, X. (2019). Delay-aware microservice coordination in mobile edge computing: A reinforcement learning approach. IEEE Transactions on Mobile Computing, 20(3), 939-951.

Shuaibing Lu is currently a lecturer at the Faculty of Information Technology of Beijing University of Technology. She received her Ph.D. degree in Computer Science and Technology from Jilin University, Changchun, in 2019. She is supported by the China Scholarship Council as a visiting scholar supervised by Prof. Jie Wu in the Department of Computer and Information Sciences at Temple University (2016-2018). She is a member of IEEE. Her current research focuses on distributed computing, cloud computing and edge computing.

Jie Wu is the Director of the Center for Networked Computing and Laura H. Carnell professor at Temple University. He also serves as the Director of International Affairs at the College of Science and Technology. He served as Chair of Department of Computer and Information Sciences from the summer of 2009 to the summer of 2016 and Associate Vice Provost for International Affairs from the fall of 2015 to the summer of 2017. Prior to joining Temple University, he was a program director at the National Science Foundation and was a distinguished professor at Florida Atlantic University. His current research interests include mobile computing and wireless networks, routing protocols, cloud and green computing, network trust and security, and social network applications. Dr. Wu regularly publishes in scholarly journals, conference proceedings, and books. He serves on several editorial boards, including IEEE Transactions on Service Computing and the Journal of Parallel and Distributed Computing. Dr. Wu was general co-chair for IEEE MASS 2006, IEEE IPDPS 2008, IEEE ICDCS 2013, ACM MobiHoc 2014, ICPP 2016, and IEEE CNS 2016, as well as program co-chair for IEEE INFOCOM 2011 and CCF CNCC 2013. He was an IEEE Computer Society Distinguished Visitor, ACM Distinguished Speaker, and chair for the IEEE Technical Committee on Distributed Processing (TCDP). Dr. Wu is a CCF Distinguished Speaker and a Fellow of the IEEE. He is the recipient of the 2011 China Computer Federation (CCF) Overseas Outstanding Achievement Award.

Pengfan Lu received his B.Sc. in Computer Science and Technology at Harbin University of Science and Technology. Currently, he is working toward a M.Sc. degree in Computer Science under the Faculty of Information Technology at Beijing University of Technology. His research interests include cloud computing and edge computing.



Ning Wang is currently an assistant professor in the Department of Computer Science at Rowan University, Glassboro, NJ. He received his Ph.D. degree from the Department of Computer and Information Sciences at Temple University, Philadelphia, PA, USA, in 2018. He obtained his B.E. degree from the School of Physical Electronics at the University of Electronic Science and Technology of China, Chengdu, Sichuan, China, in 2013. He currently focuses on communication and computation optimization problems in Internet-of-Things systems and operation optimization in Smart Cities applications. He has published nearly thirty papers in high-impact networking conferences and journals, such as, IEEE ICDCS, IEEE INFOCOM, IEEE/ACM IWQoS, IEEE Transactions on Big Data, Journal of Parallel and Distributed Computing, etc. He has served as a program committee member for top international conferences such as IEEE ICDCS, IEEE WCNC, etc., and reviewers for premier journals such as IEEE TPDS, TWC, TMC, TITS, TOIT, TITS, TSC, etc.

Haiming Liu Haiming Liu is currently a lecturer in the School of Software Engineering at Beijing Jiaotong University. He received his Ph.D. degree in Computer Science and Technology (Bioinformatics) from Jilin University, Changchun, in 2019. Before that, he received his M.S. degree in Computer Software and Theory from Jilin University, Changchun, in 2015 and B.S. degree in Computer Science and Technology from Jilin University, Changchun, in 2012. He is a member of Chinese Association for Artificial Intelligence (CAAI). His current research focuses on edge computing, data mining, and bioinformatics.

Juan Fang Juan Fang, received her M.S. degree from Jilin University of Technology, Changchun, China in 1997, and her Ph.D. degree from the College of Computer Science, Beijing University of Technology, Beijing, China, in 2005. In 1997, she joined the College of Computer Science, Beijing University of Technology. From 2015, she has been a professor at Beijing University of Technology. Her research interests include high performance computing, edge computing and big data technology.