Online Learning for Failure-Aware Edge Backup of Service Function Chains With the Minimum Latency

Chen Wang, Qin Hu, *Member, IEEE*, Dongxiao Yu, *Senior Member, IEEE*, and Xiuzhen Cheng, *Fellow, IEEE*

***Abstract*— Virtual network functions (VNFs) have been widely deployed in mobile edge computing (MEC) to flexibly and efficiently serve end users running resource-intensive applications, which can be further serialized to form service function chains (SFCs), providing customized networking services. To ensure the availability of SFCs, it turns out to be effective to place redundant SFC backups at the edge for quickly recovering from any failures. The existing research largely overlooks the influences of *SFC popularity, backup completeness,* and *failure rate* on the optimal deployment of SFC backups on edge servers. In this paper, we comprehensively consider from the perspectives of both the end users and edge system to backup SFCs for providing popular services with the lowest latency. To overcome the challenges resulted from unknown SFC popularity and failure rate, as well as the known system parameter constraints, we take advantage of the online bandit learning technique to cope with the uncertainty issue. Combining the *Prim*-inspired method with the greedy strategy, we propose a *Real-Time Selection and Deployment* (*RTSD*) algorithm. Extensive simulation experiments are conducted to demonstrate the superiority of our proposed algorithms.**

***Index Terms*— Edge computing, service function chain, virtual network function, multi-armed bandit learning.**

I. Introduction

AN EXPLOSIVE amount of service requests are generated at the network edge due to the increasing number of smart devices connecting to the Internet to run various real-time applications, demanding higher bandwidth and lower latency. To serve end users more efficiently, the emerging mobile edge computing (MEC) paradigm aims to provide a variety of computing and networking services to users in a physically-close manner [1], [2], which can greatly reduce the response delay for users' requests and the possibility of network congestion, comprehensively improving the user experience.

In recent, network virtualization has been proposed to efficiently manage networking resources, which has also been widely implemented at MEC through network function virtualization (NFV). In particular, to disengage network functions from specific hardware, virtual network functions (VNFs) are utilized in NFV to facilitate scalable and flexible operations for service providers [3], [4], [5], [6]. Further, multiple VNFs can be connected sequentially to compose a service function chain (SFC), which aims at providing specialized networking services more efficiently [7], [8]. However, due to the vulnerability of VNFs in deployment [9], it is challenging to ensure the availability of SFCs because any failed VNF component can invalidate the entire service chain [10], [11], [12], [13].

To deal with this challenge, some existing studies focus on the optimal deployment of VNFs at the beginning of the configuration process [14], [15], [16], [17], [18], [19], while others rely on the idea of preparing backups of SFCs so that they can quickly recover from any unexpected failures of VNFs [20], [21], [22], [23]. In practice, SFC backups can be placed on both the central cloud server and distributed edge servers. Considering that routing to the central cloud is generally costly in operation and time consumption, researchers suggest backing up SFCs at the network edge [11], [22], where the limitation of edge resource has to be rigorously studied. However, as a critical reflection of the users' preferences, *popularity* of SFCs has long been overlooked in the design of SFC backup scheme at the edge, which can act as an index in selecting the appropriate set of SFC backups given the limited resource available at the edge. Though a recent work [23] takes into account the user demands in deploying VNF backups at the edge, the *completeness* of SFCs is not carefully treated with scattered VNF backups on edge servers. More importantly, the *failure rate* of each SFC has rarely been considered in the backup deployment schemes, which also refers to a significant property of SFCs in availability guarantee.

In this paper, we comprehensively consider from the perspectives of both the end users and edge system, incorporating the impacts of SFC popularity, completeness, and failure rate on the deployment of SFC backups at the edge. Nevertheless,

Manuscript received 9 January 2022; revised 27 August 2022 and 2 March 2023; accepted 19 March 2023; approved by IEEE/ACM TRANSACTIONS ON NETWORKING Editor M. Zhang. This work was supported in part by the National Key Research and Development Program of China under Grant 2019YFB2102600; in part by the National Natural Science Foundation of China (NSFC) under Grant 61971269, Grant 61832012, and Grant 61771289; and in part by the U.S. NSF under Grant CNS-2105004. *(Corresponding author: Qin Hu.)*

Chen Wang, Dongxiao Yu, and Xiuzhen Cheng are with the School of Computer Science and Technology, Shandong University, Qingdao, Shandong 250100, China.

Qin Hu is with the Department of Computer and Information Science, Indiana University-Purdue University Indianapolis (IUPUI), Indianapolis, IN 46202 USA (e-mail: qinhu@iu.edu).

Digital Object Identifier 10.1109/TNET.2023.3265127

there exist two aspects of challenges from both the *unknown* and *known* factors. To be specific, the unknown challenge is due to the fact that either the popularity or failure rates of SFCs cannot be known as a priori to better select the set of SFCs being backed up on edge servers. No matter for the service requests from end users or the happening of SFC failure, it can be highly dynamic and uncertain for the system design. While the other challenge from the known factors concerns the systematic parameters, including the resource constraint of edge servers to reasonably place SFC backups, the latency minimization of all SFC backups for providing efficient responses to users, and the binary deployment variable to place any SFC backup either at the edge or the cloud. The combination of uncertainty in the unknown challenge and confirmed constraints in the known challenge results in the difficulty to appropriately deploy SFC backups at the edge for fundamentally and efficiently enhancing the service availability.

To resolve the above challenges, we utilize the online bandit learning to have an estimation of unknown information and gradually approach the real values according to the instantly realistic feedback through both exploration and exploitation. With the learning results of unknown parameters, we calculate the minimum latency for all SFCs that can be completely backed up at the edge and optimize the link latency of all SFCs based on a variant of Prim algorithm solving the *Traveling Salesman Problem* (*TSP*) problem, based on which a greedy strategy based VNF backup deployment algorithm is designed to finally place relevant VNFs composing selected SFCs on edge servers.

In summary, our major contributions in this work can be listed as follows:

- Concerning the request preferences of end users and the changing environment, we model the SFC backup placement at the edge as an optimization problem, aiming to provide popular services with the lowest latency, where the resource constraint of edge servers and backup completeness of SFCs are strictly prescribed.
- To solve the proposed optimization problem with the unknown information, we resort to the combinatorial multi-armed bandit (CMAB) problem to learn the popularity and failure rate of SFCs in an online manner. Combined with the Prim-inspired solution and greedy strategy, we propose a *Real-Time Selection and Deployment* (*RTSD*) algorithm to achieve near-optimal placement of SFC backups at the edge.
- We compare our work with three benchmark solutions and two existing works to verify the superiority of the proposed algorithm. Moreover, the performance of our scheme is testified for varying end users and edge parameters.

The remaining of this paper is organized as follows. Section II investigates the most related work about SFC backup. Section III presents the system model and problem formulation of deploying SFC backups at the edge, which is addressed by our proposed *RTSD* algorithm in Section IV. Section V displays experimental results and Section VI concludes our paper.

II. Related Work

The advancement of NFV technology has effectively solved the problems of traditional networking system, where the successful deployment of VNF is the key to realize the virtualization [3], [4], [5], [6]. To provide more sophisticated services to end users, connecting multiple VNFs to form the SFC becomes prevailing [7], [8], [18], [20]. At present, research about SFCs can be classified into two categories: *SFC deployment* and *backup*.

A. Deployment of SFC

Given a large number of SFCs, how to deploy them in the network system becomes a critical problem as simply placing all component VNFs on the central cloud server can bring huge routing cost and service delay. The emergence of edge computing provides us with new solutions, where most of the existing studies focus on the deployment and routing issues at edge servers, aiming to achieve less resource consumption [17], [18], [19], [24]. Cziva et al. [17] present a method to dynamically re-schedule the optimal placement of VNFs in the case of changing network, users and demands, which realizes the minimized end-to-end delay from VNFs to edge devices. A supervisory mechanism is proposed for a hierarchical SFC architecture in [24] to determine the selection and deployment for a group of SFCs according to a combination of topological analysis and prediction of the demands. From another angle, Jin et al. [18] try to deploy SFCs at the edge with the goal of consuming the lowest edge resource and bandwidth, which has to satisfy the constraint of total latency of each SFC. Zhu et al. [19] focus on adaptive placement strategies for different SFCs via designing the EdgePlace algorithm which helps find the first node meeting both the resource and availability constraints.

B. Backup of SFC

To ensure the availability of SFCs, using redundant technology to create backups of SFCs turns out to be an effective approach [8], [10], [11], [12], [20], [22], [23]. Based on resource awareness, Zhang et al. [8] study the optimization of backup allocation considering that VNFs may require heterogeneous resources, with the aim of minimizing the resource consumption of backup nodes under the constraint of SFC availability guarantee. Fan et al. [12] try to estimate the number of necessary SFC backups while reducing the total resource consumption. Wang et al. [23] select qualified VNFs to be backed up at the edge taking into account end users' demands, but the atomicity and completeness of SFCs, as well as the VNF failure, are not considered, which cannot comprehensively solve the availability issue of SFCs. Other works consider the impact of the chain structure in SFC backup. Shang et al. [10] propose an optimization model called partial service function chain mapping, which uses partial SFC rerouting strategy to improve the availability of SFCs and reduce the maximum load of servers. And they further propose an adaptive online solution to dynamically create and adjust backups in the cloud and edge [11]. Fan et al. [20]

TABLE I
COMPARISON WITH RELATED WORKS

Categories	Works	Availability	Dynamism	Completeness	Edge
Deployment	[17]				✓
	[18]			✓	✓
	[19]	✓		✓	✓
	[24]			✓	
Backup	[8, 10, 20]	✓		✓	
	[11]	✓		✓	✓
	[12]	✓	✓	✓	
	[22]	✓		✓	✓
	[23]	✓	✓		✓
	Ours	✓	✓	✓	✓

adopt an availability-aware SFC mapping method with off-site redundancy with the aim of reducing resource consumption. Dinh et al. [22] propose a cost-efficient deployment scheme via measuring the improvement potential of VNFs and a collaborative redundancy placement scheme via exploiting the collaboration between a fog node with other nodes and the cloud.

In order to provide users with the most effective and reliable service, we need to solve the problem of deploying complete SFC backups at the edge to meet its availability in the case of dynamic user demand and failure occurrence. Based on this, we compare our paper with related works in Table I from four aspects, namely availability, dynamism, completeness and whether SFCs are deployed at the edge. It can be seen that the existing work cannot realize all requirements.

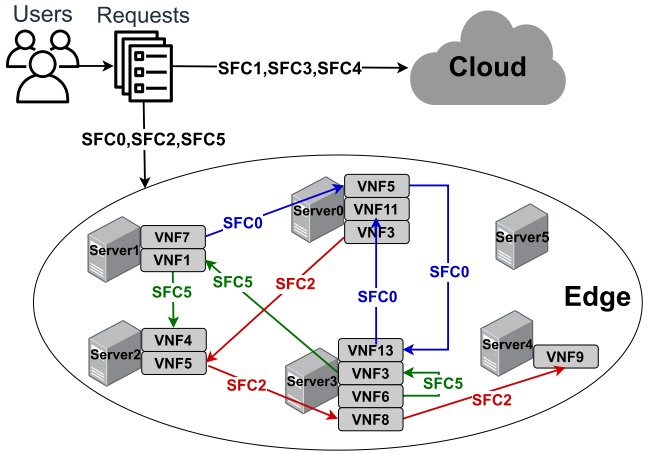


Fig. 1. An example of SFC backup deployment at the edge.

III. SYSTEM MODEL

In the context of mobile edge computing, the entire edge environment can be regarded as a network model, which can be modeled as an undirected connected graph $\mathcal{G}(\mathcal{N}, \mathcal{E})$, with the set of nodes $\mathcal{N}$ representing all servers at the edge of network and the set of edges $\mathcal{E}$ denoting the network links among servers. Fig. 1 shows a brief demonstration of entire system, which contains a central cloud server and an edge network consisting of multiple servers. In this figure, three SFCs, i.e., SFC0 (blue), SFC2 (red), and SFC5 (green), are backed up at the edge; while the backups of the remaining three SFCs are deployed at the cloud. For reference, we summarize main notations used in this paper in Table II.

TABLE II
NOTATIONS

Symbol	Explanation
$\mathcal{N}$	The set of edge servers $\mathcal{N} = \{1, 2, \cdots, n, \cdots, N\}$
M_n	The available resource of edge server n
$\mathcal{K}$	The set of users $\mathcal{K} = \{1, 2, \cdots, k, \cdots, K\}$
$\mathcal{F}$	The set of SFCs $\mathcal{F} = \{1, 2, \cdots, f, \cdots, F\}$
$\mathcal{I}$	The set of VNFs $\mathcal{I} = \{1, 2, \cdots, i, \cdots, I\}$
D_i	Resource requirements for backing up VNF i at the edge
I_f	The set of VNFs composing an SFC f
$Y_{i,f}$	Binary variable showing whether VNF i is involved in SFC f
$Q_{k,f}(t)$	Binary variable showing whether user k has a service request for SFC f during time slot t
$Q_f(t)$	Total requests for SFC f during time slot t
$q_f(t)$	The expected popularity of SFC f during time slot t
$X_f(t)$	Binary variable showing whether SFC f is placed at the edge during time slot t
$P^j_{i,n}(t)$	Binary variable showing whether VNF i is placed on edge server n during time slot t
$l(u, v)$	The latency of link between edge server u and v
$\mathcal{L}$	The set of link latency
$\phi^j_{f(u,v)}$	Count variable showing whether the jth link of chain f deployed at the edge passes through link (u, v)
$L_f(t)$	The total delay of SFC f during time slot t
$U_f(t)$	The failure rate of SFC f during time slot t
$V_i(t)$	The failure rate of VNF i during time slot t
$v_i(t)$	The expected failure rate of VNF i during time slot t
$R_f(t)$	Service hit reward brought by deploying the backup of SFC f at the edge during time slot t

A. Basic Parameters

Assuming that the collection of edge servers is represented by $\mathcal{N} = \{1, 2, \cdots, n, \cdots, N\}$ with N denoting the total number of edge servers, where the available storage resource of any edge server n to backup VNFs is defined as M_n. We denote the set of users by $\mathcal{K} = \{1, 2, \cdots, k, \cdots, K\}$ with K being the number of all users. User k will issue multiple service requests during time slot t, and each service request has to be responded by the corresponding SFC. All SFCs are included in the set $\mathcal{F} = \{1, 2, \cdots, f, \cdots, F\}$ and all VNFs are represented by the set $\mathcal{I} = \{1, 2, \cdots, i, \cdots, I\}$, in which F and I are respectively the numbers of SFCs and VNFs; and the resource requirement of backing up each VNF i on the edge server is defined as D_i. Besides, the VNFs composing an SFC f is denoted by the set $I_f \subseteq \mathcal{I}$ and there are $|I_f|$ VNFs in SFC f, which includes $|I_f| - 1$ links. Define a binary variable $Y_{i,f} \in \{0, 1\}$ to indicate whether VNF i is involved

in SFC f or not, i.e., $i \in I_f$ or not. If VNF i is a member of SFC f, we have $Y_{i,f} = 1$; otherwise, $Y_{i,f} = 0$.

B. Popularity of SFCs

Since our scheme selects SFCs to backup at the edge from the perspective of users' requests, it is necessary to obtain the service requests from users in each time slot for calculating the popularity of each SFC.

We define a binary variable $Q_{k,f}(t) \in \{0, 1\}$ to denote whether user k has a service request for SFC f in time slot t, where $Q_{k,f}(t) = 1$ if the service request sent by user k in time slot t is realized through SFC f and $Q_{k,f}(t) = 0$ when SFC f is not requested by user k. Then the total requests of all users for SFC f in time slot t, denoted by $Q_f(t)$, can be calculated as

$$Q_f(t) = \sum_{k \in \mathcal{K}} Q_{k,f}(t). \tag{1}$$

It is clear that $Q_f(t)$ is a discrete random variable in the set of $\{0, 1, 2, \cdots, k, \cdots, K\}$, which has an average mathematical expectation, denoted by $q_f(t)$, as follows,

$$q_f(t) = \mathbb{E}[Q_f(t)]. \tag{2}$$

In this article, we use this expected value $q_f(t)$ to express the popularity of SFC f during each time slot t, that is, the total expected expectation of service requests for SFC f sent by users in each time slot. We assume that the popularity of each SFC remains stable within the time slot range.

C. Backup Deployment Variables

In order to reflect whether a certain SFC is selected to be backed up at the edge, a binary variable $X_f(t) \in \{0, 1\}$ is used to represent the backup placement decision for SFC f during each time slot t. If it is decided to deploy the backup of SFC f at the edge in time slot t, $X_f(t) = 1$; otherwise, $X_f(t) = 0$.

Since SFCs are formed by linking required VNFs in sequence, the SFC backup deployment will eventually be implemented as the deployment of multiple VNFs. To indicate the backup placement decision of a certain VNF at the edge, we introduce a binary variable $P_{i,n}(t) \in \{0, 1\}$ to represent whether the backup of VNF i is placed on edge server n in time slot t. And we have $P_{i,n}(t) = 1$ if VNF i is stored on server n and $P_{i,n}(t) = 0$ if not.

To ensure that a certain SFC can provide complete functions at the edge, it is required that only when all VNFs in this chain are deployed on edge servers, can this SFC be considered to be backed up at the edge network successfully; otherwise, it will be placed in the cloud. According to this requirement, we can get the relationship between $X_f(t)$ and $P_{i,n}(t)$ as follows:

$$X_f(t) = 1 - \min\{1, \sum_{i \in I_f} (1 - P_{i,n}(t))\}. \tag{3}$$

D. Resource Constraint

To back up SFCs at the edge successfully, the resource constraint of edge servers cannot be ignored. In detail, the total resource requirement of all backups of VNFs placed on server n should not exceed the available resource of this server, i.e., M_n, which can be expressed as follows:

$$\sum_{i \in \mathcal{I}} D_i * P_{i,n}(t) \leq M_n. \tag{4}$$

E. Link Latency of SFCs

We assume that the latency of link (u, v) between the edge server u and edge server v is relatively stable and denote it as $l(u, v)$, all of which constitute a collection $\mathcal{L} = \{l(u, v), (u, v) \in \mathcal{E}\}$. For each SFC f, we use a count variable $\phi^j_{f(u,v)}$ to indicate whether the jth link of this service chain f deployed at the edge passes through the link (u, v). The value of $\phi^j_{f(u,v)}$ is set to 1 when the jth link of f passes through (u, v), and 0 otherwise. Then the total delay of SFC f deployed at the edge network in time slot t, defined as $L_f(t)$, can be calculated by:

$$L_f(t) = \sum_{j=1}^{|I_f|-1} \phi^j_{f(u,v)} * l(u, v). \tag{5}$$

F. Failure Rate Calculation

We all know that the operation of an SFC is based on the successful functioning of all VNFs composing this SFC. The failure of any one will lead to the failure of the entire SFC. Therefore, we are supposed to further consider improving the availability of SFC backups with the awareness that every SFC has a certain possibility to come into a failure. When this happens unfortunately, no matter how popular the SFC is and how low the latency it achieves, this SFC cannot successfully provide services to users and they won't get any benefits. We introduce $U_f(t)$ to represent the failure rate of SFC f in time slot t, and we have the success rate as $1 - U_f(t)$, which is jointly affected by the failure rates of the contained VNFs. Therefore, the problem is further transformed into obtaining the failure rate of each VNF.

We assume that the real-time failure rate of VNF i during time slot t is $V_i(t)$, which is a random variable within the range $[0, 1]$. After choosing one VNF to work each time, observe whether it fails (or whether it provides services successfully). By pulling this arm many times, it can be found that the distribution tends to stabilize. In other words, for VNF i, the probability of failing to realize the function has a fixed expected value, denoted by $v_i(t)$, from which we can get:

$$v_i(t) = \mathbb{E}[V_i(t)]. \tag{6}$$

Then the failure rate of VNF i is $v_i(t)$ and the success rate is $1 - v_i(t)$.

The failure of any VNF contained in an SFC will lead to the unavailability of the entire chain; in other words, only when all VNFs in the SFC run properly, can the function of this chain be successfully implemented. Therefore, we can get the probability of successful operation of SFC f as the product of

the success rates of all VNFs involved. Then, the failure rate of SFC f can be calculated as:

$$U_f(t) = 1 - \prod_{i \in I_f} (1 - v_i(t)). \tag{7}$$

In the above formula, we take into account the failure of any VNF, which helps ensure the availability of services, making our scheme more reliable.

G. Backup Reward

If an SFC corresponding to a user's request is backed up at the edge, there is no need to forward this request to the cloud. This clearly avoids additional traffic, reducing the propagation delay and users' access cost, which can be regarded as the improvement of users' service hit reward. To compare the hit reward brought by different backup schemes at the edge, we use $R_f(t)$ to represent the service hit reward brought by SFC f that satisfies the users' service request during time slot t, where the value of $R_f(t)$ is influenced by three factors, i.e., the popularity, the link delay and the failure rate of SFC f. In detail, deploying SFCs with high popularity $Q_f(t)$ at the edge can greatly reduce the cost of being routed to the cloud, especially for the frequently requested SFCs, which brings a higher service hit reward. Besides, when a certain SFC f is deployed at the edge, the lower the link delay $L_f(t)$, the faster the users' service requests can be satisfied, and the greater service hit reward would be obtained. Finally, the failure of one SFC can invalid the service provision to users, which will make the hit reward become zero.

Therefore, we define the service hit reward brought by deploying the backup of SFC f at the edge as:

$$R_f(t) = (\omega Q_f(t) - \mu L_f(t)) * X_f(t) * (1 - U_f(t)), \tag{8}$$

where $\omega, \mu > 0$ are constant scalars. Note that in the practical implementation, the relationship between these parameters can be determined by the service provider according to the specific environment and preferences.

H. Problem Formulation

According to the above description, our problem is to select the appropriate set of SFCs and deploy them at the edge to maximize time-average service hit reward with the comprehensive consideration of users' needs, SFC composition, and link latency, on the premise of satisfying edge resource constraint. Therefore, we can get the SFC backup problem formulated as:

$$\max \quad \frac{1}{T} \sum_{t=0}^{T-1} \sum_{f \in \mathcal{F}} \mathbb{E}[R_f(t)], \tag{9a}$$

$$\text{s.t.} \quad (3)(4),$$

$$P_{i,n}(t) \in \{0,1\}, \quad \forall n \in \mathcal{N}, \ \forall i \in \mathcal{I}, t, \tag{9b}$$

$$X_f(t) \in \{0,1\}, \quad \forall f \in \mathcal{F}, t. \tag{9c}$$

In the above equations, the optimization goal in (9a) is to maximize the time-average service hit reward; constraint (3) describes the relationship between the VNF backup variable $P_{i,n}(t)$ and the SFC backup variable $X_f(t)$; (4) is the resource constraint; constraints (9b) and (9c) show that SFCs and VNFs are either deployed at the edge or deployed in the cloud.

From (9a), we can know that the ultimate goal of our SFC backup problem is to maximize the time-average service hit reward. Substituting the definition of reward in (8) into the above problem, we can further specify the optimization goal as

$$\max \ \frac{1}{T} \sum_{t=0}^{T-1} \sum_{f \in \mathcal{F}} \mathbb{E}[(\omega Q_f(t) - \mu L_f(t)) * X_f(t) * (1 - U_f(t))]. \tag{10}$$

It is clear that the deployment strategy $X_f(t)$ of SFC f is independent of the request $Q_f(t)$, the link delay $L_f(t)$ and the failure rate $U_f(t)$. In addition, since the popularity $Q_f(t)$ has an expectation $q_f(t)$ from (2) and the failure rate of VNF $V_i(t)$ has an expectation $v_i(t)$ from (6), we can further transform the above equation into:

$$\max \ \frac{1}{T} \sum_{t=0}^{T-1} \sum_{f \in \mathcal{F}} \mathbb{E}[X_f(t)] * (\omega q_f(t) - \mu \mathbb{E}[L_f(t)]) * \prod_{i \in I_f} (1 - v_i(t)). \tag{11}$$

Theorem 1: SFC backup problem defined in (9) is an *NP-hard* problem.

Proof: The SFC backup problem can be transformed from the *Traveling Salesman Problem* (*TSP*), which is one of the best-known NP-Complete problems [25]. The general description of *TSP* is to find the best way of traversing all the destinations once and returning to the original city for achieving the lowest cost (or distance). We further describe it in mathematical language. Given a complete graph $\mathcal{G} = (\mathcal{H}; \mathcal{S})$ where $\mathcal{H}$ is a set of vertices and $\mathcal{S}$ is the edge set.

Let $\mathcal{M} = (m_{ij})$ be the distance from city $i \in \mathcal{H}$ to city $j \in \mathcal{H}$. Define decision variables z_{ij}, where $z_{ij} = 1$ means city j is visited immediately after city i while $z_{ij} = 0$ denotes that edge (i, j) is not selected. The goal of *TSP* can be expressed as: Minimize $\sum_{(i,j) \in \mathcal{S}} z_{ij} m_{ij}$. If there are n cities to be visited, the total number of possible ways can be inferred as $\frac{(n-1)!}{2}$, showing the time complexity of this problem is $O(n!)$. In this paper, for a specific SFC, we have to arrange all member VNFs on appropriate edge servers to realize the most efficient service provision while considering the limited resources of edge servers. That is to say, our task is to minimize the total delay of this SFC f, which is formulated as $\sum_{(u,v) \in \mathcal{E}} \phi_{f(u,v)} l(u, v)$. We can convert the classic *TSP* into our SFC backup problem. In our context, each VNF can be placed on any server that can accommodate. When we add resource constraints to destinations in *TSP* and relax the starting and ending positions to any two nodes that represent edge servers, we can reduce the *TSP* to our proposed SFC backup problem. Therefore, the time complexity of solving SFC backup problem is also $O(n!)$, indicating that it is an NP-hard problem. ■

IV. Solution of SFC Backup Problem: *RTSD*

In this section, to solve the SFC backup problem in the edge environment, we first conduct a further analysis in Section IV-A. Subsequently, the *RTSD* solution is proposed in Section IV-B to deal with the existing difficulties, and the specific implementation and detailed algorithm designs of each component in our scheme are introduced in Sections IV-C to IV-F.

A. Problem Analysis

According to the final problem in (11), we can see that the value of the reward is affected by three aspects, which is positively proportional to the popularity $q_f(t)$ of the chain f while inversely proportional to the total delay L_f of SFC f deployed at the edge and the failure rate of VNFs in each SFC. Thus, we need to find the optimal strategy to select the popular set of SFCs to deploy on the appropriate edge servers so as to obtain the biggest service hit reward. The SFC backup selection and deployment problem can be decomposed into four steps:

1) For a specific SFC, we back up all involved VNFs on edge servers with limited resources, making the link delay of this chain as low as possible;
2) Obtaining the popularity $q_f(t)$ of each SFC;
3) Obtaining the failure rate $v_i(t)$ of each VNF in each SFC;
4) According to the above results, we can get the maximum service hit reward corresponding to each SFC, and we give priority to the chain with the largest reward so that it will be backed up at the edge based on the minimum delay deployment plan until no backup resources available.

In this process, we need to address four challenges: *the dynamic and unknown popularity of SFC, the dynamic and unknown failure rate of SFC, the indeterminate latency of SFC*, and *the limited resources of edge server*.

The first two are the exploration of unknown knowledge, that is, when the current information cannot be obtained in time and the historical information is insufficient, how to obtain its estimated value more accurately so as to make a choice to maximize the reward; the last two challenges are constraints in a known environment, that is, how to perform allocation and connection actions, and place multiple VNFs forming a chain on multiple server nodes with the minimal link latency. In order to address the above challenges, we will introduce the overall idea and general process of our online learning scheme to deploy the appropriate set of SFC backups at the edge in Section IV-B. The corresponding algorithm will be designed to calculate the latency of whole chain in Section IV-C, which further solves the link completeness. For the problem of dynamic and uncertain users' preferences and VNF failure, we will propose corresponding algorithms to learn and predict the uncertain popularity and failure rate in Section IV-D and Section IV-E, respectively. Subsequently, selection and deployment are performed in Section IV-F based on the obtained results.

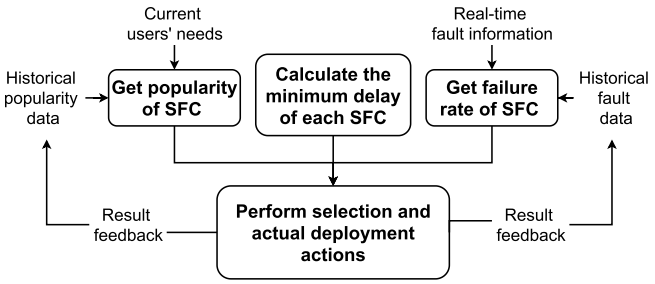


Fig. 2. An illustration of *RTSD*.

B. Real-Time Selection and Deployment

To address the issue mentioned above, inspired by the Upper Confidence Bound (UCB) algorithm, that is, deigned for solving the multi-armed bandit learning problem [26], we can analyze the historical information captured before and estimate the popularity and failure rate under the current time slot combined with real-time users' service requests and surroundings. In this process, we are faced with a challenge that cannot be ignored, that is, how to deal with the trade-off between exploration and exploitation. Among them, the former is to explore potential new arms that may generate higher returns without considering previous experience, which is a radical plan. The advantage is that we can find options with higher reward, but we cannot use the existing high-return arms. The latter develops and uses known arms that can produce high returns based on the content that has been explored to get the best strategy so far. The advantage of this scheme is that it can make full use of existing knowledge, but because of this, it is only limited to the current local optimal scheme and may ignore the better content that has not been explored, thus missing the opportunity to generate higher reward, which is a conservative choice. Therefore, we need to find a balance between two contradictions, so as to maximize the service hit reward at each time slot.

Based on the above ideas, we model our SFC backup problem with unknown popularity and failure information as a multi-armed bandit learning problem, where SFCs can be considered as multiple arms, and the service hit reward obtained by deploying a certain SFC at the edge is the reward brought by pulling an arm. During each time slot, combining historical data, we learn the current popularity of all SFCs according to users' requests and the real-time failure rate of VNFs according to current environment, calculate the reward of deploying each SFC, and then make a selection based on the calculation result to determine which SFC should be placed at the edge in the current time slot to get the highest return. The deployment result will be used as feedback information to participate in the selection and deployment of SFCs in the next time slot.

We propose an algorithm named *Real-Time Selection and Deployment* (*RTSD*) based on the above ideas, whose main idea is illustrated by Fig. 2 for a more logical presentation.

In Algorithm 1, we describe the proposed *RTSD* solution in a formal way, which is implemented with five main parts, namely *Initialization* at the beginning and *MinimizeDelay,*

Algorithm 1 *RTSD*

Input: edge server set $\mathcal{N}$, user set $\mathcal{K}$, SFC set $\mathcal{F}$, and VNF set $\mathcal{I}$

1 **Initialization**
2 **foreach** *SFC* $f \in \mathcal{F}$ **do**
3 $c_f(0) = 0, \bar{q}_f(0) = 0, \tilde{q}_f(0) = Q_f(0)$;
4 $h_i(0) = 0, \bar{v}_i(0) = 0, \tilde{v}_i(0) = V_i(0)$;
5 **end**
6 **end**
7 **foreach** *time slot* $t \in \{0, 1, \cdots\}$ **do**
8 **while** *edge resources are sufficient* **do**
9 **MinimizeDelay**
10 GETCONSUMPTION(t);
11 **end**
12 **SFCPlacement**
13 VNFDEPLOYMENT(t);
14 **end**
15 **end**
16 **Popilarity**
17 GETPOPULARITY(t);
18 **end**
19 **FailureRate**
20 GETFAILURERATE(t);
21 **end**
22 **end**

SFCPlacement, Popularity, FailureRate in every subsequent time slot. When the whole process begins, we have no historical information about SFCs and VNFs, so the value of each SFC's popularity and VNF's failure rate can only be estimated based on service requests collected from end users. For this reason, we have to set the number of times when each SFC is learned at the edge and the average value of popularity, denoted as $c_f(0)$ and $\bar{q}_f(0)$, respectively, as zero, which will participate in learning the estimate of popularity. And the popularity can be considered as the sum of service requests issued by all users for each SFC during time slot 0, that is, $\tilde{q}_f(0) = Q_f(0)$. Similarly, we set both $h_i(0)$ and $\bar{v}_i(0)$ respectively to zero and $\tilde{v}_i(0)$ to the real-time information in time slot 0, i.e., $V_i(0)$ (Lines 1-6).

In the second step, our goal is to obtain the minimum link latency of each SFC (Lines 9-11). And subsequently, we choose the best set of SFCs and deploy the corresponding VNFs on determined edge servers based on the popularity, the failure rate and the minimum delay obtained to get the biggest service hit reward (Lines 12-14). The selection and deployment actions will stop when the edge network can no longer accommodate any SFC (Line 8). After the above actions are completed, the SFC backup deployment task of the current time slot has been implemented. Then we will update the historical information according to the deployment result in this time slot. Combined with the users' requests, we get new popularity of each SFC (Lines 16-18). Next, we will also reckon the latest failure rate based on historical data (Lines 19-21) and then calculate the minimum delay again, so as to prepare for the SFC selection and deployment task in the next time slot. The above steps will be repeated for each subsequent time slot to select a set of qualified SFCs and properly deploy them on the edge.

The specific implementation of the related sub-algorithms will be elaborated in the following subsections.

C. Link Delay of SFC Backups

1) Realization Idea: In this part, our task is to arrange multiple VNFs contained in one chain on appropriate edge servers with limited resources. The goal is to find the optimal deployment scheme with the lowest total delay consumed by this SFC.

In edge network, there is a certain delay between any two edge servers, the value of which is specified in advance. Our task is to find the connection sequence of a group of available edge servers to make the link delay consumed by connecting them the lowest so that the SFC backed up in this path will be the most efficient. In order to analyze this abstract problem more clearly, we can regard edge servers as nodes in a graph. The delay between servers is materialized into the link with certain weight between two nodes, so that the edge network constitutes a weighted undirected complete graph, where each node has its own capacity (that is, the available resource of each server). Under this specific model, our problem can be seen as *finding a path with the smallest sum of weights that meets the capacity requirement from a complete graph*.

Inspired by the problem of *Traveling Salesman Problem* (*TSP*), we use the idea of Prim algorithm to solve this problem. However, it is worth noting that our problem is different from *TSP* in the following two aspects:

- We all know that the *TSP* is to find a minimum distance circuit that each vertex once and only once. But in our scenario, we do not need to include all nodes, i.e., edge servers, in the graph, and don't need to go back to the beginning. This is because it is not necessary for any certain SFC to travel through all edge servers to realize the specified services;
- Our problem has a resource constraint for each edge server, which is not involved in the traditional *TSP* problem. Specifically, each node has a certain amount of resources for backing up SFCs.

In spite of these differences, the same thing is that we all need to achieve the smallest sum of weights for the selected edges. To this end, we learn from the Prim algorithm dealing with the *Minimum Spanning Tree* (*MST*) problem, based on which we make appropriate expansions and improvements to form an effective solution for our SFC backup problem.

2) Algorithm Implementation: In Algorithm 2, we present the step of GETCONSUMPTION in detail, where the minimum delay of each SFC and its optimal deployment at the edge are calculated. For SFC f, put the contained VNFs on the determined edge servers in order. Since we need to minimize the total link delay, we give priority to the two servers with the least latency. In addition, a server with larger capacity is easier to accommodate more VNFs, which is more likely to bring zero time consumption for the corresponding links. These two strategies allow us to get the minimum delay

Algorithm 2 GETCONSUMPTION(t)

Input: the current time slot t, VNF set $\mathcal{I}$, the resource request of each VNF $\mathcal{D}$, edge link set $\mathcal{E}$, link latency set $\mathcal{L}$, the available resources of edge server $\mathcal{M}$

Output: the lowest link latency consumed by each SFC $\{L_f(t)\}$

1 Find link l' with the lowest latency;
2 Find server n' with larger resource capacity at both ends of l';
3 $\mathcal{P} \leftarrow \phi$;
4 **foreach** *SFC* $f \in \mathcal{F} \setminus \mathcal{P}$ **do**
5 $\mathbf{CurrentNode} \leftarrow n'$;
6 **foreach** $i \in I_f$ **do**
7 **while** $\mathbf{CurrentNode}$ *can hold current VNF* **do**
8 Place VNF i on $\mathbf{CurrentNode}$;
9 **end**
10 Traverse the link emanating from $\mathbf{CurrentNode}$ in the order of increasing delay;
11 $\mathbf{CurrentNode} \leftarrow$ the edge server that can accommodate VNF i at the other end of the link;
12 **end**
13 **if** *all VNF* $\in I_f$ *are palced on determined servers* **then**
14 Calculate the total delay consumed by SFC f;
15 **end**
16 $\mathcal{P} = \mathcal{P} \cup \{f\}$;
17 **end**

of each SFC in the edge environment. Therefore, we first consider placing VNFs on the larger end of the link with the smallest weight (Lines 1-2). Continue to put the VNFs in SFC f on the server until it cannot accommodate the current VNF (Lines 7-9). Learning from the idea of Prim algorithm, we diffuse outward from the current server node. Taking the consumption into account, we traverse the diverging edges of the current node in the order of increasing latency until we find a server with available resources that can be fully used to deploy the current VNF (Line 10). Update the current node (Line 11) and repeat the above process until all VNFs in this SFC have been traversed. For this chain, there are two results. First, each VNF in SFC f can be deployed on a certain server (Lines 13-15). At this time, the data chain formed by transmitting data packets in the order of VNFs consumes the minimum latency. Add delay of each link to get the total latency consumed by this SFC (Line 14). Otherwise, one or several VNFs in this SFC cannot be deployed at the edge. Specifically, no server can meet their resource requirements. In this case, (3) shows that when there are VNFs in one SFC deployed at the cloud, we believe that the backups of the entire chain need to be deployed in the cloud network. At this point, the process of solving the minimum delay of the current SFC is over, and we need to continue processing other chains. In order to prevent double calculation, we introduce a set $\mathcal{P}$ to store the traversed SFCs (Line 3), SFCs that have been judged will be put into $\mathcal{P}$ (Line 16), then we can select from the remaining SFCs for calculation (Line 4). The work of Algorithm 2 is to find a suitable edge server for each VNF in each SFC, and the time it takes is determined by the number of chains $|\mathcal{F}|$, the number of VNFs $|I_f|$ contained in each chain and the number of edge servers $|\mathcal{N}|$. Therefore, the time complexity of Algorithm 2 is $O(FI_fN^2)$.

As mentioned above, we design Algorithm 2 inspired by the *Prim* algorithm to find the minimum latency of each SFC. According to Theorem 1, the delay calculation is NP-hard, which is difficult to be resolved for acquiring an optimal solution in polynomial time. Therefore, Algorithm 2 is devised to obtain its near-optimal result. Then, we can calculate the worst-case approximate ratio of Algorithm 2 to verify its correctness. For the latency of a certain SFC, we assume that the result obtained by the optimal scheme is L^* and that by Algorithm 2 is L. For our problem, we consider the worst case where an SFC goes through all servers. Besides, when deploying each VNF of this SFC, the server directed by the link with the lowest latency cannot afford the current VNF, so we have to choose another link with larger latency. In this case, we know that the latency of the optimal solutions will not be lower than the result of the *MST* according to the definition. Inspired by the approximate ratio calculation of *TSP* [27], in the process of placing VNFs on specific servers and connecting them to form a link using Algorithm 2, we can deduce that the sum of the link latency would be less than twice the result of *MST* according to the Triangle Inequality. Therefore, the worst-case approximate ratio of Algorithm 2, denoted as ζ, satisfies:

$$\zeta = \frac{L}{L^*} < 2. \tag{12}$$

Only when all VNFs in an SFC are deployed on the edge servers can we regard that the chain would be successfully backed up at the edge. At the same time, according to the pre-deployment scheme and the calculated possible latency of each SFC, we can deploy the SFC backup with the lowest latency to provide users with timely services. This process can only be implemented when an SFC is fully deployed at the edge. Therefore, Algorithm 2 is a good solution to the challenge of ensuring minimum latency and SFC completeness in backup issues.

D. Learning Popularity of SFCs

1) Realization Idea: As analyzed at the beginning of Section IV-B, we need to make full use of historical information and combine the current data to estimate uncertain attributes. In order to make full use of the historical service request information that has been obtained, we define a variable $c_f(t)$ to represent the number of time slots when SFC f is learned in the edge network, that is, the time slots when SFC f is selected to be backed up at the edge, which can be

calculated as:

$$c_f(t) = \sum_{\tau=0}^{t-1} X_f(\tau). \tag{13}$$

In addition, we define a variable $\bar{q}_f(t)$ to record the average value of the learned popularity of SFC f up to time slot t, and its formula is described as follows:

$$\bar{q}_f(t) = \frac{\sum_{\tau=0}^{t-1} Q_f(\tau)}{c_f(t)}. \tag{14}$$

During each time slot t, the current demand $Q_f(t)$ of each SFC is obtained based on users' service requests, according to which we calculate the average value of learned popularity up to time slot t. With reference to the UCB algorithm, we then combine the historical information and current requests to estimate the popularity of SFC, expressed as $\tilde{q}_f(t)$, which is changing continuously. The calculation of $\tilde{q}_f(t)$ is as follows:

$$\tilde{q}_f(t) = \bar{q}_f(t) + K\sqrt{\frac{3\log t}{2c_f(t)}}. \tag{15}$$

According to (13)-(15), we can learn the popularity of SFC f in time slot t, and further obtain the service hit reward $R_f(t)$ of the chain, which will directly determine the deployment decision $X_f(t)$ in edge environment, i.e., determining which SFC backup to be placed at the edge. After the above actions are completed, we need to update the involved parameter values, i.e., $c_f(t)$ and $\bar{q}_f(t)$, based on the decision result so as to continue contributing to the selection and deployment process in next time slot. The corresponding update formulas are defined as follows:

$$c_f(t+1) = \begin{cases} c_f(t), & X_f(t) = 0, \\ c_f(t)+1, & X_f(t) = 1, \end{cases} \tag{16}$$

$$\bar{q}_f(t+1) = \begin{cases} \bar{q}_f(t), & X_f(t) = 0, \\ \dfrac{c_f(t)\bar{q}_f(t) + Q_f(t+1)}{c_f(t+1)}, & X_f(t) = 1. \end{cases} \tag{17}$$

2) Algorithm Implementation: We design a GETPOPULARITY algorithm to deal with the popularity which is changing dynamically, as shown in Algorithm 3. According to the description in Section IV-D.1, to estimate the popularity of each SFC in the next time slot, we first need to calculate its corresponding parameter information, that is, $c_f(t)$ and $\bar{q}_f(t)$, according to the SFC backup placement strategy of the current time slot, $X_f(t)$ (Lines 1-6). The specific parameter update process is defined in (16) and (17). After mastering the two major parameters involved in estimating popularity, we will implement the online learning step according to (15) (Lines 7-13). The estimate value of each SFC's popularity is obtained in Line 10 of Algorithm 3. The time complexity of Algorithm 3 is $O(F)$.

By solving the CMAB problem, we estimate the unknown popularity of SFC and continuously adjust it through feedback information so as to solve the challenge of dynamic demand.

Algorithm 3 GETPOPULARITY(t)

Input: the current time slot t, the number of learning slots $\{c_f(t)\}$, the average value of historical popularity $\{\bar{q}_f(t)\}$

Output: the popularity of each SFC for next time slot $\{\tilde{q}_f(t+1)\}$

1 **GetParameters**
2 **foreach** *SFC* $f \in \mathcal{F}$ **do**
3 $c_f(t+1) = \begin{cases} c_f(t), & X_f(t) = 0 \\ c_f(t)+1, & X_f(t) = 1 \end{cases}$
4 $\bar{q}_f(t+1) = \begin{cases} \bar{q}_f(t), & X_f(t) = 0 \\ \dfrac{c_f(t)\bar{q}_f(t) + Q_f(t+1)}{c_f(t+1)}, & X_f(t) = 1 \end{cases}$
5 **end**
6 **end**
7 **OnlineLearning**
8 **foreach** *SFC* $f \in \mathcal{F}$ **do**
9 **if** $c_f(t+1) > 0$ **then**
10 $\tilde{q}_f(t+1) = \bar{q}_f(t+1) + K\sqrt{\frac{3\log t}{2c_f(t+1)}}$
11 **end**
12 **end**
13 **end**

E. Learning Failure Rate of SFCs

1) Realization Idea: Similar to the SFC popularity, the failure rate of any VNF is also unknown a priori, which changes every time slot and cannot be known in advance. Therefore, we also adopt the UCB-based idea to learn its value. We first introduce a variable $h_i(t)$ to record the total number of time slots when VNF i is learned until time slot t. In Section III-C, we use $P_{i,n}(t)$ to reflect the placement decision of VNF i, based on which, we can calculate the value of $h_i(t)$:

$$h_i(t) = \sum_{\tau=0}^{t-1} \sum_{n \in \mathcal{N}} P_{i,n}(\tau). \tag{18}$$

Then, the history information denoted by $\bar{v}_i(t)$, that is, the average value of all failure rate of this VNF also needs to be obtained according to the following formula:

$$\bar{v}_i(t) = \frac{\sum_{\tau=0}^{t-1} V_i(\tau)}{h_i(t)}. \tag{19}$$

Using the above two variable values, we can estimate the value of the current failure rate in time slot t with the help of UCB algorithm. The formula is described as follows:

$$\tilde{v}_i(t) = \bar{v}_i(t) + K\sqrt{\frac{3\log t}{2h_i(t)}}. \tag{20}$$

With the consideration of SFC failure, we will get a comprehensive reward by integrating the popularity, link latency and the failure rate obtained above based on (8). According to this value, the deployment decision of each SFC, i.e., $X_f(t)$, can be determined, which will affect the popularity in the next

time slot as seen in (16) and (17). In addition, the deployment decision of VNF, i.e., $P_{i,n}(t)$, can also be found out, which serves as the feedback of the failure rate of next slot. The corresponding updates are as follows:

$$h_i(t+1) = \begin{cases} h_i(t), & \sum_{n\in\mathcal{N}} P_{i,n}(t) = 0, \\ h_i(t) + \sum_{n\in\mathcal{N}} P_{i,n}(t), & \sum_{n\in\mathcal{N}} P_{i,n}(t) > 0, \end{cases} \tag{21}$$

$$\bar{v}_i(t+1) = \begin{cases} \bar{v}_i(t), & \sum_{n\in\mathcal{N}} P_{i,n}(t) = 0, \\ \dfrac{h_i(t)\bar{v}_i(t) + V_i(t+1)}{h_i(t+1)}, & \sum_{n\in\mathcal{N}} P_{i,n}(t) > 0. \end{cases} \tag{22}$$

2) Algorithm Implementation: Using Algorithm 4, we present the calculation process of SFC failure rate. After the initialization is completed in time slot 0, we make a random selection without learning. In the subsequent time slots, the values of related parameters, i.e., the number of time slots when VNF is learned $h_i(t)$ and the average value of history data $\bar{v}_i(t)$, are updated based on the selection result of the previous time slot (Lines 2-5), which will participate in the estimation of the failure rate according to UCB Algorithm (Lines 8-12). Because the smooth function of SFC depends on the successful realization of each VNF it contains, it is necessary to integrate the data of all VNFs to obtain the failure rate of the entire SFC (Lines 14-16). Only when this SFC does not fail can it bring an ideal return, based on which we will also make the backup selection and deployment of next round. The time complexity of Algorithm 4 is $O(I)$.

The uncertain and dynamic failure rate can thus be predicted through repeated learning and exploration.

Algorithm 4 GETFAILURERATE(t)

Input: the current time slot t, the number of learning slots of failure $\{h_i(t)\}$, the average value of historical failure rate $\{\bar{v}_i(t)\}$

Output: the failure rate of each VNF for next time slot $\{\tilde{v}_i(t+1)\}$

1 **GetParameters**
2 **foreach** *VNF* $i \in \mathcal{I}$ **do**
3 $h_i(t+1) = \begin{cases} h_i(t), & \sum_{n\in\mathcal{N}} P_{i,n}(t) = 0, \\ h_i(t) + \sum_{n\in\mathcal{N}} P_{i,n}(t), & \sum_{n\in\mathcal{N}} P_{i,n}(t) > 0, \end{cases}$
4 $\bar{v}_i(t+1) = \begin{cases} \bar{v}_i(t), & \sum_{n\in\mathcal{N}} P_{i,n}(t) = 0, \\ \dfrac{h_i(t)\bar{v}_i(t) + V_i(t+1)}{h_i(t+1)}, & \sum_{n\in\mathcal{N}} P_{i,n}(t) > 0. \end{cases}$
5 **end**
6 **end**
7 **OnlineLearning**
8 **foreach** *VNF* $i \in \mathcal{I}$ **do**
9 **if** $h_i(t+1) > 0$ **then**
10 $\tilde{v}_i(t+1) = \bar{v}_i(t+1) + K\sqrt{\frac{3\log t}{2h_i(t+1)}}$;
11 **end**
12 **end**
13 **end**
14 **foreach** *SFC* $f \in \mathcal{F}$ **do**
15 $U_f(t) = 1 - \prod_{i\in I_f}(1 - v_i(t))$;
16 **end**

F. Selection and Deployment

1) Realization Idea: As shown in problem (9a), our goal is to maximize the time-average hit reward. Therefore, we need to choose and deploy backups of the most suitable SFCs at the edge with limited resources. In this case, our selection criterion is naturally the reward of each SFC.

2) Algorithm Implementation: In Algorithm 5, when the popularity, failure rate and minimum link delay of a certain SFC have been obtained by the previous algorithms, we can calculate the pre-reward of this SFC, expressed by R'_f, according to (8) (Lines 1-3). Then we will choose the SFC that can bring the greatest reward, and deploy the VNFs it contains on determined edge servers according to the pre-deployment scheme with the least delay in Algorithm 2 (Lines 4-5), which will reduce the available resources of the corresponding edge servers (Lines 6-8). The complexity of Algorithm 5 is $O(F)$.

The placement decisions of SFC backups can realize the deployment of a set of appropriate SFC backups in the edge of the network. They will be enabled when the original SFCs fail, ensuring service availability.

G. Approximate Ratio

As mentioned in Section III-H, the selection and deployment of SFC backups in edge environment is an NP-hard problem that is difficult to be resolved in polynomial time for obtaining an optimal solution. Therefore, we design the above near-optimal scheme, namely *RTSD*, to solve this problem efficiently. In this section, we will analyze the *RTSD* scheme and calculate the approximate ratio.

The general flow of the *RTSD* scheme is presented in Section IV-A, while the implementation idea and specific algorithm of each step are presented in the subsequent Sections IV-B to IV-F. In short, we first calculate the minimum latency of each SFC through a deployment algorithm, then learn and predict the unknown popularity and failure rate using the bandit learning method. The above results will jointly affect the value of the service hit reward, which is supposed to be maximized for deriving the optimal backup scheme. We can see that, in this process, both the bandit learning process and the final process of selecting the maximum reward will output the accurate value. Therefore, the key step affecting the proposed *RTSD* scheme in achieving the optimal results is the latency calculation process.

Assume that the lowest delay of chain f and its corresponding service hit reward obtained by the *RTSD* scheme

Algorithm 5 VNFDEPLOYMENT(t)

Input: the current time slot t, the learned popularity of SFCs $\{\tilde{q}_f(t)\}$, the lowest link latency consumed by each SFC $\{L_f(t)\}$
Output: the placement decision of SFC backups $\{X_f(t)\}$

foreach *SFC* $f \in \mathcal{F}$ **do**
  $R'_f(t) = (\omega Q_f(t) - \mu L_f(t)) * (1 - U_f(t))$;
end
Select SFC with the largest reward to place on the determined edge servers;
$X_f(t) \leftarrow 1$;
$P_{i,n}(t) \leftarrow 1$;
foreach *Edge Server* $n \in \mathcal{N}$ **do**
  Update the resource of each server M_n;
end

are L_f and R_f, respectively; while their values calculated by the optimal solution are L_f^* and R_f^*, respectively. Then we have

$$L_f > L_f^*, \tag{23}$$

and we need to derive the numerical relationship between R_f^* and R_f.

As mentioned above, given that the values of popularity and failure rate obtained by the *RTSD* scheme are the same as that obtained by the optimal-latency scheme, which are denoted by Q_f and U_f, respectively. According to (8), when f is backed up on the edge, that is, when $X_f = 1$, we can get that:

$$R_f = (\omega Q_f - \mu L_f) * (1 - U_f), \tag{24}$$
$$R_f^* = (\omega Q_f - \mu L_f^*) * (1 - U_f). \tag{25}$$

Then, the approximate ratio, denoted by η, can be calculated by:

$$\eta = \frac{R_f^*}{R_f} = \frac{(\omega Q_f - \mu L_f^*) * (1 - U_f)}{(\omega Q_f - \mu L_f) * (1 - U_f)} = \frac{\omega Q_f - \mu L_f^*}{\omega Q_f - \mu L_f} = 1 + \frac{\mu(L_f - L_f^*)}{\omega Q_f - \mu L_f}. \tag{26}$$

We can see that the value of η depends on the last item. For the numerator, its value is affected by the lowest delay of chain f obtained by the *RTSD* scheme and that derived by the optimal solution. According to the calculation in Section IV-C, the worst-case approximate ratio of latency $\zeta = \frac{L_f}{L_f^*} < 2$. And for the denominator, it is related to the estimated popularity, the total latency of the chain at the current moment, and the two scalar parameters. Thus, the specific value of η has to be calculated based on the parameters obtained at a specific time in the actual configuration.

V. EXPERIMENTAL EVALUATION

In this section, we evaluate our proposed solutions for SFC backup placement at the edge based on extensive simulations, which are conducted on a desktop with Intel(R) Core(TM) i7-9700 CPU @3.00GHz and 16GB RAM running Windows 10 OS.

A. Experiment Settings

We simulate an edge network with 50 edge servers. Each edge server has a limited resource capacity, which is an inherent property and is determined in advance by its own. In the experiments, we randomly assign a specific value between [10] and [20] to each edge server in advance to reflect the resource as shown in Table III. In addition, any two servers have a predetermined and fixed link latency which is determined by the current edge environment. Fluctuations of transmission medium and network configuration will cause changes in the current latency between different servers. Therefore, in our experiments, we use a random function, *RANDINT*, to generate a fixed latency of [1] and [10] for the link between any two servers.

In this paper, we also need to collect users' service requests. Assume that their requests correspond to 20 SFCs in total, consisting of 15 different VNFs. Table IV shows the affiliation between SFCs and VNFs. Similar to the edge server settings, each VNF also has its inherent properties, that is, the required resource for backing it up successfully, which is also allocated in advance as shown in Table V.

B. Superiority Evaluation of RTSD

We evaluate the superiority of *RTSD* algorithm through five sets of comparison experiments. As discussed earlier, we mainly resolve three challenges of SFC backup problem at the edge, namely the availability of SFCs, dynamic and unknown parameters, and the latency of placing SFC backups. Therefore, here we first design three benchmark solutions, namely *Failure-free* scheme, *Non-learning* scheme and *Latency-unconstrained* scheme. Besides, to further illustrate the performance, we compare our scheme with two existing ones, namely *BSPS* [23] and *Topology* [24] schemes. Specifically, five schemes for comparison can be introduced as below.

1) **Failure-free scheme.** SFC availability is not considered in this scheme, where all SFCs are assumed to be successfully implemented. While Algorithms 2 and 3 are still run to obtain the estimated popularity and minimum link latency. And we choose the SFC with the largest reward to be deployed at the edge.
2) **Non-learning scheme.** We remove the online learning method to further confirm the validity of bandit learning in Algorithms 3 and 4. Instead of adjusting the selection action based on the estimation and feedback, we select an SFC randomly to be deployed at the edge during each time slot. For the selected SFC, we also randomly select a server to determine whether it can accommodate any specific VNF. When all member VNFs satisfy the edge resource constraint, the deployment of this SFC is completed.
3) **Latency-unconstrained scheme.** When we are examining a specific server, once the resource demand of the VNF exceeds the remaining available resources of the current server, we continue to traverse and look for other qualified server. This scheme focuses on the deployment of every VNF instead of the whole SFC,

TABLE III
RESOURCE LIMITATION OF EDGE SERVERS

Server No.	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
Available resource	13	16	12	14	17	18	15	11	15	18	12	15	16	18	11	16	18	19	15	12
Server No.	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39
Available resource	11	16	18	16	14	12	17	16	17	19	13	15	11	15	16	16	13	11	16	18
Server No.	40	41	42	43	44	45	46	47	48	49										
Available resource	17	11	16	15	18	14	19	12	11	15										

TABLE IV
SFC SETTING

$SFC0$	$VNF3-VNF6-VNF9-VNF7-VNF14-VNF11-VNF10$
$SFC1$	$VNF9-VNF8-VNF0-VNF3-VNF4-VNF2$
$SFC2$	$VNF3-VNF1-VNF6-VNF10-VNF0-VNF9-VNF4$
$SFC3$	$VNF10-VNF14-VNF1-VNF4-VNF11-VNF8$
$SFC4$	$VNF0-VNF11-VNF13-VNF1-VNF4$
$SFC5$	$VNF8-VNF1-VNF12-VNF10$
$SFC6$	$VNF13-VNF9-VNF5-VNF10-VNF1-VNF6-VNF9$
$SFC7$	$VNF4-VNF2-VNF14-VNF11-VNF7-VNF6-VNF12$
$SFC8$	$VNF10-VNF2-VNF5-VNF0-VNF8$
$SFC9$	$VNF8-VNF13$
$SFC10$	$VNF10-VNF7-VNF8-VNF0$
$SFC11$	$VNF0-VNF9-VNF12-VNF5-VNF6-VNF7-VNF11-VNF1-VNF3-VNF13-VNF8$
$SFC12$	$VNF10-VNF5-VNF9-VNF0-VNF2-VNF7-VNF11$
$SFC13$	$VNF2-VNF8-VNF1-VNF0-VNF5-VNF13-VNF10-VNF7$
$SFC14$	$VNF13-VNF5-VNF0-VNF3-VNF4-VNF6$
$SFC15$	$VNF3-VNF14-VNF8-VNF4$
$SFC16$	$VNF6-VNF7-VNF10-VNF1-VNF2-VNF5-VNF8-VNF3$
$SFC17$	$VNF12-VNF5-VNF0-VNF4-VNF9$
$SFC18$	$VNF13-VNF7-VNF14-VNF0-VNF2-VNF9$
$SFC19$	$VNF14-VNF9-VNF3-VNF10$

TABLE V
VNF SETTING

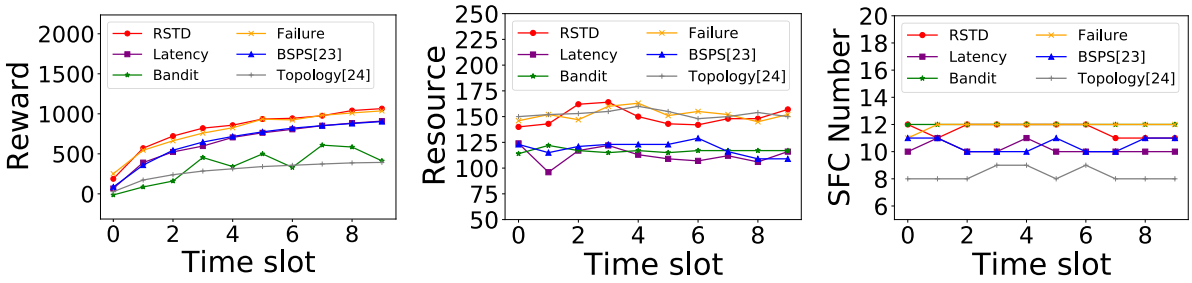
VNF No.	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
Resource requirement	8	15	12	8	9	13	10	8	7	15	11	9	13	10	6

which thus cannot ensure the minimum link latency. We still estimate its popularity and failure rate using the bandit learning method presented in Algorithms 3 and 4.

4) **BSPS scheme [23].** This method deploys individual VNF backups at the edge through learning the demand level of each VNF according to the users' requests and SFC composition. Hit reward is obtained after the operation is completed, which continues to affect the learning of demand level in the next time slot.
5) **Topology scheme [24].** A topological analysis method is used to determine a set of suitable SFCs, indicating which path tends to be condensed or not. The traffic congestion is predicted by calculating the frequency of each node in each SFC path, and the low frequency link is selected to avoid congestion. This scheme realizes the deployment of SFC from the perspective of server, ensuring the link completeness.

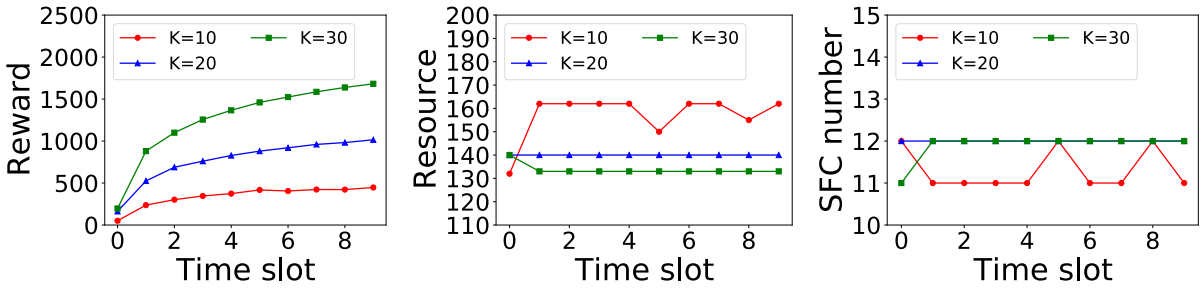
The experimental results are presented from three aspects. First of all, since our SFC backup problem considers from the users' point of view, we observe the benefits that the solution can bring to users, which is affected by three factors, i.e., popularity, failure rate, and latency. In this work, the total service hit reward is defined to integrate these three factors, making it a direct performance indicator to be examined. Besides, from edge servers' perspective, the less wasted resources at the edge, the better resource utilization, the more services can be provided efficiently, and the better the performance of our solution. Therefore, the remaining resource of the edge server is another performance metric evaluated in the experiments. In addition, the total number of SFCs deployed at the edge can reflect the services that can be implemented efficiently at the edge. So, it is also an important performance indicator to be observed.

Fig. 3 shows the comparison results, from which we can observe that the experimental results of *RTSD* scheme are basically consistent with the *Failure-free* scheme, which shows that our solution can achieve the effect that each selected SFC backup is available at the edge considering the actual failure, and thus effectively solving the availability problem. And the service hit reward of the *RTSD* scheme is significantly higher than that of the *Non-learning* scheme as shown in Fig. 3(a), which means that our online bandit learning strategy can select a better batch of SFCs for edge backup and improve users' experience. Fig. 3(b) shows that compared with the *Non-learning* scheme, the *RTSD* scheme can use fewer resources. This is because when SFCs with higher resource requirements or lower popularity fail, the reward will be correspondingly lower, which makes them less likely to be deployed at the edge according to online bandit learning



(a) Hit reward (b) Remaining resource (c) The number of SFC backups

Fig. 3. Comparison results between *RTSD* and comparison experiments.



(a) Hit reward (b) Remaining resource (c) The number of SFC backups

Fig. 4. The effect of changing the number of users on *RTSD*.

strategy. For Fig. 3(c), the *Non-learning* scheme still uses the pre-deployment idea to make full use of edge resources, so the total number of backups is not significantly different from the *RTSD* scheme. Further, we can see that the minimum latency deployment strategy helps deploy links on edge servers more reasonably, thus obtaining higher rewards and saving more edge resources from Figs. 3(a) and 3(b). However, Since the *Latency-unconstrained* scheme does not implement the overall deployment from the perspective of a complete chain, the SFC with low latency and high demand will occupy a large number of edge resources, thus reducing the total number of backups as shown in Fig. 3(c).

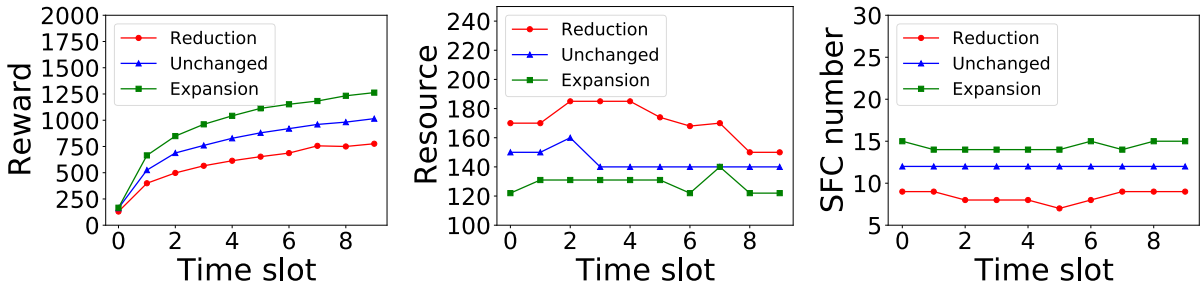
In addition, compared with our work, *BSPS* scheme does not consider the uncertainty of the actual failure rate, which makes it insufficient to cope with the changing environment. Besides, it focuses on the placement of individual VNFs rather than the deployment of complete SFCs, leading to high latency and reduced reward as implied in Fig. 3(a). Similarly, the lack of completeness guarantee will also cause the consumption of a large number of edge resources and the reduction of the number of backups in Figs. 3(b) and 3(c). Therefore, the results of *BSPS* scheme are unsatisfactory. The *Topology* scheme ignores users' requirements, so its service hit reward is bound to be low. Due to the concerns around server load, it cannot fully utilize edge resources to deploy SFCs as many as possible, so this scheme cannot achieve a comparable result as *RTSD*. Through comparison, it is clear that our proposed scheme has more advantages.

The above results show that *RTSD* solves the selection and deployment problem of SFC backup in the edge environment more effectively. In this paper, we use the online bandit learning method to learn the unknown popularity and failure rates of SFCs. By obtaining feedback continuously to adjust the selection action, we seek an optimal balance between exploration and exploitation and then address the challenge caused by the dynamic and uncertain users' preferences and VNF failure. Specifically, in the current time slot, we do not know the users' demand and whether the SFC will fail. Using the historical information obtained in the previous time slot and the hit feedback, we make predictions on these uncertain parameters and obtain the expected rewards of making different choices. Through comparison, we choose the scheme with the largest reward to complete the corresponding action. After the deployment action is completed, we get the actual users' demand and failure rate of the SFC at the current moment, which will act as feedback to update our estimated expectations for the next round of selection. Through this iteration of the continuous learning, prediction, feedback and update process, we can make the best choice at the current moment and get the biggest benefit.

C. Performance Evaluation of RTSD

In this section, we focus on investigating the impacts of different parameters on the performance of the proposed *RTSD* algorithm. This paper explores from both the perspectives of users and edge environment. On the user side, we observe the impact of end users' demands by changing the number of users in the simulation experiment; on the edge side, we observe how the edge environment affects the performance of *RTSD* algorithms by changing the resource capacity of edge servers.

First, we observe the impact of end users' needs on our solution by changing the number of users in the simulation experiment. We set the number of users sending service



(a) Hit reward (b) Remaining resource (c) The number of SFC backups

Fig. 5. The effect of changing edge resources on *RTSD*.

requests as $10, 20$, and 30, and then we get the corresponding changes in the service hit reward, remaining resources, and the number of deployed SFCs, which are shown in Fig. 4. It can be seen that as the number of users increases, the request information we can refer to becomes more sufficient, which allows us to make full use of the limited resources at the edge and further obtain higher service hit reward by deploying more suitable SFCs at the edge. But the number of SFCs that can be deployed on edge servers will not change significantly due to the limitation of the available resources as we can see in Fig. 4(c).

Subsequently, we observe the impact of changes in the edge environment on our *RTSD* algorithm. By reducing the available resources of the edge server by 50% and increasing it by 50%, Fig. 5 illustrates that when the servers' available resources increase, we can deploy more backups at the edge, which will bring less waste of resources and more service reward.

VI. Conclusion and Discussion

In this paper, we solve the problem of deploying SFC backups at the edge. To cope with the shortcomings of current research, we conduct edge backup of SFCs to provide popular services with the lowest latency considering from the perspectives of both the end users and edge system. To that aim, we propose a *Real-Time Selection and Deployment* (*RTSD*) algorithm. We first take advantage of the online bandit learning method to deal with the uncertainty of SFC popularity and failure rate dynamically changing with time. Next, inspired by the *Prim* method and combined it with the greedy strategy, we find the optimal deployment plan with the minimum latency for a specific SFC. Backup hit rewards are calculated by integrating the results of the two steps, based on which we can select popular SFC backups and properly place the corresponding VNFs contained in them on the edge network. The deployment result of the current time slot will be used as feedback information to optimize the subsequent SFC backup operation. Results of comparative experiments demonstrate the superiority of our proposed schemes.

Although we have studied and solved the optimization problem of deploying SFC backups in edge environment with the support of *UCB algorithm, Prim heuristic algorithm*, and *greedy algorithm*, there still exist several open problems that can be further addressed in implementing the proposed solution in real-world systems:

1) Servers can also fail. Our study mainly considers the SFC failure with the assumption that all edge servers can successfully operate to backup appropriate VNFs. However, servers can also fail in real situations, which can make the SFC backup problem more challenging.
2) Service providers have special preferences or requirements. Our proposed *RTSD* scheme selects suitable SFCs and back them up at the edge for a high hit reward, which is based on the assumption that all SFCs would follow the selection results and deployment suggestions. However, there may also be cases where the network service providers have special needs to place some specific SFCs at the edge instead of the cloud, resulting in invalidity of directly running the *RTSD* solution.

References

[1] Y. Mao, C. You, J. Zhang, K. Huang, and K. B. Letaief, "A survey on mobile edge computing: The communication perspective," *IEEE Commun. Surveys Tuts.*, vol. 19, no. 4, pp. 2322–2358, 4th Quart., 2017.
[2] Y. Xiao, Y. Jia, C. Liu, X. Cheng, J. Yu, and W. Lv, "Edge computing security: State of the art and challenges," *Proc. IEEE*, vol. 107, no. 8, pp. 1608–1631, Aug. 2019.
[3] J. Erfanian and B. Smith, "Network functions virtualisation-white paper on NFV priorities for 5G," ETSI White Paper, pp. 1–15, 2017.
[4] S. Van Rossem, W. Tavernier, D. Colle, M. Pickavet, and P. Demeester, "VNF performance modelling: From stand-alone to chained topologies," *Comput. Netw.*, vol. 181, Nov. 2020, Art. no. 107428.
[5] M. Peuster et al., "Introducing automated verification and validation for virtualized network functions and services," *IEEE Commun. Mag.*, vol. 57, no. 5, pp. 96–102, May 2019.
[6] M. Chiosi et al., "Network functions virtualisation-white paper#3," ETSI, Tech. Rep., 2014.
[7] X. Fei, F. Liu, H. Xu, and H. Jin, "Adaptive VNF scaling and flow routing with proactive demand prediction," in *Proc. IEEE INFOCOM Conf. Comput. Commun.*, Apr. 2018, pp. 486–494.
[8] J. Zhang, Z. Wang, C. Peng, L. Zhang, T. Huang, and Y. Liu, "RABA: Resource-aware backup allocation for a chain of virtual network functions," in *Proc. IEEE INFOCOM Conf. Comput. Commun.*, Apr. 2019, pp. 1918–1926.
[9] T. Taleb, A. Ksentini, and B. Sericola, "On service resilience in cloud-native 5G mobile systems," *IEEE J. Sel. Areas Commun.*, vol. 34, no. 3, pp. 483–496, Mar. 2016.
[10] X. Shang, Z. Li, and Y. Yang, "Partial rerouting for high-availability and low-cost service function chain," in *Proc. IEEE Global Commun. Conf. (GLOBECOM)*, Dec. 2018, pp. 1–6.
[11] X. Shang, Y. Huang, Z. Liu, and Y. Yang, "Reducing the service function chain backup cost over the edge and cloud by a self-adapting scheme," in *Proc. IEEE INFOCOM Conf. Comput. Commun.*, Jul. 2020, pp. 2096–2105.
[12] J. Fan et al., "A framework for provisioning availability of NFV in data center networks," *IEEE J. Sel. Areas Commun.*, vol. 36, no. 3, pp. 2246–2259, Oct. 2018.

[13] X. Shang, Z. Li, and Y. Yang, "Placement of highly available virtual network functions through local rerouting," in *Proc. IEEE 15th Int. Conf. Mobile Ad Hoc Sensor Syst. (MASS)*, Oct. 2018, pp. 80–88.
[14] Y. Sang, B. Ji, G. R. Gupta, X. Du, and L. Ye, "Provably efficient algorithms for joint placement and allocation of virtual network functions," in *Proc. IEEE INFOCOM Conf. Comput. Commun.*, May 2017, pp. 1–9.
[15] T. W. Kuo, B. H. Liou, K. C. Lin, and M. J. Tsai, "Deploying chains of virtual network functions: On the relation between link and server usage," *IEEE/ACM Trans. Netw.*, vol. 26, no. 4, pp. 1562–1576, Aug. 2018.
[16] R. Mijumbi, S. Hasija, S. Davy, A. Davy, B. Jennings, and R. Boutaba, "A connectionist approach to dynamic resource management for virtualised network functions," in *Proc. 12th Int. Conf. Netw. Service Manag. (CNSM)*, Oct. 2016, pp. 1–9.
[17] R. Cziva, C. Anagnostopoulos, and D. P. Pezaros, "Dynamic, latency-optimal VNF placement at the network edge," in *Proc. IEEE INFOCOM Conf. Comput. Commun.*, Apr. 2018, pp. 693–701.
[18] P. Jin, X. Fei, Q. Zhang, F. Liu, and B. Li, "Latency-aware VNF chain deployment with efficient resource reuse at network edge," in *Proc. IEEE INFOCOM Conf. Comput. Commun.*, Jul. 2020, pp. 267–276.
[19] H. Zhu and C. Huang, "EdgePlace: Availability-aware placement for chained mobile edge applications," *Trans. Emerg. Telecommun. Technol.*, vol. 29, no. 11, p. e3504, Nov. 2018.
[20] J. Fan, C. Guan, Y. Zhao, and C. Qiao, "Availability-aware mapping of service function chains," in *Proc. IEEE INFOCOM Conf. Comput. Commun.*, May 2017, pp. 1–9.
[21] Y. Kanizo, O. Rottenstreich, I. Segall, and J. Yallouz, "Optimizing virtual backup allocation for middleboxes," *IEEE/ACM Trans. Netw.*, vol. 25, no. 5, pp. 2759–2772, Oct. 2017.
[22] N. T. Dinh and Y. Kim, "An efficient availability guaranteed deployment scheme for IoT service chains over fog-core cloud networks," *Sensors*, vol. 18, no. 11, p. 3970, 2018.
[23] C. Wang, Q. Hu, D. Yu, and X. Cheng, "Proactive deployment of chain-based VNF backup at the edge using online bandit learning," in *Proc. IEEE 41st Int. Conf. Distrib. Comput. Syst. (ICDCS)*, Jul. 2021, pp. 740–750.
[24] T. Hirayama and M. Jibiki, "SFC path selection based on combination of topological analysis and demand prediction," in *Proc. 23rd Asia–Pacific Netw. Oper. Manag. Symp. (APNOMS)*, Sep. 2022, pp. 1–4.
[25] M. M. K. Jitu and M. H. Haque, "Determining best travelling salesman route of 12 cities of Europe," Tech. Rep., 2022.
[26] P. Auer, N. Cesa-Bianchi, and P. Fischer, "Finite-time analysis of the multiarmed bandit problem," *Mach. Learn.*, vol. 47, no. 2, pp. 235–256, 2002.
[27] D. P. Williamson and D. B. Shmoys, *The Design of Approximation Algorithms*. Cambridge, U.K.: Cambridge Univ. Press, 2011.

Chen Wang received the B.S. degree in computer science from the China University of Mining and Technology, Xuzhou, China, in 2010. She is currently pursuing the M.S. degree with the School of Computer Science and Technology, Shandong University. Her research interests include edge computing and reinforcement learning.

Qin Hu (Member, IEEE) received the Ph.D. degree in computer science from The George Washington University in 2019. She is currently an Assistant Professor with the Department of Computer and Information Science, Indiana University-Purdue University Indianapolis (IUPUI). Her research interests include wireless and mobile security, edge computing, blockchain, and crowdsourcing/crowdsensing.

Dongxiao Yu (Senior Member, IEEE) received the B.S. degree from the School of Mathematics, Shandong University, in 2006, and the Ph.D. degree from the Department of Computer Science, The University of Hong Kong, in 2014. He became an Associate Professor with the School of Computer Science and Technology, Huazhong University of Science and Technology, in 2016. He is currently a Professor with the School of Computer Science and Technology, Shandong University. His research interests include wireless networks, distributed computing, and data mining.

Xiuzhen Cheng (Fellow, IEEE) received the M.S. and Ph.D. degrees in computer science from the University of Minnesota Twin Cities in 2000 and 2002, respectively. She is currently a Professor with the School of Computer Science and Technology, Shandong University, China. Her current research interests include privacy-aware computing, wireless and mobile security, dynamic spectrum access, mobile handset networking systems (mobile health and safety), cognitive radio networks, and algorithm design and analysis. She has served on the editorial boards of several technical publications and the technical program committees of various professional conferences/workshops. She has also chaired several international conferences. She was the Program Director for the U.S. National Science Foundation (NSF) from April to October 2006 (full-time) and from April 2008 to May 2010 (part-time). She has published more than 300 peer-reviewed papers.