Dynamic Network Function Provisioning to Enable Network in Box for Industrial Applications

Gang Sun, *Member*, *IEEE*, Zhu Xu, Hongfang Yu, *Member*, *IEEE*, Victor Chang

***Abstract*—Network function virtualization (NFV) in 6G can use standard virtualization techniques to enable network functions via software. Resource scheduling is one of the key research areas of NFV in 6G and is mainly used to deploy service function chains (SFCs) in substrate networks. However, determining how to utilize network resources efficiently has always been a difficult problem in SFC deployment. This paper focuses on how to efficiently provision online SFC requests in NFV with 6G. We first establish a mathematical model for the problem of online SFC provisioning. Then, we propose an efficient online service function chain deployment (OSFCD) algorithm that selects the path to deploy that is close to the SFC length. Finally, we compare our proposed algorithm with three other existing algorithms by simulation experiments. The experimental results show that the OSFCD algorithm optimizes multiple performance indicators of online SFC deployment.**

***Index Terms*— Network function provisioning; Resource efficiency; Latency; Network in box; 6G**

I. Introduction

Currently, communication networks adopt the architecture of dedicated hardware and software. The dedicated equipment produces not only reliability and high performance but also some new issues. For example, resources cannot be shared, scalability is limited, and capital expenditure (CAPEX) and operating expenditure (OPEX) remain high. The proposal of network function virtualization (NFV) [1] brings new development opportunities to communication networks, including 5G and 6G. NFV can use standard virtualization technology to enable network functions via software so that they can be run on standard server virtualization software and be installed or moved to any location in the network as required without deploying new hardware equipment.

Network functions (NFs) in the traditional network are replaced by virtual network functions (VNFs), including 6G and wireless networks. When a user requests a network service from telecommunications service providers (TSPs), the network flow will pass through certain specific VNFs to reach the user. The abstract topology consisting of the TSP, specific VNFs, and the user forms the service function chain (SFC) [2]. Resource allocation is one of the key areas of NFV research. Resource allocation is mainly used to deploy SFC in the network topology and to efficiently use the network resources. The pros and cons of resource allocation will directly affect the costs of TSPs and the user experience. SFC deployment has been certified as an NP-H problem [3], which is difficult to solve in polynomial time.

Gang Sun is with Key Lab of Optical Fiber Sensing and Communications (Ministry of Education), University of Electronic Science and Technology of China, Chengdu, China; and he is also with Agile and Intelligent Computing Key Laboratory of Sichuan Province, Chengdu, China (e-mail: gangsun@uestc.edu.cn).

Zhu Xu is with Key Lab of Optical Fiber Sensing and Communications (Ministry of Education), University of Electronic Science and Technology of China, Chengdu, China (e-mail: 2215766944@qq.com).

Hongfang Yu is with Key Lab of Optical Fiber Sensing and Communications (Ministry of Education), University of Electronic Science and Technology of China, Chengdu, Chian; and is also with Peng Cheng Laboratory, Shenzhen, China (e-mail: yuhf@uestc.edu.cn)

Victor Chang is with Artificial Intelligence and Information Systems Research Group, School of Computing and Digital Technologies, Teesside University, Middlesbrough, UK (e-mail: ic.victor.chang@gmail.com).

To date, many academics have invested in SFC deployment research. Liu et al. [4] proposed a two-step algorithm G-SA for SFC deployment, first finding nodes in the network to deploy VNFs. Then, they deploy VNLs by computing the shortest path from the source node to the sink node. The authors of Ref. [5] proposed three heuristic deployment algorithms: ER, ER_CS and ER_CS_ADJ. The ER algorithm mainly considered the reliability requirements, ER_CS optimized the network load based on ER, and the ER_CS_ADJ algorithm further optimized the bandwidth resource consumption of the deployment paths.

Table 1 Performance optimization in different algorithms

Algorithm	Bandwidth	Latency	Success Ratio	Load Balance
G-SA	✓	✓	×	×
ER	×	×	×	×
ER_CS	×	×	×	✓
ER_CS_ADJ	✓	×	×	✓

Although the algorithms mentioned above have different algorithm designs for SFC deployment, neither can efficiently use the underlying network resources. In addition, the two algorithms did not consider online SFC deployment. They also centrally deployed VNFs on partial nodes, resulting in an unbalanced network load and affecting the deployment of subsequent SFCs.

However, they are not specifically designed for 6G. The 6G network can provide smart network services to users via network-as-a-service (Naas) that provisions the shared physical network resources to different users by using network slicing [6-8]. SDN, NFV and SFC orchestration work as key enablers for network slicing in 6G [9]. Therefore, the research in this work for efficient SFC provisioning can enable and drive network-in-a-box deployment for industrial applications in 6G networks.

Therefore, we propose an online service function chain deployment (OSFCD) algorithm, including 6G and wireless networks. Because the performance of the SFC deployment problem is closely related to the hops of the deployment path, the main focus of the OSFCD algorithm is to choose the path that is close to the length of SFC to deploy and efficiently use network resources. In addition, the emergence of hot nodes or links will affect the deployment of the subsequent SFCs and reduce the overall success rate. It needs to design algorithms to achieve network load balancing. The main contributions of this paper are summarized as follows:

- We establish a mathematical model for the problem of online SFC provisioning in Section III. Based on the established model, Section IV introduces an efficient deployment algorithm OSFCD. The algorithm optimizes multiple performance indicators of online SFC deployment.

- The load rate of the network is also considered to avoid hot nodes or links from appearing in the network. We prioritize the nodes or links with a smaller load rate for network load balancing in the deployment algorithm OSFCD.
- Through the simulation experiment, we compare the performance of the OSFCD algorithm with the existing approaches in Section Ⅴ.

The rest of this paper is arranged as follows. We review the related work in Section II. A mathematical model for online SFC provisioning is presented in Section Ⅲ. In Section IV, based on the established model, we introduce the OSFCD algorithm. Section Ⅴ introduces the simulation experiments and performance comparison of the OSFCD algorithm. We conclude this work in Section VI.

II. Related Work

A. Cost-efficient SFC provisioning

The deployment of SFCs has attracted the attention of many academics, who have worked to reduce the deployment costs of SFCs. Tang et al. [10] studied SFC deployment in the 5G access network. A two-stage queue-aware deployment algorithm was proposed to optimize deployment cost and improve the stability of network. Zhao et al. [11] proposed SFC deployment algorithm SFCM- FOCL, which studied the orchestration of SFC in the 5G network environment to minimize the cost of deployment. In [12], the authors studied the composition and embedding of SFC in the 5G network. Based on a greedy algorithm, a heuristic algorithm was proposed to improve the success rate and reduce the embedding cost. The authors of [13] proposed a new SFC deployment framework X-FORCE in 5G network to deploy SFC and manage the SFC life cycle. The framework improved network performance and saved network resources.

B. Low-latency SFC provisioning

End-to-end delay may affect the users' experiences. In [14], a delay-aware VNF orchestration algorithm was designed to improve the acceptance rate of SFCs by selecting VNFs with guaranteed delay. The authors of [15] studied SFC provisioning in the 5G network. A new algorithm eRESERV was proposed to improve the reliability of the 5G network under the delay constraint of SFC. In [16], a new SFC orchestration scheme was proposed to reduce the deployment path delay by optimizing the selection of VNFs and traffic control in the 5G network. The authors of [17] established an integer linear programming model for SFC orchestration. Based on the established model, a latency-aware heuristic algorithm was proposed to optimize the deployment delay of SFC requests. In [18], the authors investigated dynamic SFC orchestration under SDSN-NFV environments. They proposed a middlebox delay optimization (MDO) algorithm to reduce the transmission delay.

C. SFC deployment with machine learning

With the emergence of machine learning, an increasing number of academics have applied machine learning to SFC orchestration. Li et al. [19] utilized reinforcement learning technology to deploy SFCs with security requirements. The authors also designed a reward function to balance different optimization objectives. In [20], the authors used a partial observation Markov decision process (POMDP) to perceive the network topology. Based on POMDP, a deployment approach was designed that considered the particularity of SFC provisioning in cloud radio access networks. A deep learning model was designed to predict future virtual network function service chain (VNF-SC) requests for inter-DC networks in [21]. According to the predicted SFC requests, network resources can be predeployed. Lightpath establishment and VNF mapping can be performed accordingly. The authors of [22] combined the random cloud selection technology with the prediction model of support vector regression to improve the cost and latency of SFC provisioning.

III. Problem Statement And Modeling

A. Substrate network

We can abstract the substrate network into an undirected topology $G_P = (N_P, E_P)$, where $N_P = \{n_1, n_2, \dots, n_{|NP|}\}$ is the set of network nodes and $E_P = \{e_1, e_2, \dots, e_{|EP|}\}$ is the set of network links. $|NP|$ and $|EP|$ represent the amount of network nodes and links, respectively. A network node n_i represents a server in the network, which contains certain computing resources $a(n_i)$. We use $r(n_i)$ to denote the available computing resources. $lr(n_i)$ represents the node load rate. The calculation of $lr(n_i)$ is shown in Formula (1).

$$lr(n_i) = \frac{a(n_i) - r(n_i)}{a(n_i)} \quad \forall\, n_i \in N_P \tag{1}$$

$$e_i = \left(e_i^{n_x}, e_i^{n_y}\right) \quad \forall e_i \in E_P \tag{2}$$

For a substrate link e_i, $a(e_i)$ represents total bandwidth resources. $r(e_i)$ represents remaining bandwidth resources. The delay of link e_i is represented by $d(e_i)$. $e_i^{n_x}$ and $e_i^{n_y}$ represent the two nodes connected by link e_i. Therefore, link e_i can also be replaced by a node pair $(e_i^{n_x}, e_i^{n_y})$, as shown in Formula (2). $lr(e_i)$ represents the link load rate. The calculation of $lr(e_i)$ is shown in Formula (3).

$$lr(e_i) = \frac{a(e_i) - r(e_i)}{a(e_i)} \quad \forall\, e_i \in E_P \tag{3}$$

In addition, we use $p(n_i, n_j)$ to denote a path between nodes n_i and n_j, where $p(n_i, n_j)$ is a set that contains all substrate links on this path, which is shown in Formula (4). As shown in Formula (5), $d\big(p(n_i, n_j)\big)$ represents end-to-end delay of $p(n_i, n_j)$, which equals the sum of all link transmission delays. $h\big(p(n_i, n_j)\big)$ is used to denote the number of links on path $p(n_i, n_j)$. $b_{min}\big(p(n_i, n_j)\big)$ is used to indicate the minimum remaining bandwidth resource on path $p\big(n_i, n_j\big)$. These values are shown in Formulas (6) and (7).

$$p\big(n_i, n_j\big) = \{e_m, \dots, e_k\} \subseteq E_P \quad \forall\, n_i, n_j \in N_P \tag{4}$$

$$d\big(p(n_i, n_j)\big) = \sum_{e_k \in p(n_i, n_j)} d(e_k) \quad \forall\, n_i, n_j \in N_P \tag{5}$$

$$h\big(p(n_i, n_j)\big) = \big|p(n_i, n_j)\big| \quad \forall\, n_i, n_j \in N_P \tag{6}$$

$$b_{min}\big(p(n_i, n_j)\big) = \min_{e_k \in p(n_i, n_j)} \{r(e_k)\} \tag{7}$$

B. SFC request

We denote an SFC request as $SFC = (N_S, E_S, S, D, C_{OR})$. $N_S = \{vnf_1, vnf_2, \dots, vnf_{|NS|}\}$ represents the set of VNFs in the SFC. $|NS|$ represents the number of VNFs. The computing resources requested by vnf_i are denoted as $R(vnf_i)$. We use $N(vnf_i)$ to represent the substrate node where the VNF vnf_i is deployed. Formula (8) indicates that $Z(vnf_i, n_j)$ is a binary variable. If $Z(vnf_i, n_j) = 1$, vnf_i is deployed on n_j; otherwise, $Z(vnf_i, n_j) = 0$.

$$Z(vnf_i, n_j) \in \{0,1\}\ \forall vnf_i \in N_S, \forall n_j \in N_P \tag{8}$$

$E_S = \{vnl_1, vnl_2, \dots, vnl_{|ES|}\}$ represents the set of VNLs, and $|ES|$ represents the number of VNLs. Similarly, deploying a VNL vnl_i needs to consume bandwidth resources $R(vnl_i)$. vnl_i^{vnfx} and vnl_i^{vnfy} represent the two VNFs connected by VNL vnl_i. We use $E(vnl_i)$ to denote the path on which VNL vnl_i is deployed. $D(vnl_i)$ denotes the latency, and $B(vnl_i)$ is the bandwidth consumption for deploying the path of vnl_i. These parameters are shown in Formulas (9) to (11). Formula (12) indicates that $Y(vnl_i, e_j)$ is a binary variable. If $Y(vnl_i, e_j) = 1$, vnl_i is deployed on e_j; otherwise, $Y(vnl_i, e_j) = 0$.

$$E(vnl_i) = p\left(N\left(vnl_i^{vnfx}\right), N\left(vnl_i^{vnfy}\right)\right)\ \forall vnl_i \in E_S \tag{9}$$

$$D(vnl_i) = d\big(E(vnl_i)\big) = \textstyle\sum_{e_k \in E(vnl_i)} d(e_k)\ \forall vnl_i \in E_S \tag{10}$$

$$B(vnl_i) = R(vnl_i) * h\big(E(vnl_i)\big)\ \forall vnl_i \in E_S \tag{11}$$

$$Y(vnl_i, e_j) \in \{0,1\}\ \forall vnl_i \in E_S, \forall e_j \in E_P \tag{12}$$

The locations of TSP and the user are represented by S and D, respectively. In addition, network flows must pass through VNFs in the specified order, denoted as $C_{OR} = \{vnf_1 \xrightarrow{vnl_1} vnf_2 \xrightarrow{vnl_2} \dots \xrightarrow{vnl_{|ES|}} vnf_{|NS|}\}$. In the process of online SFC deployment, for an SFC SFC_i, we use $T(SFC_i)$ to represent the arrival time interval of SFC_i with the previous SFC, and $F(SFC_i)$ represents the service time of SFC_i. In addition, $TR(SFC_i)$ denotes the time required to respond to SFC request SFC_i.

Throughout the deployment process, we record all SFC requests in the collection L_{SFC}. For an SFC SFC_i, we use $D(SFC_i)$ to represent end-to-end delay. $B(SFC_i)$ represents the bandwidth consumption. These parameters are equivalent to the sum of transmission delay or bandwidth consumption of VNLs in SFC_i, respectively, and are shown in Formulas (13) and (14). Formula (15) indicates that $S(SFC_i)$ is a binary variable. If $S(SFC_i) = 1$, SFC_i is deployed successfully; otherwise, $S(SFC_i) = 0$.

$$D(SFC_i) = \textstyle\sum_{vnl_k \in E_S} D(vnl_k)\ \ \forall SFC_i \in L_{SFC} \tag{13}$$

$$B(SFC_i) = \textstyle\sum_{vnl_k \in E_S} B(vnl_k)\ \ \forall SFC_i \in L_{SFC} \tag{14}$$

$$S(SFC_i) \in \{0,1\}\ \ \forall SFC_i \in L_{SFC} \tag{15}$$

C. Online SFC deployment

(1) Online SFC deployment process

During the entire SFC deployment process, we use $L_{SFC} = \{SFC_1, SFC_2, \dots, SFC_{|L_{SFC}|}\}$ to record all requested SFCs. $|L_{SFC}|$ represents the amount of SFCs. $NUM_{succ}(L_{SFC})$ is used to represent the number of SFCs successfully deployed. In addition, $B_{tot}(L_{SFC})$ and $D_{tot}(L_{SFC})$ denote the total bandwidth consumption and latency, respectively. $TR_{tot}(L_{SFC})$ denotes the response time for L_{SFC}. Here, we only count the SFCs that are successfully deployed. These parameters are shown in Formulas (16) to (19).

$$NUM_{succ}(L_{SFC}) = \textstyle\sum_{SFC_K \in L_{SFC}} S(SFC_K) \tag{16}$$

$$B_{tot}(L_{SFC}) = \textstyle\sum_{SFC_K \in L_{SFC}} B(SFC_K) * S(SFC_K) \tag{17}$$

$$D_{tot}(L_{SFC}) = \textstyle\sum_{SFC_K \in L_{SFC}} D(SFC_K) * S(SFC_K) \tag{18}$$

$$TR_{tot}(L_{SFC}) = \textstyle\sum_{SFC_K \in L_{SFC}} TR(SFC_K) * S(SFC_K) \tag{19}$$

Because we are studying the problem of online SFC deployment, the dynamic arrival and departure of SFCs will be considered. We model the dynamic arrival and departure of an SFC as two Poisson processes. Therefore, the arrival time interval and service time of the SFC are independently and identically distributed and obey an exponential distribution. These processes are shown in Formulas (20) and (21), where u and v are both random numbers between 0 and 1. In addition, λ is the arrival rate. μ is the service rate.

$$T(SFC_{i+1}) = T(SFC_i) - \frac{log^u}{\lambda}\ \ u \in (0,1) \tag{20}$$

$$F(SFC_{i+1}) = F(SFC_i) - \frac{log^v}{\mu}\ \ v \in (0,1) \tag{21}$$

(2) Network resource constraints

For the VNF vnf_i and the substrate node $N(vnf_i)$, the remaining resources of $N(vnf_i)$ are required to exceed the computing resources requested by vnf_i. This requirement is shown in Formula (22). For the substrate node n_j, the consumed computing resources are required to be less than all computing resources of node n_j. This is shown in Formula (23). Formulas (24) and (25) indicate that during an SFC deployment, each VNF and substrate node are mapped one-to-one. This is to simplify the deployment schemes and to prevent the load from being concentrated on a part of the nodes and for better load balancing.

$$r\big(N(vnf_i)\big) \geq R(vnf_i)\ \ \forall vnf_i \in N_S \tag{22}$$

$$\textstyle\sum_{SFC_k \in L_{SFC}} \sum_{vnf_i \in N_S} Z(vnf_i, n_j) \times R(vnf_i) \leq a(n_j)\ \ \forall n_j \in N_P \tag{23}$$

$$0 \leq \textstyle\sum_{n_j \in N_P} Z(vnf_i, n_j) \leq 1\ \ \forall\, vnf_i \in N_S \tag{24}$$

$$0 \leq \textstyle\sum_{vnf_i \in N_S} Z(vnf_i, n_j) \leq 1\ \ \forall\, n_j \in N_P \tag{25}$$

For the VNL vnl_i and the substrate path $E(vnl_i)$, the remaining bandwidth resources of links on the path $E(vnl_i)$ are required to be greater than the bandwidth demand of VNL vnl_i. This is shown in Formula (26). For the substrate link e_j, the consumed bandwidth resources are required to be less than all bandwidth resources of link e_j. It is shown in Formula (27). In contrast to the deployment of VNFs, as shown in Formula (28), the deployment of a VNL requires only that one network link carries only one VNL during the deployment of an SFC. This is because one VNL can map into multiple substrate links.

$$b_{min}\big(E(vnl_i)\big) \geq R(vnl_i)\ \ \forall vnl_i \in E_S \tag{26}$$

$$\textstyle\sum_{SFC_k \in L_{SFC}} \sum_{vnl_i \in E_S} Y(vnl_i, e_j) \times R(vnl_i) \leq a(e_j)\ \ \forall e_j \in E_P \tag{27}$$

$$0 \leq \textstyle\sum_{vnl_i \in E_S} Y(vnl_i, e_j) \leq 1\ \ \forall\, e_j \in E_P \tag{28}$$

(3) Online SFC provisioning example

To better illustrate the online SFC provisioning, we give an example in Figures 1 to 3. These three figures represent the deployment of SFCs at three different moments. Each figure contains examples of SFC requests and the network topology. In the figures, $(i, r(n_i))$ is used to represent an attribute of the node. i and $r(n_i)$ represent the node ID and the remaining computing resources, respectively. Similarly, $(i, r(e_i), d(e_i))$ is used to represent an attribute of the link, and i is the link ID. $r(e_i)$ and $d(e_i)$ represent the remaining bandwidth resources and the delay of the link, respectively. Here, for simplicity, we assume that the total resources of each node and each link are 50 units.

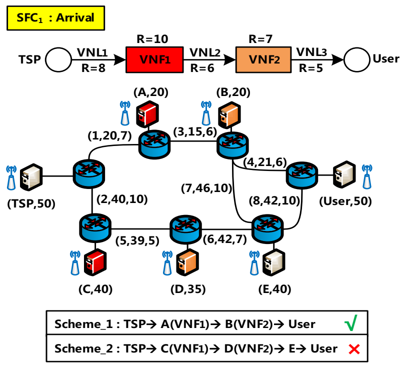

Fig. 1. Online SFC deployment at time T1.

In Figure 1, we show the deployment of an SFC at time T1. SFC_1 has just arrived. Here, we simply give two deployment schemes, $scheme\_1$ and $scheme\_2$. Because the network resources are sufficient relative to SFC_1, both schemes can successfully deploy SFC_1. However, both the bandwidth consumption and latency of $scheme\_1$ are 19, while these two indicators of $scheme\_2$ are 24 and 32, respectively. Therefore, we select $scheme\_1$ to deploy SFC_1. This example simply indicates that the performance indicators of SFC deployment are relevant to the hops of deployment paths.

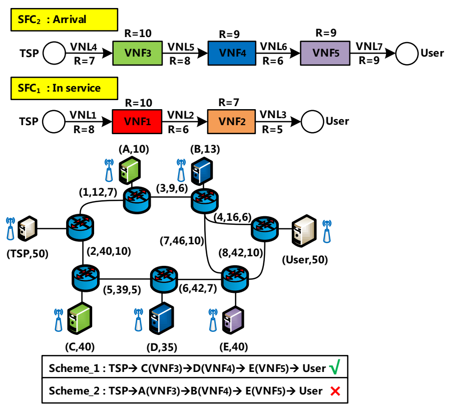

Fig. 2. Online SFC deployment at time T2.

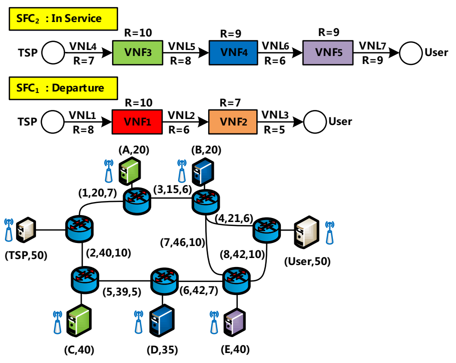

Fig. 3. Online SFC deployment at time T3.

Figure 2 shows that at time T2, SFC_2 has arrived, and SFC_1 is still in service. Here, we still give two schemes, $scheme\_1$ and $scheme\_2$. Similarly, SFC_2 can be deployed in both schemes. However, it is easy to calculate that the load rate of the nodes and links in $scheme\_2$ has already been high, while the resources in $scheme\_1$ are still abundant. Therefore, to achieve load balancing, we select $scheme\_1$ to deploy SFC_2.

Figure 3 shows that at time T3, SFC_1 completes the requested service and leaves, and the SFC_2 request is still in service. We need to return the network resources consumed by SFC_1 to the network topology. These are reflected in the remaining resources of the nodes and links in the figure.

IV. Algorithm Design

In the previous section, we have described the online SFC provisioning problem and established a mathematical model for it. To efficiently solve the researched problem, use the network resources, and optimize network load balancing, we propose the OSFCD algorithm within wired and wireless networks, including 6G.

A. Online SFC deployment

The OSFCD algorithm requires the network topology G_P and SFC request as inputs and outputs a deployment scheme for this SFC. Here, the algorithm to find the minimum k (FMK) and shortest path deployment (SPD) algorithm are described in detail in *Algorithm 2* and *Algorithm 3*, respectively. We first use the FMK algorithm to compute the length of path k between the TSP and user that is greater than or equal to the length of the SFC in the network topology. Here, we use the number of links to measure the lengths of SFC and the deployment path.

***Algorithm* 1:** Online SFC deployment (OSFCD) algorithm

```
Input: (1) Network topology $G_P = (N_P, E_P)$.
       (2) SFC request $SFC = (N_S, E_S, S, D, C_{OR})$.
Output: The deployment scheme for $SFC$.
1: $k = FMK(|ES|, S, D)$;
2: if $k < |ES|$, do
3:     End Algorithm 1.
4: end if
5: if $k > |ES|$, do
6:     Expand $SFC$ so that $|ES|$ is equal to $k$;
7: end if
8: $SPD(SFC, 0, S)$.
```

If k is less than $|ES|$, we will directly abandon the deployment of this SFC. This is because in the network, there is no path with a length greater than or equal to $|ES|$ between the TSP and user, or the existing path is too long, which will cause abundant resource consumption. If k is greater than $|ES|$, it means that there is a path in the network that can map this SFC but there are extra nodes and links in the path. Therefore, we need to simply extend the SFC. To help to understand the SFC expansion, an example is shown in Figure 4.

To make the SFC length equal to k, several VNFs and VNLs need to be added. The difference is that adding VNLs requires additional bandwidth resources, while adding VNFs does not. Therefore, to consume less bandwidth resources, we choose the VNL with the smallest $R(vnl_i)$ for expansion. As shown in Figure 4 (here, we assume $k = 5$), we need to expand two VNLs. VNL_3 is the VNL with the minimum bandwidth resource request, so we add VNL_4 and VNL_5, and their bandwidth resource requests are the same as that of VNL_3. In addition, we have added two VNFs, VNF_3 and VNF_4. Their computing resource requests are all zero.

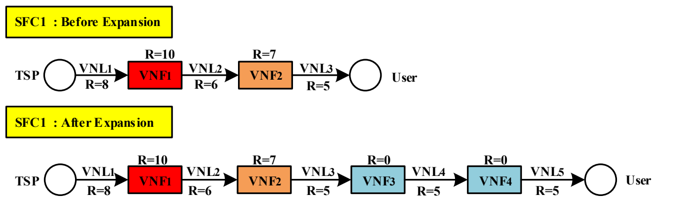

Fig. 4. Example of SFC expansion.

After completing the above work, we can ensure that $|ES|$ is equal to k. Next, we use SPD algorithm to deploy the SFC. The details of SPD algorithm are introduced in subsection C later in Section IV.

B. Find minimum k

For a network topology, we use A to represent its adjacency matrix. We can obtain the number of paths whose length is equal to n from the matrix A^n between any two nodes in the topology. Here, we just need to know whether there is a path of length k between two nodes.

***Algorithm 2*: Find minimum k (FMK)**

```
Input: (1) Adjacency matrix of network topology $A$
           and $A^1 \rightarrow A^{15}$;
       (2) The # of VNLs: $|ES|$;
       (3) The locations of TSP $S$ and user $D$.
Output: Length of path $k$.
1: for $i = |ES|$ to 15, do
2:   if $A^i[S][D] \neq 0$, do
3:     return $i$.
4:   end if
5: end for
6: return $-1$.
```

The FMK algorithm needs to obtain the adjacency matrix A of the network topology and obtain the power of A. Since the length of the SFC is generally less than 10, we have prepared the matrices A^1 to A^{15} here. In addition, *Algorithm 2* also needs the number of VNLs in the SFC as well as the locations of TSP and the user as input. The FMK algorithm outputs the length of path k between the TSP and user that is greater than or equal to $|ES|$. We traverse from $|ES|$ to 15 until $A^i[S][D]$ is not equal to zero and i is returned. If $A^i[S][D] \neq 0$ is not found during the process of traversal, we return -1 as the flag that there is no suitable path in the topology to deploy the SFC. This satisfies the condition of $k < |ES|$ in *Algorithm 1* because the length of all SFCs is greater than zero.

C. Shortest path deployment

After the preprocessing of *Algorithm 2*, the SPD algorithm is responsible for formally finding the path mapping the SFC in the network topology. The SPD algorithm deploys every VNF iteratively. In the process of deploying a VNF, the corresponding VNL is also deployed. Therefore, the SPD algorithm is not a two-step algorithm with nodes and links deployed separately. For the schemes of SFC deployment, the substrate node to host vnf_i is stored in $N(vnf_i)$, and the path on which the VNL request vnl_i is deployed is stored in $E(vnl_i)$. We can judge whether the SFC is deployed successfully or not and calculate the performance indicators.

Algorithm 3 inputs an SFC request SFC, a count variable $count$ and the substrate node n_p that maps the previous VNF. The count variable $count$ can be regarded as the number of VNFs that have been deployed. Since the source node of the SFC is known, we start the SFC deployment from the first VNF vnf_1. Therefore, in *Algorithm 1*, we initialize $count$ to 0 and n_p to S. Finally, *Algorithm 3* outputs the deployment scheme of the SFC request.

$$sf(n_a) = \alpha * lr(n_a) + (1-\alpha) * lr\left((n_p, n_a)\right) \quad (29)$$

Lines 1 to 4 of *Algorithm 3* indicate that if $count$ is equal to $|NS|+1$, then the deployment of the SFC request has been completed, and *Algorithm 3* will end. Otherwise, we traverse the adjacency list of node n_p to find the substrate node to deploy the next VNF. Before traversal, we need to sort the nodes in the adjacent linked list in ascending order according to sf. As shown in Formula (29), sf is a weighted addition of the load rate of candidate nodes and links. Here, α is a weighting factor between 0 and 1 and determines the effect of the node load rate and link load rate on sf. We can adjust the value of α according to actual needs. In the following experiments, we consider the node load rate and link load rate to be equally important.

***Algorithm 3*: Shortest path deployment (SPD)**

```
Input: (1) SFC $SFC = (N_S, E_S, S, D, C_{OR})$;
       (2) A count variable: $count$;
       (3) The substrate node that maps the previous VNF: $n_p$.
Output: The deployment scheme for $SFC$
1: if $count = |NS| + 1$, do
2:   $SFC$ is deployed successfully;
3:   End Algorithm 3.
4: end if
5: Sort all nodes in the adjacent linked list of node $n_p$ according
   to $sf$;
6: for each node $n_a$ in the adjacent linked list of node $n_p$, do
7:   if $n_a$ has been visited, do
8:     continue;
9:   end if
10:  if $n_a = D$, do
11:    if $count \neq |NS|$, do
12:      continue;
13:    else
14:      if $r\left((n_p, n_a)\right) \geq R(vnl_{|ES|})$, do
15:        $E(vnl_{count+1}) = \{(n_p, n_a)\}$;
16:        $SFC$ is deployed successfully;
17:        End Algorithm 3.
18:      end if
19:    end if
20:  else
21:    if $count = |NS|$, do
22:      continue;
23:    else
24:      if $r(n_a) \geq R(vnf_{count+1})$ &&
               $r\left((n_p, n_a)\right) \geq R(vnl_{count})$, do
25:        $N(vnf_{count+1}) = n_a$;
26:        $E(vnl_{count+1}) = \{(n_p, n_a)\}$;
27:        set $n_a$ as visited;
```

```
28:            $SPD\ (SFC, count+1, n_a)$;
29:          end if
30:      end if
31:   end if
32: end for
33: set $n_p$ as not visited.
```

When we obtain an adjacent node n_a of n_p, if node n_a has already been visited, we will skip this node and continue to visit the next node in the adjacent linked list of n_p. Next, we discuss two different cases in terms of node n_a. When n_a is the destination node D, if all VNFs have not been deployed (i.e., $count \neq |NS|$), we will skip the destination node and select the next node to map the current VNF. Otherwise, we will examine whether the link between n_a and n_p satisfies the constraint condition proposed by Formula (26). If this condition is satisfied, we will deploy the last VNL on the substrate link (n_p, n_a). Then, we can announce the successful deployment of SFC and end *Algorithm 3*.

When node n_a is not the destination node D, if all VNFs have been deployed (i.e., $count = |NS|$), this means that the node we need to find is the destination node D. Therefore, we will skip this node and look for the destination node D. If there are still VNFs that have not yet been deployed, we will examine whether the current node n_a and connected link (n_p, n_a) meet the resource constraints proposed by Formulas (22) and (26). If these constraints are met, we record the deployment scheme in $N(vnf_i)$ and $E(vnl_i)$. In addition, we set node n_a to already visited and call the SPD algorithm to deploy the next VNF. At the end of *Algorithm 3*, we set node n_p to be not visited.

D. Complexity analysis

The OSFCD algorithm is composed of the FMK algorithm and SPD algorithm. We assume that the network topology contains n nodes. The time complexity of our proposed OSFCD algorithm is analyzed as follows:

- In the FMK algorithm, we traverse from $|ES|$ to 15 until $A^i[S][D]$ is not equal to zero. Therefore, the complexity of the FMK algorithm is a constant expression, which is recorded as $O(L)$. In the process of SFC expansion, only a few virtual nodes and links are extended in SFC, so the complexity of SFC expansion can also be recorded as $O(L)$. Where L denotes the length of SFC.
- The SPD algorithm traverses the adjacent nodes of the current node in each iteration as it looks for nodes to deploy the current VNF. After traversing all adjacent nodes, if no suitable node is found, the algorithm will backtrack. Therefore, the complexity of the OSFCD algorithm is $O(n^2)$.

In summary, the time complexity of the proposed OSFCD algorithm is $O(n^2)$.

V. Simulation Results and Analysis

In this section, we carry out simulation experiments to compare our proposed OSFCD algorithm and the other three existing algorithms proposed in [5].

A. Simulation settings

We use Java to evaluate different deployment algorithms. Similar to Ref. [5], we employ the Waxman 2 model of the GT-ITM tool to randomly generate small-scale and large-scale network instances as substrate networks to prove the applicability of the OSFCD algorithm in different situations. Here, the small and large substrate networks contain 50 nodes and 200 nodes, respectively.

Similar to Ref. [11], each node or link contains 1500 units of resources. For a substrate link e_i, $d(e_i)$ obeys a uniform distribution, U (10, 20). For two different topologies, the arrival rate λ is set to 0.04. For the large topology, the service rate μ is set to 5×10^{-5}, while for the small topology, the service rate μ is 2×10^{-4}. For a VNF request vnf_i, $R(vnf_i)$ follows the uniform distribution U (10, 20). For a VNL request vnl_i, $R(vnl_i)$ is uniformly allocated, U (20, 40).

B. Optimization goals

In this paper, for online SFC deployment, our main concerns are the following performance indicators:

(1) *Success ratio*

The success ratio of SFC deployment is defined as follows. $NUM_{succ}(L_{SFC})$ is the number of SFCs successfully deployed. $|L_{SFC}|$ is the number of all SFCs.

$$Ratio_{succ} = \frac{NUM_{succ}(L_{SFC})}{|L_{SFC}|} \tag{30}$$

(2) *Average bandwidth resource consumption*

The definition is given in Formula (31). $B_{tot}(L_{SFC})$ is the total bandwidth resource consumption.

$$B_{average} = \frac{B_{tot}(L_{SFC})}{NUM_{succ}(L_{SFC})} \tag{31}$$

(3) *Average end-to-end delay*

The definition is given in Formula (32). $D_{tot}(L_{SFC})$ is the total end-to-end delay.

$$D_{average} = \frac{D_{tot}(L_{SFC})}{NUM_{succ}(L_{SFC})} \tag{32}$$

(4) *Average response time*

The average response time is defined as follows. $TR_{tot}(L_{SFC})$ is the total response time.

$$TR_{average} = \frac{TR_{tot}(L_{SFC})}{NUM_{succ}(L_{SFC})} \tag{33}$$

(5) *Maximum node load rate*

The definition is given in Formula (34). NR_{max} is used to represent the maximum node load rate.

$$NR_{max} = \max_{n_k \in N_P}\{lr(n_k)\} \tag{34}$$

(6) *Maximum link load rate*

The maximum link load rate can be expressed by Formula (35). ER_{max} is used to represent the maximum link load rate.

$$ER_{max} = \max_{e_k \in E_P}\{lr(e_k)\} \tag{35}$$

C. Experimental results and analysis

(1) *Experimental results in two different topologies*

In this section, we introduce and analyze a performance comparison between our proposed algorithm and the comparison algorithms in two different topologies.

Figure 5 shows the success ratio of SFCs s in two different topologies. Whether in the small or large topology, with sufficient network resources, the OSFCD algorithm maintains a success rate of more than 99%, and only a few SFC requests fail to be deployed. However, in the comparison algorithms, as the length of SFC increases, the success ratio continues to decrease. In both topologies, the OSFCD algorithm optimizes the success ratio by an average of 25%.

Figure 6 shows the average bandwidth resource consumption in two different topologies. The two figures show that with the increase in the length of SFC, the average bandwidth resource consumption also increases. The growth of the four algorithms is close to the average bandwidth resource requested by one VNL. However, this indicator of the OSFCD algorithm is always smaller than that of the other three algorithms. Our algorithm optimizes the average bandwidth resource consumption by 22.7% and 16% in the small and large topologies, respectively.

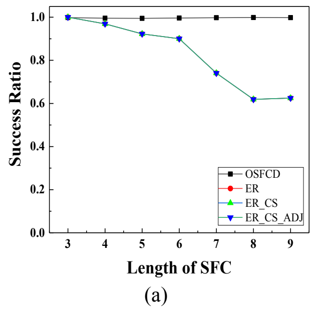
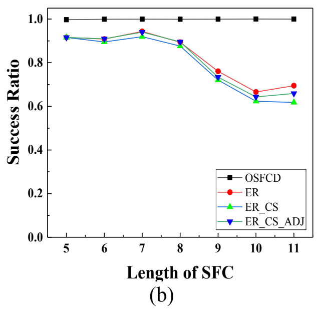
Fig. 5. The success ratio of SFCs in two topologies (a) The small-scale topology; (b) The large-scale topology.

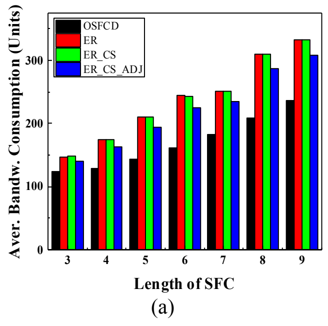
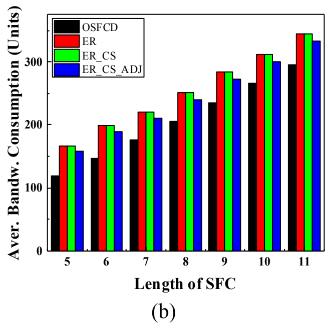
Fig. 6. Average bandwidth consumption of SFCs in two topologies (a) The small-scale topology; (b) The large-scale topology.

As shown in Figure 7, with an increase in the length of SFC, this indicator also increases. The growth rate of the four algorithms is nearly the average delay of a substrate link. The indicator of the OSFCD algorithm is always less than that of the other three algorithms, especially in the small-scale topology. This result occurs because the OSFCD algorithm tries to deploy SFCs on a path with shorter hops. Our algorithm optimizes the average end-to-end delay by 25.7% and 20% in the small and large topologies, respectively.

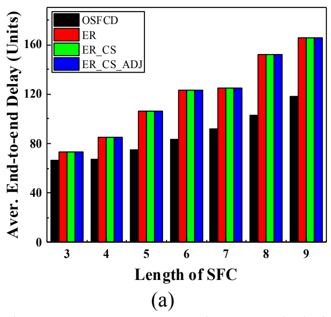
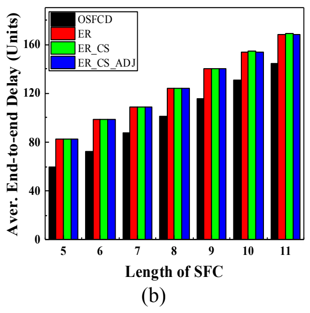
Fig. 7. Average end-to-end delay of SFCs in two topologies (a) The small-scale topology; (b) The large-scale topology.

In Figure 8, we show the maximum node load rate of SFCs in two different topologies. With an increase in the length of SFC, the maximum load rate of nodes also increases. However, the maximum node load rate and its growth rate of the OSFCD algorithm are smaller than those of the other three algorithms. As shown in Figure 8(b), the curve of the OSFCD algorithm is always under that of the other three algorithms. Finally, the maximum node load rate of comparison algorithms is nearly 1, while that of the OSFCD algorithm is always less than 0.5.

In Figure 8(a), the optimization of the OSFCD algorithm is not obvious. However, we can see the results in combination with Figure 5(a). When length of SFC is greater than 6, the success ratio of the other three algorithms is significantly reduced, while the OSFCD algorithm still maintains a high success ratio. Therefore, when the length of SFC is greater than 6, this indicator of comparison algorithms decreases because their success ratios also decrease. Nevertheless, the maximum node load rate of the OSFCD algorithm is always smaller than that of the other three algorithms.

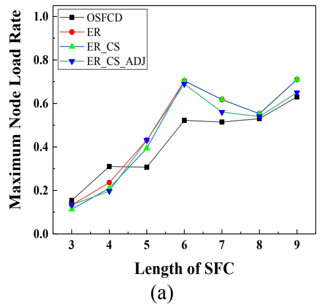
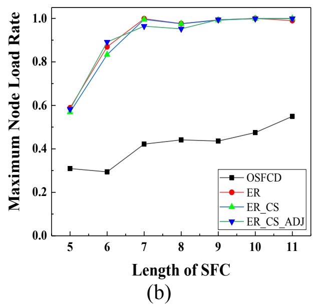
Fig. 8. Maximum node load rate of SFCs in two topologies (a) The small-scale topology; (b) The large-scale topology.

In Figure 9, we show maximum link load rate in two different topologies. As the length of SFC increases, the maximum link load rate also increases, and the curve of the OSFCD algorithm is always below the other three algorithms, especially in the large topology. Moreover, with a decrease in the success ratio, the maximum link load rate of the other three algorithms also decreases but is still greater than that of the OSFCD algorithm. This is because ER and the other algorithms concentrate the load on part of the nodes or links, resulting in the emergence of hot nodes or links. However, the OSFCD algorithm preferentially deploys VNFs or VNLs on nodes or links with low load rates.

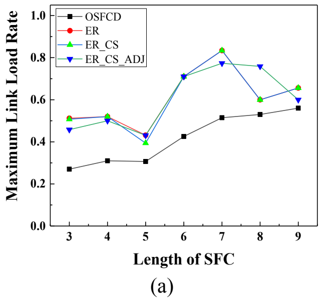
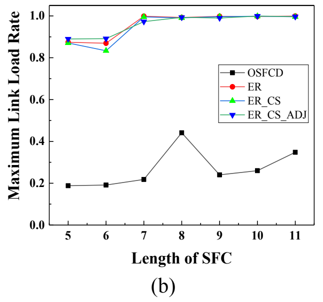
Fig. 9. Maximum link load rate of SFCs in two topologies (a) The small-scale topology; (b) The large-scale topology.

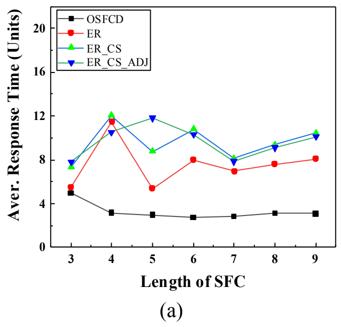
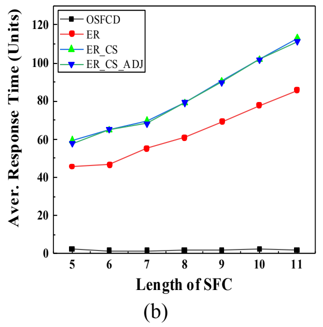
Fig. 10. Average response time of SFCs in two topologies (a) The small-scale topology; (b) The large-scale topology.

Finally, we show the average response time of SFC deployment in two different topologies in Figure 10. The average response time of the OSFCD algorithm is always smaller than that of the other three algorithms. A gap of tens or even hundreds of times is reached in large topologies. The response time of the other three algorithms in the large topology is obviously larger than that of the small topology, while the average response time of the OSFCD algorithm in the two topologies is similar. These results occur because algorithms such as ER must traverse all nodes in the topology when a VNF is deployed, while the OSFCD algorithm will end the algorithm after finding a suitable path.

(2) *Experimental results with different μ*

To examine the performance of the OSFCD algorithm in different environments, we try to change the parameters in the experiments to observe the experimental results. Here, we give the experimental results of continuously changing the service rate μ in the large topology.

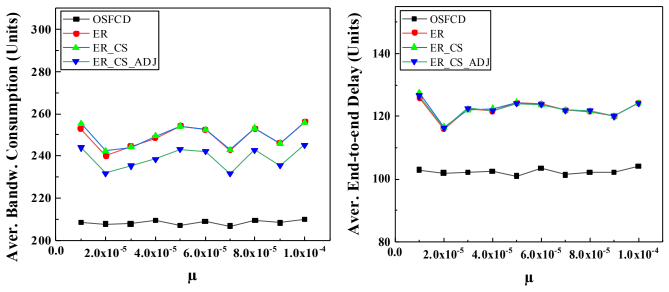

Fig. 11. Two indicators with different μ (a) Average bandwidth consumption; (b) Average end-to-end delay.

Figure 11 shows the average bandwidth resource consumption and average end-to-end delay with different μ values. With the increase of μ, both indicators of the other three algorithms fluctuate continuously, while those of the OSFCD algorithm maintain good stability. The average bandwidth resource consumption is maintained at approximately 208 and the average end-to-end delay is 104. In addition, the curve of the OSFCD algorithm is always below that of the other three algorithms. The OSFCD algorithm optimizes the average bandwidth consumption by an average of 18.23% and the average end-to-end delay by an average of 23.23%.

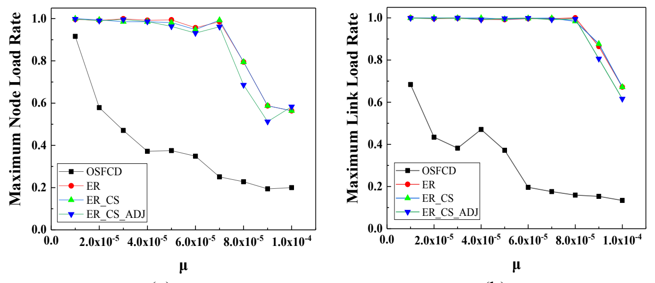

Fig. 12. Two indicators with different μ values (a) Maximum node load rate; (b) Maximum link load rate.

Figure 12 shows the maximum node load rate and maximum link load rate with different μ values, respectively. As μ increases, the both indicators of the four algorithms decrease. This is because as μ increases, the number of SFCs existing in the network topology at a certain time decreases accordingly, so the load rate will decrease. However, the curve of the OSFCD algorithm is always below that of the other three algorithms. This finding also illustrates that the OSFCD algorithm further optimizes the load balancing of the network.

Finally, in Figure 13, we show the effect of parameter α in Formula (29) on the maximum node and link load rate. When α is set as 0.2, the OSFCD algorithm tends to choose the link with lower load rate. On the other hand, when α is 0.8, the nodes with lower load rate will be preferred for VNF deployment.

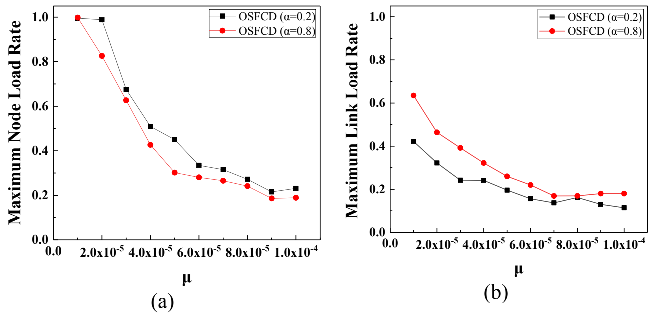

Fig. 13. Two indicators with different μ and α (a) Maximum node load rate; (b) Maximum link load rate.

VI. Conclusion

This paper studies the online SFC deployment problem in NFV for modern networks that include 6G. In the processes of the dynamic arrival and departure of SFCs, we are committed to the efficient use of network resources. We review the state-of-the-art and propose a mathematical model for online SFC provisioning. Based on the proposed model, an efficient online SFC provisioning approach, OSFCD, is proposed. The algorithm chooses a path of minimum hops to deploy the SFC. In addition, we carry out a simulation experiment and a performance comparative analysis with three other existing algorithms. The experimental results show that the OSFCD algorithm can optimize multiple performance indicators of online SFC deployment. Specifically, we reduce the bandwidth consumption and end-to-end delay by 22.7% and 25.7%, respectively.

In this paper, we do not take into account the reliability of SFC deployment in the 6G networks. In the future, we will try to utilize emerging machine learning technology to optimize SFC provisioning by considering reliability problems and design corresponding algorithms to further efficiently utilize network resources in 6G.

Acknowledgement

This research was partially supported by the National Key Research and Development Program of China (2019YFB1802800), and PCL Future Greater-Bay Area Network Facilities for Large-scale Experiments and Applications (PCL2018KP001).

References

[1] Jia Y, Wu C, Li Z, et al. Online scaling of NFV service chains across geo-distributed datacenters. IEEE/ACM Transactions on Networking, 2018, 26(2): 699-710.

[2] Sun G, Zhu G, Liao D, et al. Cost-efficient service function chain orchestration for low-latency applications in NFV networks. IEEE Systems Journal, 2019, 13(4): 3877-3888.

[3] Zheng D, Peng C, Guler E, et al. Hybrid Service Chain Deployment in Networks with Unique Function. IEEE International Conference on Communications, 2019: 1-6.

[4] Liu J, Li Y, Zhang Y, et al. Improve service chaining performance with optimized middlebox placement. IEEE Transactions on Services Computing, 2015, 10(4): 560-573.

[5] Sun J, Zhu G, Sun G, et al. A reliability-aware approach for resource efficient virtual network function deployment. IEEE Access, 2018, 6: 18238-18250.

[6] Yang P, Xiao Y, Xiao M, et al. 6G wireless communications: Vision and potential techniques. IEEE Network, 2019, 33(4): 70-75.

[7] Tariq F, Khandaker M R A, Wong K K, et al. A speculative study on 6G. IEEE Wireless Communications, 2020, 27(4): 118-125.

[8] Huang C, Hu S, Alexandropoulos G C, et al. Holographic MIMO surfaces for 6G wireless networks: Opportunities, challenges, and trends. IEEE Wireless Communications, 2020.

[9] Khan L U, Yaqoob I, Imran M, et al. 6G Wireless Systems: A Vision, Architectural Elements, and Future Directions. IEEE Access, 2020, 8: 147029-147044.

[10] Tang L, Yang H, Ma R, et al. Queue-aware dynamic placement of virtual network functions in 5G access network. IEEE Access, 2018, 6: 44291-44305.

[11] Zhao D, Ren J, Lin R, et al. On Orchestrating Service Function Chains in 5G Mobile Network. IEEE Access, 2019, 7: 39402-39416.

[12] Spinnewyn B, Isolani P H, Donato C, et al. Coordinated service composition and embedding of 5G location-constrained network functions. IEEE Transactions on Network and Service Management, 2018, 15(4): 1488-1502.

[13] Medhat A M, Carella G A, Pauls M, et al. Extensible framework for elastic orchestration of service function chains in 5g networks. IEEE Conference on Network Function Virtualization and Software Defined Networks (NFV-SDN), 2017: 327-333.

[14] Sun G, Xu Z, Yu H, et al. Low-latency and Resource-efficient Service Function Chaining Orchestration in Network Function Virtualization. IEEE Internet of Things Journal, 2020, 7(7): 5760-5772.

[15] Thiruvasagam P K, Kotagi V J, Murthy C S R. The More the Merrier: Enhancing Reliability of 5G Communication Services with Guaranteed Delay. IEEE Networking Letters, 2019, 1(2): 52-55.

[16] Cheng Y, Yang L, Zhu H. Deployment of service function chain for NFV-enabled network with delay constraint. IEEE International Conference on Electronics Technology, 2018: 383-386.

[17] Sun G, Zhou R, Sun J, et al. Energy-Efficient Provisioning for Service Function Chains to Support Delay-Sensitive Applications in Network Function Virtualization. IEEE Internet of Things Journal, 2020, 7(7): 6116-6131.

[18] Ouyang C, Wei Y, Leng S, et al. Service chain performance optimization based on middlebox deployment. IEEE International Conference on Communication Technology, 2017: 1947-1952.

[19] Li G, Zhou H, Feng B, et al. Automatic Selection of Security Service Function Chaining Using Reinforcement Learning. IEEE Globecom Workshops, 2018: 1-6.

[20] Yang Y, Chen Q, Zhao G, et al. The Stochastic-Learning-Based Deployment Scheme for Service Function Chain in Access Network. IEEE Access, 2018, 6: 52406-52420.

[21] Li B, Lu W, Liu S, et al. Deep-learning-assisted network orchestration for on-demand and cost-effective vNF service chaining in inter-DC elastic optical networks. IEEE/OSA Journal of Optical Communications and Networking, 2018, 10(10): D29-D41.

[22] Gupta L, Samaka M, Jain R, et al. COLAP: A predictive framework for service function chain placement in a multi-cloud environment. IEEE 7th Annual Computing and Communication Workshop and Conference, 2017: 1-9.