Copyright ⓒ 2022 KSII

Traffic Forecast Assisted Adaptive VNF Dynamic Scaling

Hang Qiu[1], Hongbo Tang[1*], Yu Zhao[1], Wei You[1], and Xinsheng Ji[1,2]
[1] Institute of Information Technology, Information Engineering University
Zhengzhou, 450002 China
[e-mail : hangsoon@foxmail.com, tahobo@sina.com]
[2] Purple Mountain Laboratories
Nanjing, 211111, China
[e-mail : jixs@pmlabs.com.cn]
*Corresponding author: Hongbo Tang

Received May 13, 2022; revised August 31, 2022; accepted October 23, 2022; published November 30, 2022

Abstract

NFV realizes flexible and rapid software deployment and management of network functions in the cloud network, and provides network services in the form of chained virtual network functions (VNFs). However, using VNFs to provide quality guaranteed services is still a challenge because of the inherent difficulty in intelligently scaling VNFs to handle traffic fluctuations. Most existing works scale VNFs with fixed-capacity instances, that is they take instances of the same size and determine a suitable deployment location without considering the cloud network resource distribution. This paper proposes a traffic forecasted assisted proactive VNF scaling approach, and it adopts the instance capacity adaptive to the node resource. We first model the VNF scaling as integer quadratic programming and then propose a proactive adaptive VNF scaling (PAVS) approach. The approach employs an efficient traffic forecasting method based on LSTM to predict the upcoming traffic demands. With the obtained traffic demands, we design a resource-aware new VNF instance deployment algorithm to scale out under-provisioning VNFs and a redundant VNF instance management mechanism to scale in over-provisioning VNFs. Trace-driven simulation demonstrates that our proposed approach can respond to traffic fluctuation in advance and reduce the total cost significantly.

Keywords: NFV, VNF scaling, Traffic forecasting, New instance deployment, Redundant instance management.

1. Introduction

Compared with the previous generations of mobile network technologies, the fifth-generation (5G) mobile network is foreseen to realize the internet of everything such as vehicles, drones, wearables, and machinery, especially spurring the development lots of vertical industries (e.g., automatic driving, smart cities, and industrial control, etc.) [1]. The ever-increasing number of novel applications and services in the 5G network, along with the growing demand for data traffic, presses an urgent need for mobile network operators (MNOs) to promote the evolution of the network architecture.

Network Function Virtualization (NFV) plays a critical role in the transformation of the 5G network [2-4]. It decouples the network function from traditional dedicated element as hardware-independent software called virtualized network function (VNF), which provides remarkable deployment and management flexibility. Especially, NFV offers a new model representing 5G network services and applications by way of Service Function Chains (SFCs), which contain a series of VNFs interconnected in a predefined order [5,6].

In NFV, most works investigate one-stage SFC mapping and the capacity of SFC instance is fixed [7-9]. In reality, mobile traffic demand is frequently fluctuating, and MNOs have to dynamically adjust the capacity of the related VNFs efficiently [10]. For example, tidal effects, large gatherings, hot events, etc., MNOs need to adjust the capacity of the related application VNFs (e.g. UPF) in the 5G user plane to ensure the quality of service expected by users and to optimize resource usage efficiency. This is where the VNF scaling comes into play, and the VNF scaling includes vertical scaling and horizontal scaling. Vertical scaling refers to adding or reducing the processing resources of the existing VNF instances, while horizontal scaling is done by increasing or decreasing the number of the same VNF instances [11]. However, it is a serious task to determine which type of VNF scaling and how much instance capacity to perform since there are some parameters (e.g., the VNF type, and its resource requirements) to consider.

To solve this issue, the existing works [12-14] propose some reactive VNF horizontal scaling methods to achieve dynamic resource provisioning. Since spawning a new VNF instance (e.g., copying VM image, booting VM) may incur unpredictable delay, the reactive way may degrade the quality of service and even cause packet loss. In [15,16], the traffic forecasting approaches are exploited to forecast the future traffic demands to guide the dynamic VNF scaling. Furthermore, Fei et al. [17] proposed an adaptive VNF scaling and flow routing algorithm with demand prediction, where one instance with residual demand can be created after launching some instances with the maximum capacity. In summary, all the above works are to determine the capacity of new instances before they are deployed, which limits the joint optimization of instance capacity and deployment location. In a cloud network, MNO needs to provide multiple network services with different requirements, resulting in the uneven distribution of resource usage in each data center. if the VNF instance capacity is fixed before deployment, and the remaining node resources cannot meet the larger instance capacity, the fragmented node resources cannot be fully utilized. However, the adaptive VNF instance capacity can allow for fine-grained VNF scaling with node resources, which achieves higher deployment flexibility to improve resource utilization.

The main contributions of this paper can be summarized as follows:

- We propose a traffic forecasting approach based on LSTM to realize proactive VNF scaling and discuss the impact of window size on prediction accuracy.
- We model the VNF scaling as integer quadratic programming and then propose an adaptive VNF scaling approach. For under-provisioning VNFs, we design a resource-

aware new VNF instance deployment algorithm achieving best-fit capacity with node.
- We also design a redundant VNF instance management mechanism to efficiently scale in over-provisioning VNFs, including how to delete or reload idle instances.
- We conduct simulation experiments using a trace-driven approach and prove that our proposed PAVS approach achieves less total cost.

The rest of this paper is structured as follows. Section 2 briefly reviews the related works of this paper, and Section 3 presents the network model and problem formulation in detail. Then, in Section 4, we discuss how to solve the traffic forecasting problem and VNF scaling problem, and propose a redundant VNF instance management mechanism. In Section 5, we describe the resource-aware new VNF instance deployment algorithm based on matrix coded genetic algorithm. Large-scale real trace-driven simulations and performance evaluations are performed in Section 6. We conclude this paper in Section 7.

2. Related Works

2.1 Optimal deployment of SFC

With the rapid development of NFV, there have been many works on the optimal deployment of SFCs, and the SFC deployment is defined as a resource allocation problem. The study in [18] models the SFC deployment as an integer linear programming (ILP) considering the time-varying workloads, and designs a two-stage heuristic solution to solve the ILP. You et al. [19] consider load balancing as the optimization objective, and propose a new load-balancing policy termed constrained min-max placement. To improve the scalability of SFC deployment, [20] designs a heuristic algorithm with an improved multi-stage graph to effectively achieve VNF placement. The authors of [21] establish a Markov decision process model of the SFC deployment problem and propose a deep reinforcement learning (DRL) based scheme to handle the complexities of the large-scale network. Furthermore, in [22], the robust optimization model is utilized to overcome the resource demand uncertainty of SFC deployment. Li et al. [23] devise a near-optimal approximation algorithm for the robust SFC provisioning in mobile edge computing (MEC), by adopting the Markov approximation technique. The above studies all assume that the traffic demand of network services is a deterministic value or an approximate interval, and does not consider the follow-up processing when traffic fluctuates widely.

2.2 VNF Scaling

To address traffic fluctuation and take advantage of NFV, many researchers have paid a lot of attention to the topic of VNF scaling. The works [24-26] evaluate the performance and cost of horizontal and vertical scaling in cloud computing, and conclude the optimal scaling strategy for different scenarios. Wang et al. [13] investigate the maximum supportable traffic rate based on past information to obtain a feasible VNF pre-planning deployment solution, and then propose a VNF horizontal scaling method based on the ski-rental algorithm. The work presented by Zhao et al. [27] proposes a VNF scaling management mechanism based on the threshold and optimized VNF deployment algorithm, which focuses on transmission delay and resource utilization. Pei et al. [28] investigate the SFC embedding with the VNF dynamic release problem, which efficiently deploys SFC requests and optimizes the number of placed instances. However, these studies are reactive in nature. To proactively achieve VNF scaling, Tang et al. [15] adopt the ARMA model to derive the traffic estimation, and design two VNF placement algorithms to guide the dynamic VNF scaling for SFC Run-to-Complete in a Rack

and Cross Rack Pipelined. The approach in [16] adopts a Fourier-Series-based forecasting method and 3-σ principle to ensure the network has enough resources and proposes an online VNF instance deployment method. Moreover, Zhai et al. [29] propose a fine-grained dynamic VNF scaling approach, which can efficiently improve the success ratio and reduce the resource cost. The authors of [30] study the problem of a joint user association, SFC placement, and VNF scaling on the 5G core network and MEC. In general, the above works have done a good job of VNF scaling, but the way they first determine the instance capacity and then deploy them limits the flexibility of VNF deployment.

3. Problem Description and Formulation

3.1 Cloud Network and Service Function Chain

The cloud network is represented as an undirected graph $G=(N,E)$. Each substrate node $n \in N$ could consist of one or more servers that support virtualization, and its available processing resource capacity is denoted as C_n, which are used to instantiate the VNFs. Note that all substrate nodes are connected by high-speed optical fiber and the physical links are full-duplex. The physical link $e_{m,n} \in E$ connecting the substrate nodes $m,n \in N$ has available bandwidth $B_{m,n}$.

The network operators could deploy and operate a set of VNFs denoted by F (such as firewall, NAT, DPI, load balance, customized VNFs, etc.) in the NFV infrastructure (NFVI), and they deal with the arrived service requests set R in the fashion of time-slotted, that is batch processing in a fixed time interval. A service request $r_i \in R$ arriving at t_i is described as a 4-tuple $\langle s_i, d_i, \alpha_i(t), sfc_i \rangle$ and lasts Δt_i in the form of the SFC, where the $s_i \in N$ and $d_i \in N$ are the source and the destination nodes respectively, and sfc_i is the VNF sequence of the service request r_i. We use $\alpha_i(t)$ to denote the traffic demand of service requests r_i at t for $t_i \le t \le t_i + \Delta t_i$, and the traffic demand changes from time to time. For each VNF $f \in F$, c_f indicates the processing resources required by the VNF f instance to handle a unit traffic demand, which reflects the VNF diversity in processing traffic. What's more, the VNF instance can be in three states: ACTIVE, IDLE, and DELETED. ACTIVE means that the instance is working to process traffic flows, IDLE means that the instance is still on the substrate node but not working, and DELETED means that the instance will be released soon.

We assume that traffic flows belonging to different SFCs can share the same type-f VNF instances. We use $x_{f,n}$ to indicate whether there is a type-f VNF instance assigned on the substrate node $n \in N$ ($x_{f,n}=1$, at least one instance is assigned on the node) or not ($x_{f,n}=0$). At the beginning of each time slot t, the residual resource capacity of the substrate node n is denoted by $C_n(t)$ and the residual bandwidth of the physical link $e_{m,n} \in E$ is represented by $B_{m,n}(t)$.

3.2 VNF Scaling and Traffic Routing

In the initial state $t=0$, the processing capacity $C_f(0)$ of type-f VNF instances and the routing bandwidth requested by the in-service network services have been satisfied. However, because of varing traffic demands and the arrival of new service requests, the processing

capacity of VNFs and the bandwidth requirements may not able to cover the demands in successive time slots. In addition, once a VNF instance is created, revising the computing and memory size on-the-fly is not supported in the cloud computing platform (e.g., Openstack).

To ensure the quality of service and improve resource utilization efficiency, we employ proactive traffic forecasting of each SFC to guide VNF scaling ahead of time. The VNF scaling includes the redundant instance management of over-provisioning VNFs and new instance deployment of under-provisioning VNFs. When traffic demand decreases, we convert the redundant instances into an idle state instead of deleting them immediately, which can be reactivated to avoid frequent creation and deletion. While the traffic demand increases, we need to create the new VNF instances with an adoptive capacity, where the capacity of new instances can be best-fit with node rather than the fixed capacity in other horizontal scaling works [27,29], which is supported by the current cloud computing platform. We use $c_f^{\max}$ to denote the maximum traffic that a VNF f instance can process in a time slot. With the obtained traffic demand $\alpha_i(t)$ along each SFC $r_i \in R$ in t, the processing resources that need to be allocated to VNF f instances can be calculated by (1).

$$C_f(t) = \sum_{r \in R: t \in [t_i, t_i + \Delta t_i]} c_f \alpha_i(t) \tag{1}$$

For traffic flow routing along a SFC, Split/Merge [31] enables efficient, load-balanced elasticity to support VNF scaling. The work [32] studies the dependency between VNFs and the traffic changing impact. Here, we do not study the impact of traffic changes between VNFs and assume that the traffic change ratio is 1 for all VNFs. Let variable $y^i_{g,f,m,n}(t)$ as routing variable denoting the amount of traffic forwarded from the type-g instances on the node m to the type-f instances on the node n belonging to r_i in time slot t. Then, the processing capacity requested by VNF f on the node n is:

$$C_{f,n}(t) = \sum_{r_i \in R} \sum_{f \in F} \sum_{m \in N} c_f y^i_{g,f,m,n}(t) \tag{2}$$

Thus, the number of VNF f instances that need to be newly deployed on the node n in t is:

$$z^{new}_{f,n}(t) = \begin{cases} 0 & C_{f,n}(t) \le C_{f,n}(t-1) \\ \left\lceil \dfrac{C_{f,n}(t) - C_{f,n}(t-1)}{c_f^{\max}} \right\rceil & other \end{cases} \tag{3}$$

Where the symbol $\lceil \bullet \rceil$ means round up. When $C_{f,n}(t) \le C_{f,n}(t-1)$, this does not need to create a new type-f VNF instance on the node n. When $0 < C_{f,n}(t) - C_{f,n}(t-1) \le c_f^{\max}$ and the node resource is more than the demand, there will be one new instance with capacity $C_{f,n}(t) - C_{f,n}(t-1)$. In other cases, we have to create $z^{new}_{f,n}(t) - 1$ instances with capacity $c_f^{\max}$ and one instance with $C_{f,n}(t) - C_{f,n}(t-1) - \left(z^{new}_{f,n}(t) - 1\right) c_f^{\max}$. After new VNF instances are spawned, traffic flow along an SFC can be split through many instances of the same type VNF and merged at the next VNF (i.e., successor VNF).

3.3 Cost Minimization Problem

The goal of MNOs is to minimize the total resource consumption of provisioning all network services. To avoid performance degradation and even service interruption caused by traffic

fluctuation, the related VNFs have to be scaled proactively. In t, MNOs forecast the traffic demand $\alpha_i(t+1)$ of each in-service SFC $r_i \in R$ at the next moment $t+1$. According to the forecasted results and newly arrived service requests in $t+1$, the MNO checks whether the processing capacity of existing VNF instances covers the upcoming traffic demands. If not, MNOs need to scale out the under-provisioning VNF by deploying new instances to handle the traffic flows in $t+1$. If the capacity of existing VNF instances is over-provisioning, the corresponding number and size of active VNF instances will be set to the idle state. Thus, we focus on the following two types of cost.

(1) Operating Cost: Let ϕ_f denote the operating cost of the VNF f instance renting a unit of processing resource per time slot. The total operating cost for running all VNFs of SFCs in t is:

$$C_{op}(t) = \sum_{n \in N} \sum_{f \in F} \phi_f C_{f,n}(t) \tag{4}$$

(2) Deployment Cost: Let φ_f be the resource usage cost for deploying a new type- f VNF instance. To create a new instance, the system needs to copy the VNF's image to the server node, and then spawn a VM/docker with the image. The total deployment cost for creating new VNF instances in t can be calculated by (5):

$$C_{de}(t) = \sum_{n \in N} \sum_{f \in F} \varphi_f z_{f,n}^{new}(t) \tag{5}$$

Table 1 lists significant notation in this section.

Table 1. Notation

Notation	Description
N	Set of substrate nodes
E	Set of physical links
C_n	The capacity of the substrate node n
$B_{m,n}$	The available bandwidth of the physical link $e_{m,n}$
F	Set of VNFs $f \in F$
c_f	Amount of processing resources required by the VNF f to process a unit traffic
R	Set of service requests
s_i	The source node of the service request r_i
d_i	The destination node of the service request r_i
$\alpha_i(t)$	Traffic demand of the service request r_i
sfc_i	VNF sequence of the service request r_i
Δt_i	Duration of the service request r_i
$x_{f,n}$	VNF f deployed on n
$y_{f,g,m,n}^{i}$	Amount of traffic forwarded from instances of VNF g on the node m to instances of VNF f on the node n
$c_f^{\max}$	Maximum processing capacity of a instance of VNF f
$C_f(t)$	Processing capacity requested by type- f VNF in time slot t
$C_{f,n}(t)$	Total processing capacity requested by VNF f on node n
$z_{f,n}^{new}(t)$	Number of newly created instances of VNF f on the node n in t

ϕ_f	Operating cost of the VNF f instance renting unit processing capacity per time slot
φ_f	The deployment cost of launching a new VNF f instance

Therefore, the optimization objective of this problem is to minimize the total resource costs over the time span T:

$$\min \sum_{t \in T} \left(C_{op}(t) + C_{de}(t) \right) \tag{6}$$

The VNF scaling problem can be formulated as an Integer Quadratic Programming (IQP), and the constraints are described as follows:

$$(1), (2), (3)$$

$$\sum_{f \in F} C_{f,n}(t) x_{f,n} \le C_n(t), \forall n \in N \tag{7}$$

$$\sum_{f \in F} \sum_{r_i \in R} \sum_{g \in F} \sum_{m \in N} y^i_{g,f,m,n}(t) \le B_{m,n}(t), \forall e_{m,n} \le E \tag{8}$$

$$\sum_{n \in N} C_{f,n}(t) = C_f(t), \forall f \in F \tag{9}$$

$$\sum_{f \in F} \sum_{m \in N} y^i_{f,g,m,n}(t) = \sum_{f \in F} \sum_{m \in N} y^i_{g,f,n,m}(t), \forall r_i \in R, g \in F, n \in N \tag{10}$$

$$x_{f,n} \in \{0,1\}, z^{new}_{f,n}(t) \in Z^+, y^i_{f,g,m,n}(t) \ge 0 \tag{11}$$

Constraint (7) guarantees that the total processing capacity required by the VNF instances assigned on the node $n \in N$ is not more than the residual capacity of the node n in t. Constraint (8) ensures that the total traffic carried by the physical link $e_{m,n} \in E$ cannot exceed its residual capacity $B_{m,n}$ in t. The equality between the total processing capacity of type-f VNF instances assigned on all nodes and the required processing capacity of VNF f in t is established by constraint (9). Constraint (10) represents flow conservation for type-f instances on the node n. Constraint (11) specifies the range of variables $x_{f,n}$, $z^{new}_{f,n}(t)$ and $y^i_{f,g,m,n}(t)$.

4. Proactive Adoptive VNF Scaling Approach

In this section, we first design an effective traffic forecasting method based on LSTM to proactively forecast traffic demands, and then describe the adaptive VNF scaling algorithm in detail. Finally, we propose a redundant instance management mechanism including deleting and reloading idle instances.

4.1 Traffic Forecasting Using LSTM

The main reason we use traffic forecasting is to achieve proactive VNF scaling, thus the accuracy of traffic prediction is very important for VNF scaling. To realize effective forecasting of traffic demand, we use a deep learning method called LSTM. As shown in **Fig. 1**, the LSTM is a special Recurrent Neural Network (RNN) based on long and short-term memory. It introduces the long and short-term memory cell structure to replace the ordinarily hidden neurons in the general RNN. It can dynamically change the weights of the input, state and output information in the cell structure at the current time step, and dynamically adjust the influence of historical input data on the prediction results on the time scale to solve the long-term dependence problem in the traditional RNN.

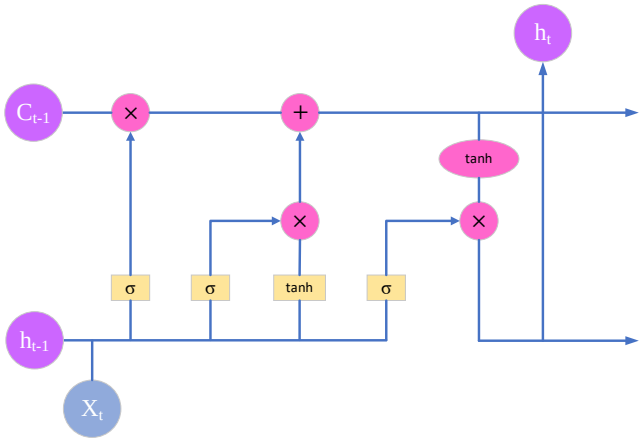

Fig. 1. LSTM cell framework

Our goal is to use historical traffic data information to forecast the upcoming traffic demand $\alpha_i(t)$ of each SFC r_i, and then use the traffic demand information to guide the VNF scaling. LSTM supports multiple input and single output mode, so we use a vector $\left[\alpha_i(t-\mathrm{T}),\alpha_i(t-\mathrm{T}+1),...,\alpha_i(t-1),\alpha_i(t)\right]$ as the input sequence and the traffic demand of next slot as the output result. Moreover, we can improve the prediction accuracy by controlling the size of the sliding window (i.e., the number of retrospective historical data).

4.2 Proactive Adaptive VNF Scaling Algorithm

Based on traffic forecasting, the MNO can perform the VNF scaling in advance to cope with the traffic fluctuation. The vertical scaling needs to restart the running VNF instances, hence we adopt horizontal scaling to adjust the total processing capacity of each VNF.

In this part, we describe the Proactive Adaptive VNF Scaling (**PAVS**) algorithm in detail, and the pseudocode is shown in **Algorithm 1**. **PAVS** uses the above LSTM to forecast the traffic demand $\alpha_i(t+1)$ of the SFC r_i at the next slot $t+1$. Based on $\alpha_i(t+1)$, **PAVS** can compute the processing capacity $C_f(t+1)$ required by each VNF f in the time slot $t+1$. If the required processing capacity in the next slot $t+1$ is more than the current processing capacity, the algorithm will scale out the corresponding VNFs according to the residual node resource and link bandwidth. It derives the processing capacity $C_{f,n}(t+1)$ on each node n, and then calculates the number of newly created instances for each VNF f. Finally, these new instances will be spawned and assigned to serve the traffic flows of SFCs.

Especially, if the required processing capacity in the next slot $t+1$ is less than the current processing capacity, we convert corresponding active instances to the idle state and manage the lifecycle of redundant instances by the ski-rental algorithm. The idle instances can be reactivated in the idle period to work. Beginning with $t=2$ (line 2), the algorithm checks the lifecycle of idle instances and deletes those instances after they have been idle for the "deadline" number of time slots. From $t=1$, we monitor the real traffic rates (line 6). If the estimated VNF f capacity cannot cover the actual demand, we will reactivate the idle instances to process the unserved traffic (lines 8-11). If the idle instances cannot cover the capacity requirement, it will call subprogram **RVID** to create new instances and routing paths (lines 12-13). We then update the residual node capacity and link bandwidth.

Next, we explain how to perform adaptive VNF scaling in SFCs based on traffic forecasting in detail. Based on the forecast traffic demands $\alpha_i(t+1)$, and the new arrival requests (lines 17-18), the processing capacity of each VNF f in the next slot $t+1$ can be estimated. Comparing the capacity demand at the next slot with the current capacity, we can determine whether the current VNF capacity needs to be adjusted. If the current processing capacity cannot meet the demand at the next slot, **PAVS** calls a subprogram **RVID** to jointly determines the capacity and deployment location of new instances on available nodes (lines 20-22). **RVID** will be given in detail in section 5. When the required processing capacity decreases, we will select the redundant VNF f instances on the node n and turn them into the idle state in the next slot. More importantly, we set a maximum lifecycle for each idle instance using the ski-rental algorithm (line 24).

Algorithm 1. Proactive Adaptive VNF Scaling (PAVS) algorithm

Input: $G=(N,E),R,F,\phi(f),\varphi(f)$

Output: $x_{f,n}, z_{f,n}^{new}(t), y_{f,g,m,n}^{i}(t)$

1. **for** $t=1,2,\ldots,T$ **do**
2. **for** $f=1,2,\ldots,F$ and $t\geq 2$ **do**
3. The remaining lifecycle of idle instances -1
4. Delete those instances after they have been idle for the "deadline" number of time slots
5. **end for**
6. Monitor real traffic rates $\alpha_i^*(t)$, $\forall r_i \in R$
7. **for** $f=1,2,\ldots,F$ **do**
8. Compute the required processing capacity $C_{f,n}^*(t)$ on each node n
9. **if** $C_{f,n}^*(t)\geq C_{f,n}(t)$ **then**
10. Reactive type- f VNF instances that are in IDLE state
11. **end if**
12. **if** The idle instances cannot cover the added capacity **then**
13. Call RVID to deploy new instances and routing paths
14. **end if**
15. **end for**
16. Update the residual node capacity $C_n(t)$ and bandwidth $B_{m,n}(t)$
17. Derive forecasted traffic demand $\alpha_i(t+1)$ by LSTM, $\forall r_i \in R: t+1 \in [t_i, t_i+\Delta t]$
18. Add new arrival service requests $\alpha_i(t+1), \forall r_i \in R: t+1 \in [t_i, t_i+\Delta t]$
19. **for** $f=1,2,\ldots,F$ **do**
20. Compute the processing capacity $C_f(t+1)$ and compare it with the last slot
21. **if** $C_f(t+1) > C_f(t)$ **then**
22. Call RVID to deploy instances and allocate the routing path
23. **else**
24. Convert the redundant VNF f instances to idle and set a lifecycle for idle instances
25. **end if**
26. **end for**
27. Update residual node capacity $C_n(t)$ and bandwidth $B_{m,n}(t)$
28. **end for**

4.3 Redundant VNF Instance Management Mechanism

For the over-provisioning VNF instances, we choose to keep them in the system for some time instead of directly deleting them. That is because the traffic fluctuation and new arrival service requests may cause the frequent deletion and creation of VNF instances, which will incur significant deployment costs. Inspired by literature [13], We use the ski-rental algorithm to set a maximum lifetime for every idle instance to optimize resource usage efficiency. When the instance capacity of VNF f on the node n is over-provisioning, we select the corresponding redundant instances and convert them to the idle state. The idle instances can save operating costs (e.g., energy consumption) compared with the active instances. Meanwhile, for each idle instance, its maximum lifetime "deadline" j follows the given distribution:

$$P_j = \left(\frac{\Delta_f - 1}{\Delta_f}\right)^{\Delta_f - j} \frac{1}{\Delta_f \left(1 - \left(1 - 1/\Delta_f\right)^{\Delta_f}\right)} \tag{12}$$

Where $\Delta_f = \left[\varphi_f / \left(\phi_f c_f\right)\right]$. When an instance has been idle for the "deadline" number of time slots, it will be deleted from the node. If an idle instance needs to be reloaded again, it will be reactivated to process traffic flows. In this way, the algorithm can effectively balance deployment and operating expenses to reduce total costs. Meanwhile, it has been proved that the ski-rental algorithm can achieve a competitive ratio of $e/e-1$ [33].

5. Resource-aware VNF Instance Deployment Algorithm

We now present the subprogram Resource-aware VNF Instance Deployment (**RVID**) algorithm. Given the traffic demands $\alpha_i(t+1)$ and the substrate network resource status, the problem of deploying new instances and setting routing paths remained to be solved to serve the increased traffic demands of SFCs.

Algorithm 2. RVID based on Matrix Coded Genetic Algorithm

Input: $\alpha_i(t+1)$-$\alpha_i(t), r_i, C_n(t), B_{m,n}(t)$

Output: VNF deployment decision $X^*(t+1)$

1. NUMPOP=100, MAXGEN = 500, P_c=0.75, P_m=0.05
2. Calculate the quantity of VNFs to be scaled in the time slot $t+1$
3. Initialize population NUMPOP, the chromosome of each individual is encoded by a matrix
4. **for** gen = 1: MAXGEN **do**
5. **if** The chromosomes satisfy the constraint (7)-(10) **then**
6. Calculate the fitness of chromosomes, the fitness is the reciprocal of resource costs
7. **else**
8. The fitness value is the small penalty
9. **end if**
10. Calculate the priority of chromosomes participating based on fitness
11. Crossover candidate chromosomes with probability P_c
12. Variant candidate chromosomes with probability P_m
13. Generate the kid population, repeat steps 4-8
14. Replace chromosomes with high fitness in the kids with chromosomes with low fitness in the parent and generate a new population
15. **end for**
16. Keep the best chromosomes and return to their corresponding VNF deployment strategy

The SFC deployment has been proved to be an NP-hard problem [34,35]. When the network function capacity is expanded, it is necessary to deploy new instances and add new routing paths, which can be regarded as the deployment of a service function chain containing only one VNF. Therefore, the new instance deployment and flow routing path allocation can be seen as a simplified variant of SFC deployment. To reduce the computational complexity, we design an **RVID** algorithm based on the matrix-coded genetic algorithm, and the specific process is given in **Algorithm 2**. RVID first calculates the quantity of VNFs to be scaled in the time slot $t+1$, which determines the number of columns in the matrix. We then generate an initial population with the number of NUMPOP, where the chromosome of each individual adopts a matrix coding method (line 3). As shown in **Fig. 2**, the number of rows in the matrix is determined by the number of available substrate nodes, and the value of the corresponding position indicates the traffic that the VNF needs to process at the node. The sum of each row of the matrix represents the processing capacity that this node needs to allocate for each VNF, and the total amount cannot exceed the residual resources of the node. The sum of each column of the matrix indicates the capacity of each VNF that needs to be scaled out. Each individual's chromosome represents a VNF instance deployment scheme, which needs to be checked whether the constraints are met. If the chromosome satisfies all constraints (7)-(10), the fitness is equal to the reciprocal of resource usage costs, which consist of node resource and bandwidth usage cost, else its fitness will be a very small penalty value and it is eliminated (lines 5-9). We choose these individuals with larger fitness values to participate in genetic selection and perform crossover and mutation operations (lines 10-12). The individuals with higher fitness in the newly generated kids replace the individuals with lower fitness in the parent to generate a new population. Finally, after the loop is over, the best-performing individual is selected, and the VNF instance deployment scheme corresponding to its chromosome is decoded.

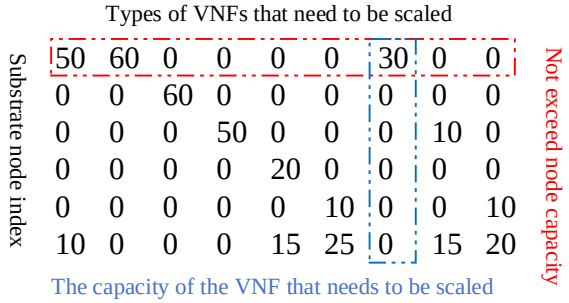

Fig. 2. Schematic diagram of individual coding

When performing crossover and mutation operations, the **RVID** algorithm adopts the matrix coding method, which makes the traditional single-point crossover or mutation no longer applicable. Thus, we design a crossover and mutation method based on the column. When different chromosomes are crossover, the crossover is carried out in the basic unit of the column, while the mutation of a single chromosome is mutated at the same time in the same column. In this way, the generation of individuals violating the constraints can be avoided, and the rate of convergence of the algorithm can be accelerated.

6. Performance Evaluation

6.1 Simulation Setup

The experimental cloud data center is built using Openstack and contains 15 RH2288 V3 servers. Network equipment includes 1 10 GB traditional switch and 3 V580-32X SDN switches, the specific server configuration parameters are shown in **Table 2**. The traffic demands of service function chains are derived from the data center access gateway server by trace-driven. The time interval of traffic statistics in the data set is 1 hour, and the statistical duration spans over 1500 hours. We divide data center traffic into different flows according to different service requirements, such as video, web services, instant messaging, etc. Each flow goes through an SFC containing related VNFs.

Table 2. Server Parameters

CPU	Memory(GB)	Hard Disk(TB)
28 CPUs x Intel(R) Xeon(R) CPU E5-2660 v4 @ 2.00GHz	1024	10

We use the PyTorch framework 3.8 to build the LSTM network. The experimental parameters of traffic forecast based on LSTM are selected as follows: the time step is set to 25, the learning rate is 0.01, the learning period is 10000, the loss function is the minimum mean square error, and the optimizer selects Adam. Select 70% of the traffic data is used as the training set, and 30% of the data is used as the test set. In the genetic algorithm, the total population size is 100, the crossover probability is 0.75, the mutation probability is 0.05, and the maximum evolutionary generation is 500.

6.2 Effectiveness of Traffic Forecast

We first evaluate the performance of the traffic forecasting method based on LSTM and adopt Mean Squared Error (MSE) as the performance indicator. We first investigate the impact of window size on prediction accuracy in **Fig. 3**. The input window size is selected from [5, 10, 15, 20, 25, 30], and the MSE is the smallest when the window size is 25. Thus, we use 25 as the window size for the subsequence experiments. Meanwhile, the real traffic demand fluctuation is shown in **Fig. 4**. **Fig. 5** shows the forecast error of the traffic forecasting method at each moment. The forecast errors between the forecasted traffic sequence and the original sequence are less than 0.5. Thus, the forecast result can verify that the LSTM network can effectively forecast the trend of traffic changes and the forecasted result can be used to guide the dynamic VNF scaling.

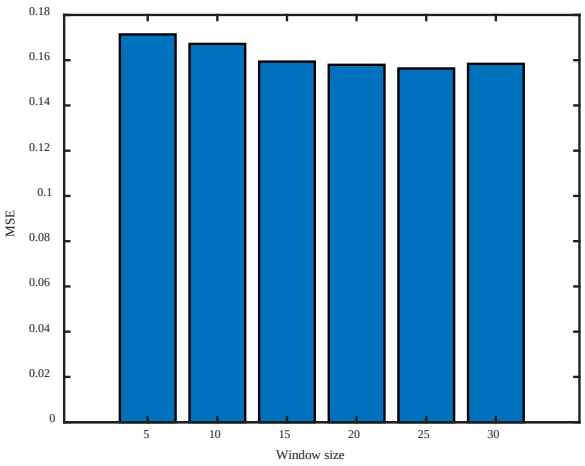

Fig. 3. Forecast error with window size

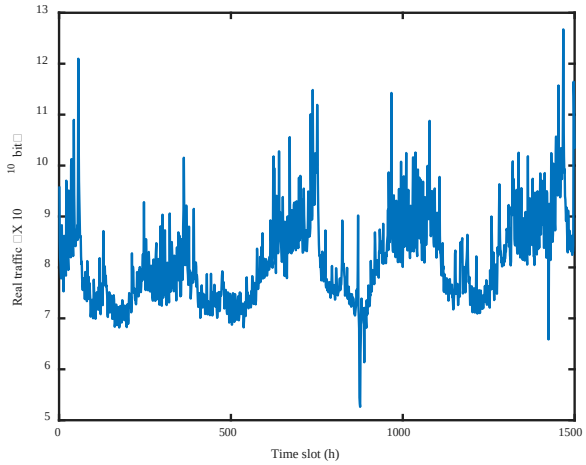

Fig. 4. Dataset traffic

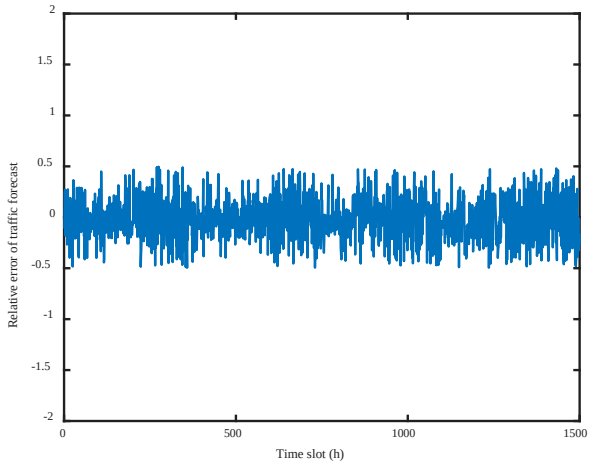

Fig. 5. Relative error of traffic forecast

6.3 Impact of Redundant VNF Instance Management Mechanism

To evaluate the effectiveness of the redundant VNF instance management mechanism, we compare our proposed ski-rental model (SRM) with the random lifecycle (RLC) and static lifecycle (SLC) methods. The RLC randomly configures the lifecycle of the idle instances in the interval of [1, 7]. The SLC always sets the lifecycle of the idle instances to a fixed value (i.e., 7). As shown in **Fig. 6**, because the SLC sets the lifecycle of the idle instances to a maximum fixed value, the idle instances remain in the system for a long time resulting in the highest operating cost. According to Equation (12), the SRM tends to configure the lifecycle of the idle instances with a larger value, so the SRM will incur more operating costs than the RLC. The RLC has the lowest operating costs because the average lifecycle of those idle instances is the shortest.

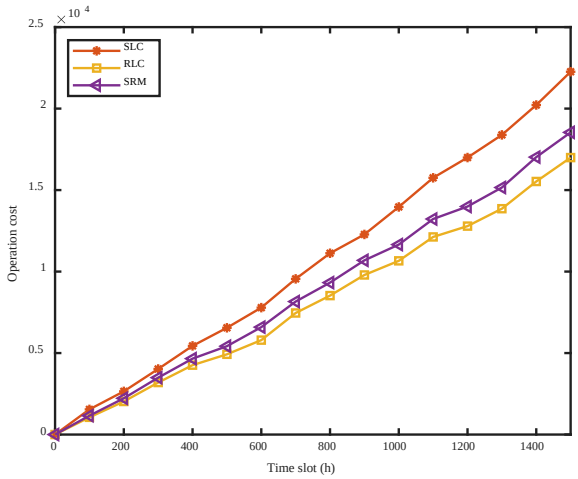

Fig. 6. Operating costs with different lifecycle management

On the other hand, idle instances can be reactivated to continue to provide services, which can reduce the creation of new instances to decrease deployment costs. The idle instances with a longer lifecycle remain in the system longer, which has a greater probability of being reactivated. Thus, as shown in **Fig. 7**, the deployment cost of SLC is the least, and the SRM is less than the RLC during the long-time operation of the system.

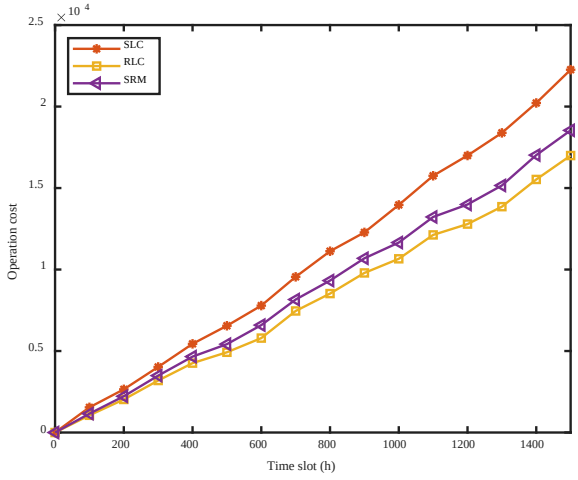

Fig. 7. Deployment cost with different lifecycle management

Fig. 8 depicts the total cost of the three methods over time. As we can see, the total cost of the SRM is the smallest. That is because the SRM can effectively balance the retention lifecycle of idle instances according to the deployment and operating costs of VNF instances, thereby improving resource utilization. What's more, as time increases, the difference in cost between the three methods is bigger and bigger.

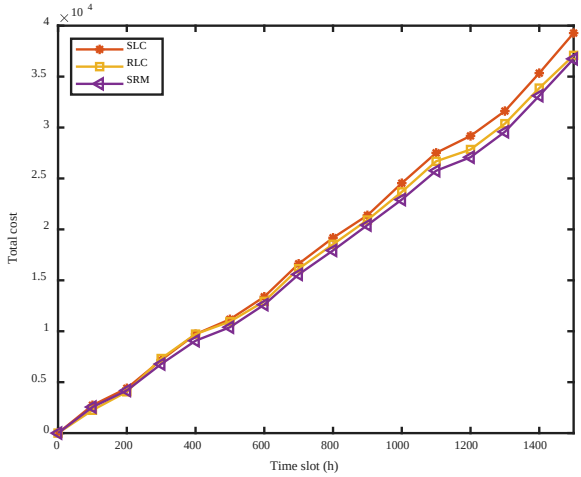

Fig. 8. Total cost with different lifecycle management

6.4 Comparison with Other Approaches

In this part, we investigate the performance of the proposed **PAVS** algorithm compared with the other two algorithms called Best-fit and VPCM [17]. The Best-fit algorithm creates new VNF instances only with the same capacity and places the newly created VNF instances on the used node with the maximum residual capacity, that is it adopts a greedy algorithm. It should be noted that all the total costs in the simulation are calculated based on the real traffic requirement with different VNF scaling schemes by different algorithms, and all three algorithms add our proposed redundant instance management mechanism. We first verify the solution efficiency of the **RVID** algorithm based on the matrix-coded genetic algorithm. It can be seen from **Fig. 9** that the proposed **RVID** algorithm can achieve fast convergence when solving the approximate optimal solution.

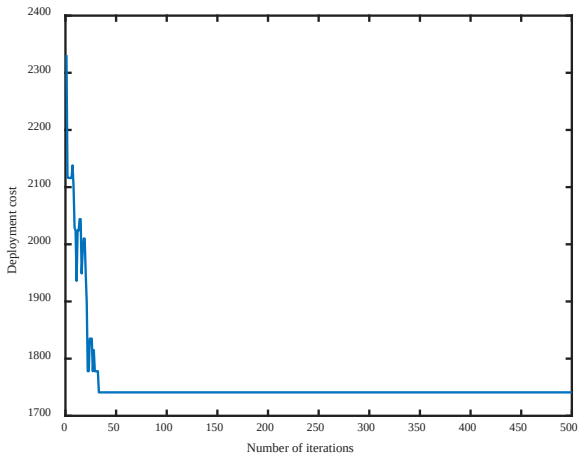

Fig. 9. Rate of convergence of RVID algorithm

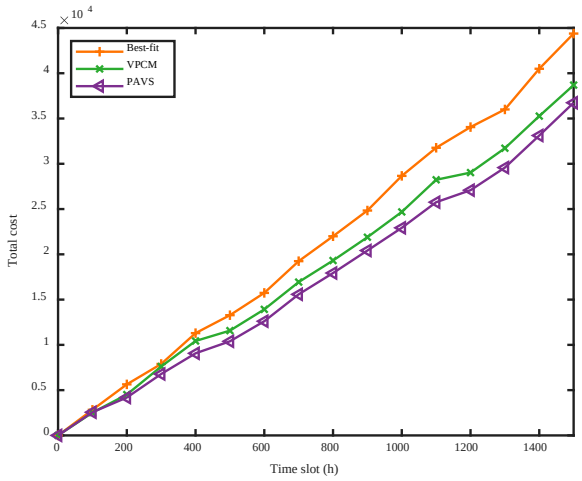

Fig. 10. Total cost under different algorithms

Next, the experiment results shown in **Fig. 10** describe the total costs of the three algorithms changing over 1500 time slots. We can clearly see that **PAVS** achieves the lowest total cost compared with Best-fit and VPCM. This is due to our proposed **PAVS** algorithm jointly resolves the capacity and deployment location of new instances, rather than determining instance capacity before deploying. According to the initial deployment of SFCs, joint decision-making can make full use of the residual resource of substrate nodes and links, while the other method may lead to circuitous paths due to resource constraints. Moreover, Best-fit prioritizes the scaled VNF instance on the node with the most residual resources, which may miss an optimal location that satisfies capacity constraints instead of the maximum residual capacity. Thus, the Best-fit produces the highest total cost.

7. Conclusion

We investigate the VNF scaling problem taking account of the traffic fluctuation of service function chains in the paper. We first model this problem as an integer quadratic programming, which minimizes the total resource consumption incurred by VNF deployment and operating. And then we propose a proactive adaptive VNF scaling approach. The approach first employs a traffic forecasting method based on LSTM to forecast the traffic demand. When the current capacity cannot cover the next slot demand, a resource-aware new VNF instance deployment algorithm based on a matrix coding genetic algorithm is devised to make full use of substrate network resources and decrease deployment costs. Meanwhile, when traffic demands decrease, we propose a redundant instance management mechanism to avoid the frequent creation and deletion of instances. Trace-driven simulation further demonstrates that the overall cost can be reduced significantly by our method and the good performance. In future work, we will study the dynamic extension of already deployed SFCs or VNF forward graphs to meet the users' new requirements, such as the security level.

Acknowledgement

The authors sincerely thank the anonymous reviewers for their valuable comments that helped improve the readability and clarity of this paper. This work was supported by the National Key R&D Program of China (Grant No. 2020YFB1806607).

References

[1] Alliance, NGMN, "5G white paper," white paper Version 1.0. March 2015.
[2] ETSI GS NFV-MAN, "Network Functions Virtualisation (NFV); Management and Orchestration". 2014.
[3] ETSI GS NFV, "Network Functions Virtualisation (NFV); Architectural Framework". 2013.
[4] System architecture for the 5G system, 3GPP TS 23.501 V15. 3.0, 2018.
[5] Service function chaining (SFC) architecture, No. rfc7665, 2015.
[6] Hantouti, Hajar, Nabil Benamar, and Tarik Taleb, "Service Function Chaining in 5G & Beyond Networks: Challenges and Open Research Issues," *IEEE Network*, vol. 34, no. 4, pp. 320-327, July 2020. Article (CrossRef Link)
[7] Yao Hong, et al, "Joint optimization of function mapping and preemptive scheduling for service chains in network function virtualization," *Future Generation Computer Systems*, vol. 108, pp. 1112-1118, July 2020. Article (CrossRef Link)
[8] Sun Gang, et al, "Low-latency and resource-efficient service function chaining orchestration in network function virtualization," *IEEE Internet of Things Journal*, vol. 7, no. 7, pp. 5760-5772, July 2020. Article (CrossRef Link)
[9] Hang Qiu, Hongbo Tang, and Wei You, "Online Service Function Chain Deployment Method Based on Deep Q Network," *Journal of Electronics and Information Technology*, vol. 43, no. 11, pp. 3122-3130, Nov. 2021. Article (CrossRef Link)
[10] Yu Hui, Jiahai Yang, and Carol Fung, "Elastic network service chain with fine-grained vertical scaling," in *Proc. of 2018 IEEE Global Communications Conference (GLOBECOM)*, 2018. Article (CrossRef Link)
[11] Pre-deployment Testing; Report on Validation of NFV Environments and Services, ETSI GS NFV-TST 001 V 1.1.1, 2016.
[12] Ghaznavi, Milad, et al., "Elastic virtual network function placement," in *Proc. of 2015 IEEE 4th International Conference on Cloud Networking (CloudNet)*, 2015. Article (CrossRef Link)
[13] Wang Xiaoke, et al., "Online VNF scaling in datacenters," in *Proc. of 2016 IEEE 9th International Conference on Cloud Computing (CLOUD)*, 2016. Article (CrossRef Link)
[14] Houidi Omar, et al., "An efficient algorithm for virtual network function scaling," in *Proc. of 2017 IEEE Global Communications Conference*, 2017. Article (CrossRef Link)
[15] Tang Hong, Danny Zhou, and Duan Chen, "Dynamic network function instance scaling based on traffic forecasting and VNF placement in operator data centers," *IEEE Transactions on Parallel and Distributed Systems*, vol. 30, no. 3, pp. 530-543, March 2019. Article (CrossRef Link)
[16] Yao Yifu, et al, "Forecasting assisted VNF scaling in NFV-enabled networks," *Computer Networks*, vol. 168, no. 26, p. 107040, Feb. 2020. Article (CrossRef Link)
[17] Fei Xincai, et al., "Adaptive VNF scaling and flow routing with proactive demand prediction," in *Proc. of 2018 IEEE Conference on Computer Communications*, 2018. Article (CrossRef Link)
[18] Li Defang, Peilin Hong, and Kaiping Xue, "Virtual network function placement considering resource optimization and SFC requests in cloud datacenter," *IEEE Transactions on Parallel and Distributed Systems*, vol. 29, no. 7, pp. 1664-1677, July 2018. Article (CrossRef Link)
[19] You Chaoqun, "Efficient load balancing for the VNF deployment with placement constraints," in *Proc. of 2019 IEEE International Conference on Communications (ICC)*, 2019. Article (CrossRef Link)

[20] Qi Dandan, Subin Shen, and Guanghui Wang, "Towards an efficient VNF placement in network function virtualization," *Computer Communications*, vol. 138, pp. 81-89, Apr. 2019. Article (CrossRef Link)
[21] Fu Xiaoyuan, et al, "Dynamic service function chain embedding for NFV-enabled IoT: A deep reinforcement learning approach," *IEEE Transactions on Wireless Communications*, vol. 19, no. 1, pp. 507-519, Jan. 2020. Article (CrossRef Link)
[22] Nguyen Minh, Mahdi Dolati, and Majid Ghaderi, "Deadline-aware SFC orchestration under demand uncertainty," *IEEE Transactions on Network and Service Management*, vol. 17, no. 4, pp. 2275-2290, Dec. 2020. Article (CrossRef Link)
[23] Li Jing, Weifa Liang, and Yu Ma, "Robust service provisioning with service function chain requirements in mobile edge computing," *IEEE Transactions on Network and Service Management*, vol. 18, no. 2, pp. 2138-2153, June 2021. Article (CrossRef Link)
[24] Sedaghat Mina, Francisco Hernandez-Rodriguez, and Erik Elmroth, "A virtual machine re-packing approach to the horizontal vs. vertical elasticity trade-off for cloud autoscaling," in *Proc. of the 2013 ACM Cloud and Autonomic Computing Conference*, pp. 1-10, 2013. Article (CrossRef Link)
[25] Hwang Kai, Yue Shi, and Xiaoying Bai, "Scale-out vs. scale-up techniques for cloud performance and productivity. in *Proc. of 2014 IEEE 6th International Conference on Cloud Computing Technology and Science*, 2014. Article (CrossRef Link)
[26] Wang Wenting, Le Xu, and Indranil Gupta, "Scale Up vs. scale out in cloud storage and graph processing systems," in *Proc. of IEEE International Conference on Cloud Engineering*, 2015. Article (CrossRef Link)
[27] Zhao Xin, Xuan Jia, and Yanpei Hua, "An Efficient VNF Deployment Algorithm for SFC Scaling-out Based on the Proposed Scaling Management Mechanism," in *Proc. of 2020 Information Communication Technologies Conference (ICTC)*, 2020. Article (CrossRef Link)
[28] Pei Jianing, et al, "Efficiently embedding service function chains with dynamic virtual network function placement in geo-distributed cloud system," *IEEE Transactions on Parallel and Distributed Systems*, vol. 30, no. 10, pp. 2179-2192, Oct. 2019. Article (CrossRef Link)
[29] Zhai D, Meng X, Yu Z, et al, "A fine-grained and dynamic scaling method for service function chains," *Knowledge-Based Systems*, vol. 228, p. 107289, Sep. 2021. Article (CrossRef Link)
[30] Harutyunyan Davit, Rasoul Behravesh, and Nina Slamnik-Kriještorac, "Cost-efficient placement and scaling of 5G core network and MEC-enabled application VNFs," in *Proc. of 2021 IFIP/IEEE International Symposium on Integrated Network Management (IM)*, 2021. Article (CrossRef Link)
[31] Rajagopalan Shriram, et al., "{Split/Merge}: System Support for Elastic Execution in Virtual Middleboxes," in *Proc. of 10th USENIX Symposium on Networked Systems Design and Implementation (NSDI 13)*., 2013. Article (CrossRef Link)
[32] Ma Wenrui, et al., "Traffic aware placement of interdependent NFV middleboxes," in *Proc. of 2017 IEEE Conference on Computer Communications*, 2017. Article (CrossRef Link)
[33] Lotker Zvi, Boaz Patt-Shamir, and Dror Rawitz, "Ski rental with two general options," *Information processing letters*, vol. 108, no. 6, pp. 365-368, Nov. 2008. Article (CrossRef Link)
[34] Hawilo Hassan, Manar Jammal, and Abdallah Shami, "Network function virtualization-aware orchestrator for service function chaining placement in the cloud," *IEEE Journal on Selected Areas in Communications*, vol. 37, no. 3, pp. 643-655, March 2019. Article (CrossRef Link)
[35] Herrera Juliver Gil, and Juan Felipe Botero, "Resource allocation in NFV: A comprehensive survey," *IEEE Transactions on Network and Service Management*, vol. 13, no. 3, pp. 518-532, Sept. 2016. Article (CrossRef Link)

Hang Qiu received the B.E. degree from Nanjing University of Post and Telecommunication, Nanjing, China, in 2016, and M.E. degree in Information and Communication Systems from Information Engineering University, Zhengzhou, China, in 2019. He is currently working towards the Ph. D degree in Information Engineering University. His current interest includes Next-generation mobile communication technology and mobile communication network security.

Hongbo Tang received the B.E. degree from Huazhong University of Science and Technology in 1989, Wuhan, China, and M.E. degree in Engineering from National University of Defense Technology, Changsha, China, in 1995. He is now a professor of Information Engineering University, China. His current interest includes Next-generation on mobile communication and mobile communication network security.

Yu Zhao received the M.S. and Ph.D degrees in Information and Communication Systems from Information Engineering University, Zhengzhou, China. He is now a lecturer of Information Engineering University. His current interest includes Next-generation on mobile communication and mobile communication network security.

Wei You received the M.S. and Ph.D degrees in cryptology from Information Engineering University, Zhengzhou, China. He is currently an associate professor of Information Engineering University. His major research interests include New-generation mobile communication systems, and mobile communication network security.

Xinsheng JI, received his BE degree in Fudan University, Shanghai, China, in 1988, and his MS degree in Information Engineering University, Zhengzhou, China, in 1991. He is currently a Chief Engineer of the China National Digital Switching System Engineering and Technological R&D Center (NDSC). He is a member of the National 6G Technology R&D General Expert Group, a Chief Scientist of the wireless security field of the Collaborative Innovation Center for Wireless Communication, a Deputy Director of the National Engineering Laboratory for Mobile Network Security, and an Academic Leader of the National Science Foundation Innovation Corps. His major research interests include next-generation mobile communication and cyber space security.