Resource Aware Routing for Service Function Chains in SDN and NFV-Enabled Network

Jianing Pei, Peilin Hong, Kaiping Xue, *Senior Member, IEEE*, and Defang Li, *Student Member, IEEE*

Abstract—Owing to the Network Function Virtualization (NFV) and Software-Defined Networks (SDN), Service Function Chain (SFC) has become a popular service in SDN and NFV-enabled network. However, as the Virtual Network Function (VNF) of each type is generally multi-instance and flows with SFC requests must traverse a series of specified VNFs in predefined orders, it is a challenge for dynamic SFC formation to optimally select VNF instances and construct paths. Moreover, the load balancing and end-to-end delay need to be paid attention to, when routing flows with SFC requests. Additionally, fine-grained scheduling for traffic at flow level needs differentiated routing which should take flow features into consideration. Unfortunately, traditional algorithms cannot fulfill all these requirements. In this paper, we study the Differentiated Routing Problem considering SFC (DRP-SFC) in SDN and NFV-enabled network. We formulate the DRP-SFC as a Binary Integer Programming (BIP) model aiming to minimize the resource consumption costs of flows with SFC requests. Then a novel routing algorithm, Resource Aware Routing Algorithm (RA-RA), is proposed to solve the DRP-SFC. Performance evaluation shows that RA-RA can efficiently solve the DRP-SFC and surpass the performance of other existing algorithms in acceptance rate, throughput, hop count and load balancing.

Index Terms—Service function chain, software-defined networks, network function virtualization, differentiated routing, flow feature

1 Introduction

Network Function Virtualization (NFV) has been an arising technology decoupling the software from hardware devices recently. It has the potential to significantly reduce the Operating Expenses (OPEX) and Capital Expenses (CAPEX) and facilitate the flexibility of new services deployment with increased agility and faster time-to-value [1], [2]. Software-Defined Networks (SDN) is a new network paradigm which decouples the control plane and data plane. According to centralized control and flexible management, SDN controller can efficiently control the network forwarding among Network Functions (NFs) based on the acquired information about the network [3]. Owing to the technologies of NFV and SDN, many NFs such as Firewall (FW), Deep Package Inspection (DPI), Intrusion Detection System (IDS), Intrusion Prevention System (IPS) and Wide Area Network (WAN) optimizers can be software-oriented, programmed and deployed flexibly on Commercial-Of-The-Shelf (COTS) devices [4], [5], [6], which are known as the Virtual Network Functions (VNFs).

Benefitting from NFV and SDN, Service Function Chain (SFC) has been proposed as a popular service paradigm. An SFC defines an ordered or partially ordered set of VNFs and ordering constraints that must be applied to packets, frames and/or flows selected as a result of classification [7], [8]. SFC provides the means so that the traffic can naturally pass through a set of specified VNF instances sequentially without the intervention imposed by different services residing at different physical devices [9], [10]. With the application of SFC, high acceleration of traffic performance will be provided by more intelligent traffic routing strategies in today's Internet Service Provider (ISP) networks.

Even though SFC is hopeful to enhance the flexibility and cost efficiency in ISP networks [9], [11], however, a set of new challenges come correspondingly, which should be well addressed. First, the VNF of each type is generally multi-instance and flows with SFC requests must pass through a series of VNF instances in predefined orders to satisfy the requirements of users. For example, in Fig. 1, there are four types of VNFs deployed in the network and each of them contains multiple instances. *VNF11*, *VNF12* and *VNF13* indicate the first, second and third instances of *VNF1*, respectively. The rest of VNF instances satisfy the same rule. Supposing that a flow with SFC request starts from node A and needs to traverse the instances of *VNF1*, *VNF2*, *VNF3* and *VNF4* before arriving at node J. Nevertheless, in the network, there exist many paths (such as the dotted lines marked with different colors) that traverse different VNF instances and can satisfy the requirement. Therefore, it is a challenge for dynamic SFC formation to make an optimal strategy selecting VNF instances from multi-instance NFV environment and routing flows with SFC requests to traverse these selected VNF instances in predefined orders.

Second, as the bandwidth on links should not be the only resource to be considered in SDN and NFV-enabled network, flow table entries on SDN switch nodes which are resolved in Ternary Content Addressable Memory (TCAM) [12], [13],

• *The authors are with the Key Laboratory of Wireless-Optical Communications, Chinese Academy of Sciences, School of Information Science and Technology, University of Science and Technology of China, Hefei 230027, China. E-mail: {jianingp, ldf911}@mail.ustc.edu.cn, {plhong, kpxue}@ustc.edu.cn.*

Manuscript received 20 Aug. 2017; revised 9 Apr. 2018; accepted 18 June 2018. Date of publication 22 June 2018; date of current version 5 Aug. 2021. (Corresponding author: Peilin Hong.)
Digital Object Identifier no. 10.1109/TSC.2018.2849712

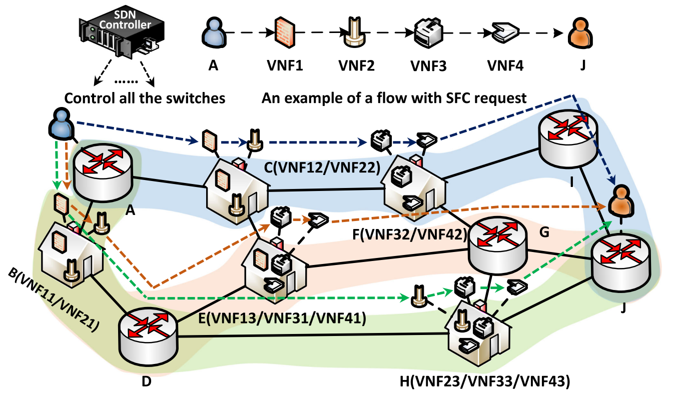


Fig. 1. VNF instance selection and path construction for flows with SFC requests in SDN and NFV-enabled network.

the CPU on function nodes which hold VNF instances and end-to-end delay cannot be neglected as well [14]. Hence, a mechanism should be designed to make a trade off among multiple kinds of resources and pay attention to end-to-end delay, which can reduce the network congestion and keep the network with high performance. Third, to deal with increasingly heterogeneous traffic in the network, a differentiated routing strategy with the consideration of flow features is hoped by network operators to achieve fine-grained scheduling for traffic at flow level [9], [15].

Given the challenges introduced by flows with SFC requests in SDN and NFV-enabled network, there are two problems in the following to be solved: (i) how to make differentiated routing strategy for different kinds of flows with SFC requests to optimally select VNF instances and construct the paths without violating predefined orders. (ii) how to achieve load balancing among multiple kinds of resources, when routing flows with SFC requests. These problems are denoted as the *Differentiated Routing Problem considering SFC (DRP-SFC)*. The contributions of this paper are listed as follows.

- We make a detailed analysis of the DRP-SFC and formulate it as a Binary Linear Programming (BIP) model aiming to minimize the resource consumption costs of flows with SFC requests.
- We separate the flows into different kinds based on resource preferences, and define relative cost to balance the resource consumption and route heterogeneous traffic at flow level differentiatedly in SDN and NFV-enabled network.
- Considering multi-resource constraints (bandwidth, flow table entries on switch nodes, CPU on function nodes and end-to-end delay) and flow features comprehensively, we propose a novel routing algorithm, Resource Aware Routing Algorithm (RA-RA), to solve the DRP-SFC in SDN and NFV-enabled network. Detailed simulation results show that, comparing with other algorithms in existing literatures, RA-RA can efficiently solve the DRP-SFC and obtain higher performance in acceptance rate, throughput, hop count and load balancing.

The rest of the paper is organized as follows: we review the related works in Section 2. The system model is presented in Section 3. In Section 4, we give the definition of the relative cost and formulate the DRP-SFC as a BIP model. Section 5 describes the RA-RA algorithm. In Section 6, we validate the effectiveness of RA-RA and compare it with some existing algorithms. Finally, Section 7 concludes the paper.

2 Related Works

Recently, advancements in the field of NFV and SDN make SFC drawn significant attention in both the standardization organizations and research communities.

The Service Chaining Working Group in Internet Engineering Task Force (IETF) completed a set of related SFC use-cases drafts referred to the SFC architecture [7], mobile networks [16] and datacenters [17]. The VNF Forwarding Graph (VNFFG) was proposed by European Telecommunications Standards Institute (ETSI) to describe the connectivity between VNFs [18].

Based on NFV and SDN, new SFC architectures are proposed to interconnect different VNF instances in specified orders [19], [20]. Moreover, with an increasing number of tenants launching their applications in clouds, it is also advocated by Cloud Service Providers (CSPs) to construct SFC architectures in clouds to meet the demands of tenants, promote the cloud performance and reduce the OPEX/CAPEX [1], [21].

Medhat et al. [22] came up with a service function selection algorithm for the service function instance selection and service function path creation problem. In order to realize load balancing among VNF instances, the authors selected the specified VNF instances by trading off the delay feature of flows, load conditions of VNF instances and distance in the network. And the Shortest Path (SP) algorithm is used to produce a complete path. Based on the euclidean distance and SP algorithm, Oh et al. proposed a Virtual Machine (VM) selection algorithm to create optimal service-chain paths for flows with SFC requests [23]. When creating the optimal path, this algorithm first constructs a 3D vector space based on the requirement of a flow and the statements of VMs in the network. Then the right VMs are selected by calculating the euclidean distance from the requirement point, and the SP algorithm is used to concatenate these selected VMs. Mechtri et al. [24] proposed a novel eigendecomposition based approach to cope with the VNF placement and chaining problem in distributed cloud environments. This algorithm first needs to extend the adjacent matrix of a VNFFG to the same dimension of physical network's, then execute Umeyama's eigendecomposition approach to select the VNF instances and construct paths.

All the papers in [22], [23], [24] route the flows with SFC requests by two-stage algorithms which need to complete the selection of VNF instances at the first stage, then construct the paths concatenating the selected VNF instances as the predefined order in the second stage. When routing the flows with SFC requests in the network, both the selection of VNF instances and the construction of paths have influence on the network performance. As splitting the relationship between the selection of VNF instances and path construction, these two-stage algorithms will lead to sub-optimal solution for the routing of flows with SFC requests. Distinct from the two-stage algorithms, RA-RA is a one-stage algorithm which finishes these two processes meanwhile. And RA-RA can efficiently make strategies for flows with SFC requests by eliminating sub-optimal paths.

Considering flows with SFC requests online routing problem in SDN framework, Cao et al. [25] proposed a novel algorithm named Competitive Online Algorithm for Traffic Steering (COATS). In COATS, the authors iteratively updated the costs on links and routed flows with SFC requests based on a layered graph. Bari et al. [26] defined the VNF deployment and online routing problem as VNF Orchestration Problem (VNF-OP). In the paper, all the nodes in the network can support VNF instances, and the costs of VNF deployment, energy, data forwarding, resource fragments and Service Level Objective (SLO) violation are considered. Then based on Viterbi algorithm [27], the ProvisionTraffic is proposed to select the path with the lowest cost to route the flow with SFC request in the network. Facing the node-constrained service chain routing problem in SDN framework, Dwaraki et al. [28] proposed an adaptive service routing algorithm to solve this problem. In the algorithm, the network graph is transformed to a layered graph considering process steps. Then the path with the minimum end-to-end delay is obtained to route the flow with SFC request by executing the conventional SP algorithm on layered graph. Pei et al. [14] studied the VNF Selection and Chaining Problem (VNF-SCP), and proposed to solve it based on deep learning technology. The papers in [29] explicitly states that the largest open source SDN controller, OpenDaylight, has supported the algorithms including Random, Round Robin, Load Balance and Shortest Path for the selection of VNF instances in SDN framework.

To solve the DRP-SFC, the tradeoff among multiple kinds of resources and end-to-end delay needs to be considered comprehensively. In SDN and NFV-enabled network, all kinds of resources including the bandwidth, flow table entries and CPU and end-to-end delay have influence on the network performance. Unbalanced utilization of resources will lead to low network performance due to network congestion. And long end-to-end delay also results in low Quality of Service (QoS). In [25], the authors only paid attention to the bandwidth on links, when routing flows with SFC requests. In [26], the authors routed the flows with SFC requests by minimizing the OPEX cost and resource fragments. And only the end-to-end delay and CPU resource are taken into account, respectively, in [14], [28] and [29]. In RA-RA, all the resources including bandwidth, flow table entries and CPU and end-to-end delay are considered in the meantime, when solving DRP-SFC.

The off-line optimal algorithms to route flows with SFC requests are studied in [26] and [30]. With the help of CPLEX, the off-line optimal solution to route flows with SFC requests is realized in [26]. Guo et al. [30] studied a joint optimization of MiddleBox Selection and Routing (MBSR) problem. In order to solve MBSR, the authors formulated this problem as an integer programming model to maximize the throughput for a specified set of sessions with SFC requests in SDN network. Then a polynomial algorithm using the Markov approximation technique is proposed, which adjusts the selected middleboxes for sessions with SFC requests randomly and iterates to find the best result.

Nevertheless, most of the mentioned works neglect to achieve the differentiated routing for flows with SFC requests. In the paper, we take the flow features into consideration and formulate the DRP-SFC as a BIP model with the objective to minimize the resource consumption costs for flows with SFC requests. Then, the routing algorithm named RA-RA is proposed to solve the DRP-SFC in SDN and NFV-enabled network. To the best of our knowledge, this work is the first effort that not only manages to make efficient differentiated routing strategies for flows with SFC requests, but also achieves load balancing among multiple kinds of resources in SDN and NFV-enabled network.

3 System Model

3.1 Physical Network with VNF Instances

We consider the physical network as an undirected graph $G = (\mathcal{V}, \mathcal{L})$, where $\mathcal{V}$ and $\mathcal{L}$ indicate the node set and link set, respectively. $u, v \in \mathcal{V}$ are physical nodes. $uv \in \mathcal{L}$ stands for the physical link connecting the physical nodes u and v. There exist two kinds of nodes in the network. One is switch node that is responsible to forward data to neighbor nodes based on the control signals from SDN controller. And the other kind is function node which not only takes charge of information forwarding but also holds VNF instances to process flows with SFC requests. We define $\mathcal{V}_{fn} \subset \mathcal{V}$ as the set of function nodes and $\mathcal{V}_{sn} \subset \mathcal{V}$ as the set of switch nodes. $\mathcal{M}$ represents the set of all the VNF instances and $m \in \mathcal{M}$ represents the mth VNF instance deployed in the network.

In the paper, $SFCR_i$ is used to represent the SFC request of flow i. We use C_u^{ft} to symbolize the flow table capacity on node u. The ratio of remaining flow table entries on node u, when routing $SFCR_i$, is represented by $r_{i,u}^{ft}$. The CPU capacity on node u is C_u^{cpu}, and the bandwidth capacity on link uv is C_{uv}^{bw}. When routing $SFCR_i$, the $r_{i,u}^{cpu}$ represents the ratio of remaining CPU on node u, and $r_{i,uv}^{bw}$ stands for the ratio of remaining bandwidth on link uv.

It is worth noting that, VNF instances are only allowed to be deployed on function nodes, and switch nodes do not need to process the flows with SFC requests, so we neglect the CPU consumption on switch nodes. Moreover, as micro datacenters and cloud datacenters can serve as function nodes [18], [31], comparing with switch nodes, we do not consider the flow table consumption on function nodes as well.

3.2 Flows with SFC Requests

In an SDN and NFV-enabled network, the flows originating from users should always traverse a set of VNF instances concatenated in predefined orders to satisfy their demands. In the paper, we assume that all the flows are with SFC requests and each SFC request consists of an ingress node, an egress node and a series of VNF requests. We use a six-tuple to represent the SFC request of a flow. For $SFCR_i$ in Eq. (1), S_i represents the ingress node and T_i represents the egress node. The sequence of VNF requests of $SFCR_i$ is defined as Ω_i. $l = |\Omega_i|$ represents the length of $SFCR_i$, which indicates the total number of VNF requests of an SFC request. $\Omega_i(j)$ stands for the jth VNF request, $j = 1, 2, \ldots, l$. For $SFCR_i$, the bandwidth and CPU consumptions and the maximum tolerated delay are represented by F_i^{bw}, F_i^{cpu} and F_i^{delay}, respectively. However, for an SFC request, the assumption that the bandwidth and CPU consumptions on links and VNF instances are set as fixed and the same values, respectively, is just for concise, and it is easy to extend the following formulation to support distinctive ones

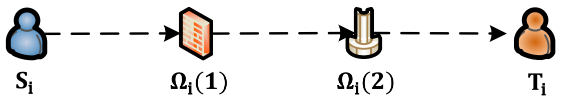


Fig. 2. An example of service function graph.

$$SFCR_i = \{S_i, T_i, \Omega_i, F_i^{bw}, F_i^{cpu}, F_i^{delay}\},$$
$$\Omega_i = \{\Omega_i(1), \Omega_i(2), \ldots, \Omega_i(l)\}, l = |\Omega_i|. \quad (1)$$

In the system model, the service function graph $\bar{G}_i = (\bar{\mathcal{V}}_i, \bar{\mathcal{L}}_i)$ is used to depict Ω_i. The service function graph is a digraph, where $\bar{\mathcal{V}}_i$ and $\bar{\mathcal{L}}_i$ represent the set of nodes and links, respectively. $\bar{u}, \bar{v} \subset \bar{\mathcal{V}}_i$ represent two nodes and $\bar{u}\bar{v} \subset \bar{\mathcal{L}}_i$ indicates the link connecting nodes $\bar{u}$ and $\bar{v}$ on $\bar{G}_i$. Fig. 2 shows a service function graph for $SFCR_i$, where S_i, T_i, $\Omega_i(1)$ and $\Omega_i(2)$ are nodes and $SFCR_i$ must traverse S_i, $\Omega_i(1)$ and $\Omega_i(2)$ in order before arriving at T_i.

3.3 Classification of Flows Based on Flow Features

For the purpose of cost-efficient Traffic Engineering (TE) in the network, it is important for ISPs to improve network performance and achieve load balancing with differentiated routing strategy based on flow features [15], [32], [33]. In the network, elephant flows carry the most of the traffic volume, while their number is small. Though mice flows are short-lived and carry a small number of packets, there exist large numbers of them, which also have important impact on the network performance and cannot be neglected [32]. Moreover, considering the computation consumption, the computationally intensive workload is defined in [33] which consumes lots of computation resources on servers but requires small bandwidth during data transmission. According to these flow features, several values of thresholds have been proposed to differentiate flows in the network [34].

However, as the data plan and control plan are coupled and integrated in today's network architecture, existing TE technologies are prevented to achieve truly differentiated services to adapt to uneven and high variable traffic patterns [35]. On the contrary, in SDN, the controller can achieve centralized network monitoring and management. Based on the techniques of packet-based sampling, flow statistics, hardware/software modification and data stream mining [36], [37], it is hopeful to efficiently obtain flow features in SDN and NFV-enabled network to deal with increasingly heterogeneous traffic with fine-grained scheduling at flow level.

In the paper, we take the resource preferences of flows as flow features and classify the flows with SFC requests into different kinds. On the perspective of bandwidth preference, all the flows with SFC requests can be divided into three kinds which are mice flow, dog flow, and elephant flow. Mice flows are such flows that are short-lived and consume little bandwidth. Elephant flows are on the contrary, which are long-lived and consume large amounts of bandwidth. And dog flows are defined between mice flows and elephant flows. On the perspective of CPU preference, all the flows are divided into two kinds which are computationally sparse flow and computationally dense flow. Therefore, based on the consumptions of bandwidth and CPU, there are six kinds of flows in the network, which are computationally sparse mice flow, computationally sparse dog flow, computationally sparse elephant flow, computationally dense mice flow, computationally dense dog flow and computationally dense elephant flow.

4 Problem Formulation

In this section, we first give the definition of relative cost, then formulate the DRP-SFC as a BIP model.

4.1 Definition of Relative Cost

In our work, the relative costs are defined to indicate the resource conditions in the network. $v_{i,uv}^{bw}$, $v_{i,u}^{ft}$ and $v_{i,u}^{cpu}$ represent the relative costs of bandwidth, flow table and CPU, when routing $SFCR_i$, respectively. The relative costs have reciprocal relationships to the remaining resources. For example, in Eq. (2a), the numerator represents the maximum bandwidth capacity in the network, and the denominator represents the difference between the remaining bandwidth on link uv and the bandwidth consumption of $SFCR_i$. The value range of the relative cost on each link is uniform between $(1, +\infty)$. According to the form of Eq. (2a), the less bandwidth remains on a link, the bigger relative cost of the link will be. And the relative cost grows very fast, if the remaining bandwidth on a link approaches zero. Eqs. (3a) and (4a) abide by the similar rule of Eq. (2a). Therefore, a link or node can be determined as a bottleneck, if its relative cost is big.

$$v_{i,uv}^{bw} = \begin{cases} \dfrac{\max\limits_{uv\in\mathcal{L}} C_{uv}^{bw}}{r_{i,uv}^{bw} C_{uv}^{bw} - F_i^{bw}} & F_i^{bw} > \mu, \quad (2a) \\ 0 & F_i^{bw} \le \mu. \quad (2b) \end{cases}$$

$$v_{i,u}^{ft} = \begin{cases} \dfrac{\max\limits_{u\in\mathcal{V}_{sn}} C_{u}^{ft}}{r_{i,u}^{ft} C_{u}^{ft} - 1} & F_i^{bw} < \nu, \quad (3a) \\ 0 & F_i^{bw} \ge \nu. \quad (3b) \end{cases}$$

$$v_{i,u}^{cpu} = \begin{cases} \dfrac{\max\limits_{u\in\mathcal{V}_{fn}} C_{u}^{cpu}}{r_{i,u}^{cpu} C_{u}^{cpu} - F_i^{cpu}} & F_i^{cpu} > \omega, \quad (4a) \\ 0 & F_i^{cpu} \le \omega. \quad (4b) \end{cases}$$

Additionally, flow features are considered in the definition of relative cost and the thresholds μ, ν and ω are used to differentiate the flows with SFC requests. The flows with bandwidth consumption larger than ν are set as elephant flows, and the flows with bandwidth consumption smaller than μ are set as mice flows ($\mu < \nu$). A flow is set as dog flow, if the bandwidth consumption is between μ and ν. The flows are set as computationally dense flows, if the CPU consumption is larger than ω. And the flows with CPU consumption smaller than ω are set as computationally sparse flows.

Since the bandwidth consumption of elephant flows is huge and the number of them is small, it is better to pay more attention on the remaining bandwidth rather than the remaining flow table entries on switch nodes. On the contrary, due to the fact that the bandwidth consumption of mice flows is negligible but the number of them is large, we should pay more attention on the remaining flow table entries on switch nodes, rather than the remaining bandwidth on links. Therefore, for elephant flows, the relative costs of bandwidth are calculated according to Eq. (2a) and

TABLE 1
Notations

Physical Network	
$G = (\mathcal{V}, \mathcal{L})$	Network graph G with the sets of nodes $\mathcal{V}$ and links $\mathcal{L}$, $u, v \in \mathcal{V}$, $uv \in \mathcal{L}$.
$\mathcal{V}_{sn}$, $\mathcal{V}_{fn}$	Sets of switch nodes and function nodes $\mathcal{V} = \mathcal{V}_{sn} \cup \mathcal{V}_{fn}$.
C^{bw}_{uv} ,C^{ft}_{u}, C^{cpu}_{u}	Capacities of bandwidth, flow table and CPU.
$r^{bw}_{i,uv}$, $r^{ft}_{i,u}$, $r^{cpu}_{i,u}$	Ratios of remaining bandwidth, flow table entries and CPU, when routing $SFCR_i$.
$v^{bw}_{i,uv}$, $v^{ft}_{i,u}$, $v^{cpu}_{i,u}$	Relative costs of bandwidth, flow table and CPU, when routing $SFCR_i$.
$d_{i,uv}$	Delay on $uv \in \mathcal{L}$, when routing $SFCR_i$.
$\mathcal{M}$	Set of VNF instances in the network, $m \in \mathcal{M}$.
Service Function Chains	
$\bar{G}_i = (\bar{\mathcal{V}}_i, \bar{\mathcal{L}}_i)$	Service function graph $\bar{G}_i$ with the sets of nodes $\bar{\mathcal{V}}_i$ and links $\bar{\mathcal{L}}_i$ of $SFCR_i$, $\bar{u}, \bar{v} \in \bar{\mathcal{V}}_i$, $\bar{u}\bar{v} \in \bar{\mathcal{L}}_i$.
S_i, T_i	The ingress node and egress node of $SFCR_i$.
Ω_i	The sequence of VNF requests of $SFCR_i$; $\Omega_i = \{\Omega_i(1), \Omega_i(2), \ldots, \Omega_i(l)\}$, $l = \lvert\Omega_i\rvert$.
F^{bw}_i, F^{cpu}_i, F^{delay}_i	The bandwidth and CPU consumptions and maximum tolerated delay of $SFCR_i$.
Binary Variables	
$x^{\bar{u}}_{i,m}$	Whether $\bar{u}$ is served by $m \in \mathcal{M}$ for $SFCR_i$.
y^{m}_{u}	Whether $m \in \mathcal{M}$ is hosted on $u \in \mathcal{V}_{fn}$.
$z^{\bar{u}\bar{v}}_{i,uv}$	Whether $\bar{u}\bar{v} \in \bar{\mathcal{L}}_i$ traverses $uv \in \mathcal{L}$ for $SFCR_i$.
$z^{\bar{u}\bar{v}}_{i,u}$	Whether $\bar{u}\bar{v} \in \bar{\mathcal{L}}_i$ traverses $u \in \mathcal{V}$ for $SFCR_i$.

the relative costs of flow table entries are set as 0 due to Eq. (3b). And, for mice flows, the relative costs of bandwidth are set as 0 due to Eq. (2b) and the relative costs of flow table entries are calculated according to Eq. (3a). For dog flows, because the bandwidth consumption and the number of them are medium-sized, both the bandwidth and flow table entries should be taken into account, and the relative costs are computed according to Eqs. (2a) and (3a), respectively.

As the CPU consumption of computationally dense flows is much larger than computationally sparse flows', we should pay more attention to the CPU consumption of computationally dense flows. Therefore, we compute the relative costs of CPU consumption for computationally dense flows according to Eq. (4a), and set it as 0 due to Eq. (4b) for computationally sparse flows.

4.2 BIP Formulation

Next, we formulate the DRP-SFC in detail and the notations used in this part are described in Table 1.

The binary variable $x^{\bar{u}}_{i,m}$ represents whether VNF request $\bar{u} \in \bar{\mathcal{V}}_i$ is served by VNF instance $m \in \mathcal{M}$.

$$x^{\bar{u}}_{i,m} = \begin{cases} 1, & \bar{u} \text{ is served by VNF instance } m, \\ 0, & \text{otherwise.} \end{cases} \tag{5}$$

We use binary variable y^m_u to represent whether VNF instance $m \in \mathcal{M}$ is hosted on function node $u \in \mathcal{V}_{fn}$.

$$y^m_u = \begin{cases} 1, & \text{VNF instance } m \text{ is hosted on } u, \\ 0, & \text{otherwise.} \end{cases} \tag{6}$$

The next two binary variables represent whether $\bar{u}\bar{v} \in \bar{\mathcal{L}}_i$ traverses link $uv \in \mathcal{L}$ or node $u \in \mathcal{V}$, respectively:

$$z^{\bar{u}\bar{v}}_{i,uv} = \begin{cases} 1, & \bar{u}\bar{v} \text{ traverses link } uv, \\ 0, & \text{otherwise.} \end{cases} \tag{7}$$

$$z^{\bar{u}\bar{v}}_{i,u} = \begin{cases} 1, & \bar{u}\bar{v} \text{ traverses node } u, \\ 0, & \text{otherwise.} \end{cases} \tag{8}$$

For link uv, the bandwidth consumption of $SFCR_i$ cannot exceed the remaining bandwidth on physical links.

$$\sum_{\bar{u}\bar{v} \in \bar{\mathcal{L}}_i} F^{bw}_i z^{\bar{u}\bar{v}}_{i,uv} \le r^{bw}_{i,uv} C^{bw}_{uv}, \forall uv \in \mathcal{L}. \tag{9}$$

The flow table consumption of $SFCR_i$ cannot exceed the remaining flow table entries on physical nodes, so the following constraint must be satisfied:

$$\sum_{\bar{u}\bar{v} \in \bar{\mathcal{L}}_i} z^{\bar{u}\bar{v}}_{i,u} \le r^{ft}_{i,u} C^{ft}_{u}, \forall u \in \mathcal{V}_{sn}. \tag{10}$$

Besides, all the CPU consumption of $SFCR_i$ cannot exceed the remaining CPU on the selected function nodes.

$$\sum_{\bar{u} \in \bar{\mathcal{V}}_i} \sum_{m \in \mathcal{M}} F^{cpu}_i x^{\bar{u}}_{i,m} y^m_u \le r^{cpu}_{i,u} C^{cpu}_{u}, \forall u \in \mathcal{V}_{fn}. \tag{11}$$

For $SFCR_i$, the end-to-end delay must be smaller than the maximum tolerated delay.

$$\sum_{uv \in \mathcal{L}} \sum_{\bar{u}\bar{v} \in \bar{\mathcal{L}}_i} d_{i,uv} z^{\bar{u}\bar{v}}_{i,uv} \le F^{delay}_i. \tag{12}$$

Once a link is selected, both the end points of this link must be selected as well.

$$z^{\bar{u}\bar{v}}_{i,u} z^{\bar{u}\bar{v}}_{i,v} = \begin{cases} 1, & z^{\bar{u}\bar{v}}_{i,uv} = 1, \forall u, v \in \mathcal{V}, \forall uv \in \mathcal{L}, \forall \bar{u}\bar{v} \in \bar{\mathcal{L}}_i, \\ 0, & \text{otherwise.} \end{cases} \tag{13}$$

Eq. (14) ensures that the routing path of $SFCR_i$ is consecutive and cannot be split.

$$\sum_{v \in \mathcal{V}} \sum_{\bar{u}\bar{v} \in \bar{\mathcal{L}}_i} (z^{\bar{u}\bar{v}}_{i,uv} - z^{\bar{u}\bar{v}}_{i,vu}) = \begin{cases} 1, & u = S_i, \\ -1, & u = T_i, \\ 0, & \text{otherwise.} \end{cases} \tag{14}$$

In order to ensure that the routing path traverses all the function nodes which contain the selected VNF instances for $SFCR_i$, the next equation must be satisfied:

$$\begin{gathered} x^{\bar{u}}_{i,m} y^m_u \le z^{\bar{u}\bar{v}}_{i,u}, \forall u \in \mathcal{V}_{fn}, \\ \forall \bar{u} \in \bar{\mathcal{V}}_i, \forall \bar{u}\bar{v} \in \bar{\mathcal{L}}_i, \forall m \in \mathcal{M}. \end{gathered} \tag{15}$$

Eq. (16) is used to ensure that each VNF request $\bar{u} \in \bar{\mathcal{V}}_i \backslash \{S_i, T_i\}$ on $\bar{G}_i$ can only be served by one VNF instance.

$$\sum_{m \in \mathcal{M}} x^{\bar{u}}_{i,m} = \begin{cases} 1, & \bar{u} \text{ is served by VNF instance } m, \\ 0, & \text{otherwise.} \end{cases} \tag{16}$$

Due to the fact that each VNF instance can only be hosted on one function node, then the next uniqueness constraint is satisfied:

$$\sum_{u\in\mathcal{V}_{fn}} y_u^m = 1, \forall m \in \mathcal{M}. \tag{17}$$

In the paper, we use the relative costs $v_{i,uv}^{bw}$, $v_{i,u}^{ft}$ and $v_{i,u}^{cpu}$ to evaluate whether the links and nodes are congested and to route flows with SFC requests based on flow features differentiatedly. The resource consumption cost of the routing path for $SFCR_i$ is calculated as below:

$$\begin{aligned}\mathbb{R} = &\sum_{uv\in\mathcal{L}}\sum_{\bar{u}\bar{v}\in\bar{\mathcal{L}}_i} v_{i,uv}^{bw} z_{i,uv}^{\bar{u}\bar{v}} + \sum_{u\in\mathcal{V}_{sn}}\sum_{\bar{u}\bar{v}\in\bar{\mathcal{L}}_i} v_{i,u}^{ft} z_{i,u}^{\bar{u}\bar{v}} \\ &+ \sum_{u'\in\mathcal{V}_{fn}}\sum_{\bar{u}\in\bar{\mathcal{V}}_i}\sum_{m\in\mathcal{M}} v_{i,u'}^{cpu} x_{i,m}^{\bar{u}} y_{u'}^{m}.\end{aligned} \tag{18}$$

As the relative costs are used to indicate the resource conditions on links and nodes, we also use the sum of relative costs to indicate the resource condition of a path. If the resource consumption cost of a path is small, we can determine that there are abundant remaining resources and small number of bottleneck links or nodes on this path, and routing flows with SFC requests on this path will not incur network congestion. On the contrary, if the resource consumption cost of a path is very big, we can determine that there exist bottleneck links or nodes on this path, and another path with smaller resource consumption cost should be found to avoid network congestion and achieve load balancing. Therefore, our objective in Eq. (19) is to minimize the resource consumption costs for flows with SFC requests under the constraints of Eqs. (9)–(17).

$$\begin{aligned}&\mathcal{M}inimize\ \mathbb{R} \\ &s.t.\ Eq.(9)-(17).\end{aligned} \tag{19}$$

5 Proposed Algorithm

In this section, we propose a novel routing algorithm named RA-RA. RA-RA can efficiently solve the DRP-SFC by transforming the original network graph to Logical Function Graph (LFG). When executing RA-RA, we first need to calculate the relative costs on the links, switch nodes and function nodes. Based on the relative costs and the sequence of VNF requests, we search for the candidate VNF instances in the network and arrange them in order to construct the LFG. LFG is a digraph, where each path from the ingress node to egress node along with the link direction satisfies the predefined order. Finally, the routing paths for flows with SFC requests are obtained based on a modified k-shortest path algorithm.

5.1 Constructing LFG

In the paper, we solve the DRP-SFC by constructing LFG. LFG is a digraph which is comprised of the ingress node, egress node and candidate VNF instances for a flow with SFC request. Then, we construct the LFG for $SFCR_i$ according to the original network graph. The LFG is denoted as $\widehat{G}_i = (\widehat{\mathcal{V}}_i, \widehat{\mathcal{L}}_i)$. $\widehat{u}$, $\widehat{v} \in \widehat{\mathcal{V}}_i$ represent two nodes and $\widehat{uv} \in \widehat{\mathcal{L}}_i$ stands for a link on LFG. When constructing LFG, the first process is to find the ingress node, egress node and all the candidate VNF instances and arrange them as the predefined order of $SFCR_i$. Above all, the ingress node S_i is selected and placed in the 1st column. Then, due to the sequence of VNF request $\Omega_i = \{\Omega_i(1), \Omega_i(2), \ldots, \Omega_i(l)\}, l = |\Omega_i|$, the VNF instances belonging to the same type of $\Omega_i(1)$ are selected and placed in the 2nd column. Next, we execute the same operation sequentially to the VNF instances which belong to the same type of $\Omega_i(2)$ to $\Omega_i(l)$. Finally, the egress node T_i is found and placed in the $(l+2)$th column. When finishing the first process, the second process is to produce the links on LFG. For each two adjacent columns, we connect each node in the last column to all the nodes in the next column. And the link direction is from the node in the last column to the node in the next column.

On the LFG, each link is corresponding to a path which is obtained by executing SP algorithm such as Dijkstra on network graph. Here, the relative costs defined in Eqs. (2a)-(4b) are set as the costs on physical links and nodes. For two VNF instances hosted on different function nodes, a flow with SFC request needs to traverse a complete path to pass through these two VNF instances. Therefore, the link connecting the VNF instances on different function nodes is corresponding to a path on network graph. Nevertheless, because an function node is allowed to deploy multiple VNF instances, for the SFC request served by the VNF instances on one function node, the link will be corresponding to a function node on the network graph.

According to the relationship between network graph and LFG, the relative costs of bandwidth, flow table and CPU on network graph are all transformed to the links on LFG. For $SFCR_i$, the link cost of $\widehat{uv}$ is set as $v_{i,\widehat{uv}}$. The binary variables, $z_{i,uv}^{\widehat{uv}}$ and $z_{i,u}^{\widehat{uv}}$, indicate whether $\widehat{uv} \in \widehat{\mathcal{L}}_i$ traverses $uv \in \mathcal{L}$ and $u \in \mathcal{V}$, respectively. $z_{i,uv}^{\widehat{uv}} = 1$, if $\widehat{uv} \in \widehat{\mathcal{L}}_i$ traverses $uv \in \mathcal{L}$ and $z_{i,u}^{\widehat{uv}} = 1$, if $\widehat{uv} \in \widehat{\mathcal{L}}_i$ traverses $u \in \mathcal{V}$. In addition, an mapping function $\Pi(\cdot)$ is used to obtain the function node that a VNF instance is deployed on and $v_{i,\Pi(\widehat{u})}^{cpu}$ indicates the relative cost of CPU on the function node $\Pi(\widehat{u})$. Eq. (20) shows the calculation of $v_{i,\widehat{uv}}$, where the three parts in Eq. (20) represent the relative costs of bandwidth, flow table and CPU, respectively:

$$\begin{aligned}v_{i,\widehat{uv}} = &\sum_{uv\in\mathcal{L}} v_{i,uv}^{bw} z_{i,uv}^{\widehat{uv}} + \sum_{u\in\mathcal{V}_{sn}} v_{i,u}^{ft} z_{i,u}^{\widehat{uv}} \\ &+ \frac{v_{i,\Pi(\widehat{u})}^{cpu} + v_{i,\Pi(\widehat{v})}^{cpu}}{2}, \forall \widehat{uv} \in \widehat{\mathcal{L}}_i, \forall \widehat{u}, \widehat{v} \in \widehat{\mathcal{V}}_i\end{aligned} \tag{20}$$

On LFG, the path with the minimum resource consumption cost is calculated based on Eq. (21). $z_{i,\widehat{uv}}^{\bar{u}\bar{v}}$ is a binary variable and represents whether $\bar{u}\bar{v} \in \bar{\mathcal{L}}_i$ on $\bar{G}_i$ traverses $\widehat{uv} \in \widehat{\mathcal{L}}_i$ on $\widehat{G}_i$. And $z_{i,\widehat{uv}}^{\bar{u}\bar{v}} = 1$, only when $\bar{u}\bar{v} \in \bar{\mathcal{L}}_i$ traverses $\widehat{uv} \in \widehat{\mathcal{L}}_i$.

$$\mathcal{M}inimize \sum_{\bar{u}\bar{v}\in\bar{\mathcal{L}}_i}\sum_{\widehat{uv}\in\widehat{\mathcal{L}}_i} v_{i,\widehat{uv}}\, z_{i,\widehat{uv}}^{\bar{u}\bar{v}}. \tag{21}$$

As for LFG, all the candidate VNF instances are arranged in predefined order. Then, each path from the ingress node to egress node satisfies the demand of SFC request. If the path derived from Eq. (21) satisfies all the constraints of Eqs. (9)–(17), we will get the final solution to route $SFCR_i$ by transforming this path to network graph.

For example, Fig. 3 shows the LFG for $SFCR_i$. In the figure, the flow starts from the ingress node A and needs to traverse

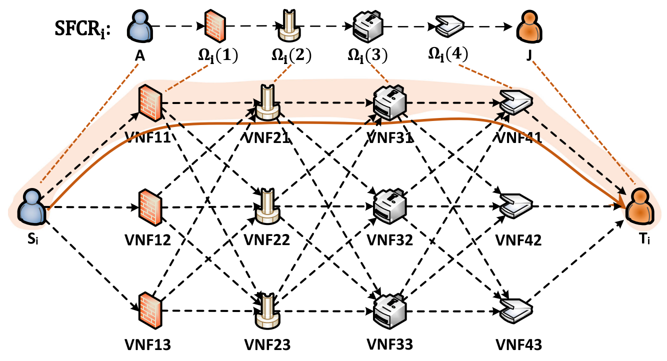


Fig. 3. Solving DRP-SFC on LFG.

the instances of $\Omega_i(1)$, $\Omega_i(2)$, $\Omega_i(3)$, and $\Omega_i(4)$ in order before arriving at node J. Assuming that the path with the minimum resource consumption cost calculated in Eq. (21) is $A \rightarrow VNF11 \rightarrow VNF21 \rightarrow VNF31 \rightarrow VNF41 \rightarrow J$, then due to Fig. 1, we know that $SFCR_i$ will be processed by $VNF11$ and $VNF21$ on B and $VNF31$ and $VNF41$ on E. On network graph, if the path derived from LFG is $A \rightarrow B \rightarrow VNF11 \rightarrow VNF21 \rightarrow E \rightarrow VNF31 \rightarrow VNF41 \rightarrow G \rightarrow J$ and all the constraints of Eqs. (9)–(17) are satisfied, then this path is selected to route $SFCR_i$.

Algorithm 1. RA-RA

```
1: Input: Network graph: $G = (\mathcal{V}, \mathcal{L})$,
       Resource capacities: $C^{bw}_{uv}$, $C^{ft}_{u}$, $C^{cpu}_{u}$,
       Remaining ratios: $r^{bw}_{i,uv}$, $r^{ft}_{i,u}$, $r^{cpu}_{i,u}$,
       Flow with SFC request: $SFCR_i$,
       Thresholds: $\mu$, $\nu$, $\omega$,
       Iteration times: $K$;
2: Output: $\Phi$;
3: Initialize $k = 1$;
4: Set $\Phi$ and $\widehat{\Phi}$ as ø;
5: Remove all the links and nodes with less resources to serve
   $SFCR_i$;
6: Find $\max_{uv\in\mathcal{L}} C^{bw}_{uv}$, $\max_{u\in\mathcal{V}_{sn}} C^{ft}_{u}$ and $\max_{u\in\mathcal{V}_{fn}} C^{cpu}_{u}$ on $G$;
7: Calculate $v^{bw}_{i,uv}$, $v^{ft}_{i,u}$, $v^{cpu}_{i,u}$ based on Eqs. (2a)-(4b);
8: Construct LFG; ⇒ Function 1
9: while $k \leq K$ do
10:    $\widehat{\Phi}$ = Calculate the $k$th shortest path on LFG based on
       Eq. (21);
11:    if~Isempty($\widehat{\Phi}$) then
12:        $\Phi$ = Transform $\widehat{\Phi}$ from $\widehat{G}_i$ to $G$;
13:        if $\Phi$ satisfies all the constraints of Eqs. (9)–(17) then
14:            Receive $SFCR_i$;
15:            Update $r^{bw}_{i,uv}$, $r^{ft}_{i,u}$ and $r^{cpu}_{i,u}$ ;
16:            return $\Phi$;
17:        end
18:    else
19:        return Routing Failed;
20:    end
21:    $k = k + 1$;
22: end
23: return Routing Failed;
```

On LFG, since the resource consumption cost is used to indicate the resource condition of a path, we can achieve load balancing and avoid bottleneck and congestion by finding the path with the minimum resource consumption cost in Eq. (21) for each flow with SFC request. Additionally, noting that, on LFG, there is only one path for each combination of VNF instances which can satisfy the demand of the SFC request, the path set of LFG is only a subset of original network graph. Therefore, LFG provides a simplified view of the network topology, which makes the path computation efficient by eliminating sub-optimal solutions.

Function 1. Construct LFG

```
1: Input: Network graph: $G = (\mathcal{V}, \mathcal{L})$,
       Relative costs: $v^{bw}_{i,uv}$, $v^{ft}_{i,u}$, $v^{cpu}_{i,u}$,
       Flow with SFC request: $SFCR_i$;
2: Output: $\widehat{G}_i = (\widehat{\mathcal{V}}_i, \widehat{\mathcal{L}}_i)$;
3: for $j = 1 : l$ do
4:     $\eta(j+1)$ = Find the VNF instances belonging to the same
       type of $\Omega_i(j)$ on $G$;
5: end
6: $\eta(1) = S_i$;
7: $\eta(l+2) = T_i$;
8: for $j = 1 : (l+1)$ do
9:     for $\widehat{u} \in \eta(j)$ do
10:        for $\widehat{v} \in \eta(j+1)$ do
11:            Calculate $v_{i,\widehat{uv}}$ based on Eq. (20); ⇒ Dijkstra
12:        end
13:    end
14: end
15: Construct $\widehat{G}_i = (\widehat{\mathcal{V}}_i, \widehat{\mathcal{L}}_i)$ based on $v_{i,\widehat{uv}}$, $\widehat{uv} \in \widehat{\mathcal{L}}_i$;
16: return $\widehat{G}_i = (\widehat{\mathcal{V}}_i, \widehat{\mathcal{L}}_i)$;
```

5.2 RA-RA Routing Algorithm

RA-RA provides an efficient way to achieve load balancing and differentiated routing for flows with SFC requests. There are two steps included in RA-RA. The first step is to construct the LFG according to the relative costs, and the second step is to run a modified k-shortest path algorithm on LFG to find the path that has the minimum resource consumption cost and satisfies all the constraints in Eqs. (9)–(17) for each flow with SFC request.

The pseudocode of RA-RA is described below. First, we initialize the current iteration times k and empty Φ and $\widehat{\Phi}$ which are used to record the paths on network graph and LFG, respectively (lines 3-4 in Algorithm 1). Then, all the nodes and links with less resources to serve $SFCR_i$ are removed from the network graph (line 5 in Algorithm 1). Next, we compute the maximum capacities of bandwidth, flow table and CPU in the network (line 6 in Algorithm 1). After that, we calculate the values of $v^{bw}_{i,uv}$, $v^{ft}_{i,u}$ and $v^{cpu}_{i,u}$ which represent the relative costs of bandwidth, flow table and CPU (line 7 in Algorithm 1).

Function 1 presents the construction of LFG based on the relative costs $v^{bw}_{i,uv}$, $v^{ft}_{i,u}$ and $v^{cpu}_{i,u}$. The VNF instances belonging to the same type of Ω_i are recorded in η sequentially (lines 3-5 in Function 1). According to LFG, we add the ingress node S_i and egress node T_i to $\eta(1)$ and $\eta(l+2)$ (lines 6-7 in Function 1). Then, the SP algorithm, Dijkstra, is used to produce the links for LFG (lines 8-14 in Function 1). The

LFG is constructed based on the values of $v_{i,\widehat{uv}}$ (line 15 in Function 1). And we return the LFG in line 16 of Function 1.

We modify the k-shortest path algorithm to get the solution on LFG (lines 9-22 in Algorithm 1). K represents the maximum iteration times. We first compare whether the current iteration times k is bigger than K (line 9 in Algorithm 1). If $k \leq K$, the path with the first minimum resource consumption cost is calculated on LFG and if found, this candidate path will be recorded in $\widehat{\Phi}$ (line 10 in Algorithm 1). Then we check whether $\widehat{\Phi}$ is empty (line 11 in Algorithm 1). If there is candidate path in $\widehat{\Phi}$, this path will be transformed from LFG to network graph and recorded in Φ (line 12 in Algorithm 1). Next, the path recorded in Φ will be checked whether it satisfies all the constraints of Eqs. (9)–(17) (line 13 in Algorithm 1). If all the constraints are satisfied, this flow will be received and routed by the path recorded in Φ (line 14 in Algorithm 1). After that, we update the ratios of remaining bandwidth, flow table entries and CPU on links, switch nodes and function nodes, respectively (line 15 in Algorithm 1). If there is no available path derived from LFG, this flow is denied to be served (line 19 in Algorithm 1). If the path with the first minimum resource consumption cost returned from LFG cannot satisfy all the constraints of Eqs. (9)–(17), then we set $k = k + 1$ to find the path with the next minimum resource consumption cost on LFG (line 21 in Algorithm 1). The RA-RA will stop and deny this flow until k exceeds the maximum iteration times K (line 23 in Algorithm 1).

5.3 Complexity Analysis

In this section, we give a detailed complexity analysis of RA-RA in the worst situation.

When executing RA-RA, first, we need to calculate the relative costs on links and nodes and the complexity is $O(|\mathcal{V}| + |\mathcal{L}|)$. In the DRP-SFC, the worst situation is that each function node deploys all types of VNF instances. Under this circumstance, we need to execute the SP algorithm at most $\frac{1}{2}|\mathcal{V}_{fn}|^2 + 2|\mathcal{V}_{fn}|$ times to produce all the links on LFG. The complexity of the SP algorithm on $G = (\mathcal{V}, \mathcal{L})$ is $O(|\mathcal{L}| + |\mathcal{V}|log|\mathcal{V}|)$, then the complexity of constructing LFG results in $O(|\mathcal{V}_{fn}|^2(|\mathcal{L}| + |\mathcal{V}|log|\mathcal{V}|))$. On LFG, there are at most $l|\mathcal{V}_{fn}| + 2$ nodes and $(l-1)|\mathcal{V}_{fn}|^2 + 2|\mathcal{V}_{fn}|$ links, then the worst situation is to iterate K times to get the solution, which runs in $O(Kl^2|\mathcal{V}_{fn}|^2(|\mathcal{V}_{fn}| + logl))$. The complexity of path transformation and resource update is $O(1)$. Because l, which represents the length of SFC request, is a finite and small value, then the complexity to solve the DRP-SFC by RA-RA at the worst situation is $O(Kl^2|\mathcal{V}_{fn}|^3 + |\mathcal{V}_{fn}|^2(|\mathcal{L}| + |\mathcal{V}|log|\mathcal{V}|))$.

6 Performance Evaluation

This section depicts the simulation settings and the performance comparison between the RA-RA and existing algorithms including the COATS [25], SP [29] and Eigen-decomposition [24] algorithms. All the algorithms are implemented with MATLAB 2016a and performed on a computer with Intel(R) Core(TM) i7-4790 CPU 3.60 GHz and 32 GB RAM.

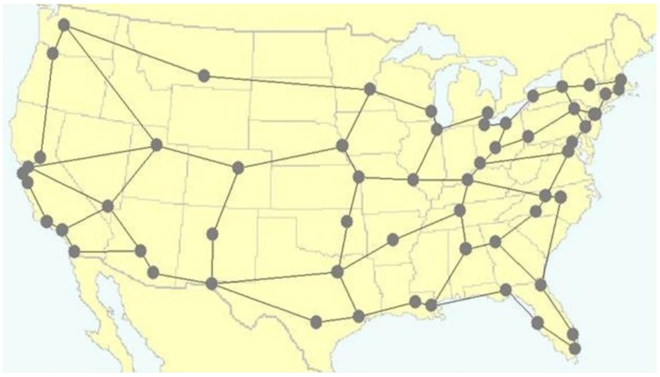


Fig. 4. CORONET CONUS topology.

6.1 Simulation Settings

6.1.1 Topology Settings

The network graph we use is a US carrier networks topology named CORONET CONUS Topology (shown in Fig. 4), which composes of 60 nodes and 79 links [30], [38]. In the network, we select function nodes based on the node degree. All the nodes are sorted in descending order according to the node degree and the top 30 percent of nodes are set as function nodes. There are 20 types of VNF instances deployed in the network. And we deploy 8 types of VNF instances on each function node. Therefore, there are 18 function nodes and 144 VNF instances belonging to 20 different VNF types [39] in the network.

The bandwidth capacity of each link is set as 1,200 Mbps [30]. The capacities of flow table and CPU on the switch nodes and function nodes are set as 800 units [40] and $8{,}000$ MIPS [39], respectively. In the simulation, $d_{i,uv}$ is calculated according to Eq. (22), where the first part represents the queuing delay and the second part d_{uv}^{prop} represents the propagation delay on link uv [28]. d_{uv}^{tx} stands for the transmission delay on link uv and we set it as 10 μs [41]. The propagation delay d_{uv}^{prop} is calculated according to the length between the nodes u and v. All the network parameters used in this paper are described in Table 2.

$$d_{i,uv} = \frac{1 - r_{i,uv}^{bw}}{r_{i,uv}^{bw}} d_{uv}^{tx} + d_{uv}^{prop}, \forall uv \in \mathcal{L}. \tag{22}$$

6.1.2 SFC Request and Flow Distribution Settings

For each flow, we set F_i^{bw} as a constant which randomly falls in (0, 10] Mbps [14]. F_i^{cpu} is proportional with F_i^{bw}. The unit of F_i^{cpu} is MIPS and F_i^{cpu} equals the product between the value of F_i^{bw} and a constant distributed in 0 and 10 randomly [33]. F_i^{delay} is set between 50-100 ms [42].

We classify all the flows into six kinds based on the bandwidth and CPU consumptions [32], [33]. The distribution of flows satisfies the law of two to eight, where elephant flows account for 20 percent, mice flows account for 50 percent and dog flows account for 30 percent. The thresholds to differentiate flows with SFC requests are set as $\mu = 0.1$ Mbps, $\nu = 1$ Mbps, $\omega = 5$ MIPS. The bandwidth and CPU consumptions of these flows are set as below:

- computationally sparse mice flow: The CPU consumption is no more than 5 MIPS and the bandwidth consumption is between 0 and 0.1 Mbps.

TABLE 2
Simulation Parameter Settings

Description		Value
Network topology		CORONET CONUS Topology
Proportion of function node		30%
Proportion of mice, dog, and elephant flows		50%, 30% and 20%
Total VNF types		20
Number of VNF types per function node		8
Maximum iteration times of RA-RA		5
Parameters	**Description**	**Value**
C_{uv}^{bw}	Bandwidth capacity on link uv	1,200 Mbps
C_u^{ft}	Flow table capacity on switch node u	800 units
C_u^{cpu}	CPU capacity on function node u	8,000 MIPS
F_i^{bw}	Bandwidth consumption	(0, 10] Mbps
F_i^{cpu}	CPU consumption	(0, 100] MIPS
F_i^{delay}	Maximum toleration delay	[50, 100] ms
d_{uv}^{tx}	Transmission delay on link uv	10 μs
d_{uv}^{prop}	Propagation delay on link uv	defined on network topology
μ, ν, ω	Thresholds of flow classification	0.1 Mbps, 1 Mbps and 5 MIPS
$\lvert\mathcal{M}\rvert$	Total VNF instances	144
$\lvert\Omega_i\rvert$	Length of SFC request	4

- computationally sparse dog flow: The CPU consumption is no more than 5 MIPS and the bandwidth consumption is between 0.1 and 1 Mbps.
- computationally sparse elephant flow: The CPU consumption is no more than 5 MIPS and the bandwidth consumption is between 1 and 10 Mbps.
- computationally dense mice flow: The CPU consumption is more than 5 MIPS and the bandwidth consumption is between 0 and 0.1 Mbps.
- computationally dense dog flow: The CPU consumption is more than 5 MIPS and the bandwidth consumption is between 0.1 and 1 Mbps.
- computationally dense elephant flow: The CPU consumption is more than 5 MIPS and the bandwidth consumption is between 1 and 10 Mbps.

Moreover, the length of SFC request in this simulation is 4, and the maximum iteration times of RA-RA is 5. In the simulation, each experiment is repeated 20 times.

6.1.3 *Introduction of Comparing Algorithms*

In the simulation, the performance of RA-RA is compared with the COATS, SP and Eigendecomposition algorithms. SP is realized and integrated in the OpenDaylight platform which is the largest open source SDN controller to schedule flows with SFC requests in service function selection framework. It is worth noting that we use Eigen to represent the Eigendecomposition algorithm in the paper.

Before introducing the simulation results, we would like to give a brief description to these comparing algorithms.

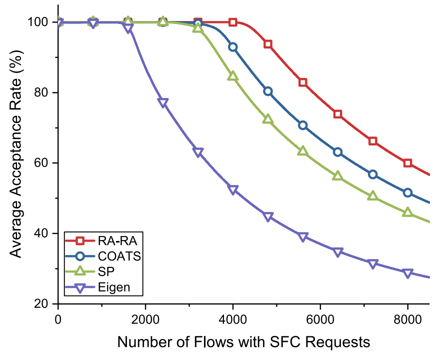
- COATS: Each link keeps a cost which is calculated based on the remaining bandwidth on the link. Then COATS constructs a layered graph and selects the path with the lowest cost to route a flow with SFC request.
- SP: The Dijkstra algorithm is used to select appropriate VNF instances and route a flow with SFC request from the ingress node to egress node passing through the selected VNF instances with the shortest path sequentially.
- Eigen: First, Eigen constructs the adjacent matrixes of SFC request and network topology, respectively. Then, the adjacent matrix of SFC request is extended with the same dimension of network topology's. Finally, Umeyama's eigendecomposition approach and widest-shortest path routing algorithm are used to compute the optimal matching between the SFC request and network topology.

6.2 Simulation Results

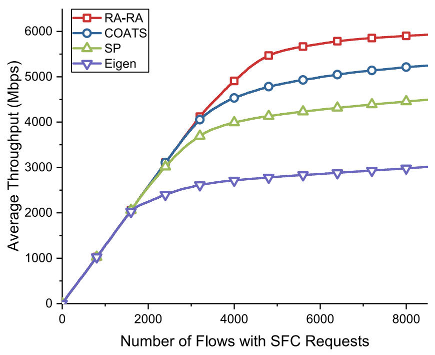
6.2.1 *Comparison of Average Acceptance Rate, Throughput and Hop Count*

In Fig. 5, we present the comparison between the RA-RA and COATS, SP and Eigen in terms of average acceptance rate, throughput and hop count.

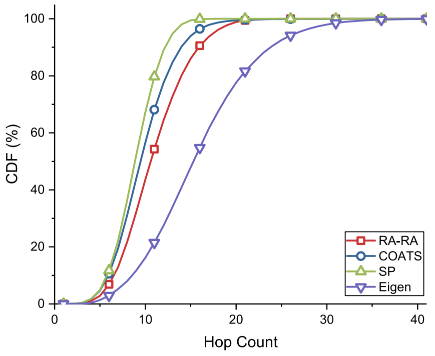
Fig. 5a shows the average acceptance rate of these algorithms. Average acceptance rate reflects the flows served by the network accounting for the total arrival ones. In the simulation, RA-RA performs the best, which gets about 10 percent higher in average acceptance rate than that of COATS. And the performance of COATS is about 5 percent higher than that of SP and 20 percent higher than that of Eigen. In RA-RA, we achieve load balancing among multiple kinds of resources and manage to route flows with SFC requests differentiatedly. Therefore, RA-RA does better than other comparing algorithms. As for COATS, on the one hand, it is a variation of the SP algorithm which is beneficial to reduce the consumptions of bandwidth and flow table in the network. On the other hand, COATS balances the consumption of bandwidth by defining the cost due to the remaining bandwidth on links. Comparing with the SP and Eigen, COATS can serve more flows in the network. However, for the reason that COATS



(a) Average acceptance rate

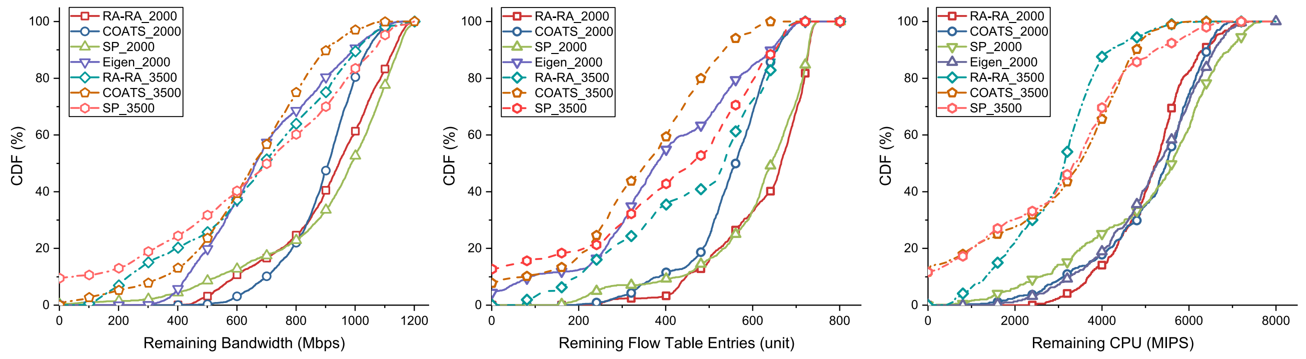


(b) Average throughput



(c) Average hop count

Fig. 5. The comparison of average acceptance rate, throughput and hop count.



(a) CDF of average remaining bandwidth (b) CDF of average remaining flow table entries (c) CDF of average remaining CPU

Fig. 6. CDF of average remaining bandwidth, flow table entries and CPU.

neglects the capacities of flow table and CPU, it will result in the resource exhaustion on the nodes with fewer resources, which leads to worse performance comparing with RA-RA. As for Eigen, Umeyama's eigendecomposition approach cannot ensure to get the optimal path and the widest-shortest path routing algorithm results in longer paths in the chaining solution, so there are more resource consumptions, which leads to the worst performance in the simulation.

Fig. 5b presents the average throughput of these four algorithms. The average throughput reflects the total bandwidth of flows received successfully in the network. In this part, the throughput of RA-RA is about 700 Mbps higher than that of COATS, 1,500 Mbps higher than that of SP and 3,000 Mbps higher than that of Eigen. Eigen results in the lowest throughput, because the longer paths consume more bandwidth and flow table entries. The performances of COATS and the SP are lower than that of RA-RA because of unbalanced utilization of the resources in the network. On the contrary, RA-RA realizes the load balancing and differentiated routing by fine-grained flow scheduling. Therefore, RA-RA can avoid bottlenecks on the links and nodes, which results in higher throughput comparing with other algorithms.

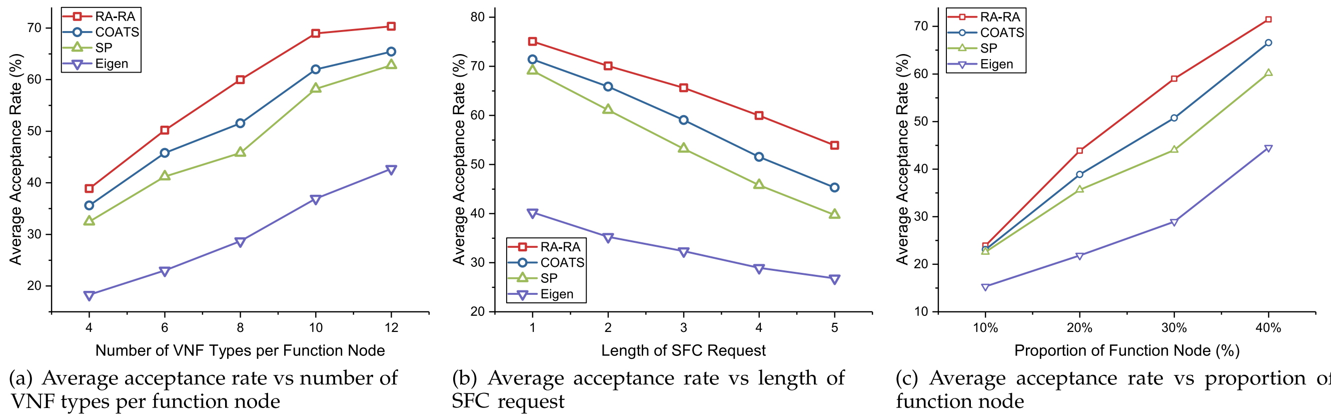
Fig. 5c presents the average hop count of these four algorithms. The average hop count represents the number of nodes a flow with SFC request needs to traverse before reaching the egress node. According to the result, SP performs the best, the COATS surpasses RA-RA and Eigen is the worst. Due to the fact that Eigen tends to route flows with SFC requests with long paths, the paths with hop counts distributing from 15 to 35 account for about 55 percent, which is only about 15 percent for RA-RA, 8 percent for COATS and 3 percent for SP. When routing flows with SFC requests, RA-RA takes bandwidth, flow table, CPU and flow features into consideration at the same time, so it incurs a little longer paths than COATS's. As SP always finds the paths with the shortest hop counts for flows, the performance of SP in this simulation is the best among other algorithms. In addition, from Fig. 5c, we get that the RA-RA, COATS and SP are prone to produce short paths, which are beneficial to reduce end-to-end delay and satisfy low-delay demands.

6.2.2 *Comparison of Average Remaining Bandwidth, Flow Table Entries and CPU*

The CDF curves in Fig. 6 present the comparison of these algorithms in terms of average remaining bandwidth, flow table entries and CPU.

Fig. 6a presents the CDF of the average remaining bandwidth on links when 2,000 and 3,500 flows with SFC requests are successfully received. The performance of COATS is higher than that of all the other algorithms. The performance of RA-RA outperforms that of SP, and Eigen falls to balance the bandwidth consumption on links. Because of longer paths and unbalanced resource utilization for Eigen, when receiving 2,000 flows, the proportion of the paths with remaining bandwidth less than 600 Mbps is about 37 percent, which is only 12 percent for SP, 10 percent for RA-RA and 3 percent for COATS. When receiving 3,500 flows, the proportion of bottleneck links of SP increases to 10 percent, while there are no bottleneck links for RA-RA and COATS. This is because SP fails to achieve load balancing on the bandwidth, which leads to network congestion. Nevertheless, when receiving 3,500 flows, for RA-RA, there are about 40 percent of links with the remaining bandwidth between 400 and 800 Mbps, while, for COATS, it is about 60 percent. Therefore, the distribution of remaining bandwidth for COATS on links is more balanced comparing with that of RA-RA. The reason is that, for COATS, only the bandwidth on links is considered when routing flows with SFC requests. And RA-RA needs to balance the consumptions of bandwidth, flow table and CPU on links and nodes, so COATS can get better performance in this simulation.

Fig. 6b illustrates the CDF of average remaining flow table entries on switch nodes. Here, the performance of RA-RA outperforms that of other algorithms and the performances of COATS and SP are both better than that of Eigen. Because of more resource consumption, when receiving 2,000 flows, there are about 5 percent of switch nodes becoming bottlenecks for Eigen. When receiving 3,500 flows, there are about 9 and 12 percent of switch nodes becoming bottlenecks for COATS and SP, while there is no switch node running out of flow table entries for RA-RA. Moreover, for COATS and SP, there are about 40 percent of switch nodes with remaining flow table entries fewer than 300 and 400 units, respectively, while the proportions for RA-RA are only about 22 and 35 percent. This is because the relative costs defined in Eqs. (2a)-(4b) can indicate the resource conditions on links and nodes. If the resources on links or nodes are going to be used up, the relative costs will increase quickly, which can protect them from being exhausted. So RA-RA can efficiently avoid congestion and achieve load balancing by minimizing the resource consumption costs for flows with SFC requests in the network.



(a) Average acceptance rate vs number of VNF types per function node

(b) Average acceptance rate vs length of SFC request

(c) Average acceptance rate vs proportion of function node

Fig. 7. Comparison of average acceptance rate in different scenarios.

Fig. 6c describes the CDF of average remaining CPU on function nodes. In this part, the performance of RA-RA surpasses that of other algorithms, while Eigen dose better than COATS and the SP performs the worst. When receiving 2,000 flows, there are no bottleneck nodes that are short of CPU resource for these four algorithms. However, when receiving 3,500 flows, the bottleneck nodes of COATS and SP increase obviously. This is because both the COATS and SP neglect to optimize the CPU on function nodes. Furthermore, for COATS and SP, the utilization of CPU on function nodes is unbalanced comparing with RA-RA. As shown in Fig. 6c, when receiving 3,500 flows, the proportion of function nodes of which remaining CPU are from 2,000 to 4,000 MIPS is about 70 percent for RA-RA, which is only about 35 percent for COATS and SP. Though, Eigen gets balanced distribution of CPU resource, the huge amounts of bandwidth and flow table consumptions lead to low network performance.

6.2.3 *Comparison of Average Acceptance Rate in Different Scenarios*

In Fig. 7, we change the number of VNF types per function node, the length of SFC requests and the proportion of function node to compare the average acceptance rate of RA-RA with comparing algorithms'. The average acceptance rate is evaluated under 8,000 flows with SFC requests.

Fig. 7c shows the average acceptance rate under different number of VNF types per function node. In the simulation, the average acceptance rates of these tested algorithms grow quickly when increasing the number of VNF types per function node. This is because, when the number of VNF types on function nodes increases, it is more possible for SFC requests to be served by one function node instead of being split on the VNF instances distributed on several function nodes. Therefore, it is beneficial for the reduction of bandwidth and flow table consumptions by increasing the number of VNF types per function node. In addition, for RA-RA, COATS and SP, the average acceptance rates grow slowly, when increasing the number of VNF types per function node from 10 to 12. The reason is that, comparing with the bandwidth and flow table entries, CPU on function nodes becomes scarce which prevents the network from receiving flows with SFC requests. This can be proven by the curve of Eigen. Generally, comparing with CPU on function nodes, the bandwidth and flow table entries are more scarce for Eigen. When increasing the number of VNF types, the consumptions of bandwidth and flow table reduce and there are more flows with SFC requests can be served in the network. Therefore, the curve of Eigen grows quickly with the increasement of VNF types per function node.

Fig. 7b shows the average acceptance rate under different length of SFC request. The longer the length of SFC request is, the more resource consumption will be in the network. Therefore, these curves drop quickly when increasing the length of SFC request. Due to the fact that COATS and SP neglect to balance the consumptions on bandwidth, flow table and CPU at the same time and cannot differentiatedly route flows with SFC requests based on flow features, the performance gaps between these two algorithms and RA-RA become obvious, when the length of SFC request increases.

Fig. 7c shows the average acceptance rate under different proportion of function nodes. More proportion of function nodes means more flow table entries and more CPU resources in the network. When the proportion of function node is small, there are few VNF instances deployed in the network. Then, there are few choices for an SFC request to select VNF instances to satisfy its predefined order. Therefore, when the proportion of function node stays low between 10 and 20 percent, the performance gaps among these tested algorithms are small. And the performance gaps become obvious, when increasing the proportion of function nodes between 20 to 40 percent.

7 Conclusion

In the paper, to make a differentiated routing strategy with the optimal dynamic SFC formation and load balancing among multiple resources for flows with SFC requests, we study the DRP-SFC in SDN and NFV-enabled network. This problem is formulated as a BIP model with the aim to minimize the resource consumption costs for flows with SFC requests. In order to solve the DRP-SFC, we have proposed a novel routing algorithm named RA-RA. RA-RA makes efficient selection of VNF instances and find the associated paths for flows with SFC requests by transforming the network graph to LFG. In RA-RA, relative costs are used to balance the resource consumptions and avoid congestion in the network. Moreover, we take the resource preference as flow features and classify all the flows into different kinds to achieve differentiated routing for flows with SFC requests. The performance evaluation shows that RA-RA can efficiently solve the DRP-SFC and obtain higher network performance in terms of average acceptance rate, throughput, hop count and load balancing, comparing with other algorithms in existing literatures.

In the future work, we intend to extend our approach in a number of ways. We want to extend our approach to deal with VNF deployment problem in both ISP network and datacenter. We intend to design high-performance differentiated routing algorithm for flows with SFC requests to reduce the computation complexity and enhance the time efficiency of routing computation. And we also want to achieve fast failure resilience for flows with SFC requests in SDN and NFV-enabled network in the future.

Acknowledgments

This work is supported by the National Natural Science Foundation of China (NSFC) under Grant No. 61671420, 61672484.

References

[1] R. Mijumbi, J. Serrat, et al., "Network function virtualization: State-of-the-art and research challenges," *IEEE Commun. Surveys Tuts.*, vol. 18, no. 1, pp. 236–262, Jan.–Mar. 2016.

[2] C. Pham, N. H. Tran, and S. Ren, "Traffic-aware and Energy-efficient vNF placement for service chaining: Joint sampling and matching approach," *IEEE Trans. Services Comput.*, 2017.

[3] OpenFlow Switch Specification: Version 1.5.1. [Online]. Available: https://www.opennetworking.org/images/stories/downloads/sdn-resources/onf-specifications/openflow/openflow-switch-v1.5.1.pdf

[4] J. Martins, M. Ahmed, C. Raiciu, et al., "ClickOS and the art of network function virtualization," in *Proc. 11th USENIX Conf. Netw. Syst. Des. Implementation.*, 2014, pp. 459–473.

[5] J. Sherry, S. Hasan, C. Scott, et al., "Making middleboxes someone else's problem: Network processing as a cloud service," *SIGCOMM Comput. Commun. Rev.*, vol. 42, no. 4, pp. 13–24, 2012.

[6] J. Liu, Y. Li, and Y. Zhang, "Improve service chaining performance with optimized middlebox placement," *IEEE Trans. Services Comput.*, vol. 10, no. 4, pp. 560–573, Jul./Aug. 2017.

[7] J. Halpern and C. Pignataro, "Service function chaining (SFC) architecture," Informational RFC, RFC 7665. 2015. [Online]. Available: https://tools.ietf.org/rfc/rfc7665.txt

[8] P. Quinn and T. Nadeau, "Problem statement for service function chaining," *Informational RFC, RFC 7498*, 2015. [Online]. Available: http://www.rfc-editor.org/rfc/rfc7498.txt

[9] W. John, K. Pentikousis, G. Agapiou, et al., "Research directions in network service chaining," in *Proc. IEEE SDN Future Netw. Services*, 2013, pp. 1–7.

[10] D. Li, P. Hong, K. Xue, and J. Pei, "Virtual network function placement considering resource optimization and SFC requests in cloud datacenter," *IEEE Trans. Parallel Distrib. Syst.*, vol. 29, no. 7, pp. 1664–1677, Jul. 2018.

[11] J. G. Herrera and J. F. Botero, "Resource allocation in NFV: A comprehensive survey," *IEEE Trans. Netw. Service Manage.*, vol. 13, no. 3, pp. 518–532, Sep. 2016.

[12] H. Huang, S. Guo, P. Li, et al., "Cost minimization for rule caching in software defined networking," *IEEE Trans. Parallel Distrib. Syst.*, vol. 27, no. 4, pp. 1007–1016, Apr. 2016.

[13] A. Dixit, F. Hao, S. Mukherjee, et al., "ElastiCon: An elastic distributed SDN controller," in *Proc. ACM/IEEE Symp. Archit. Netw. Commun. Syst.*, 2014, pp. 17–28.

[14] J. Pei, P. Hong, and D. Li, "Virtual network function selection and chaining based on deep learning in SDN and NFV-enabled networks," *2018 IEEE Int. Conf. Commu. Workshops (ICC Workshops)*, Kansas City, MO, USA, 2018, pp. 1–6.

[15] T. Wood, K. K. Ramakrishnan, J. Hwang, et al., "Toward a software-based network: Integrating software defined networking and network function virtualization," *IEEE Netw. Mag.*, vol. 29, no. 3, pp. 36–41, May/Jun. 2015.

[16] W. Haeffner, J. Napper, et al., "Service function chaining use cases in mobile networks," *IETF Draft, draft-ietf-sfc-use-case-mobility-08*, 2018. [Online]. Available: https://www.ietf.org/id/draft-ietf-sfc-use-case-mobility-08.txt

[17] W. Liu, H. Li, O. Huang, et al., "Service function chaining (SFC) general use cases," *IETF Draft, draft-liu-sfc-use-cases-08*. 2014. [Online]. Available: https://www.ietf.org/archive/id/draft-liu-sfc-use-cases-08.txt

[18] D. Bhamare, R. Jain, M. Samaka, et al., "A survey on service function chaining," *J. Netw. Comput. Appl.*, vol. 75, pp. 138–155, 2016.

[19] G. Cheng, H. Chen, H. Hu, et al., "Enabling network function combination via service chain instantiation," *Comput. Netw.*, vol. 92, no. 2, pp. 396–407, 2015.

[20] I. Trajkovska, M.-A. Kourtis, C. Sakkas, et al., "SDN-based service function chaining mechanism and service prototype implementation in NFV scenario," *Comput. Standards Interfaces*, vol. 54, no. 4, pp. 247–265, 2017.

[21] R. Yu, G. Xue, V. T. Kilari, et al., "Network function virtualization in the multi-tenant cloud," *IEEE Netw.*, vol. 29, no. 3, pp. 42–47, May/Jun. 2015.

[22] A. M. Medhat, G. Carella, C. Lück, et al., "Near optimal service function path instantiation in a multi-datacenter environment," in *Proc. IEEE Int. Conf. Netw. Service Manage.*, 2015, pp. 336–341.

[23] H. Oh, D. Yu, Y.-H. Choi, et al., "Design of an efficient method for identifying virtual machines compatible with service Chain in a virtual network environment," *Int. J. Multimedia Ubiquitous Eng.*, vol. 9, no. 11, pp. 197–208, 2014.

[24] M. Mechtri, C. Ghribi, and D. Zeghlache, "A scalable algorithm for the placement of service function chains," *IEEE Trans. Netw. Service Manage.*, vol. 13, no. 3, pp. 533–546, Sep. 2016.

[25] Z. Cao, M. Kodialam, T. V. Lakshman, et al., "Traffic steering in software defined networks: Planning and online routing," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 44, no. 4, pp. 65–70, 2014.

[26] M. F. Bari, S. R. Chowdhury, R. Ahmed, et al., "Orchestrating virtualized network functions," *IEEE Trans. Netw. Service Manage.*, vol. 13, no. 4, pp. 725–739, Dec. 2016.

[27] A. Viterbi, "Error bounds for convolutional codes and an asymptotically optimum decoding algorithm," *IEEE Trans. Inf. Theory*, vol. 13, no. 2, pp. 260–269, Apr. 1967.

[28] A. Dwaraki and T. Wolf, "Adaptive service-chain routing for virtual network functions in software-defined networks," in *Proc. ACM Workshop Hot Topics Middleboxes Netw. Function Virtualization*, 2016, pp. 32–37.

[29] OpenDaylight. [Online]. Available: https://media.readthedocs.org/pdf/opendaylight/latest/opendaylight.pdf

[30] H. Huang, S. Guo, J. Wu, et al., "Joint middlebox selection and routing for software-defined networking," in *Proc. IEEE Int. Conf. Commun.*, 2016, pp. 1–6.

[31] F. Jalali, K. Hinton, and R. Ayre, "Fog computing may help to save energy in cloud computing," *IEEE J. Sel. Areas Commun.*, vol. 34, no. 5, pp. 1728–1739, May 2016.

[32] T. Benson, A. Akella, and D. A. Maltz, "Network traffic characteristics of data centers in the wild," in *Proc. ACM SIGCOMM Conf. Internet Meas.*, 2010, pp. 267–280.

[33] D. Kliazovich, P. Bouvry, and S. U. Khan, "GreenCloud: A packet-level simulator of energy-aware cloud computing data centers," *J. Supercomput.*, vol. 62, no. 3, pp. 1263–1283, 2012.

[34] R. Trestian, K. Katrinis, and G.-M. Muntean, "OFLoad: An OpenFlow-based dynamic load balancing strategy for datacenter networks," *IEEE Trans. Netw. Service Manage.*, vol. 14, no. 4, pp. 792–803, Dec. 2017.

[35] I. F. Akyildiz, A. Lee, P. Wang, et al., "Research challenges for traffic engineering in software defined networks," *IEEE Netw.*, vol. 30, no. 3, pp. 52–58, May/Jun. 2016.

[36] S.-C. Chao, K. C.-J. Lin, and M.-S. Chen, "Flow classification for software-defined data centers using stream mining," *IEEE Trans. Services Comput.* 2016. [Online]. Available: https://doi.org/10.1109/TSC.2016.2597846

[37] D. Adami, G. Antichi, R. G. Garroppo, et al., "Towards an SDN network control application for differentiated traffic routing," in *Proc. IEEE Int. Conf. Commun.*, 2015, pp. 5827–5832.

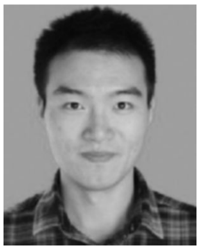
[38] Monarch Network Architects, "Sample Optical Network Topology Files". [Online]. Available: http://www.monarchna.com/topology.html

[39] T.-W. Kuo, B.-H. Liou, K. C.-J. Lin, et al., "Deploying chains of virtual network functions: On the relation between link and server usage," in *Proc. IEEE INFOCOM*, 2016, pp. 1–9.

[40] K. He, J. Khalidy, A. Gember-Jacobson, et al., "Measuring control plane latency in SDN-enabled switches," in *Proc. ACM SIGCOMM Symp. Softw. Defined Netw. Res.*, 2015, pp. 1–6.

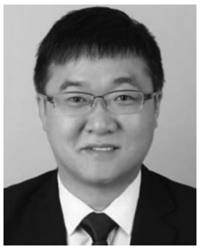
[41] R. Ramaswamy, N. Weng, and T. Wolf, "Characterizing network processing delay," in *Proc. IEEE Global Telecommun. Conf.*, 2004, pp. 1629–1634.

[42] L. Pantel and L. C. Wolf, "On the impact of delay on real-time multiplayer games," in *Proc. ACM Int. Workshop Netw. Operating Syst. Support Digit. Audio Video*, 2002, pp. 23–29.



Jianing Pei received the BS degree from the Department of Information and Electrical Engineering (IEE), China University of Mining and Technology (CUMT), in 2015 and he is working toward the PhD degree at the University of Science and Technology of China (USTC) with his advisor Peilin Hong now. His research interests include machine learning, SDN, NFV, and the network resource orchestration and management.

Peilin Hong received the BS and MS degrees from the Department of Electronic Engineering and Information Science (EEIS), University of Science and Technology of China (USTC), in 1983 and 1986, respectively. Currently, she is a professor and advisor for PhD candidates with the Department of EEIS, USTC. Her research interests include next-generation Internet, policy control, IP QoS, and information security. She has published two books and more than 100 academic papers in several journals and conference proceedings.



Kaiping Xue (M09-SM15) received the BS degree from the Department of Information Security, University of Science and Technology of China (USTC), in 2003 and the PhD degree from the Department of Electronic Engineering and Information Science (EEIS), USTC, in 2007. Currently, he is an associate professor with the Department of Information Security and Department of EEIS, USTC. His research interests include next-generation Internet, distributed networks, and network security. He is a senior member of the IEEE.

Defang Li received the BS degree from the Department of Electronic Engineering and Information Science (EEIS), University of Science and Technology of China (USTC), in 2014 and he is working toward the PhD degree at the University of Science and Technology of China with his advisor Peilin Hong now. His research interests include SDN, NFV, and the network resource orchestration and management. He is a student member of the IEEE.