IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received October 7, 2019, accepted October 25, 2019, date of publication October 30, 2019, date of current version November 11, 2019.

Digital Object Identifier 10.1109/ACCESS.2019.2950424

Resource Aware Chaining and Adaptive Capacity Scaling for Service Function Chains in Distributed Cloud Network

JIACHEN ZU, GUYU HU, YANG WU, DONGSHENG SHAO, AND JIAJIE YAN
Institute of Command and Control Engineering, Army Engineering University, Nanjing 210007, China
Corresponding author: Guyu Hu (huguyu@189.cn)

ABSTRACT With the development of network technology such as software-defined network (SDN) and network function virtualization (NFV), Internet service providers (ISPs) are increasingly placing the virtual network function(VNF) instances at the network edge to provide network service. However, there are some issues to be tackled in the distributed SDN/NFV enabled cloud. Firstly, VNF instances require to be chained in predefined order to provide network services. It is a challenge to optimally select and chain VNF instances from the multi-instances. Moreover, due to the capacity limitation of the distributed edge nodes. The capacity of the Virtual Machines (VMs) that host VNFs should be proactively adjusted to cope with traffic demands. Since most existing works ignore the vertical capacity scaling problem in routing commodities with Service Function Chain (SFC) requests. In this paper, a fine-grained scheduling scheme at VM-level is proposed. Firstly, we formulate the SFC chaining problem as an Integer Linear Programming (ILP) model aiming to embed SFC requests with minimum estimated latency cost. Furthermore, we formulate the adaptive VNF resource allocation (VNF-AR) problem as a convex optimization. The theoretical optimal capacity for each VM can be derived from the Karush-Kuhn Tucker (KKT) conditions. At last, a novel joint optimization approach of VNF chaining and adaptive scaling (VNF-CAS) is proposed to efficiently embed the SFC requests. Performance evaluation shows that VNF-CAS can achieve better performance in SFC requests acceptance rate, average effective throughput, average load utilization and VM load balancing when it is compared with other algorithms in existing works.

INDEX TERMS Network function virtualization, service function chain, distributed cloud network, resource optimization.

I. INTRODUCTION

In traditional, the Internet service providers (ISPs) use the dedicated hardware equipment to offer different network functions such as Firewalls, Proxies, Network Address Translators (NATs) and Intrusion Detection Systems (IDSs), this can result in high cost and inflexible management of ISP's network. To reduce the Operating Expenditures (OPEX) and Capacity Expenditures (CAPEX), network function virtualization (NFV) [1] was proposed to migrate network functions from the hardware-based equipment to software-defined instances and allow scalable and flexible deployment of network functions. In NFV, different network functions are executed in virtual machines (VMs) or containers on standardized servers. In general, the NFV architecture [2] is composed of three main components, Virtual Network Functions (VNF), Network Function Virtualization Infrastructure (NFVI), management and orchestration architectural framework (NFV MANO). The VNFs are controlled and managed by MANO according to software-defined networking (SDN) paradigm [3]. Typically, NFV is used in the data center network, which brings great advantages in flexibility and cost-efficiency. However, the centralized orchestration of a large number of VNFs becomes a problem. To reduce the complexity of orchestration, the ISPs can place a few VNFs in distributed Micro-Data Centers (MDCs) [4] with the network edge computing technology [5]. These distributed MDCs can be deployed in buildings or neighbors near users to achieve better QoS and lower latency. In addition, NFV is applied to 5G network slice technology, which provides diversified

The associate editor coordinating the review of this manuscript and approving it for publication was Xiaodong Xu.

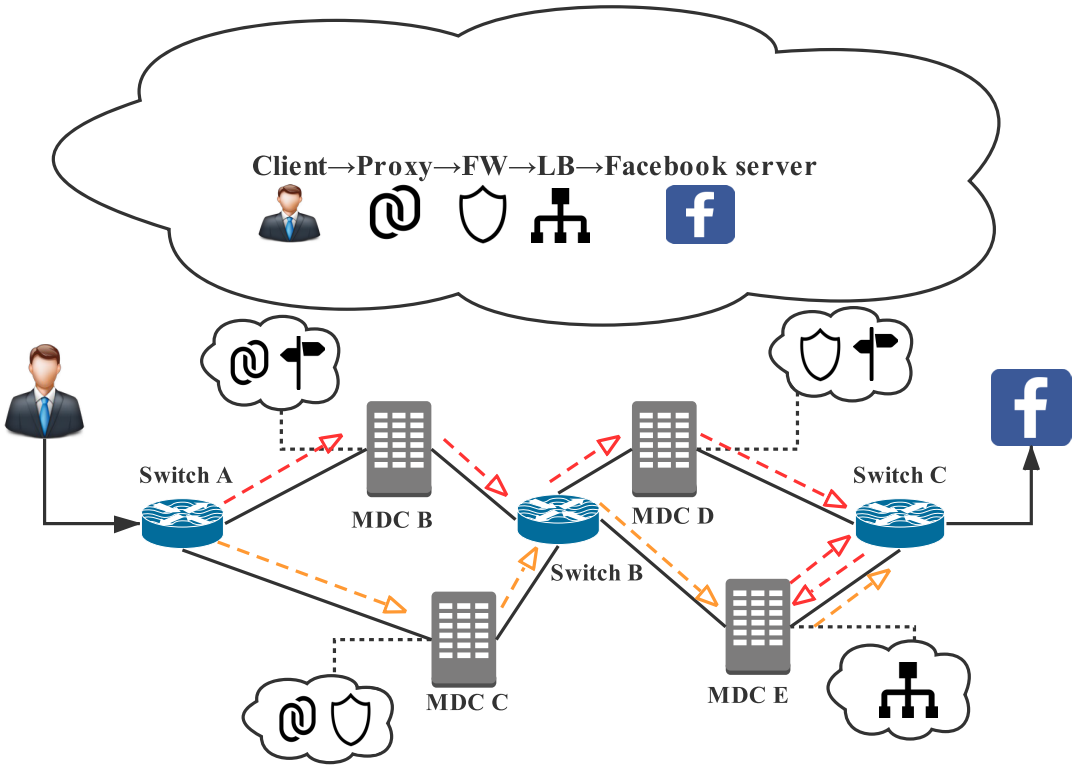


FIGURE 1. VNF selection and chaining in distributed cloud network.

and customized network services to users through on-demand network slicing [6].

In fact, there are several issues to be investigated on the orchestration of VNFs. Firstly, how to concatenate different VNFs effectively within the ordering constraints. In real scenarios, the service requests from clients often consist of several network functions. Network flows need to traverse through a series of VNFs in sequence to achieve the specific objective. The interconnection of VNFs which provide end-to-end Internet service is defined as a service function chain (SFC) [7], [8]. As different chain composition affects the performance of the network service. Authors in [9] defined a model for formalizing the chaining of network functions using a context-free language. In [10], a topological dependence sorting method was used to jointly optimize the processes of SFC designing and deployment. Secondly, a lot of researchers have focused on how to deploy the SFC optimally to achieve a good network performance. In [11], an approximation algorithm based on GAP (Generalized Assigned Problem) was applied to solve the NFV location problem of small distributed cloud nodes. The work in [12] studied the joint problem of VNF placement and path selection to better utilize network resource, it was verified to serve more demands by optimizing the VM reuse factor and path length. Besides, Energy-aware SFC placement strategy was proposed to reduce power consumption in data center scenarios [13], [14].

Since the different stages of SFC orchestration are relevant to each other. (*e.g.* SFC chaining and SFC placement). A growing number of studies are trying to consider many stages together and jointly optimize them. Therein, the optimal steering and resource allocation of NFV has attracted attention from both academia and industry [15]. Once given the available locations of VNF instances, there are usually multiple instances to choose from. How to optimally select the VNF instance and construct the routing path without violating constraints becomes a problem, especially in the distributed network. Fig.1 shows a simple instance of the SFC in the distributed cloud system: a client needs to get social network service (*e.g.* logging in Facebook server through an SFC). This SFC is composed of 3 types of VNFs, the Web Proxy, the Firewall (FW) and the Load Balancer (LB). There are 3 switch nodes and 4 MDC nodes in the substrate network. And four types of VNFs are deployed on MDC nodes B, C, D, E respectively. Each type of VNF usually has multiple instances located in different MDC nodes. In that case, there are many paths that can satisfy the predefined order for the orchestrator to choose. For example, Path I: Switch A $\Rightarrow$ MDC B $\Rightarrow$ Switch B $\Rightarrow$ MDC D $\Rightarrow$ Switch C $\Rightarrow$ MDC E $\Rightarrow$ Switch C $\Rightarrow$ Facebook server or Path II: Switch A $\Rightarrow$ MDC C $\Rightarrow$ Switch B $\Rightarrow$ MDC E $\Rightarrow$ Switch C $\Rightarrow$ Facebook server. Path I traverses VNFs in different MDCs, which may result in much bandwidth usage and many hop counts. Path II traverses VNFs by sharing MDC node C, which may be helpful to reduce hop counts and bandwidth usage. However, it can lead to load imbalance when too many phsical resources are occupied in a single node. Therefore, due to the limited physical resources (*i.e.* CPU, memory and

bandwidth) in the substrate network, it is a challenge to find the best compromise routing path to achieve load balancing and reduce resource bottlenecks.

On the side of the client's needs, the clients want to get better Internet QoS such as low end-to-end delay and low packet loss rate. Different network services also have different preferred network metrics. For example, the real-time video service typically requires more bandwidth whereas the VoIP service is more sensitive to latency. There is a need to design a resource-aware VNF chaining algorithm related to clients' demand. On the other side of the network provider's requirements, the network is ought to accommodate flows as many as possible under the capacity constraints to achieve throughput-optimal. There should be a mechanism to dynamically adjust resource allocation to balance the load based on the historical traffic. But little research has been done in the joint optimization of VNF chaining and scaling. In this paper, a joint VNF chaining and adaptive scaling optimization in the distributed cloud system is proposed. There are mainly two problems in the following to be solved: (1) How to optimally select and concatenate VNF instances based on the resource information of the substrate network. (2) How to dynamically adjust resource allocation of VMs according to the traffic demand. The contributions of this paper are listed as follows:

- We formulate the VNF chaining (VNF-C) problem as an ILP model aiming at minimizing the estimated delay cost. Taking the bandwidth of links, the load of VMs into account, a Service Function Graph method is proposed to chain VNFs in a fine-grained VM level.
- Instead of simply allocating resources fairly to VMs of different VNFs (*e.g.* in [12]), we first formulate the adaptive VNF resource allocation (VNF-AR) problem into a convex optimization and give the optimal capacity in theoretical analysis.
- Furthermore, by combining the optimization results above. We design a novel VNF chaining and adaptive scaling (VNF-CAS) algorithm. The joint VNF-CAS algorithm is proven to provide good performance guarantees by real trace-driven simulation. The performance evaluation results show that our proposed algorithm achieves better performance in acceptance rate, effective throughput, average load utilization and load balancing than the existing algorithms [16]–[18] in different topologies.

The rest of the paper is organized as follows: The related work is presented in Section II. We explain the detailed model in Section III and formulate it into an ILP model in Section IV. Section V gives the description of the proposed algorithms including VNF-C, VNF-AR, and VNF-CAS. In section VI, we validate our proposed algorithm through real trace-driven simulation and compare it with other existing algorithms in different topologies. Finally, Section VII gives the conclusion of the paper.

II. RELATED WORK

To solve virtual network function chaining and routing problem. Bari *et al.* [18] formulated this problem as an ILP model, the costs of VNF deployment, energy, and Service Level Objective (SLO) violation are considered with the aim of minimum OPEX and fragmentation. A heuristic-based on the Viterbi algorithm was proposed to embed SFC requests in a multi-stage graph. Mechtri *et al.* [19], [20] proposed a matrix-based approach, using eigendecomposition of adjacency matrices to cope with the VNF placement and chaining problem in distributed cloud environments. Khebbache *et al.* [21] also adopted matrix-based optimization and multi-stage graph method to find solutions in polynomial times. Using perfect 2 matching method at the VNF chain placement stage is helpful to enhance resolution time and guarantee scalability. Nevertheless, studies in [18]–[20] select VNFs in a greedy way at each stage of the multi-stage graph. Though the heuristic method may be conducive to reduce the execution time of solution and improve scalability. These algorithms can lead to a sub-optimal solution for the routing of SFC flows.

Tang *et al.* [22] raised a dynamic queuing model to deal with the VNF placement in 5G access network. With Lyapunov optimization technique, a genetic algorithm-based heuristic SFC scheduling and mapping algorithm (QDPVNF) was presented. In addition, other methods like dynamic programming (DP) [23], Column Generation (CG) [24], Monte Carlo Tree Search (MCTS) [25] are also applied in the VNF placement and chaining.

Different from the research works above, some studies formulate the SFC placement problem from the perspective of multi-path routing (MRP). Cao *et al.* [16] proposed a novel algorithm named Competitive Online Algorithm for Traffic Steering (COATS) based on segmentation and lazy dual. In COATS, the flows are given one by one and the objective is to steer the flows to maximize the total amount of traffic. Dwaraki *et al.* [26] proposed an adaptive service routing algorithm to solve the online problem. In the algorithm, the network graph is transformed into a layered graph. The author proposed a way to calculate end-to-end delay from utilization measurement on links and servers. Conventional SP algorithm is executed to get the optimal solution and the feasibility of this approach is verified through an emulated prototype implementation.

Based on the research method of [16], [26], Pei *et al.* [17] proposed a differentiated routing algorithm aiming to minimize resource consumption costs of flows with SFC requests. In this algorithm, the flows are classified into different kinds based on resource preference. The costs of different flows are defined differentially to balance resource consumption. In [27], Sallam *et al.* proposed a transformation of the network graph which can reduce the computational complexity of works in [16], [26]. Then they formulated this SFC-constrained maximum flow problem as a fractional

multicommodity to achieve throughput-optimal routing. Similarly, Pei *et al.* [28] designed a routing scheme based on deep learning to reduce the routing computation time of the layered method. Performance evaluation shows that the deep learning method can get almost the same performance in acceptance rate and end-to-end delay, but enhance time efficiency dramatically.

Considering the more complicated scenario, due to the highly dynamic of resource usage and limited duration time of SFC requests. The SFC requests are dynamic and result in the variation of network load. Zhou [29] proposed a primal-dual placement scheme that makes on-spot decisions upon the arrival of SFC. Pei *et al.* [30] studied the SFC embedding problem to optimize the number and placement of VNF instances according to the dynamic load in geo-distributed cloud system. A novel SFC embedding approach (SFC-MAP) and a VNF dynamic release algorithm (VNF-DRA) were proposed to efficiently embed SFC requests and optimize the number of placed VNF instances. Furthermore, with the traffic history, the upcoming traffic can be predicted. VNF instances should be dynamically scaled up or down. Fei *et al.* [31] employed an efficient online method called follow-the-regularized-leader (FTRL) to minimize the error in predicting SFC demands. With the prediction results, the processing capacity can be derived to adaptively assign VNF instances. The joint online algorithm is proven to provide good performance guarantees by both theoretical analysis and trace-driven simulation.

Nevertheless, most of the mentioned works neglect the fact that different VNFs are segregated in different VMs to share the processing-resource. Each VM has a limited computational capacity and the load imbalance of VMs will cause network performance degradation. In that case, there should be an efficient processing-resource allocation mechanism to accommodate VNF demands. Buh *et al.* [32] defined a novel adaptive traffic distribution among multi-core architectures, this method was experimentally validated by tests on Linux Bridge networking devices. Savi *et al.* [33] considered two penalties include context switching costs (causing by repeated context loading/saving on the same CPU) and upscaling costs (causing by load-balancing needs of VNFs among multiple CPU cores). Both of them were verified to affect the embedding of SFCs. Kulkarni *et al.* [34] proposed a VNF scheduling and service chain management framework named VNFnice to achieve fair resource allocation of CPU. In their work, the VNF load was calculated as the product of the arrival rate and estimated service time. An extending backpressure approach at packet granularity was applied to manage congestion. A DPDK based platform [35] that runs VNFs in containers was used to facilitate deployment. In this paper, we mainly focus on the optimal selection and chaining problem of SFC in the distributed system. A joint optimization method of SFC chaining and VNF adaptive scaling is proposed.

III. SYSTEM MODEL

A. SUBSTRATE NETWORK

The network is generally modeled as an undirected graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, where $\mathcal{V}$ is the node set and $\mathcal{E}$ is the link set. The node number $|\mathcal{V}|$ and the link number $|\mathcal{E}|$ are both limited. Here, we assume two kinds of nodes in the distributed cloud network, $\mathcal{V} = \mathcal{V}_{se} \cup \mathcal{V}_{sw}$ where $\mathcal{V}_{se}$ is the server node set and $\mathcal{V}_{sw}$ is the switch node set. Switch nodes are responsible to forward data to neighbors. Server nodes are not only responsible to forward data but also to process the traffic of SFC demands. Both of switches and servers work under the control instructions of the orchestrator. Since the switches do not need to process the flows, we neglect the communication cost on switch nodes. In the NFV MANO network architecture, the NFV orchestrators can be seen as a kind of SDN controllers responsible to collect the network status and manage the VNF orchestration.

For each link $(v, w) \in \mathcal{E}$, its bandwidth capacity is C_{vw}. We use η_{vw} to represent the relative bandwidth utilization of link $(v, w) \in \mathcal{E}$. For each node $v \in \mathcal{V}_{se}$, we use C_v to represent the computational capacity. While $C_v = 0$ for $v \in \mathcal{V}_{sw}$. In most NFV scenarios, the CPU capacity is regarded to be the most important metric to evaluate computational resources in MIPs (million instructions per second), whereas other hardware resource like memory is relatively sufficient. Multiple VMs can be installed to hold VNF instances on each server node. We use η_v^m to represent the relative load utilization of the m^{th} VM on server node $v \in \mathcal{V}_{se}$ and η_v to represent the load utilization of the server node. In NFV-enabled network, the network can support a set of VNFs F (with maximum number $|F|$). Each $f \in F$ is associated with a specific kind of VNF like the firewall. We assume that every VM is able to execute one VNF $f \in F$. The VNF types in different servers can be the same, but the VNF types of VMs on the same server must be different from each other. In addition, since the orchestrators have no aprior knowledge of the traffic distribution of VNF instances. At the Initial phase, we assume the CPU capacity of each VM C_v^m on the same server is allocated equally according to server's C_v and VM number M_v.

B. SERVICE FUNCTION CHAIN REQUESTS

In this network model, all data that enter the network are associated with a particular commodity with SFC request. A seven-tuple is defined to symbolize the SFC $S_i = \left\{o_i, d_i, \varphi_i, R_i^{cpu}, R_i^{bw}, R_i^D, \widetilde{\mathbb{G}}_i\right\}$. Given a set of SFC commodities, we use $|S|$ to represent the total number of the existed commodities in the network. Let o_i and d_i represent the source and destination address of S_i respectively. Each S_i is composed of a series of VNFs in a specific order which is defined as $\varphi_i = \{\varphi_{i1}, \varphi_{i2}, \ldots, \varphi_{il}\}$. φ_{ij} represents the j^{th} VNF request of S_i, where $j = 1, 2, 3, \ldots l$, $l = |\varphi_i|$ is the maximum VNF number of S_i. For each S_i, the CPU demand, bandwidth consumption and maximum tolerated delay are defined as R_i^{cpu}, R_i^{bw}, R_i^D. Both of these metrics depend on the

specific demand of SFC. It's important to note that since there exsit non-prioritized flows like IoT traffic, these traffic can be delayed and their tolerated delay is assumed to be infinite. For each SFC commodity, to simplify its complexity, we assume the bandwidth consumption and CPU demand of VNFs to be the same value. As for $\widetilde{\mathbb{G}}_i = \left(\widetilde{\mathbb{V}}_i, \widetilde{\mathbb{E}}_i\right)$, it is a directed graph to describe the available routing path of S_i. $\widetilde{\mathbb{V}}_i$ is the set of available virtual nodes that can host the required VNF and $\widetilde{\mathbb{E}}_i$ is the set of virtual links to connect adjacent VNFs. The direction of links must satisfy the constraints in φ_i and the weight of links represents the resource consumption cost between adjacent nodes. Routing graph $\widetilde{\mathbb{G}}_i = \left(\widetilde{\mathbb{V}}_i, \widetilde{\mathbb{E}}_i\right)$ is the subset of the Service Function Graph $\mathbb{G}_i = (\mathbb{V}_i, \mathbb{E}_i)$, which is described in detail in Section V.

To better utilize resources, we allow different flows to share the same VNF instance if its capacity C_v^m is sufficient. In addition, any two different VNFs in one SFC can be served on the same server, so that the communication between them becomes intra-node. In this case, the bandwidth consumption of the corresponding virtual link is regarded to be zero. It is beneficial to reduce the end-to-end latency. Last but not least, though service function chains can greatly enrich the diversity of NFV, the actual form of the SFC is usually not arbitrary in the real scenario. Taking the Web service as an example, the requests of a user usually need to sequentially go through the web proxy, the load balancer and the firewall to access to the Internet. Therefore, It is assumed that there is a set Ω to cover all the SFC forms, and we merge the service function chains that have the same ingress node o_i, egress node d_i and the same VNF forms into one group Ω_k.

IV. PROBLEM FORMULATION

A. PROBLEM DESCRIPTION

In this paper, once the network topology $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ and a set of SFC commodities Ω are given. To realize the SFC optimal routing in distributed cloud network, the network sevice provider needs to (1) collect the VNFs' location and resource usage status of the substrate network (2) construct a multi-layer graph for each SFC and calculate the weights of corresponding edges (3) select a path and judge whether the required VNFs can be placed or not. The optimal VNF selection and chaining problem can be seen as a multi-path routing problem (MRP) with SFC constraints. From another perspective, the relationship between server nodes and SFCs can be compared to the facilities and clients. This problem can also be formulated as a facility location problem (FLP), which is aiming at minimizing the total cost of matching the facilities to clients.

B. DETAILED FORMULATION

In this section, we formulate the optimal VNF selection and chaining problem into an ILP model, which can be seen as the combination model of MRP and FLP. The main notions are listed in TABLE 1.

There are many constraints for SFC in the NFV enabled network. Firstly, the routing path of S_i starts from o_i and ends at d_i, it can not be split. If S_i is accepted and placed, then it should origin from o_i to d_i as:

$$\sum_{v\in\mathcal{V}}\sum_{(\mathbb{v},\mathbb{w})\in\mathbb{E}_i}\left(z_{i,wv}^{\mathbb{v},\mathbb{w}} - z_{i,vw}^{\mathbb{v},\mathbb{w}}\right) = \begin{cases} 1 & \mathbb{w} = o_i \\ -1 & \mathbb{w} = d_i \\ 0 & \text{otherwise} \end{cases} \tag{1}$$

Here, we use binary variable $z_{i,vw}^{\mathbb{v},\mathbb{w}}$ to indicate whether virtual link $(\mathbb{v}, \mathbb{w}) \in \mathbb{E}_i$ traverses through the link $(v, w) \in \mathcal{E}$. The selected routing path π_i must be able to traverse through VNFs following the specified order in S_i, *i.e.* $I\left(\varphi_{i,j}\right) \leq I\left(\varphi_{i,j+1}\right) \forall i, j. I\left(\varphi_{i,j}\right)$ indicates the sequence index of the node in selected path π_i.

TABLE 1. Symbol and variables.

Substrate Network	
$\mathcal{G} = (\mathcal{V}, \mathcal{E})$	Network topology with node set $\mathcal{V}$ and link set $\mathcal{E}$.
$\mathcal{V}_{se}, \mathcal{V}_{sw}$	Node set of servers and switches, where $\mathcal{V} = \mathcal{V}_{se} \cup \mathcal{V}_{sw}$.
C_v, C_v^m	CPU capacity of server node $v \in \mathcal{V}_{se}$ and its m^{th} VM.
C_{vw}	Bandwidth capacity of link $(v, w) \in \mathcal{E}$
$\eta_v, \eta_v^m, \eta_{vw}$	Relative load utilization of server node $v \in \mathcal{V}_{se}$, its m^{th} VM and bandwidth utilization of link $(v, w) \in \mathcal{E}$
D_{vw}^{prop}	Propagation delay of $\text{link}(v, w) \in \mathcal{E}$
D_{vw}^{queue}	Queuing delay of $\text{link}(v, w) \in \mathcal{E}$
$D_{v,m}^{proc}$	Processing delay of the m^{th} VM on server node $v \in \mathcal{V}_{se}$
M_v	The VM number of server node $v \in \mathcal{V}_{se}$
F	Supported VNF set of the substrate network
Ω	Supported SFC set of the substrate network
Service Function Chain(S_i)	
o_i, d_i	The ingress node and egress node of S_i
φ_i	The form of VNFs in S_i, $\varphi_i = \{\varphi_{i1}, \varphi_{i2}, \ldots, \varphi_{il}\}$, $l = \lvert\varphi_i\rvert$ which represents the VNF number of S_i
$R_i^{cpu}, R_i^{bw}, R_i^D$	The CPU demand,bandwidth consumption, and maximum tolerated delay of S_i
$\widetilde{\mathbb{G}_i} = \left(\widetilde{\mathbb{V}_i}, \widetilde{\mathbb{E}_i}\right)$	Routing Graph $\widetilde{\mathbb{G}_i}$ with the node set $\widetilde{\mathbb{V}_i}$ and link set $\widetilde{\mathbb{E}_i}$.
π_i	Actual routing path of S_i
Binary Variable	
$x_{v,m}^f$	Boolean variable that equals 1 if the VNF $f \in F$ is hosted on the m^{th} VM of server node $v \in \mathcal{V}_{se}$ and 0 otherwise
$x_{v,m}^{\varphi_{ij}}$	Boolean variable that equals 1 if the VNF φ_{ij} of S_i is hosted on the m^{th} VM of server node $v \in \mathcal{V}_{se}$ and 0 otherwise
$z_{i,v,m}^{\mathbb{v},\mathbb{w}}$	Boolean variable that equals 1 if the virtual link $(\mathbb{v}, \mathbb{w}) \in \widetilde{\mathbb{E}}_i$ traverses through the m^{th} VM of node $v \in \mathcal{V}_{se}$ and 0 otherwise
$z_{i,vw}^{\mathbb{v},\mathbb{w}}$	Boolean variable that equals 1 if the virtual link $(\mathbb{v}, \mathbb{w}) \in \widetilde{\mathbb{E}}_i$ traverses through the link $(v, w) \in \mathcal{E}$ and 0 otherwise.

Secondly, the total bandwidth consumption on each physical link $(v, w) \in \mathcal{E}$ would not exceed bandwidth capacity as:

$$\sum_{i=1}^{|S|} \sum_{(\mathbb{v},\mathbb{w})\in\mathbb{E}_i} R_i^{bw} \cdot z_{i,vw}^{\mathbb{vw}} \leq C_{vw} \quad \forall (v, w) \in \mathcal{E} \tag{2}$$

As for the capacity constraints of nodes, the required computational resources on each VM of node would not exceed its CPU capacity. Secondly, the total capacities of VMs would not exceed their server's CPU capacity. Here, $y_{v,m}^{\varphi_{ij}}$ is symbolized to indicate whether the VNF φ_{ij} of S_i is hosted on the m^{th} VM of server node $v \in \mathcal{V}_{se}$.

$$\sum_{i=1}^{|S|} \sum_{j=1}^{|\varphi_i|} R_i^{cpu} \cdot y_{v,m}^{\varphi_{ij}} \leq C_v^m \quad \forall m,\ \forall v \in \mathcal{V}_{se} \tag{3}$$

$$\sum_{m}^{M_v} C_v^m \leq C_v \quad \forall v \in \mathcal{V}_{se} \tag{4}$$

For any server nodes in the network, the VM of a server node is assumed to host only one type of VNF $f \in F$. Here, $x_{v,m}^f$ is symbolized to judge whether the VNF $f \in F$ is hosted on the m^{th} VM of server node $v \in \mathcal{V}_{se}$.

$$\sum_{f\in F} x_{v,m}^f = 1 \quad \forall m,\ \forall v \in \mathcal{V}_{se} \tag{5}$$

The VNF types of VMs on the same server node are supposed to be different from each other as:

$$\sum_{f\in F} x_{v,m}^f \cdot x_{v,m'}^f = 0 \quad \forall m \neq m',\ \forall v \in \mathcal{V}_{se} \tag{6}$$

If different SFCs traverse the same VM in a server node, they must request the same type of VNF:

$$y_{v,m}^{\varphi_{ij}} = y_{v,m}^{\varphi_{i'j'}} = 1 \text{ if } \varphi_{ij} = \varphi_{i'j'} \quad \forall m,\ \forall i, i',\ \forall j, j',\ \forall v \in \mathcal{V}_{se} \tag{7}$$

For each S_i, (8) ensures that different VNFs in S_i can be served on one server, but the VNF instance number can not exceed the maximum VM number. So there may be some loops in the routing path.

$$\sum_{m=1}^{M_v} \sum_{j=1}^{|\varphi_i|} y_{v,m}^{\varphi_{ij}} \leq M_v \quad \forall i, v \in \mathcal{V}_{se} \tag{8}$$

At last, the end-to-end delay of selected path must be smaller than SFC's maximum tolerated delay. In this paper, we consider the end-to-end delay of S_i consists of three components: queuing delay D_{vw}^{queue}, processing delay $D_{v,m}^{proc}$ and propagation delay D_{vw}^{prop}.

$$\sum_{(\mathbb{v},\mathbb{w})\in\mathbb{E}_i} \sum_{v\in\mathcal{V}_{se}} \sum_{m=1}^{M_v} D_{v,m}^{proc} \cdot z_{i,v,m}^{\mathbb{v},\mathbb{w}} + \sum_{(\mathbb{v},\mathbb{w})\in\mathbb{E}_i} \sum_{(v,w)\in\mathcal{E}} \left(D_{vw}^{prop} + D_{vw}^{queue}\right) \cdot z_{i,vw}^{\mathbb{vw}} \leq R_i^D \quad \forall i \tag{9}$$

C. DEFINITION OF THE RESOURCE COST

From a multi-path routing perspective, we define the objective function of the optimal VNF chaining as the resource cost. In order to meet the requirements of network service providers and clients, the resource cost is ought to be relevant to bandwidth utilization and load utilization. In addition, we hope to balance the workload and reduce the resource bottlenecks by minimizing the defined resource cost. So we consider using the definition of estimated end-to-end delay in [26] to represent the cost of a routing path.

$$\widetilde{D_{vw}^{prop}} = \frac{\mathrm{Dis}(v, w)}{c_{medium}} \tag{10}$$

$$\widetilde{D_{vw}^{queue}} = \frac{\eta_{vw}}{1 - \eta_{vw}} \cdot d_{tx} \tag{11}$$

$$\widetilde{D_{v,m}^{proc}} = \frac{\eta_v^m}{1 - \eta_v^m} \cdot t_{proc} \tag{12}$$

The resource cost for traversing a link consists of two components: estimated propagation delay $\widetilde{D_{vw}^{prop}}$ as in (10) and estimated queuing delay $\widetilde{D_{vw}^{queue}}$ as in (11). The resource cost for traversing a VM consists of the estimated processing delay $\widetilde{D_{v,m}^{proc}}$ as in (12).

The estimated propagation delay is calculated by the physical length $Dis(v, w)$ of link and propagation speed of signals in that medium. c_{medium} is generally measured to be $5\mu s/km$ in optical fiber.

The estimated queuing delay is dependent on the bandwidth utilization η_{vw} of link. Using a simple M/M/1 queuing model with an expected service time, d_{tx} represents the transmission delay which is dependent on the transmitted packet size and link bandwidth. Similarly, the estimated processing delay of VM is depending on its load utilization η_v^m. t_{proc} is the per packet processing time.

The resource costs defined in (11) and (12) are closely relevant to the load of links and nodes. Both of them are increasing and convex functions of load utilization. This kind of function captures nearly linear growth in latency at light traffic load, but it witnesses a sharp increase when the resource consumptions are nearly approaching the maximum capacity. It is noted that the estimated processing delay in (12) is only considered on the server node $v \in \mathcal{V}_{se}$.

In our hypothesis, the resource cost is calculated by the estimated end-to-end delay. However, the real end-to-end delay is dynamic and stochastic in the network. It is difficult to evaluate with the resource usage information. But the resource utilization of network can be a way to estimate the delay cost. Furthermore, this method is tested to be effective to match with dynamic network data as in [26]. So in this paper, we use the sum of estimated delay on the routing path to indicate the resource cost. The objective function of each

SFC request is:

$$\min \sum_{(\mathbb{v},\mathbb{w})\in\mathbb{E}_i}\sum_{v\in\mathcal{V}_{se}}\sum_{m=1}^{M_v}\frac{\eta_v^m}{1-\eta_v^m}\cdot t_{proc}\cdot z_{i,v,m}^{\mathbb{v},\mathbb{w}} + \sum_{(\mathbb{v},\mathbb{w})\in\mathbb{E}_i}\sum_{(v,w)\in\mathcal{E}}\left(\frac{Dis(v,w)}{c_{medium}}+\frac{\eta_{vw}}{1-\eta_{vw}}\right)\cdot z_{i,vw}^{\mathbb{vw}}$$
$$\text{Subject to} \quad Eq.(1)-Eq.(9) \tag{13}$$

The aim of the ILP model is to find the optimal solutions of $z_{i,v,m}^{\mathbb{v},\mathbb{w}}$ and $z_{i,vw}^{\mathbb{v},\mathbb{w}}$ to minimize the estimated end-to-end delay cost. Because the objective function defined in (13) can reflect the load status of links and nodes on the path. If the resource cost is a very big number, we can infer that something bad happens on the routing path. Maybe the resources are exhausted in some bottleneck links or nodes. Maybe the end-to-end delay of the flow is beyond its tolerated upbound. In this way, an optimal routing path can be found to achieve load balance and avoid congestion.

V. PROPOSED ALGORITHM

In this section, we propose the novel optimal VNF chaining and adaptive scaling algorithm (VNF-CAS). It is composed of two parts, the first part is the optimal VNF chaining algorithm (VNF-C). When executing VNF-C, for each SFC flow, the candidate VNFs are selected and rearranged in order. The original network graph is transformed into a multi-layer graph. Based on the defined latency cost on each virtual link of multi-layer graph, the optimal path can be obtained through a modified shortest path (SP) algorithm. Considering the capacity limitation in the distributed cloud edge node. The second part is a VNF resource allocation scheme (VNF-AR) to adjust the capacity of VMs according to different traffic demands, We formulate the resource allocation problem as a convex optimization model. The optimal allocated capacity of different VMs can be derived by the KKT conditions. Orchestrators are responsible to give instructions to re-allocate the CPU resource of each VM based on the optimal result. In this algorithm, the network traffic is assumed to be steady without huge fluctuations, every SFC request is handled one by one. Details of VNF-CAS are presented in the following subsections.

A. OPTIMAL VNF CHAINING ALGORITHM (VNF-C)

1) CONSTRUCTING SERVICE FUNCTION GRAPH

In order to solve the optimal selection and chaining problem of SFC, instead of finding exact numerical solutions by an analytical method which suffers from combinatorial complexity and high time complexity, we consider using the approach of constructing the Service Function Graph. It is modified from the Route Segmentation [16] in finer granularity at VM-level. For each SFC flow S_i, the Service Function Graph is constructed as a multi-layer directed graph $\mathbb{G}_i = (\mathbb{V}_i, \mathbb{E}_i)$, where $\mathbb{V}_i$ is the set of virtual nodes and $\mathbb{E}_i$ is the set of virtual links.

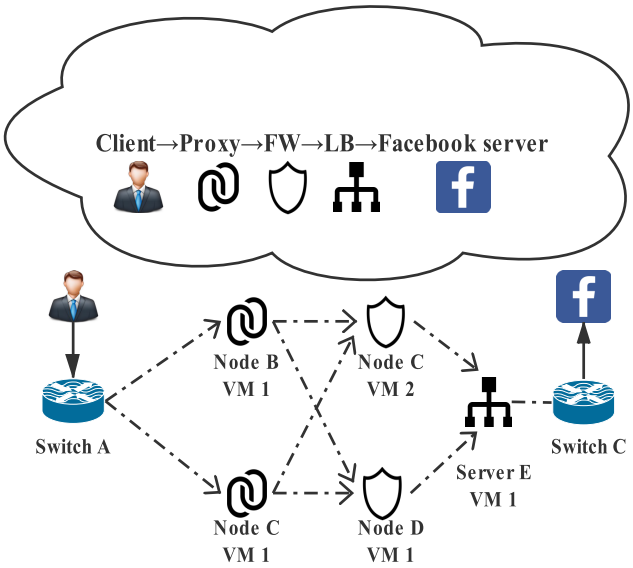


FIGURE 2. Service function graph.

Definition 1 (Service Function Graph): The Service Function Graph $\mathbb{G}_i = (\mathbb{V}_i, \mathbb{E}_i)$ associated with an SFC flow is constructed as follows:

1. The ingress node o_i is selected and arranged in the 1^{st} layer as the 1^{st} virtual node of the Graph.
2. According to the VNF types in $\varphi_i = \{\varphi_{i1}, \varphi_{i2}, \ldots, \varphi_{il}\}$. The VMs that host the j^{th} type of VNF φ_{ij} are selected and placed in the $(j+1)^{th}$ layer sequentially. The operation is repeated until the VMs hold the final VNF φ_{il} have been placed.
3. The egress node d_i is selected and arranged in the $(l+2)^{th}$ layer as the last virtual node of the Graph.
4. There are arcs between every virtual node in adjacent layers which represent the virtual links. The arc direction is from virtual node in the last layer to virtual node in the next layer.

Figure 2 shows the example of Service Function Graph with a service function chain S_i. The substrate network of the example is the same in Fig.1. It can be seen that there are 4 available routing paths for the SFC flow composed of the Proxy, FW and LB. The Routing Graph $\widetilde{\mathbb{G}}_i = \left(\widetilde{\mathbb{V}}_i, \widetilde{\mathbb{E}}_i\right)$ is the path with the least cost in $\mathbb{G}_i = (\mathbb{V}_i, \mathbb{E}_i)$. Assuming the virtual routing path with minimum resource cost is Switch A ⇒ Node C VM1 ⇒ Node C VM2 ⇒ Node E VM1 ⇒ Switch C. It can be mapped in Switch A ⇒ MDC C ⇒ Switch B ⇒ MDC E⇒ Switch C in the substrate network if all the constraints are satisfied.

Considering using SP to solve the objective function in (13). The estimated latency cost of substrate network should be transformed into the weight of virtual links in the Service Function Graph. For each SFC flow S_i, the weight of the virtual link from $\mathbb{v}(j)$ to $\mathbb{w}(j+1)$ is denoted as $\mathbb{C}_{i,\mathbb{v}(j),\mathbb{w}(j+1)}$ in (14). Here, $\mathbb{v}(j)$ represents virtual node $\mathbb{v} \in \mathbb{V}_i$ in the j^{th} layer and $\mathbb{w}(j+1)$ represents virtual node $\mathbb{w} \in \mathbb{V}_i$ in the next $(j+1)^{th}$ layer. The binary variable $z_{i,vw}^{\mathbb{v}(j),\mathbb{w}(j+1)}$ indicates whether virtual link $(\mathbb{v}(j), \mathbb{w}(j+1)) \in \mathbb{E}_i$ traverses

through link $(v, w) \in \mathcal{E}$. Noting that since there exists special case when adjacent required VNFs are mapped into one server node. (*e.g.* Path: Switch A⇒Node C VM1⇒Node C VM2⇒Node E VM1⇒Switch C). $z_{i,vw}^{\mathbb{v}(j),\mathbb{w}(j+1)}$ is assumed to be zero when adjacent virtual nodes $\mathbb{v}(j)$ and $\mathbb{w}(j+1)$ are mapped into the same server node. Furthermore, a mapping function $\mathcal{F}(\cdot)$ is defined in (15) transforming the virtual node $\mathbb{v}(j)$ to its corresponding VM's load utilization.

$$\mathbb{C}_{i,\mathbb{v}(j),\mathbb{w}(j+1)} = \sum_{(v,w)\in\mathcal{E}} \left(\frac{\mathrm{Dis}(v,w)}{c_{medium}} + \frac{\eta_{vw}}{1-\eta_{vw}} \right) \cdot z_{i,vw}^{\mathbb{v}(j),\mathbb{w}(j+1)} + \frac{1}{2} \cdot t_{proc} \cdot \left(\frac{\mathcal{F}[\mathbb{v}(j)]}{1-\mathcal{F}[\mathbb{v}(j)]} + \frac{\mathcal{F}[\mathbb{w}(j+1)]}{1-\mathcal{F}[\mathbb{w}(j+1)]} \right) \tag{14}$$

$$\mathcal{F}[\mathbb{v}(j)] = \begin{cases} \eta_v^m & 2 \le j \le l+1 \\ 0 & \text{otherwise} \end{cases} \tag{15}$$

Based on the weight of virtual links, the virtual path with the minimum resource cost can be obtained by executing the shortest path algorithm in the Service Function Graph. Therefore, the Service Function Graph has the benefit of simplifying the substrate network topology, which improves the efficiency of path computation. The pseudocode of VNF-C is presented in Algorithm 1.

Algorithm 1 VNF-C

Input: Network topology: $\mathcal{G} = (\mathcal{V}, \mathcal{E})$;
Network utilization: η_v^m, η_{vw};
Commodity with SFC request: S_i;
Output: Service Function Graph $\mathbb{G}_i = (\mathbb{V}_i, \mathbb{E}_i)$;

Initialize: k = 2 ;
$\mathbb{v}_{i,1} = o_i$;
for $j = 1 : l$ **do**
  for *each* $v \in V_{se}$ **do**
    $v, m \leftarrow$ find VMs on v that holds the same type of φ_{ij};
    if $v, m \neq \emptyset$ **then**
      $\mathbb{v}_{i,k} \leftarrow \{v, m\}$;
      k = k + 1;
$\mathbb{v}_{i,k} = d_i$;
for $j = 1 : l + 1$ **do**
  for *each virtual node* $\mathbb{w}$ *in* $(j+1)^{th}$ *layer* **do**
    for *each virtual node* $\mathbb{v}$ *in* j^{th} *layer* **do**
      $(\mathbb{v}, \mathbb{w}) \leftarrow$ Connect $\mathbb{v}$ to $\mathbb{w}$;
      $\mathbb{C}_{i,\mathbb{v}(j),\mathbb{w}(j+1)} \leftarrow$ Calculate the link weight of $\mathbb{G}_i$ according to Eq.(14)-(15);
return Service Function Graph $\mathbb{G}_i = (\mathbb{V}_i, \mathbb{E}_i)$

2) COMPLEXITY ANALYSIS

In VNF-C, Firstly the complexity of calculating the resource cost on nodes and links is at most $O(|\mathcal{V}| + |\mathcal{E}|)$. Since the construction of the Service Function Graph needs to copy the avaliable function nodes at each layer. Considering the worst case, all types of VNFs can be deployed on every server node. Therefore, there are no more than $l_{\max}|\mathcal{V}_{se}| + 2$ virtual nodes and $2|\mathcal{V}_{se}| + (l_{\max} - 1)|\mathcal{V}_{se}|^2$ virtual links on the Service Function Graph. l_{max} indicates the maximum number of the VNFs in the SFC commodity. In general, the time complexity of the SP algorithm (*e.g.* Dijkstra algorithm) is $O(|\mathcal{E}| + |\mathcal{V}|\log|\mathcal{V}|)$ on $\mathcal{G} = (\mathcal{V}, \mathcal{E})$. In the Service Function Graph, the weight of virtual links are calculated through the SP with complexity of $O\left(|\mathcal{V}_{se}|^2(|\mathcal{E}| + |\mathcal{V}|\log|\mathcal{V}|)\right)$. So the complexity of VNF-C is at most $O\left(|\mathcal{V}_{se}|^2(|\mathcal{E}| + |\mathcal{V}|\log|\mathcal{V}|)\right)$.

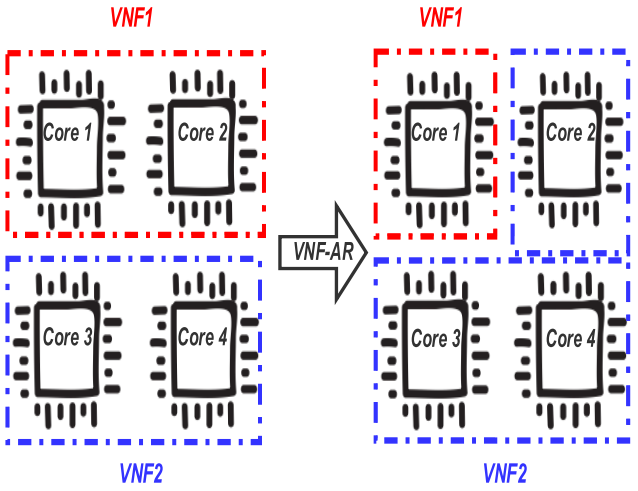


FIGURE 3. CPU resource allocation among VNFs.

B. ADAPTIVE VNF RESOURCE ALLOCATION (VNF-AR)

Due to the processing resource sharing of hardware among different VNFs, the unbalanced resource sharing results in congestion and additional processing latency. VNF-AR focuses on the efficient CPU resource allocation on VMs according to the traffic demands.

1) OPTIMAL RESOURCE ALLOCATION FORMULATION

In each server node $v \in \mathcal{V}_{se}$, the CPU cores are shared among VNFs hosted by different VMs. Fig.3 shows a simple example of a 4-core CPU are shared by 2 different VNFs. Alternatively, if the VNF2 requires more computational resources, while the VNF1 requires few computational resources. The server node should have an adaptive resource allocation scheme to balance the load, which can adjust capacity allocation for different VMs according to VNF's traffic demand. (*i.e.* Re-allocating the CPU resources of VNF1 from 2 cores to one core, but increasing the CPU resources of VNF2 from 2 cores to 3 cores).

VNF-AR formulates this problem into a convex optimization with nonlinear programming. The objective function is defined to minimize the estimated processing delay cost for each server node $v \in \mathcal{V}_{se}$:

$$\min_{C_v^1, C_v^2, \ldots, C_v^{M_v}} D_v^{proc} = \sum_{m=1}^{M_v} \widetilde{D_{v,m}^{proc}}\left(C_v^m\right) \tag{16}$$

Subject to: $\widetilde{D_{v,m}^{proc}}\left(C_v^m\right)$

$$= \frac{\overline{C_v^m}}{C_v^m - \overline{C_v^m}} \cdot t_{proc} \quad 1 \le m \le M_v \tag{17}$$

$$0 \le \eta_v^m < 1, \quad \eta_v^m = \frac{\overline{C_v^m}}{C_v^m} \quad 1 \le m \le M_v \tag{18}$$

$$\sum_{m=1}^{M_v} C_v^m = C_v \tag{19}$$

In VNF-AR, We use $\overline{C_v^m}$ to represent the possessed CPU resource of the m^{th} VM on server node $v \in \mathcal{V}_{se}$. It is a visible metric that can be measured and collected on each node. This optimization problem is a convex optimization, the reason is as follow: Firstly, the objective function $\widetilde{D_{v,m}^{proc}}$ is a convex function which is already proved and used in [36]. The sum of $\widetilde{D_{v,m}^{proc}}$ over VMs is also a convex function. Secondly, Inequality constraint function in (18) is convex. And the equality constraint in (19) is affine. According to the theorems in convex optimization, considering the dual to this problem, we associate dual variables μ for the constraint (17), λ_1 and λ_2 for the constraint (18). The Lagrange dual function is defined as follow:

$$g\left(\lambda_1, \lambda_2, \mu\right) = \inf_{C_v^m} \left(\begin{array}{c} \sum_{m=1}^{M_v} \frac{\overline{C_v^m}}{C_v^m - \overline{C_v^m}} \cdot t_{proc} - \\ \sum_{m=1}^{M_v} \left[\lambda_{1m} C_v^m + \lambda_{2m}\left(C_v^m - \overline{C_v^m}\right)\right] \\ + \mu \cdot \left(\sum_{m=1}^{M_v} C_v^m - C_v\right) \end{array} \right) \tag{20}$$

For any convex optimization, the KKT conditions provide necessary and sufficient conditions for optimality. In VNF-AR, $C_v^{m^*}$ and $\left(\lambda_1^*, \lambda_2^*, \mu^*\right)$ that satisfy the KKT conditions are primal and dual optimal, and have zero duality gap. By applying the KKT condition:

It can be derived that $C_v^{m^*}$ is optimal if it does not violate the constraints in (17)-(19) and there exist $\lambda_1, \lambda_2 \in \mathbb{R}^{M_v}$ and $\mu \in \mathbb{R}$ satisfy:

$$\lambda_1 \ge 0, \lambda_2 \ge 0 \tag{21}$$

$$\lambda_{1m} \cdot C_v^{m^*} = 0 \quad 1 \le m \le M_v \tag{22}$$

$$\lambda_{2m}\left(C_v^{m^*} - \overline{C_v^m}\right) = 0 \quad 1 \le m \le M_v \tag{23}$$

$$\frac{\overline{C_v^m} \cdot t_{proc}}{\left(C_v^{m^*} - \overline{C_v^m}\right)^2} + \lambda_{1m} + \lambda_{2m} = \mu \tag{24}$$

Considering the practical definition of C_v^m should be a positive number, $\lambda_{1\,m}^*$ and $\lambda_{2\,m}^*$ are assumed to be 0. Thus, the optimal C_v^{m*} is given by:

$$C_v^{m^*} = \overline{C_v^m} + \sqrt{\frac{\overline{C_v^m} \cdot t_{proc}}{|\mu|}} \tag{25}$$

Combining the capacity constraint in (19), the optimal μ^* can be derived as well.

$$\mu^* = \left(\frac{\sum_{m=1}^{M_v} \sqrt{\overline{C_v^m} \cdot t_{proc}}}{C - \sum_{m=1}^{M_v} \overline{C_v^m}} \right)^2 \tag{26}$$

In the end, the final optimal $C_v^{m^*}$ can be obtained if we substitute μ^* for μ in (25).

2) COMPLEXITY ANALYSIS

The interior-point method is applied to calculate the optimal allocated capacity. For each server node, the time complexity of computation is $O\left(\sqrt{M_v}\log\left(M_v\right)\right)$. During every update of the network status, the complexity of the VNF resource optimal allocation is at most $O\left(|\mathcal{V}_{\text{se}}|\sqrt{M}\log(M)\right)$ where M indicate the maximum VM number.

C. OPTIMAL VNF CHAINING AND ADAPTIVE SCALING ALGORITHM(VNF-CAS)

In VNF-CAS, based on the mentioned two parts (*i.e.* VNF-C and VNF-AR), an efficient resource-aware and load balancing VNF embedding algorithm is proposed. For each SFC, Service Function Graph gives a simple perspective to describe the SFC constraints. A modified shortest path algorithm can be executed to obtain the optimal path with the minimum resource cost. Furthermore, by accumulating a number of traffic requests, the capacity of VMs on the server node can be scaled after serving every batch of traffic T.

1) ALGORITHM DESCRIPTION

The pseudocode of VNF-CAS is described in Algorithm 2. In general, given the network parameters and SFC commodities, VNF-CAS finds a path for every SFC to chain required VNFs. Firstly, we initialize the capacity scaling batch size T and re-routing threshold coefficient σ. After accumulating every batch of traffic, the server node can scale its VMs' capacity to accommodate new SFC requests. Line 3 is to prune the nodes and links that can't accommodate the new incoming SFC request. In line 4, VNF-C is used to construct corresponding Service Function Graph $\mathbb{G}_i = (\mathbb{V}_i, \mathbb{E}_i)$, the resource cost $\mathbb{C}_{i,\mathbb{v}(j),\mathbb{w}(j+1)}$ of each virtual link is calculated. In line 5, the SP algorithm(*e.g.* Dijkstra algorithm) based on the above weights is executed to obtain the Routing Graph $\widetilde{\mathbb{G}}_i = \left(\widetilde{\mathbb{V}}_i, \widetilde{\mathbb{E}}_i\right)$. Considering the objective function with minimum embedding cost, the optimal solutions of variables $z_{i,v,m}^{\mathbb{v},\mathbb{w}}$ and $z_{i,vw}^{\mathbb{v},\mathbb{w}}$ are obtained by transforming the Routing Graph $\widetilde{\mathbb{G}}_i = \left(\widetilde{\mathbb{V}}_i, \widetilde{\mathbb{E}}_i\right)$ into the substrate network $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ in line 6. From line 7 to line 12, the selected path must be checked whether it satisfies all the constraints in equation (1)-(9). If all the constraints are satisfied in the physical path, this SFC request is accepted successfully and occupies the resource in the substrate network. Then the network resource utilization status (*i.e.* η_v, η_v^m and η_{vw}) are updated after the SFC being embedding in line 8. Otherwise, if the solutions violate any constraints, the SFC request is denied to be accepted in

Algorithm 2 VNF-CAS

Input: Network topology: $\mathcal{G} = (\mathcal{V}, \mathcal{E})$;
Network capacity: C_v, C_v^m, C_{vw};
Network utilization: η_v, η_v^m, η_{vw};
Commodity with SFC request: S_i;
Output: Selected path π_i of every $S_i \in \Omega$;

1 **Initialize:** T, σ;
2 **for** *each* S_i *in* Ω **do**
3 $G = (V, E) \leftarrow$ Pruning the nodes, links with fewer resources than the demand of S_i;
4 $\mathbb{G}_i = (\mathbb{V}_i, \mathbb{E}_i) \leftarrow$ Construct Service Function Graph of S_i according to ***VNF-C***;.
5 $\widetilde{\mathrm{G}}_i = \left(\widetilde{\mathrm{V}}_i, \widetilde{\mathrm{E}}_i\right) \leftarrow$ Execute the shortest-path algorithm on $\mathbb{G}_i$;
6 $\pi_i \leftarrow$ Transform $\widetilde{\mathrm{G}}_i$ from $\mathbb{G}_i$ to $\mathcal{G} = (\mathcal{V}, \mathcal{E})$;
7 **if** π_i *satisfy the constraints of Eq.(1)-(9)* **then**
8 $|S| = |S| + 1 \leftarrow S_i$ is accepted successfully;
9 $\eta_v, \eta_v^m, \eta_{vw} \leftarrow$ Update the network resource utilization;
10 **return** π_i;
11 **else**
12 **return** S_i routing failed;
13 **if** $|S| == p \cdot T (p = 1, 2, 3, \dots)$ **then**
14 **for** *each* $v \in \mathcal{V}_{se}$ **do**
15 **for** m $= 1 : M_v$ **do**
16 $C_v^m \leftarrow$ Adjust C_v^m according to ***VNF-AR***;
17 $\eta_v, \eta_v^m \leftarrow$ Update the resource utilization;
18 **for** *each* S_i *in* $|S|$ **do**
19 **if** *Esitimated latency of* $S_i \geq \sigma R_i^D$ **then**
20 $\pi_i \leftarrow$ Repeat procedures of line 3 to 6;
21 **if** π_i *satisfy the constraints of Eq.(1)-(9)* **then**
22 S_i is accepted successfully;
23 **Update** π_i;
24 **else**
25 $|S| = |S| - 1$;
26 **return** S_i routing failed;
27 $\eta_v, \eta_v^m, \eta_{vw} \leftarrow$ Update the resource utilization;

line 12. In lines 13-17, when the number of existed SFC requests $|S|$ reaches the capacity scaling batch size, VNF-AR is applied in line 16 to find the optimal capacity for each VM. During the same time, Lines 18-27 are aimed to re-calculate the routing path π_i for the existed SFC requests whose estimated delay exceeds the threshold. There is a threshold cofficient σ in view of the gap between the estimated delay and the actual delay. In this paper, we assume the capacity scaling batch size T as a fix number chosen by network operators. Furthermore, Adjusting T dynamically according to the fluctuations of traffic volumes is an interesting research challenge beyond the scope of this work. Maybe we will pursue to solve it in the future work.

2) COMPLEXITY ANALYSIS

VNF-CAS is composed of VNF-C and VNF-AR.For each SFC flow, the complexity of constructing Service Function Graph in VNF-C is $O\left(|\mathcal{V}_{se}|^2 (|\mathcal{E}| + |\mathcal{V}| \log |\mathcal{V}|)\right)$. Then the SP is executed to obtain the optimal path, the complexity of the VNF chaining is thus $O\left(l_{\max} |\mathcal{V}_{se}|^2\right)$. $l_{\max}$ indicates the maximum number of the VNFs in the SFC commodity. The complexity of transformation and resource update is $O(1)$. Because $l_{\max}$ is quite a small number when compared with $|\mathcal{E}|$. Taking the total number of the SFC commodities Ω into account. The complexity of the VNF chaining and re-routing part is at most $O\left(|\Omega| |\mathcal{V}_{se}|^2 (|\mathcal{E}| + |\mathcal{V}| \log |\mathcal{V}|)\right)$. As for the adaptive resource allocation, the update times is at most $\left\lfloor \frac{|\Omega|}{T} \right\rfloor$. During every update of the network status, the complexity of VNF resource optimal allocation is at most $O\left(|\mathcal{V}_{se}| \sqrt{M} \log(M)\right)$, where M represents the maximum VM number. Therefore, the complexity of the adaptive scaling is $O\left(\left\lfloor \frac{|\Omega|}{T} \right\rfloor |\mathcal{V}_{Se}| \sqrt{M} \log(M)\right)$. Finally, by combaining the two parts, the complexity of VNF-CAS at the worst situation is $O\left\{|\Omega|| |V_{se}| \left[\sqrt{M} \log(M) + |V_{se}| (|\mathcal{E}| + |\mathcal{V}| \log |\mathcal{V}|)\right]\right\}$.

In addition, the complexity of VNF-C can be further reduced by merging the SFCs that belong to the same Ω_k into one group. During the construction of the Service Function Graph, the service function chains that have the same ingress node o_i, egress node d_i and VNF forms usually belong to the same Service Function Graph. There is no need to calculate $\mathbb{G}_i = (\mathbb{V}_i, \mathbb{E}_i)$ for every SFC and it can be just calculated one time for each group.

VI. PERFORMANCE EVALUATION

In this section, we demonstrate the performance of our proposed algorithms. Firstly, the simulation configuration is illustrated. Then, we compare the performance of VNF-CAS with modified VNF-C without adaptive scaling and three other existing algorithms [16]–[18]. All the simulations are implemented with MATLAB 2018A and performed on a computer with Intel(R) Core(TM) i5-8250 CPU @ 1.60 GHz and 8 GB RAM.

A. SIMULATION CONFIGURATION

In our simulation, the network topology that we use is two WAN topologies in SNDlib [37] and a data center network [38]. (1) Abilene is a small-sized network topology with 12 nodes and 15 links. (2) Germany50 is a middle-sized network topology which is composed of 50 nodes and 88 links. (3) EDU1 is a university data center network with 22 device nodes and 41 links.

Similarly in [30], the server nodes in WAN topologies are selected according to the node degree. In Abilene topology, we select the top 40% nodes as the server nodes with the

TABLE 2. Network parameters settings.

Parameters	Value		
Network topology	Abilene	Germany	EDU1
Node number	12	50	22
Link number	15	88	41
Server number	5	15	18

Parameters	Description	Value
C_v	CPU capacity on server node $v \in \mathcal{V}_{se}$	15000 MIPs
C_{vw}	Bandwidth capacity of link $(v, w) \in \mathcal{E}$	1000 Mbps
M_v	VM number on each server node $v \in \mathcal{V}_{se}$	4
C_v^m	Initial CPU capacity of each VM on server node $v \in \mathcal{V}_{se}$	3750 MIPs
d_{tx}	Transmission delay on links	10 μs
t_{proc}	Per-packet processing time	1 ms
c_{medium}	Propagation speed of links	$5\mu s/km$
T	Capacity scaling batch size	50
σ	Re-routing threshold coefficient	1.5

number of 5. And in Germany topology, we select the top 30% nodes as the server nodes with the number of 15. For the data center network, we attach each edge/ToR device to a server node. For simplicity, the CPU capacity of each server node $v \in \mathcal{V}_{se}$ is set uniformly as 15000 MIPs. The VM numbers which are activated on the server nodes are all set to be 4. Therefore, the initial CPU capacity of each VM on server node is set as 3750 MIPs. Besides, the bandwidth capacity of each link is uniformly set as 1000 Mbps.

Considering the latency and processing parameters of the estimated delay cost, the transmission delay d_{tx} in Eq.(11) is set as $10\mu s$ and the per packet processing time in Eq.(12) is set as 1 ms [39]. In addition, the propagation speed of links is set to be $5\mu s/km$. The propagation delay $\widehat{D_{vw}^{prop}}$ of each link is derived from the real length Dis(v, w) in the used dataset. During the updated re-allocation of the capacity, the scaling batch size T is set as 50. All the network parameters used in the simulation are presented in TABLE 2.

In this simulation, all the node pairs generate traffic according to the avaliable traces in [37], [38], which means the ingress and egress nodes of the SFC request is the same as the distribution in real traces. Moreover, we consider six different VNFs that can be chained to provide four different network services [40], [41] as in TABLE 3 (*i.e.* Web Service, VoIP, Video Streaming and Online Gaming) And they are generated according to the corresponding probability in [41]. TABLE 3 shows the performance requirements in terms of bandwidth consumption R_i^{bw} and maximum tolerated latency R_i^D for each SFC. In our performance evaluation, for simplicity, we assume every link of the SFC's path occupies the same bandwidth. As for the CPU demands of VNFs, we assume the CPU demands of different VNFs in one SFC are the same.

TABLE 3. SFC parameters settings [41].

SFC	Chained VNFs	Tolerated latency	Bandwidth consumption	%traffic
Web Service	NAT-FW-TM-WOC-IDPS	500 ms	100 kb/s	18.2%
VoIP	NAT-FW-TM-FW-NAT	100 ms	64 kb/s	11.8%
Video Streaming	NAT-FW-TM-VOC-IDPS	100 ms	4 Mb/s	69.8%
Online Gaming	NAT-FW-VOC-WOC-IDPS	60 ms	4 Mb/s	0.2%

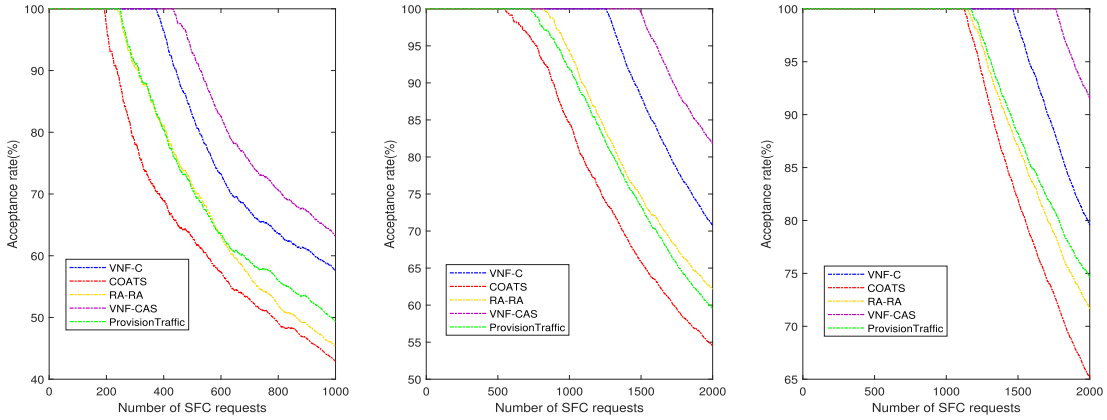
For each SFC, we set $R_i^{cpu} = 10 \cdot R_i^{bw}$ with probability 90% and $R_i^{cpu} = 10 \cdot R_i^{bw} \ln R_i^{bw}$ with probability 10% which is used in [12]. By combining the synthetical generated SFC types and real traffic traces, we get the final experimental data in our performance evaluation.

B. INTRODUCTION OF THE COMPARED ALGORITHM

In the simulation, the performance of VNF-CAS is compared with four other algorithms. All of the three algorithms are executed by constructing the multi-layer graph. The first one is the modified VNF-C algorithm without capacity scaling which is already introduced in this paper. The second one is the algorithm named Competitive Online Algorithm for Traffic Steering (COATS) proposed in [16]. In COATS, the resource cost is calculated based on the remaining bandwidth on the link. With the lazy dual update, COATS selects the shortest path with the lowest cost to route a flow with SFC request.

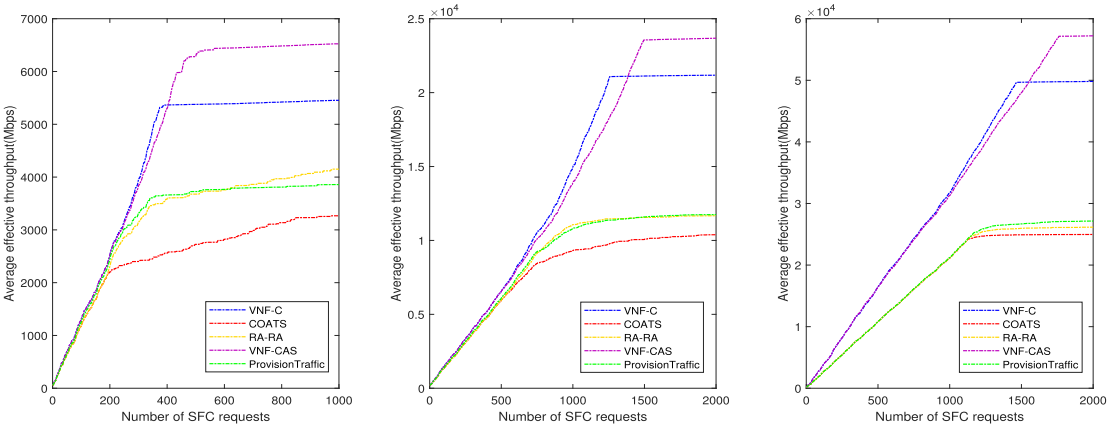
The third algorithm named Resource Aware Routing Algorithm (RA-RA) is introduced in [17]. In RA-RA, the flows are classified into different kinds based on the bandwidth and CPU consumptions. Different kind of flow has different relative cost defined on the virtual link. Different from the fine-grained VM-level schedule in VNF-CAS, the resource cost in is only defined at the node-level. Considering the experimental data in our paper, we choose the bandwidth threshold as 1 Mb/s and CPU consumption threshold as 10 MIPs to differentiate flows. Therefore, the SFC commodities in the simulation are classified into four kinds. And their relative costs are defined according to the corresponding equations. Finally, a modified K-shortest path algorithm is used to find the available path.

The fourth algorithm named ProvisionTraffic is proposed in [18]. ProvisonTraffic adopts the Viterbi algorithm to place and steer VNF instances to minimize the OPEX and resource fragmentation. Based on a multi-stage graph (similar to Service Function Graph), the link cost is related to the VNF deployment cost, server's energy consumption, traffic forwarding cost and penalty for Service Level Objective violation. We use the server data in [18] to caculate the corresponding cost.



(a) Abilene topology (b) Germany topology (c) EDU1 topology

FIGURE 4. The performance evaluation of acceptance rate in different topologies.



(a) Abilene topology (b) Germany topology (c) EDU1 topology

FIGURE 5. The performance evaluation of average effective throughput in different topologies.

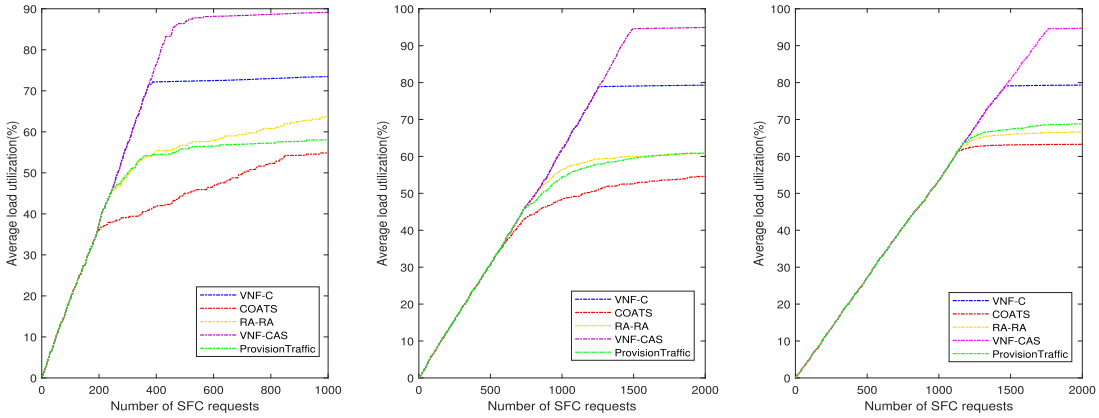
C. SIMULATION RESULTS

1) COMPARISON OF ACCEPTANCE RATE, AVERAGE EFFECTIVE THROUGHPUT AND AVERAGE LOAD UTILIZATION

Based on the above simulation configuration, we first compare VNF-CAS with VNF-C, COATS, RA-RA and ProvisionTraffic in terms of average acceptance rate, load utilization and effective throughput in different topologies.

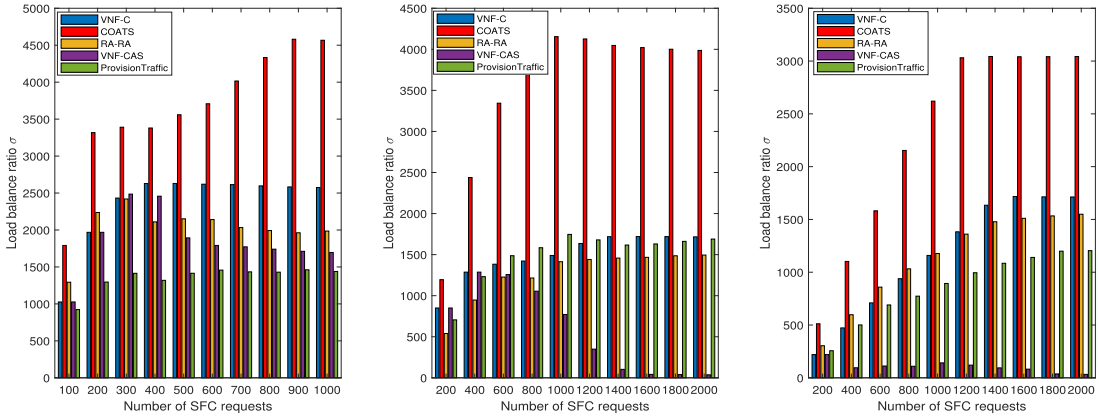
Figure 4 provides an overview of the acceptance rate evaluation of these algorithms in different topologies. The acceptance rate reflects the served flow number accounting for the total arriving flow number. It is an important metric to evaluate system capacity. We record the actual acceptance rate after handling every new SFC request. In Fig.4(a), It can be seen from the chart that VNF-CAS achieves the highest acceptance rate among the five algorithms, which is about 7 % higher than VNF-C, 15% higher than ProvisionTraffic, 18% higher than RA-RA and 20% higher than COATS in heavy traffic load. Similarly, VNF-CAS also performs best in Fig.4(b) and Fig.4(c). The acceptance rate of RA-RA is close to ProvisionTraffic in light traffic load. After handling a number of SFC requests, RA-RA performs better than ProvisionTraffic in Germany but performs worse than ProvisionTraffic in Abilene and EDU1. The COATS performs worst among these algorithms, the main reason is that though COATS can balance the consumption of bandwidth. But it neglects the consumption of server node and latency, which can result in the resource exhaustion on the node with few remaining CPU resources. While RA-RA, ProvisionTraffic, and VNF-C all consider the CPU consumption on the server node. RA-RA mainly considers the load of server node when serving the dense flow. ProvisionTraffic considers the server's energy consumption and VNF-C considers the VM load to balance the consumption of CPU. As for VNF-CAS, since it is built on VNF-C and further adjusts the VM's capacity periodically. This algorithm can accommodate more SFC flows and performs best.

Figure 5 describes the performance evaluation of the average effective throughput among the five algorithms in



(a) Abilene topology (b) Germany topology (c) EDU1 topology

FIGURE 6. The performance evaluation of average load utilization in different topologies.



(a) Abilene topology (b) Germany topology (c) EDU1 topology

FIGURE 7. The performance evaluation of load balance in different topologies.

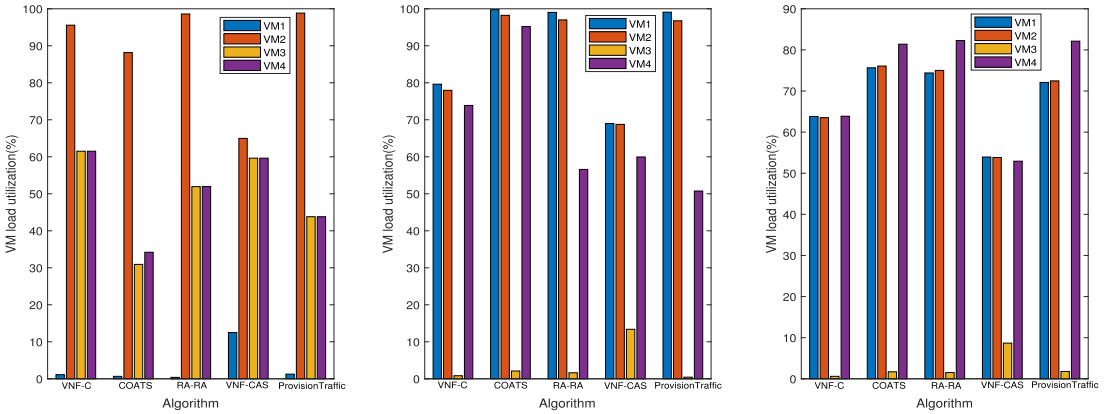
different topologies. The average effective throughput is the total occupied bandwidth of flows received successfully in the network. Since accepting more flows will take up more bandwidth, the average effective throughput is related to the acceptance rate of the SFC requests. As shown in Fig.5(a)(b)(c), the maximum throughput of VNF-CAS is nearly more than twice of COATS, which can reach nearly 6500 Mbps in Abilene, 24000 Mbps in Germany and 57000 Mbps in EDU1. Under light traffic load condition, the average effective throughput of VNF-C is more than that of VNF-CAS. The reason may be the adaptive capacity scaling causes the choice of shorter path in VNF-CAS which results in less consumption of the bandwidth. RA-RA and ProvisonTraffic basically achieve similar effective throughput under different loads, although there are small differences in the three topologies.

Figure 6 presents an evaluation of the average load utilization among these algorithms in different topologies. The average load utilization shows the average CPU utilization of all server nodes, which can reflect the maximum processing capacity of the network. In light traffic load condition, the performance of different algorithms can be the same because all the flows are accepted successfully. In extreme case, it's obvious that VNF-CAS ranks first with the average load utilization of 90% in the three topologies. Due to the lack of effective resource re-allocation mechanism, VNF-C, RA-RA, ProvisonTraffic, and COATS are just able to reach under 80% CPU utilization.

2) COMPARISON OF LOAD BALANCE RATIO AND VM LOAD UTILIZATION

Secondly, we compare VNF-CAS with COATS, RA-RA, VNF-C and ProvisionTraffic in regard to the node load balance ratio and VM load utilization of a specific node in different topologies.

In Fig.7, we evaluate the load balance ratio's variation when the SFC requests increase in different topologies. The load balance ratio σ is calculated as in Eq.(27). It reflects the load difference between different server nodes. The lower the load balance ratio is, the better the load balance of the algorithm. It can be seen in Fig.7(b)(c) that VNF-CAS goes



(a) Abilene topology(400) (b) Germany topology(1000) (c) EDU1 topology(1000)

FIGURE 8. The performance evaluation of VM load utilization in different topologies.

through an up-down fluctuation and gets the lowest σ in Germany and EDU1. In fact, all of the five algorithms have the ability to balance the load. However, COATS performs worst since it only considers the bandwidth consumptions. The performance of RA-RA outperforms that of VNF-C. The reason is that the relative cost defined in RA-RA mainly indicates the resource conditions on nodes and links. If the CPU resources on nodes are going to be exhausted, the relative resource costs will surge dramatically to avoid to be used. So RA-RA can achieve a good load balance at node-level. As for VNF-CAS, because the adaptive capacity scaling approach can achieve fine-grained load balance on VMs. VNF-CAS has the best performance in load balance under the heavy traffic load condition. But the performance of VNF-CAS in Fig.7(a) seems not better than ProvisionTraffic, ProvisionTraffic can balance the load based on the server's energy status. When the SFC forms are generated in proportion, the balance ratio may be related to the network topology and traffic volumes.

$$\sigma = \sqrt{\frac{1}{|\mathcal{V}_{se}|-1}\sum_{i=1}^{|V_{se}|}\left(C_{v_i}\cdot\eta_{v_i} - \frac{1}{\mathcal{V}_{se}}\sum_{j=1}^{|V_{se}|}C_{v_j}\cdot\eta_{v_j}\right)^2} \quad \forall v_i, v_j \in \mathcal{V}_{se} \qquad (27)$$

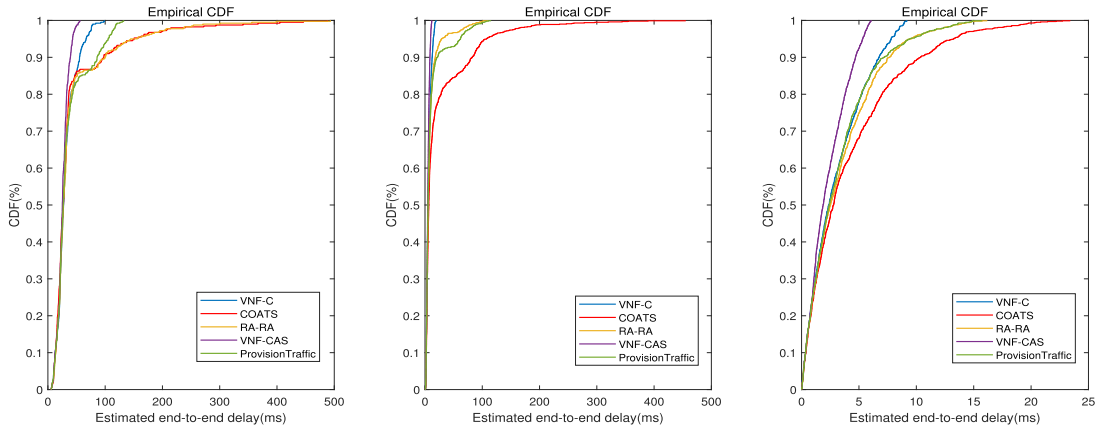
The performance of VM load utilization is evaluated in Fig.8. Since there are many VMs to choose from the network. Referring to TABLE 3, we choose the server nodes that host the VNF WOC in different topologies, which is prone to reflect the imbalance of VM load due to this VNF is rarely used. In Fig.8(a), we record the VM load utilization when 400 SFC requests are accepted. WOC is deployed in VM1 in advance. The utilization of VM1 is always around 1% among VNF-C, RA-RA, COATS and ProvisionTraffic, but VM2's load utilization comes close to 95% of these algorithms. Only in VNF-CAS, the utilization of VM1 is around 10%. Similarly in Fig.8(b)(c), the VM load utilization is depicted when 1000 SFC requests are accepted. With the help of VNF-AR, the utilization of VM3 that hosts WOC can reach 10% or so. While the other VMs' load utilization can witness a moderate decrease, which is conducive to avoid capacity bottlenecks. The performance evaluation proves that VNF-CAS has the ability to migrate the resources from the inactive VM to the frequently used VM. (i.e. In EDU1 topology, it migrates the residual resources from VM3 to other VMs so that the load utilization of VM1, VM2 and VM4 can be reduced).

3) COMPARISON OF ESTIMATED END-TO-END DELAY AND REMAINING BANDWIDTH

In the end, we compare VNF-CAS with COATS, RA-RA, VNF-C and ProvisionTraffic in the field of estimated end-to-end delay and remaining bandwidth in different topologies.

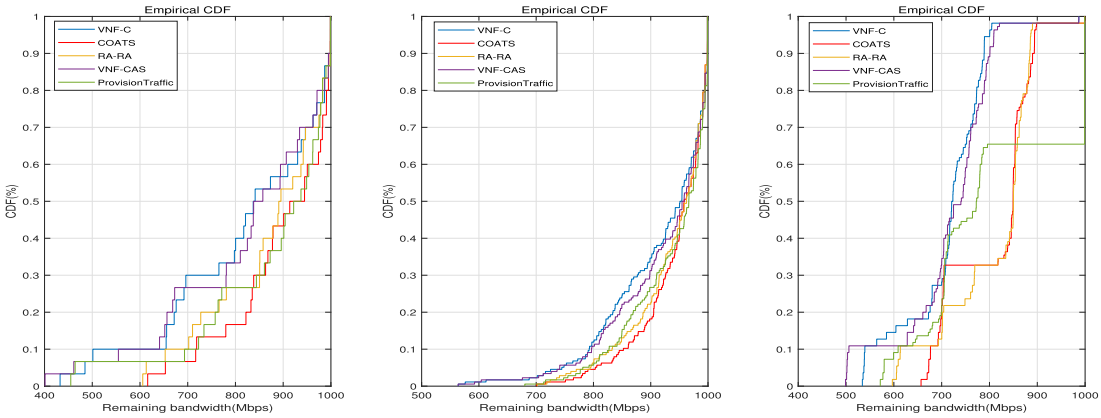
Figure 9 illustrates the CDF of the estimated end-to-end delay when 400 SFC requests are accepted in Abilene, 1000 SFC requests are accepted in Germany and EDU1. Among the five algorithms, VNF-CAS gets the best performance with the estimated latency under 10 ms in EDU1, 20 ms in Germany and 50 ms in Abilene. VNF-C also gets a good latency performance under 100 ms in different topologies. This is because the objective functions of VNF-CAS and VNF-C are both the estimated end-to-end delay when solving the SFC chaining problem. RA-RA also pays attention to avoid bottlenecks to reduce the queuing delay and processing delay. But COATS can only balance the bandwidth consumption which is beneficial to reduce a part of latency, so it performs worst in the simulation.

In Fig.10, the CDF of the average remaining bandwidth among the five algorithms is depicted. When receiving 400 SFC requests in Fig.10(a), the proportion of the links with remaining bandwidth less than 700Mbps is around 7% for COATS, 10% for RA-RA and ProvisionTraffic but around 27% for VNF-CAS and 30% for VNF-C. When receiving 1000 SFC requests in Fig.10(b), the proportion of the links



(a) Abilene topology(400) (b) Germany topology(1000) (c) EDU1 topology(1000)

FIGURE 9. The performance evaluation of estimated end-to-end delay in different topologies.



(a) Abilene topology(400) (b) Germany topology(1000) (c) EDU1 topology(1000)

FIGURE 10. The performance evaluation of remaining bandwidth in different topologies.

with remaining bandwidth between 800Mbps to 1000Mbps is about 90% for VNF-CAS and VNF-C, but 95% for the other algorithms. Only VNF-CAS and VNF-C have the remaining bandwidth less than 600Mbps in Germany. Similarly in Fig.10(c), only VNF-CAS and VNF-C have remaining bandwidth less than 550Mbps with a proportion of 10% in EDU1. It can be seen from the comparison that the proportion of the bottleneck links in VNF-CAS and VNF-C is higher than the other three algorithms. The distribution of the remaining bandwidth is more balanced in RA-RA and COATS. Because COATS mainly consider the bandwidth consumption to avoid congestion, RA-RA also considers to balancing the bandwidth consumption for elephant flows. While in VNF-C and VNF-CAS, more resource elements like latency are taken into account. They sacrifice the performance in bandwidth balance to get a better performance in the load balance. Our proposed method is more helpful to solve the load imbalance problem under the condition of high traffic volumes.

VII. CONCLUSION

In this paper, we study the optimal VNF chaining and resource allocation problem in the distributed cloud network. Based on the fine-grained VM scheduling in the multi-layer graph, the VNF-C algorithm is proposed to embed SFC requests with minimum estimated delay. Since the estimated delay cost is relevant to the resource status in the network. The NFV orchestrators are able to calculate the optimal path for each SFC request depending on the resource utilization. Moreover, considering the on-demand VNF placement is usually proactive and dynamic. The VNF-CAS algorithm with a fine-tune in VM capacity at each service provision time is proposed to efficiently make use of the CPU resource on server nodes. Performance evaluation results show that VNF-CAS can achieve approximately 90% average load utilization, and obtain performance improvement in acceptance rate, average effective throughput, VM load balance and estimated end-to-end delay compared with the existing algorithms in other literatures.

In the future, we plan to extend our work from the offline mechanism to online mechanism which is adapted to dynamic work loads. In the more complicated situation, the duration time of the SFC and the VM migration for VNF should be considered. In addition, In view of the high computation complexity to embed the SFC requests. A heuristic approach in the large-sized network should be designed to improve the network scalability.

REFERENCES

[1] *Network Functions Virtualization*. Accessed: Oct. 17, 2014. [Online]. Avaliable: https://portal.etsi.org/Portals/0/TBpages/NFV/Docs/NFV_White_Paper3.pdf

[2] R. Mijumbi, J. Serrat, J. Gorricho, N. Bouten, F. De Turck, and R. Boutaba, "Network function virtualization: State-of-the-art and research challenges," *IEEE Commun. Surveys Tuts.*, vol. 18, no. 1, pp. 236–262, 1st Quart., 2016.

[3] *SDN Architecture Overview*. Accessed: Oct. 11, 2014. [Online]. Avaliable: https://www.opennetworking.org/images/stories/downloads/sdn-resources/technical-reports/TR_SDN-ARCH-Overview-1.1-111120-14.02.pdf

[4] W. Xiao, W. Bao, X. Zhu, and L. Liu, "Cost-aware big data processing across geo-distributed datacenters," *IEEE Trans. Parallel Distrib. Syst.*, vol. 28, no. 11, pp. 3114–3127, Nov. 2017.

[5] J. Xu, B. Palanisamy, H. Ludwig, and Q. Wang, "Zenith: Utility-aware resource allocation for edge computing," in *Proc. IEEE Int. Conf. Edge Comput. (EDGE)*, Jun. 2017, pp. 47–54.

[6] D. Zhao, J. Ren, R. Lin, S. Xu, and V. Chang, "On orchestrating service function chains in 5G mobile network," *IEEE Access*, vol. 7, pp. 39402–39416, 2019.

[7] *Service Function Chaining (SFC) Architecture*. Accessed: Oct. 2015. [Online]. Available: http://www.rfc-editor.org/info/rfc7665

[8] H. Hantouti, N. Benamar, T. Taleb, and A. Laghrissi, "Traffic steering for service function chaining," *IEEE Commun. Surveys Tuts.*, vol. 21, no. 1, pp. 487–507, 1st Quart., 2018.

[9] S. Mehraghdam, M. Keller, and H. Karl, "Specifying and placing chains of virtual network functions," in *Proc. IEEE 3rd Int. Conf. Cloud Netw. (CloudNet)*, Oct. 2014, pp. 7–13.

[10] M. Jalalitabar, E. Guler, D. Zheng, G. Luo, L. Tian, and X. Cao, "Embedding dependence-aware service function chains," *IEEE/OSA J. Opt. Commun. Netw.*, vol. 10, no. 8, pp. 64–74, Aug. 2018.

[11] R. Cohen, L. Lewin-Eytan, J. S. Naor, and D. Raz, "Near optimal placement of virtual network functions," in *Proc. IEEE Conf. Comput. Commun. (INFOCOM)*, Apr. 2015, pp. 1346–1354.

[12] T. W. Kuo, B. H. Liou, K. C. Lin, and M. J. Tsai, "Deploying chains of virtual network functions: On the relation between link and server usage," *IEEE/ACM Trans. Netw.*, vol. 26, no. 4, pp. 1562–1576, Aug. 2018.

[13] K. Yang, H. Zhang, and P. Hong, "Energy-aware service function placement for service function chaining in data centers," in *Proc. IEEE Global Commun. Conf. (GLOBECOM)*, Dec. 2016, pp. 1–6.

[14] O. Soualah, M. Mechtri, C. Ghribi, and D. Zeghlache, "A green VNFs placement and chaining algorithm," in *Proc. IEEE/IFIP Netw. Oper. Manage. Symp. (NOMS)*, Apr. 2018, pp. 1–5.

[15] 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.

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

[17] J. Pei, P. Hong, K. Xue, and D. Li, "Resource aware routing for service function chains in SDN and NFV-enabled network," *IEEE Trans. Services Comput.*, to be published.

[18] F. Bari, S. R. Chowdhury, R. Ahmed, R. Boutaba, and O. C. M. B. Duarte, "Orchestrating virtualized network functions," *IEEE Trans. Netw. Service Manage.*, vol. 13, no. 4, pp. 725–739, Dec. 2016.

[19] M. Mechtri, C. Ghribi, and D. Zeghlache, "VNF placement and chaining in distributed cloud," in *Proc. IEEE 9th Int. Conf. Cloud Comput. (CLOUD)*, Jun. 2016, pp. 376–383.

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

[21] S. Khebbache, M. Hadji, and D. Zeghlache, "Virtualized network functions chaining and routing algorithms," *Comput. Netw.*, vol. 114, pp. 95–110, Feb. 2017.

[22] L. Tang, H. Yang, R. Ma, L. Hu, W. Wang, and Q. Chen, "Queue-aware dynamic placement of virtual network functions in 5G access network," *IEEE Access*, vol. 6, pp. 44291–44305, Aug. 2018.

[23] C. Ghribi, M. Mechtri, and D. Zeghlache, "A dynamic programming algorithm for joint VNF placement and chaining," in *Proc. ACM Workshop Cloud-Assisted Netw. (CAN)*, New York, NY, USA, 2016, pp. 19–24.

[24] A. Jarray and A. Karmouch, "Decomposition approaches for virtual network embedding with one-shot node and link mapping," *IEEE/ACM Trans. Netw.*, vol. 23, no. 3, pp. 1012–1025, Jun. 2015.

[25] O. Soualah, M. Mechtri, C. Ghribi, and D. Zeghlache, "An efficient algorithm for virtual network function placement and chaining," in *Proc. 14th IEEE Annu. Consum. Commun. Netw. Conf. (CCNC)*, Jan. 2017, pp. 647–652.

[26] A. Dwaraki and T. Wolf, "Adaptive service-chain routing for virtual network functions in software-defined networks," in *Proc. Workshop Hot Topics Middleboxes Netw. Function Virtualization (HotMIddlebox)*, New York, NY, USA, 2016, pp. 32–37.

[27] G. Sallam, G. R. Gupta, B. Li, and B. Ji, "Shortest path and maximum flow problems under service function chaining constraints," in *Proc. IEEE Conf. Comput. Commun. (INFOCOM)*, Apr. 2018, pp. 2132–2140.

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

[29] R. Zhou, "An online placement scheme for VNF chains in Geo-distributed clouds," in *Proc. IEEE/ACM 26th Int. Symp. Qual. Service (IWQoS)*, Jun. 2018, pp. 1–2.

[30] J. Pei, P. Hong, K. Xue, and D. Li, "Efficiently embedding service function chains with dynamic virtual network function placement in geo-distributed cloud system," *IEEE Trans. Parallel Distrib. Syst.*, vol. 30, no. 10, pp. 2179–2192, Oct. 2018.

[31] X. Fei, F. Liu, H. Xu, and H. Jin, "Adaptive VNF scaling and flow routing with proactive demand prediction," in *Proc. IEEE Conf. Comput. Commun. (INFOCOM)*, Apr. 2018, pp. 486–494.

[32] T. Buh, R. Trobec, and A. Ciglič, "Adaptive network-traffic balancing on multi-core software networking devices," *Comput. Netw.*, vol. 69, pp. 19–34, Aug. 2014.

[33] M. Savi, M. Tornatore, and G. Verticale, "Impact of processing-resource sharing on the placement of chained virtual network functions," *IEEE Trans. Cloud Comput.*, to be published.

[34] S. G. Kulkarni, W. Zhang, J. Hwang, S. Rajagopalan, K. K. Ramakrishnan, T. Wood, M. Arumaithurai, and X. Fu, "NFVnice: Dynamic backpressure and scheduling for NFV service chains," in *Proc. Conf. ACM Special Interest Group Data Commun. (SIGCOMM)*, New York, NY, USA, Aug. 2017, pp. 71–84.

[35] I. Cerrato, M. Annarumma, and F. Risso, "Supporting fine-grained network functions through intel DPDK," in *Proc. 3rd Eur. Workshop Softw. Defined Netw. (EWSDN)*, Sep. 2014, pp. 1–6.

[36] R. Gallager, "A minimum delay routing algorithm using distributed computation," *IEEE Trans. Commun.*, vol. COM-25, no. 1, pp. 73–85, Jan. 1977.

[37] S. Orlowski, R. Wessäly, M. Pióro, and A. Tomaszewski, "SNDlib 1.0—Survivable network design library," *Netw., Int. J.*, vol. 55, pp. 276–286, May 2010.

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

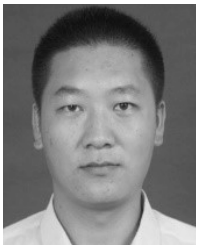
[39] R. Ramaswamy, N. Weng, and T. Wolf, "Characterizing network processing delay," in *Proc. IEEE Global Telecommun. Conf. (GLOBECOM)*, vol. 3, Nov./Dec. 2004, pp. 1629–1634.

[40] R. Shea, J. Liu, E. C.-H. Ngai, and Y. Cui, "Cloud gaming: Architecture and performance," *IEEE Netw.*, vol. 27, no. 4, pp. 16–21, Jul./Aug. 2013.

[41] A. Gupta, B. Jaumard, M. Tornatore, and B. Mukherjee, "A scalable approach for service Chain mapping with multiple SC instances in a wide-area network," *IEEE J. Sel. Areas Commun.*, vol. 36, no. 3, pp. 529–541, Mar. 2018.

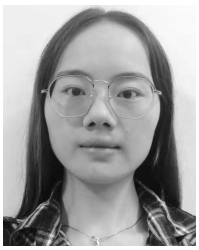
JIACHEN ZU received the B.S. degree in electronic science and technology from Shanghai Jiao Tong University, in 2017. He is currently pursuing the Ph.D. degree with the Department of Network Engineering, Army Engineering University, Nanjing, China. His research interests include NFV, service function chain, and satellite networks.

GUYU HU received the B.S. degree in radio communication from Zhejiang University, Hangzhou, China, in 1983, and the M.Sc. degree in computer application technology and the Ph.D. degree in communications and information systems from the Nanjing Institute of Communication, Nanjing, China, in 1989 and 1992, respectively. In 1990, he devotes to the research on network management. Since 1997, he has been a Full Professor with Army Engineering University, Nanjing. His research interests include computer networks, maintenance and administration of the satellite networks, and intelligent network management.



YANG WU received the B.S. degree in computer science and technology from the PLA University of Science and Technology, in 2015, and the M.Sc. degree in computer science and technology from Army Engineering University, Nanjing, China, in 2017, where he is currently pursuing the Ph.D. degree with the Department of Network Engineering. His research interests include satellite networks and computer networks.

DONGSHENG SHAO received the B.S. degree in electronic science and technology from Shanghai Jiao Tong University, in 2017. He is currently pursuing the master's degree with the Department of Network Engineering, Army Engineering University, Nanjing, China. His research interests include satellite networks and network management.



JIAJIE YAN received the B.S. degree in the Internet of Things from the Yunnan University of Finance and Economics, in 2017. She is currently pursuing the master's degree with the Department of Network Engineering, Army Engineering University, Nanjing, China. Her research interest includes network management and simulation.