IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received July 31, 2018, accepted October 4, 2018, date of publication October 10, 2018, date of current version November 30, 2018.

Digital Object Identifier 10.1109/ACCESS.2018.2875124

Towards Resource-Efficient Service Function Chain Deployment in Cloud-Fog Computing

DONGCHENG ZHAO[1], DAN LIAO[1], GANG SUN[2], AND SHIZHONG XU[1]

[1]Key Laboratory of Optical Fiber Sensing and Communications, Ministry of Education, University of Electronic Science and Technology of China, Chengdu 611731, China

[2]Key Laboratory of Optical Fiber Sensing and Communications, Center for Cyber Security, Ministry of Education, University of Electronic Science and Technology of China, Chengdu 611731, China

Corresponding authors: Dan Liao (dliao.uestc@gmail.com) and Gang Sun (gangsun@uestc.edu.cn)

This work was supported in part by the Natural Science Foundation of China under Grant 61571098 and in part by the 111 Project under Grant B14039.

ABSTRACT Most studies on network function virtualization are based on cloud computing environments. Fog computing has been proposed as a supplement to cloud computing. When deploying the service function chain (SFC), the consumption of network resources can be effectively reduced by taking advantage of a combination of cloud and fog computing. However, few SFC studies are based on fog computing environments. Moreover, the problem of combining SFCs for the support of live online services to reduce network congestion and save network resources has not been considered. To effectively take advantage of cloud-fog computing and thus achieve the goal of saving resources and reducing network congestion, in this paper, we study the SFC combination and deployment problem in cloud-fog computing environments. To solve this problem, we present an efficient SFC combination and deployment algorithm. Finally, we conduct extensive simulations to evaluate the performance of our proposed algorithm. The results show that our proposed algorithm can effectively reduce network resource consumption and effectively resolve network congestion caused by live online services.

INDEX TERMS Cloud-fog computing, combination, deployment, network function virtualization, service function chain.

I. INTRODUCTION

In traditional networks, the network functions are executed by specialized hardware. For example, before the user's packets arrive at the user, the user's data packets may be filtered through the firewall hardware. However, with continuous increases in the number of users, traditional hardware-based network functions have been unable to meet the user's demands, and with the increasingly large scale of networks, the network ossification problem is increasingly serious. Moreover, joining new hardware functions may be difficult or costly to implement [1]. To solve these challenges, researchers have proposed the technology of network function virtualization (NFV) [2], [3]. Through NFV, cloud computing resources are virtualized into virtual network functions (VNFs), which allow network functions performed by specialized hardware to be executed by software running on virtual machines, thus achieving the purpose of reducing Operational Expenditures (OPEX) and Capital Expenditures (CAPEX) [4], [5]. When a user requests a service, a service function chain (SFC) is generated to connect the user and the service terminal that the server of the service provider to realize communication, and it is composed of a plurality of VNFs according to a specific sequence to realize the corresponding strategy. For instance, to realize a user's security policy, the SFC may be: user → firewall → content filters → terminal. To achieve communication between the user and terminal, the SFC will be deployed into the corresponding cloud computing environment.

As the resource demand for cloud computing increases, the abilities of centralized cloud computing have also been challenged [6]. For example, a large number of user requests may lead to the core network becoming congested and centralized cloud computing may not be able to meet the constraints of users with high delay constraints. Therefore, in 2011, Cisco proposed the concept of fog computing [7] as a supplement to cloud computing rather than a replacement of cloud computing. Fog computing is implemented via edge networks at the user's location that consist of micro or nano

data centers [8], and it is called the cloud on the ground. Namely, fog computing is a distributed computing environment located closest to the user. Because fog computing is proposed as a supplement to cloud computing, it is usually combined with cloud computing to provide services to users. The combined architecture of distributed fog computing and centralized cloud computing is shown in Fig. 1.

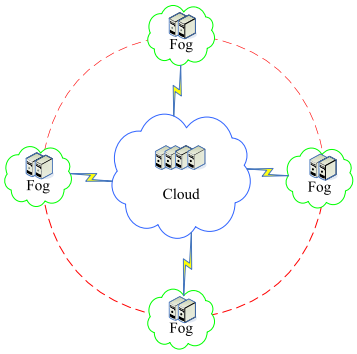


FIGURE 1. Combination architecture of the distributed fog computing and centralized cloud computing environment.

The combination of fog computing and cloud computing can effectively solve the challenges facing cloud computing. For instance, by using these fog computing edge networks, core network congestion is reduced and delay-sensitive services can be run. Due to these advantages, fog computing has become the research direction of radio access networks [9], computer access networks [10], the Internet of Things [11] and vehicle networks [12]. In [9]–[12], although the authors research fog computing, they do not consider combining NFV with fog computing to study. At present, researchers have combined fog computing and SDN networks to improve the use of network resources [13], and this work indicates the feasibility of combining NFV with fog computing as an area of study [13]. Therefore, we can combine fog computing and NFV technology to evaluate whether such combinations can improve SFC deployment and more effectively utilize network resources.

At present, most of the studies on NFV are based on cloud computing [14]–[19]. In [14], a myopia algorithm is designed to implement a packet schedule with low complexity and space efficiency in a NFV environment, and it utilizes the regular pattern of network flow to ensure resource fairness between the low time and space costs. To guarantee the timely provision of resources, Li *et al.* [15] study the resource provision problem for NFV and propose a real-time resource provisioning system (NFV-RT). In [16], the near optimal deployment problem of VNFs is studied and near optimal approximation algorithms are used to minimize the overall network cost. Mehraghdam *et al.* [17] research the deployment problem of SFCs so that server resources are fully exploited, and this research considers sharing and reusing these existing VNFs. To minimize the overall network consumption, Kuo *et al.* [18] consider optimizing link and server resource utilization and sharing and reusing the existing VNFs. To minimize the bandwidth resource consumption, Ye *et al.* [19] jointly study the topology design and placement problem of SFC and merge VNFs in advance if the link resource requirements between these VNFs are larger to achieve the goal of minimizing bandwidth resources.

Although Li and Qian [14], Li *et al.* [15], and Cohen *et al.* [16] study the deployment problem of VNFs/SFCs to minimize the consumption of network resources, they do not consider the combination problem of SFC. In [17]–[19], although sharing and reusing existing VNFs is considered, combining fog computing and NFV technology is not considered. However, if we use the advantages of fog computing to conduct research on the deployment of SFCs, then we can save more network resources. Moreover, the combination problem of total SFCs has not been considered in the cited studies. A real-world problem is the generation of SFC requests to support various services such as news on demand, video on demand and live online. These SFC requests usually arrive in batches. Live online services (i.e., NBA live online, European Cup live online and World Cup live online) are special because each user that receives the same service content at the same time can be considered the same user if we do not consider delay and the same live online service content originates from the same service terminal; thus, live online services will produce a large number of SFC requests and network traffic. Existing algorithms are not effective at solving this problem; therefore, the network will experience congestion.

In this paper, we study the static deployment problem of SFCs. In these given SFCs, a subset of SFCs support live online services. We consider classifying all SFCs and appropriately combining SFCs, and fog computing is used to deploy all SFCs, including the combined and uncombined SFCs, to save more network resources. The main contributions of this paper are as follows.

- To solve the static deployment problem of SFCs, we first model the static deployment problem of SFCs as an integer linear programming problem.
- Since the deployment problem of SFCs is an NP-hard problem, to achieve effective deployment solutions, we present a SFC mapping algorithm that classifies and combines homogeneous SFCs, i.e., the SFCM-CC, to solve the problem of deploying SFCs.
- To classify and combine homogeneous SFCs, we propose a sub-algorithm of the SFCM-CC algorithm, CCHSFC, to classify all SFCs and combine the SFCs if the SFCs are homogeneous.
- To achieve the deployment of SFCs, we propose a sub-algorithm of the SFCM-CC algorithm, SFCM, to deploy all SFCs into the cloud-fog computing environment by taking advantage of the combination of fog computing and cloud computing to deploy SFCs, thus saving more network resources.
- Finally, to estimate the performance of the SFCM-CC algorithm, we use the combined distributed fog

computing and the core network environment, i.e., USANET, as the substrate network to conduct extensive simulations.

The remainder of this paper is arranged as follows. In section II, we summarize the current relevant studies. In section III, we describe and model the research problem and model. In section IV, we propose a heuristic algorithm to solve our research problem. In section V, we conduct extensive simulations to evaluate the performance of our proposed algorithm. Finally, we present the conclusions.

II. RELATED WORK

A. FOG COMPUTING

To solve the challenges of centralized cloud computing, Cisco proposed the concept of fog computing, which is a distributed computing approach located in an edge network that extends cloud computing rather than replacing it. The proposed fog computing environment can reduce the congestion of the core network, run certain delay-sensitive services, save network resources and reduce the energy consumption of the entire network. Due to these advantages, fog computing has become a popular research direction, and studies [6], [9]–[13], [20]–[25] have focused on the 5G radio access network, computer access networks, the Internet of Things and vehicle networks.

To solve the challenges of radio access networks, [9] proposes a radio access network architecture that is software defined and virtualized and includes fog computing and designs an OpenPipe to enable network-level virtualization. To improve energy and spectral efficiency, in [20], an F-RAN architecture is proposed based on fog computing for the 5G radio wireless communication system. To improve the performance of the F-RAN system [21], Yan *et al.* derive the coverage probability and ergodic rate for both F-AP users and device-to-device users.

In [10], the energy consumption of nano data centers for implementing fog computing is evaluated to help reduce the energy consumption of the entire network. In [22], fog computing is comprehensively defined for sensor networks peer-to-peer networks, etc., and the main challenges of fog computing are highlighted. To solve data privacy issues and improve resource management efficiency in a fog computing environment, a mechanism to safely delete duplicate data and efficiently manage resources in fog storage is presented in [23]. To improve the performance of cloud-fog computing, Deng *et al.* [24] research the tradeoff between delay and power consumption to reduce transmission latency and save communication bandwidth.

Xia *et al.* [6] indicate that the characteristics of fog computing make the fog platform appropriate for a number of Internet of Things services. In [11], Chiang and Zhang summarize the challenges and opportunities of fog computing in the context of the Internet of Things.

To solve the challenges of vehicular computation and communication, Hou *et al.* [12] present a vehicular fog computing architecture that utilizes a number of collaborative end users or the edge of the equipment to conduct communication and computing. To provide highly responsive geo-localized services to users and achieve vehicle interactions with city-level smart objects, Bruneo *et al.* [25] propose a platform for smart city applications, Stack4Things, which is based on fog computing.

In [6], [9]–[12], and [20]–[25], although the authors research fog computing, these works do not consider the combination of NFV and fog computing. Therefore, these studies are not suitable for NFV scenarios. In [13], Zeng *et al.* studied the deployment of SDN in the fog computing environment [13]; therefore, this study indicates the feasibility of combining NFV with fog computing as an area to study.

B. SFC DEPLOYMENT

To reduce OPEX and CAPEX, NFV technology is used to virtualize the resources of cloud computing into VNFs, in which the network functions of specialized hardware are performed by software running on virtual machines. With the development of NFV technology, a number of studies have focused on VNF deployment [1], [4], [14]–[19], [26]–[33].

To achieve load balancing of the network and server, Thai *et al.* [1] study the deployment problem of SFCs in a datacenter environment and present the 2-phase algorithm NF-LGT. To reduce network congestion, Elias *et al.* [26] analyzed the causes of network congestion and presented innovative NFV scheduling methods to deploy VNFs. Pham *et al.* [27] research the joint optimization problem of operational costs and network traffic cost and present the SAMA method to achieve the optimization goal. Due to the use of additional methods to select traffic paths that lead to additional east-west traffic, Chi *et al.* [28] study the deployment problem of VNFs in data center networks and propose a heuristic algorithm to deploy VNFs and schedule traffic according to traffic flows and user demands. Jalalitabar *et al.* [4] research the problem of efficiently constructing the service function graph and present the SFG_PD mapping algorithm to deploy the SFG into the substrate network. To minimize the overall network cost, Cohen *et al.* [16] study the near optimal deployment problem of VNFs and present a near optimal approximation algorithm. Due to physical hardware resources and VNF susceptibility to malicious attacks and natural disasters, Liu *et al.* [29] propose a framework to evaluate the reliability of NFV deployment and the physical or logical nodes. To meet the security demands of users, Shameli-Sendi *et al.* [30] present a VNF deployment method with network security defense patterns.

To implement a packet schedule with low complexity and space efficiency in NFV environments, Li and Qian [14] propose the myopia algorithm, which utilizes the regular pattern of network flow to ensure resource fairness between the low time and space costs. Li *et al.* [15] study the resource provision problem for NFVs and propose the real-time resource provisioning system NFV-RT to guarantee the timely provision of resources. To minimize the scheduling latency of the overall VNFs, Qu *et al.* [31] study the VNF scheduling

problem and formulate the problem as a MILP, and they then propose a genetic algorithm to solve the problem. Because the SFC deployment problem is NP-hard, Khebbache *et al.* [32] present a matrix-based optimization and a multistage graph method of deploying SFCs that achieves cost efficiency and improves scalability. To achieve the goal of minimizing the response time and the inter-cloud traffic, Bhamare *et al.* [33] propose an affinity-based heuristic algorithm to solve the problem for a multi-cloud scenario.

Mehraghdam *et al.* [17] research the deployment problem of SFCs and show that server resources can be fully exploited by sharing and reusing the existing VNFs. Kuo *et al.* [18] consider optimizing link and server resource utilization and sharing and reusing existing VNFs to minimize the overall network consumption. To minimize bandwidth resource consumption, Ye *et al.* [19] study the topology design and placement problem of SFCs and merge VNFs in advance if the link resource requirements between these VNFs are larger.

Although research has been performed on the deployment problem of VNFs/SFCs [1], [4], [14]–[16], [26]–[33], these studies do not consider the combination problem of SFCs. Mehraghdam *et al.* [17], Kuo *et al.* [18], and Ye *et al.* [19] consider sharing and reusing existing VNFs only in a cloud computing environment; however, fog/edge computing can be exploited for the deployment of SFCs to save additional network resources. Moreover, the previously cited studies do not consider the combination problem of the total SFCs. Therefore, heuristic algorithms are presented for the deployment of SFCs in a cloud-fog computing environment in this paper.

III. PROBLEM DEFINITION AND FORMULATION

A. PROBLEM DEFINITION

A real-world problem includes the generation of SFC requests to support various services, such as news on demand, video on demand and live online (i.e., NBA live online, European Cup live online and World Cup live online). These SFC requests usually arrive in batches. We consider classifying all SFCs and appropriately combining SFCs, and fog computing is used to deploy all SFCs, including the combined and uncombined SFCs, to save more network resources. For a group of SFC requests, user locations and service terminal locations of each SFC request and a substrate network composed of a fog computing environment and a cloud computing environment, the problem is how to efficiently classify, combine and deploy all the SFCs such that the total mapping cost, VNF mapping cost, link mapping cost and blocking ratio are minimized.

B. SFC REQUESTS

We model a SFC request as an undirected weighted graph $g = (N_V, E_V, L_V)$ in which the set of all SFC requests is $G_V = \{g_1, g_2, \ldots, g_m\}$, where the *i-th*SFC request is $g_i = (NiV, EiV, LiV)$, $NiV = \{V_1, V_2, \ldots, V_n i\}$ indicates the set of VNFs in the *i-th*SFC request g_i, and n^i denotes the number of VNFs in the *i-th*SFC request g_i. In this paper, we divide the SFC links into SFC main links and SFC user links, with the SFC user links denoting the SFC links connecting users and the SFC main links indicating all SFC links except the SFC user links.$E_V^i = \{e_1, e_2, \ldots, e_{|E^i|}\}$ represents the set of SFC main links, and $|E^i|$ indicates the number of SFC main links. $L_V^i = \{l_1, l_2, \ldots, l_{|L^i|}\}$ represents the set of SFC user links, and $|L^i|$ indicates the number of SFC user links.

1) DEPLOYMENT CONSTRAINT

We define $PC = \{PC_1, PC_2, \ldots, PC_m\}$ as the deployment constraints of all SFC requests, where the deployment constraint of the *i-th* SFC request is $PC_i = (C_N^i, C_E^i, C_L^i, LC_N^i, LC_U^i, LC_T^i, Type^i)$.

2) VNFs RESOURCE CONSTRAINT

$C_N^i = \{\varepsilon(V_1), \varepsilon(V_2), \ldots, \varepsilon(V_n i)\}$ denotes the computing resource demands of all VNFs in the *i-th*SFC request g_i.

3) SFC LINKS RESOURCE CONSTRAINT

$C_E^i = \{\varepsilon(e_1), \varepsilon(e_2), \ldots, \varepsilon(e_{|E^i|})\}$ indicates the bandwidth resource demands of all SFC main links in the *i-th*SFC request g_i. $C_L^i = \{\varepsilon(l_1), \varepsilon(l_2), \ldots, \varepsilon(l_{|L^i|})\}$ denotes the bandwidth resource demands of all SFC user links in the *i-th*SFC request g_i.

4) VNF, USER AND SERVICE TERMINAL LOCATION CONSTRAINT

$LC_N^i = \{LC(V_1), LC(V_2), \ldots, LC(V_n i)\}$ represents the location constraint of all VNFs in the *i-th*SFC request g_i. LC_U^i represents the location constraint of the user in the *i-th*SFC request g_i, i.e., the user's current location. LC_T^i denotes the location constraint of the service terminal in the *i-th*SFC request g_i. We assume that there is only one user in each SFC request when the SFC requests are not combined and each user achieves service through the fog access network.

5) SFC TYPE

$Type^i$ denotes the business type of the *i-th*SFC request g_i, such as news on demand, video on demand or live online. We assume that the business types of all non-live online requests are numbered zero and the business types of different live online requests are numbered from 1 to π. So the number of business types is $\pi + 1$.

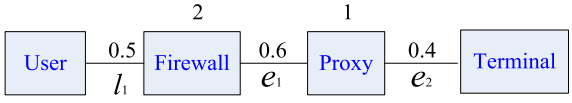


FIGURE 2. SFC request.

Fig. 2 shows an example of the SFC request, where l_1 indicates the SFC user link and e_1 and e_2 denote the SFC main links. The numbers above links represent the bandwidth resource demands of links, and the numbers above VNF indicate computing resource demands of VNF.

C. SUBSTRATE NETWORK

The substrate network is composed of the distributed fog access network and the centralized cloud computing environment/core network. Similarly, the substrate network can be modeled as an undirected weighted graph $G^S = (N^S, E^S)$, where N^S indicates the set of physical computing nodes and E^S denotes the set of physical links.

1) SUBSTRATE NETWORK RESOURCE CONSTRAINT

We define $SC = (C^N, C^E, LC^N)$ as substrate network resource constraints.

2) SUBSTRATE NODE RESOURCE ATTRIBUTES

C^N indicates the set of the substrate node resource attributes, which include the capacity of the substrate node resources $b(n_s)$ and the cost per unit resource of the substrate node $p(n_s)$.

3) SUBSTRATE LINK RESOURCE ATTRIBUTES

C^E indicates the set of the substrate link resource attributes, which include the bandwidth capacity $b(e_s)$ and the cost per unit resource of the substrate link $p(e_s)$.

4) SUBSTRATE NODE LOCATION CONSTRAINT

LC^N represents the set of the location constraints of all substrate network nodes.

We assume that the computing nodes in the distributed fog computing environment have the capacity to provide service for the SFC requests, although the capacity is weaker than the computing nodes of the centralized cloud computing environment. An example of a substrate network is shown in Fig. 3.

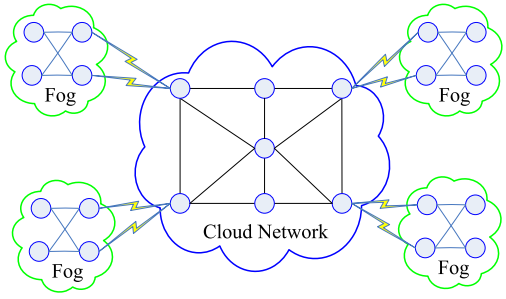


FIGURE 3. Example of a substrate network.

D. HOMOGENEOUS SFC COMBINATION

The given SFC requests are generated to support various services, such as news on demand, video on demand and live online. Live online services will produce a large number of SFC requests and network traffic, which usually leads to network congestion. To save network resources and reduce network congestion, in this paper, we propose a strategy for combining homogeneous SFCs. We first define homogeneous SFCs, which must meet the following conditions:

H.1: Homogeneous SFCs must be the same live online business;

H.2: Homogeneous SFCs must originate from the same service terminal;

H.3: The number of VNFs of the homogeneous SFCs must be exactly the same;

H.4: The number of links of the homogeneous SFCs must be exactly the same;

H.5: The type of corresponding VNFs of the homogeneous SFCs must be exactly the same;

H.6: The resource demands of the corresponding VNFs and links of homogeneous SFCs must be exactly the same;

H.7: The location of the users must originate from the same fog access network.

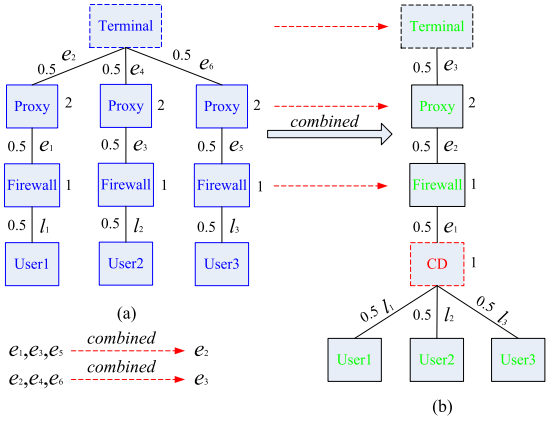


FIGURE 4. Homogeneous SFCs combination.

SFCs that satisfy all conditions **H.1**-**H.7**are called homogeneous SFCs. For example, when users request the same live online business from the same service terminal with the same clarity and these users belong to the same fog access network, the SFC requests are homogeneous. Because the video content of each user of the same live online service at the same time can be considered the same user if we ignore delay, we can combine the homogeneous SFCs according to the method shown in Fig. 4. Fig. 4 (a) shows three original SFCs, which are independent of each other except that they originate from the same service terminal, and their computing resources and link resources are also independent of each other. Additionally, the three original SFCs need to transmit three identical videos. Fig. 4 (b) shows the combined SFCs, which share computing resources and link resources. Fig. 4 shows that the combined SFC generates a new VNF *CD* between the firewall and the user, and this new VNF *CD* is responsible for caching and distributing videos to each user. A new SFC main link e_1 is generated to connect the new VNF *CD*and the firewall. Three firewalls of the three original SFCs are combined into a firewall; three proxies of the three original SFCs are combined into a proxy; SFC main links e_1, e_3 and e_5 are combined into a SFC main link e_2; and SFC main links e_2, e_4 and e_6 are combined into a SFC main link e_3. To maximize saving resources, in this paper, we deploy the new VNF *CD* into the fog access network to reduce the length of the mapping paths for the user links. Fig. 4 shows that

the resource requirements of the combined SFC are much lower than that of the three original SFCs because the three original SFCs need to transmit three identical videos while the combined SFC only needs to transmit a video from the service terminal to the new VNF *CD* and then the new VNF *CD* caches and distributes the video to each user.

E. SFC DEPLOYMENT

In the deployment process of SFCs, we try to deploy a VNF into each substrate node at every turn and map the corresponding link at the same time. Finally, we choose a deployment solution based on our strategy. Then, we allocate computing resources for the VNFs and bandwidth resources for SFC links. The SFC deployment procedure can be formulated as follows.

(1) *VNF Deployment:*

The deployment process of VNFs can be formulated as follows:

$$\begin{aligned}
&M_N : (N_V^i, C_N^i) \xrightarrow{M_N} (N^{S1}, C^{N1}),\\
&\quad M(V_i) \in N^{S1}, \quad \forall V_i \in N_V^i,\\
&\quad R(M(V_i)) \geq \varepsilon(V_i), \quad \forall V_i \in N_V^i,\\
&Z_{V_i}^w \in \{0,1\}, \quad \forall V_i \in N_V^i,\ \forall w \in \{0,1,...,W\},\\
&\quad LC(M(V_i)) \in \{0,1,2,...,Y\}, \quad \forall V_i \in N_V^i,\\
&\quad Z_{V_i}^{LC(M(V_i))} = 1, \quad \forall V_i \in N_V^i,
\end{aligned}$$

where $N^{S1} \subset N^S$ denotes the set of substrate nodes for hosting all VNFs; C^{N1} represents the computing resources that are allocated to all VNFs; $M_N = \{M(V_1), M(V_2), \ldots, M(V_n i)\}$ denotes the set of the deployment solutions of all VNFs, where $M(V_i)$ indicates the substrate node for hosting the *i-th* VNF V_i; and $R(M(V_i))$ denotes the remaining resources of the substrate node for hosting the *i-th* VNF V_i. Since users are located in different fog access networks, the fog access networks can only perform small-scale calculations, and the VNFs also have the corresponding position constraints, we divide the entire substrate network into different network areas according to the different fog access networks and the core network. We use $w \in 0, 1, \ldots, W$ to represent the network area number. $Z_{Vi}^w = 1$ denotes that the *i-th* VNF V_i can be deployed in the network area, and $Z_{Vi}^w = 0$ indicates that the *i-th* VNF V_i cannot be deployed in the network area. To save network resources, we request that the VNF *CD* must be deployed into the fog access network where the users are located.$LC(M(V_i))$ denotes the network area number of the substrate node $M(V_i)$, and a substrate node can only belong to a network area. $Z_{Vi}^{LC(M(Vi))} = 1$ represents that the substrate node $M(V_i)$ meets the location constraint of the *i-th* VNF V_i.

(2) *SFC Link Deployment:*

The deployment process of SFC links can be denoted as follows:

$$\begin{aligned}
&M_E : (E_V^i, L_V^i, C_E^i, C_L^i) \xrightarrow{M_E} (P', C^{E'}),\\
&M(e_i) = p_{e_i}, \quad \forall e_i \in E_V^i,\ \exists p_{e_i} \in P',\\
&M(l_i) = p_{l_i}, \quad \forall l_i \in E_V^i,\ \exists p_{l_i} \in P',
\end{aligned}$$

$$\begin{aligned}
&B(p_{e_i}) = \min_{e_s \in p_{e_i}} \{b(e_s)\} \geq \varepsilon(e_i), \quad \forall p_{e_i} \in P',\\
&B(p_{l_i}) = \min_{e_s \in p_{l_i}} \{b(e_s)\} \geq \varepsilon(l_i), \quad \forall p_{l_i} \in P',
\end{aligned}$$

where $P^{'}$ denotes the set of substrate paths for hosting all SFC links and each substrate path is a subset of E^S. C^{E1} denotes the link resources that are allocated to all SFC links. $M_E = \{M(e_1), M(e_2), \ldots, M(e_{|E^i|}), M(l_1), M(l_2), \ldots, M(l_{|L^i|})\}$ indicates the set of the mapping paths of all SFC links, and $M(e_i)$ represents the mapping path for hosting the SFC main link e_i. $M(l_i)$ represents the mapping path for hosting the SFC user link l_i. $B(p_{ei})$ indicates the available bandwidth resources of the substrate path p_{ei}, and $B(p_{li})$ indicates the available bandwidth resources of the substrate path p_{li}.

$$\begin{aligned}
&\min \Big(\sum_{e_i \in E_V^i} \sum_{e_s \in p_{e_i}} P(e_s)\varepsilon(e_i) + \sum_{l_i \in L_V^i} \sum_{e_s \in p_{l_i}} P(e_s)\varepsilon(l_i) \Big)\\
&\min \sum_{V_i \in N_V^i} P(M(V_i))\varepsilon(V_i)\\
&s.t.\ R(M(V_i)) \geq \varepsilon(V_i), \quad \forall V_i \in N_V^i\\
&\quad B(p_{e_i}) = \min_{e_s \in p_{e_i}} \{b(e_s)\} \geq \varepsilon(e_i), \quad \forall e_i \in E_V^i\\
&\quad B(p_{l_i}) = \min_{e_s \in p_{l_i}} \{b(e_s)\} \geq \varepsilon(l_i), \quad \forall l_i \in L_V^i\\
&\quad Z_{V_i}^w \in \{0,1\}, \quad \forall V_i \in N_V^i,\ \forall w \in \{0,1,...,W\}\\
&\quad LC(M(V_i)) \in \{0,1,2,...,Y\}, \quad \forall V_i \in N_V^i\\
&\quad Z_{V_i}^{LC(M(V_i))} = 1, \quad \forall V_i \in N_V^i
\end{aligned} \tag{1}$$

In this paper, we jointly optimize the cost of bandwidth resources and the cost of computing resources, i.e., *i*) the cost of the bandwidth resources is minimized; and *ii*) the cost of the computing resources is minimized. Therefore, the deployment problem of SFCs in a cloud-fog network can be formulated as the following linear programming model (1). Where $b(e_s)$ denotes the bandwidth capacity and $p(e_s)$ denotes the cost of per unit resource of substrate links.

The first objective is used to minimize the cost of bandwidth resources, i.e., shortening the mapping path for the SFC request as much as possible. The second objective is used to minimize the cost of computing resources as much as possible. In addition, the constraints in linear programming (1) are used to ensure the following condition.

Constraint 1 ensures that the substrate nodes being used satisfy the computing resource requirements of VNFs.

Constraints 2, 3 ensure that the substrate links being used satisfy the bandwidth resources requirements of SFC links.

Constraints 4, 5 and 6 ensure that the substrate nodes being used satisfy the location constraints of VNFs.

IV. HEURISTIC ALGORITHM

Since the deployment problem of SFCs in a cloud-fog network is an NP-hard problem, to achieve effective deployment solutions, we present the SFCM-CC mapping algorithm for classifying and combining homogeneous SFCs to solve the problem of deploying SFCs. In this paper, we assume that

a group of SFC requests are given and the location of the user and the service terminal of each SFC request are also randomly given. The group of SFC requests are stored in the set of all SFC requests $G_V = \{g_1, g_2, \ldots, g_m\}$. We first call the CMHSFC procedure for classifying and combining homogeneous SFCs to achieve the new set of all SFC requests after business classification $G_{V1} = \{g_1, g_2, \ldots, g_{m1}\}$. Then, we call the MSFC procedure for mapping each SFC request in the new set of all SFC requests G_{V1}. In addition, as we map each SFC request, we allocate the computing resources and the bandwidth resources for the SFC request at the same time. Finally, we update the substrate network resources when we successfully map each SFC request. If a user in the SFC request mapping has failed, we store the blocked users in the set of blocked users $User_{blo}$. The pseudocode of the presented SFC mapping algorithm is shown in *Algorithm* 1.

Algorithm 1 SFC Mapping Algorithm for Classifying and Combining Homogeneous SFCs(SFCM-CC)

Input:1. Substrate network $G^S = (N^S, E^S)$ and resource constraints $SC = (C^E, C^N, L^N)$;
2. All SFC requests $G_V = \{g_1, g_2, \ldots, g_m\}$.
Output: Mapping cost$M$ *total cost* and the set of blocked users, $User_{blo}$.
1: Initialization: **let**M *total cost* =0 and $User_{blo} = \emptyset$;
2: Call the CMHSFC procedure to achieve all SFC requests after business classification G_{V1};
3: **for** each SFC request $g_i \in G_{V1}$, **do**
4: Call MSFC procedure for mapping the SFC request g_i;
5: **if** a mapping solution$M$is found for g_i, **then**
6: $M_{cost}^{total} = M_{cost}^{total} + M_{SFC}$, update substrate network resources;
7: **end if**
8: Update $User_{blo}$ according to the mapping solution$M$;
9: **end for**
10: **return** M_{cost}^{total}, $User_{blo}$.

The CCHSFC procedure is responsible for classifying and combining homogeneous SFCs to achieve the new set of all SFC requests after business classification $G_{V1} = \{g_1, g_2, \ldots, g_{m1}\}$ as shown in *Procedure*1. In the CCHSFC procedure, we first store the SFC requests in U_k, $k \in 0, 1, \ldots, \pi$ according to the business type of each SFC request, where all SFC requests in U_0 are non-live online requests, all SFC requests in U_k, $k \in 1, \ldots, \pi$ perform the same business and the serial number of their business types are also k. Then, we divide the SFC requests in U_k, $k \in 1, 2, \ldots, \pi$ into different sets of homogeneous SFCs $Gi\ V1$according to the conditions of homogeneous SFCs **H.1**-**H.7**and sort all classified homogeneous SFC sets in descending order according to $|G_{V1}^i|$. The sorted homogeneous SFC sets are $\{G1V1, G2V1, \ldots, GT\ V1\}$. Finally, we combine the homogeneous SFCs according to the method shown in Fig. 4 and update the related parameters. In the combined SFCs, more than one user is included. Therefore, through calling the CMHSFC procedure, we can classify and combine homogeneous SFCs and achieve the new set of all SFC requests after business classification $G_{V1} = \{g_1, g_2, \ldots, g_{m1}\}$. In the new set of all SFC requests $G_{V1} = \{g_1, g_2, \ldots, g_{m1}\}$, the combined SFCs are stored in front of the new set of all SFC requests G_{V1} so that when we deploy all SFC requests, we can achieve a smaller user blocking ratio. The pseudocode of the CCHSFC procedure is shown in *procedure*1.

Procedure 1 Classifying and Combining Homogeneous SFCs (CCHSFC)

Input: All SFC requests $G_V = \{g_1, g_2, \ldots, g_m\}$, deployment constraints $PC = \{PC_1, PC_2, \ldots, PC_m\}$, $PC_i = (C_N^i, C_E^i, C_L^i, LC_N^i, LC_U^i, LC_T^i, Type^i)$ and the number of business classification $\pi + 1$.
Output: All SFC requests after business combination G_{V1}.
1: **for** each SFC request $g_i \in G_V$, **do**
2: **for** $k \in 0, 1, \ldots, \pi$, **do**
3: **if** $Type^i == k$, **then**
4: Store g_i in U_k;
5: **end if**
6: **end for**
7: **end for**
8:$G_{V1} \leftarrow \emptyset$, $G_{V1}^0 \leftarrow U_0$, $i = T = 0$;
9: **for** $k \in 1, 2, \ldots, \pi$, **do**
10: **while** $U_k \neq \emptyset$, **do**
11: Take the first SFC request g_1 in U_k, i++, T++, $SFC_i = g_1$, $G_{V1}^i \leftarrow G_{V1}^i \cup \{g_1\}$, $U_k \leftarrow U_k \{g_1\}$;
12: **for** each SFC request $g_j \in U_k$, **do**
13: **if** g_j and SFC_i are homogeneous, **then**
14: $G_{V1}^i \leftarrow G_{V1}^i \cup \{g_j\}$,$U_k \leftarrow U_k \{g_j\}$;
15: **end if**
16: **end for**
17: **end while**
18: **end for**
19: Sort all classified business sets in descending order according to $|G_{V1}^i|$, and the sorted businesses are $\{G_{V1}^1, G_{V1}^2, \ldots, G_{V1}^T\}$;
20: **for** each $i \in 1, 2, \ldots, T$, **do**
21: Combine all SFCs in $Gi\ V1$ into SFC g_i according to the method shown in Fig. 4, and update $Type^i = i$ and $G_{V1} \leftarrow G_{V1} \cup \{g_i\}$;
22: **end for**
23: $G_{V1} \leftarrow G_{V1} \cup \{G0V1\}$ and update the deployment constraints $PC = \{PC_1, PC_2, \ldots, PC_{m1}\}$
24: **return** $G_{V1} = \{g_1, g_2, \ldots, g_{m1}\}$.

The MSFC procedure is responsible for mapping a SFC request to achieve an effective solution as shown in *Procedure*2. When we call the MSFC procedure to map a SFC request, we first try to deploy the first VNF V_1 into each available substrate node and map all user links at the same time. We select the solution of the first VNF V_1 with the

Procedure 2 Mapping a SFC (MSFC)

Input: 1. Substrate network $G^S = (N^S, E^S)$ and resource constraints $SC = (C^E, C^N, L^N)$;
2. SFC request $g_i = (N_V^i, E_V^i, L_V^i)$ and deployment constraints $PC_i = (C_N^i, C_E^i, C_L^i, LC_N^i, LC_U^i, LC_T^i, Type^i)$.

Output: Mapping solution M.

1: Initialization: $U^S \leftarrow N^S$, take the first VNF $V_1 \in N_V^i$;
2: **for** each $n_j \in U^S$, **do**
3: **if** $Z_{V1}^{LC(n_j)} == 1$, **then**
4: Map V_1 into the substrate node n_j and find the minimal cost path $p^1(n_j, LC_T^i)$, and calculate and record $Cost(V_1 \rightarrow n_j)$ according to Equation (2);
5: **if** $VCost(V_1 \rightarrow n_j) < \infty$, **then**
6: **for** each $l_x \in L_V^i$, **do**
7: Find the minimal cost path p_{lx}, calculate and record $LCost(p_{lx})$ according to Equation (5);
8: **end for**
9: Calculate and record $TCost(V_1 \rightarrow n_j)$ according to Equation (6);
10: **end if**
11: **end if**
12: **end for**
13: Find the mapping solution of V_1 with the maximum user acceptance ratio and the minimum total mapping cost, and store the mapping solutions of V_1 and each $l_x \in L_V^i$ in M;
14: **for** each VNF $V_k \in N_V^i$, $k = 2, 3, \ldots, |N_V^i|$ **do**
15: **for** each $n_j \in U^S$, **do**
16: **if** $Z_{V_k}^{LC(n_j)} == 1$, **then**
17: Try to place V_k into substrate node n_j, calculate and record $Cost(V_k \rightarrow n_j)$ in Equation (7);
18: Find the minimal cost path pe_{k-1} $\mathrm{p_{e_{k-1}}}$ and $p^k(n_j, LC_T^i)$, calculate and record $LCost(pe_{k-1}\mathrm{p_{e_{k-1}}})$ according to Equation (8), and calculate and record the total mapping cost $TCost(V_k \rightarrow n_j)$ in Equation (10);
19: **end if**
20: **end for**
21: Find the mapping solution of V_k with the minimal total mapping cost $TCost(V_k \rightarrow n_j)$, and store the mapping solutions of V_k and SFC main link e_{k-1} in M;
22: **end for**
23: updating the final mapping cost into M;
24: **return** M.

maximum user acceptance ratio as the final solution of the first VNF V_1. When there are several solutions with the maximum user acceptance ratio, we select the solution with the minimum mapping cost as the final solution of the first VNF V_1. Then, we deploy other VNFs and SFC links. In the final solution of the SFC request, some users may be blocked due to a lack of network resources. When a VNF or a SFC main link map fails, all users in the SFC request will be blocked. However, when a user link map fails, other users are not affected. Therefore, when we calculate the blocking ratio, we should calculate the blocking ratio according to the number of blocked users rather than the number of the blocked SFC requests.

The mapping cost of the first VNF V_1 is defined as follows:

$$VCost(V_1 \rightarrow n_j) = Cost(V_1 \rightarrow n_j) + Cost(p^1(n_j, LC_T^i)), \tag{2}$$

where n_j indicates a substrate node; the substrate node resource cost$Cost(V_1 \rightarrow n_j)$ and the link cost$Cost(p^1(n_j, LC_T^i))$ can be calculated according to Equation (3) and (4).

$$Cost(V_1 \rightarrow n_j) = p(n_j)\varepsilon(V_1) \tag{3}$$

$$Cost(p^1(n_j, LC_T^i)) = \sum_{e_s \in p^1(n_j, LC_T^i)} p(e_s)\varepsilon(e_1) \tag{4}$$

where $p^1(n_j, LC_T^i)$ is the substrate path connecting the substrate node n_j and the substrate node where the service terminal is located and the bandwidth resource demands of the substrate path $p^1(n_j, LC_T^i)$ is $\varepsilon(e_1)$.

The mapping cost of each user link l_x can be calculated according to Equation (5).

$$LCost(p_{l_x}) = \sum_{e_s \in p_{l_x}} p(e_s)\varepsilon(l_x) \tag{5}$$

The total mapping cost of the first VNF V_1 is defined in Equation (6).

$$TCost(V_1 \rightarrow n_j) = VCost(V_1 \rightarrow n_j) + \sum_{l_x \in L_V^i} LCost(p_{l_x}). \tag{6}$$

The mapping cost of the *k-th* VNF V_k can be calculated according to Equation (7).

$$Cost(V_k \rightarrow n_j) = p(n_j)\varepsilon(V_k) \tag{7}$$

The mapping cost of each SFC main link e_k is defined as follows:

$$LCost(p_{e_k}) = \sum_{e_s \in p_{e_k}} p(e_s)\varepsilon(e_k) + Cost(p^k(n_j, LC_T^i)), \tag{8}$$

where the link cost$Cost(p^k(n_j, LC_T^i))$ can be calculated as in Equation (9).

$$Cost(p^k(n_j, LC_T^i)) = \sum_{e_s \in p^k(n_j, LC_T^i)} p(e_s)\varepsilon(e_k) \tag{9}$$

where $p^k(n_j, LC_T^i)$ is the substrate path connecting the substrate node n_j and the substrate node where the service terminal is located and the bandwidth resource demands of the substrate path $p^k(n_j, LC_T^i)$ is $\varepsilon(e_k)$.

The total mapping cost of the *k-th* VNF V_k is defined as follows:

$$TCost(V_k \rightarrow n_j) = Cost(V_k \rightarrow n_j) + LCost(p_{e_k}). \tag{10}$$

V. SIMULATION RESULTS

A. SIMULATION ENVIRONMENT

In our simulations, we use the USANET network as the substrate core network. The USANET network (shown in Fig. 5) has 46 nodes and 76 links, and 15 fog access networks are connected to the USANET network. In addition, the fog access networks connect to the red nodes 0, 5, 7, 12, 14, 16, 20, 23, 25, 29, 32, 34, 36, 42 and 44 as shown in Fig. 5.

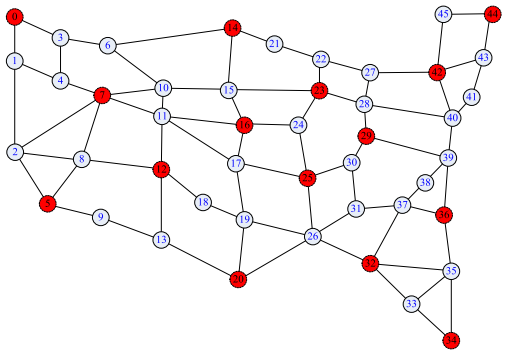


FIGURE 5. USANET network.

In our simulations, we use an unlimited resource capacity scenario and a limited resource capacity scenario to estimate the performance of the SFCM-CC algorithm. When the resource capacity is limited, we assume that the capacity of the substrate node resources of the core network nodes follows a uniform distribution U(3000, 3500). Since the fog access networks have the capacity to provide service for the SFC requests but the capacity is weaker than that of the computing nodes of centralized cloud computing environments, we assume that the capacity of the substrate node resources of the fog access network nodes will follow a uniform distribution U(30, 35). Meanwhile, we assume that the capacity of the bandwidth resource of all substrate network links follows a uniform distribution U(3000, 3500). To calculate the related costs, we assume that the cost per unit resource of each substrate node is also 1 unit. Moreover, we assume that a group of SFC requests are given, i.e., the SFC requests are static; the number of VNFs in each SFC request is varied among 5, 6, 7 and 8; the computing resource requirement of each VNF in each SFC request follows a uniform distribution U(1, 9.5), U(1, 8), U(1, 6.5) or U(1, 5.5); the bandwidth resource demand of each link in the same SFC request is same and the bandwidth resource demand in the different SFC requests follows a uniform distribution U(1, 9.5), U(1, 8), U(1, 6.5) or U(1, 5.5); the location of the user of each SFC request is randomly given in the fog access networks; and the location of the service terminal of each SFC request is randomly given in the USANET network.

In our simulations, to demonstrate the performance of our proposed algorithm, we compare our proposed algorithm with the**PATH-EXTENSION**algorithm proposed in [18].

B. PERFORMANCE METRICS

In our simulations, we estimate the total mapping cost, the VNF mapping cost, and the link mapping cost when the resource capacity is unlimited. Furthermore, we estimate the blocking ratio when the resource capacity is limited.

(1) *Total Mapping Cost:* this cost can be calculated as in Equation (11), and it denotes the total cost of using the node and bandwidth resources of the substrate network for mapping all SFC requests.

$$M_{cost}^{total} = \sum_{|G_{V1}|} M_{SFC} \tag{11}$$

where M_{SFC} indicates the cost of using the node and bandwidth resources of the substrate network for mapping a SFC request and $|G_{V1}|$ denotes the number of the SFC requests.

(2) *VNF Mapping Cost:* this cost denotes the total cost of using the node resources of the substrate network for mapping all VNFs of all SFC requests, and it is defined as follows:

$$M_{cost}^{VNF} = \sum_{|G_{V1}|} M_{VNF}, \tag{12}$$

where M_{VNF} represents the cost of using the node resources of the substrate network for mapping all VNFs in a SFC request.

(3) *Link Mapping Cost:* this cost denotes the total cost of using the bandwidth resources of the substrate network for mapping all links of all SFC requests, and it can be calculated according to Equation (13).

$$M_{cost}^{Link} = \sum_{|G_{V1}|} M_{Link} \tag{13}$$

where M_{Link} indicates the cost of using the bandwidth resources of the substrate network for mapping all links in a SFC request.

(4) *Blocking Ratio:* this ratio represents the ratio of the number of blocked users to the number of total users, and it can be calculated via Equation (14):

$$P_b = \frac{|User_{blo}|}{|User_{all}|} \tag{14}$$

where $|User_{blo}|$ and $|User_{all}|$ denote the numbers of blocked users and total users, respectively. Note that more than one user is included in a combined SFC, and even if certain users are mapped as fails, other users may be mapped successfully in the combined SFC; therefore, we calculate the blocking ratio of users, i.e., the blocking ratio of SFC requests, before combining.

C. SIMULATION RESULTS AND ANALYSIS

Fig. 6 shows the VNF mapping costs of the PATH-EXTENSION algorithm and the SFCM-CC algorithm when the percentage of live online services (i.e., L) is 10%, 20% and 30% and the number of VNFs in the original SFC request (i.e., n) is varied among 5, 6, 7 and 8. Fig. 6 shows that the VNF mapping costs of the SFCM-CC algorithm are lower than that of the PATH-EXTENSION algorithm and the VNF

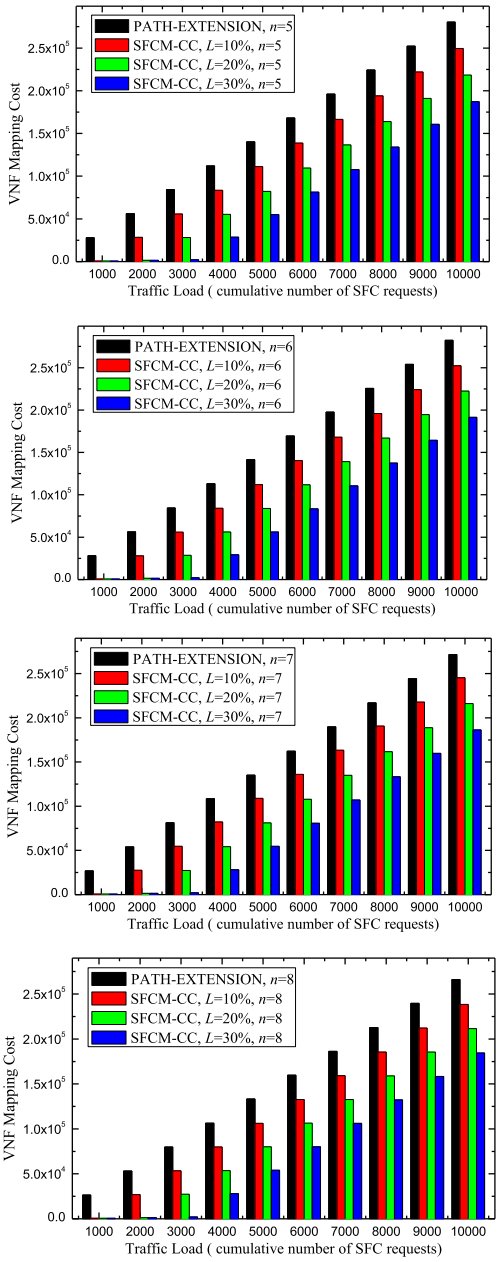


FIGURE 6. VNF mapping cost.

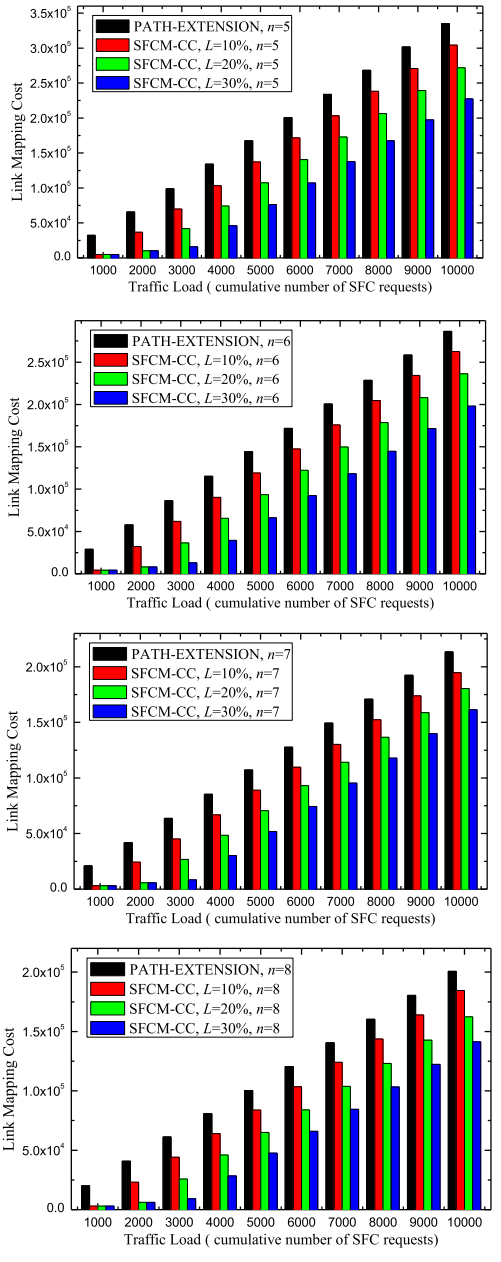


FIGURE 7. Link mapping cost.

mapping cost decreases as the percentage of live online services increases. When the percentage of live online services is increased by 10%, the VNF mapping cost is reduced by approximately 10% because the SFCM-CC algorithm first classifies the SFC requests for supporting live online services into multiple sets of homogeneous SFCs and then combines each set of homogeneous SFCs into a SFC. Therefore, the number of VNFs in the combined SFC is far less than the number of VNFs in the set of homogeneous SFCs before combining. Additionally, the combined SFC only needs to transmit a video from the service terminal to the new VNF *CD* and then the new VNF *CD* caches and distributes the video to each user; therefore, the VNF mapping cost of the SFCM-CC algorithm is lower than that of the PATH-EXTENSION algorithm. Moreover, because the number of VNFs in the combined SFC requests for supporting live online services is far less than the number of VNFs in the SFC requests before combining, when the percentage of live online services is increased by 10%, the VNF mapping costs are reduced by 10%.

Fig. 7 compares the link mapping costs of the PATH-EXTENSION algorithm and the SFCM-CC algorithm when the percentages of live online services are 10%, 20% and 30% and the number of VNFs in the original SFC request is varied among 5, 6, 7 and 8. Fig. 7 shows that the link mapping cost of the SFCM-CC algorithm is lower than that of the

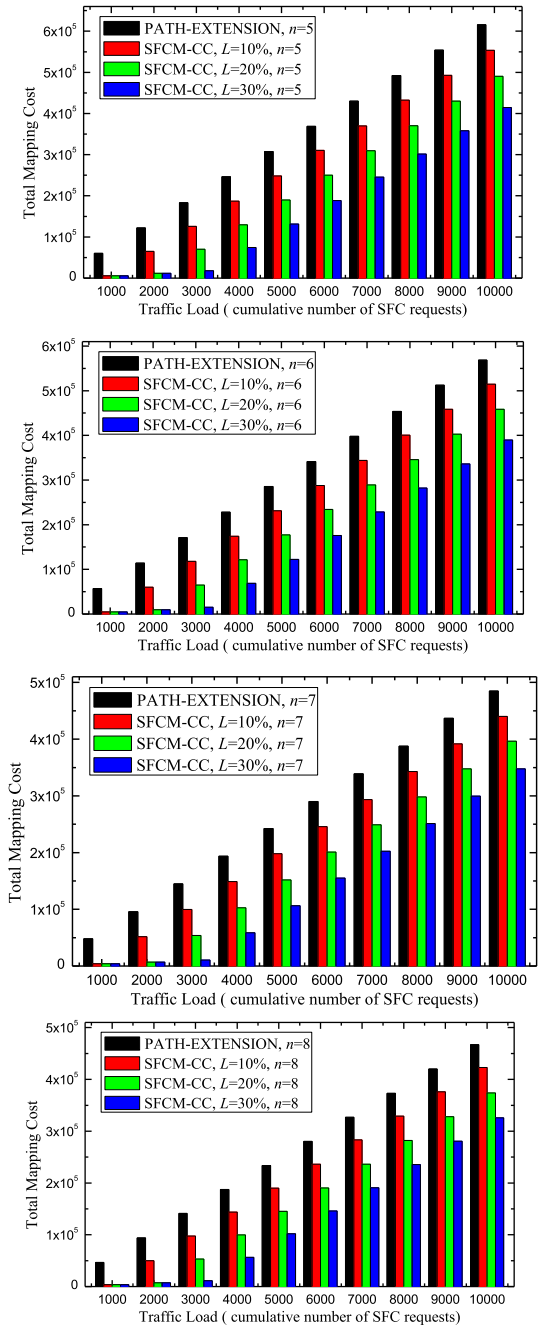


FIGURE 8. Total mapping cost.

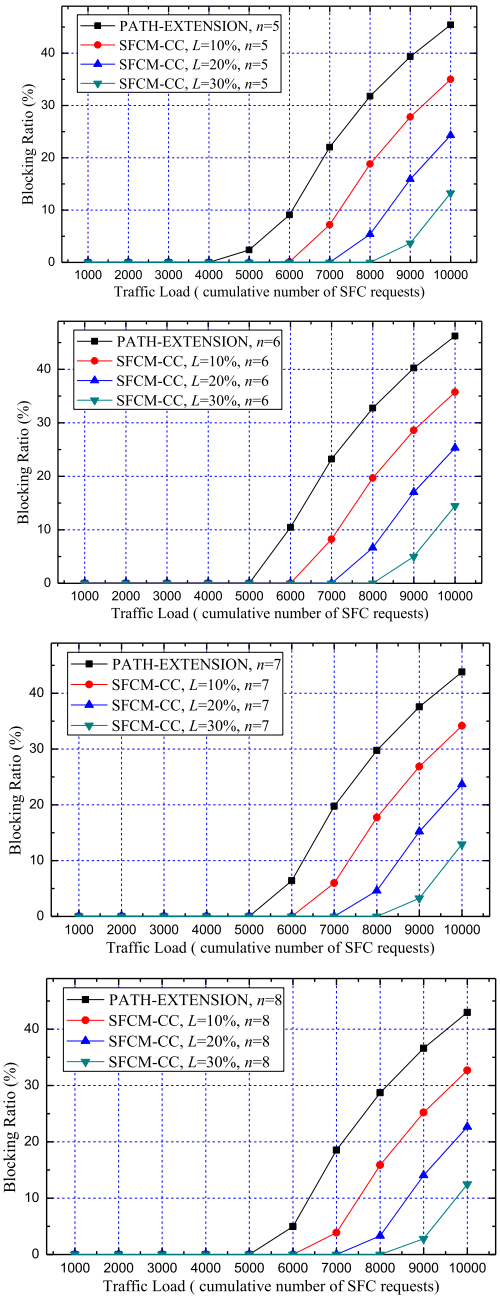


FIGURE 9. Blocking ratio.

PATH-EXTENSION algorithm, and the link mapping cost decreases as the percentage of live online services increases. When the percentage of live online services is increased by 10%, the link mapping cost is reduced by approximately 10% because in the SFCM-CC algorithm, the number of SFC links in the combined SFC are far less than the number of SFC links in the set of homogeneous SFCs before combining. Therefore, the bandwidth resource demands of the SFC links in the combined SFC are far lower than that of the SFC links in the set of homogeneous SFCs before combining. Thus, the link mapping cost of the SFCM-CC algorithm is lower than that of the PATH-EXTENSION algorithm, and when the percentage of live online services is increased by 10%, the link mapping costs of the SFCM-CC algorithm are reduced by approximately 10%.

Fig. 8 shows the total mapping costs of the PATH-EXTENSION algorithm and the SFCM-CC algorithm when the percentage of live online services is 10%, 20% and 30% and the number of VNFs in the original SFC request is varied among 5, 6, 7 and 8. Fig. 8 shows that when the percentages of live online services are 10%, 20% and 30%, the total mapping costs of the SFCM-CC algorithm are lower than those of the

PATH-EXTENSION algorithm, and the total mapping cost decreases as the percentage of live online services increases. The total mapping costs are reduced by approximately 10% when the percentage of live online services is increased by 10% because the total mapping cost is the sum of the VNF mapping cost and the link mapping cost, and in the SFCM-CC algorithm, the SFC requests for supporting live online services are classified into multiple sets of homogeneous SFCs and then each set of homogeneous SFCs is combined into a SFC. Because the combined SFC only needs to transmit a video to the new VNF *CD*, the resource requirements of the combined SFC are far lower than that of the set of homogeneous SFCs before combining. Therefore, the total mapping cost of the SFCM-CC algorithm is lower than that of the PATH-EXTENSION algorithm, and the total mapping cost decreases as the percentage of live online services increases.

Fig. 9 demonstrates the blocking ratios of the PATH-EXTENSION algorithm and the SFCM-CC algorithm when the percentage of live online services is 10%, 20% and 30% and the number of VNFs in the original SFC request is varied among 5, 6, 7 and 8. Fig. 9 shows that the blocking ratio of the SFCM-CC algorithm is lower than that of the PATH-EXTENSION algorithm. When the percentage of live online services is increased by 10%, the blocking ratio of the SFCM-CC algorithm is reduced by 10% because the SFCM-CC algorithm can combine the SFC requests for supporting live online services according to the classified homogeneous SFCs so that the resource requirements of the combined SFC are far less than the resource requirements of the set of homogeneous SFCs before combining. Therefore, the blocking ratio of the SFCM-CC algorithm is lower than that of the PATH-EXTENSION algorithm. In real-world applications, live online services often result in considerable network traffic and network congestion. Fig. 9 shows that our proposed method can effectively combine and deploy these services and help solve network congestion.

VI. CONCLUSION

In this paper, we research the SFC combination and deployment problem in cloud-fog computing environments. To solve this problem, we first model the deployment problem of SFCs as integer linear programming and then present the SFCM-CC algorithm, which is an efficient SFC combination and deployment algorithm. In our proposed SFCM-CC algorithm, we assume that a group of SFC requests are given and the locations of the user and the service terminal of each SFC request are also randomly given. In the SFCM-CC algorithm, we first call the CMHSFC procedure for classifying and combining homogeneous SFCs. Then, we call the MSFC procedure for mapping each SFC request. While we map each SFC request, we also allocate the computing resources and the bandwidth resources for the SFC request at the same time. Finally, we update the substrate network resources when we successfully map each SFC request. Last, we use the USANET network as the substrate core network and implement fog access networks to conduct extensive simulations to estimate the performance of the SFCM-CC algorithm. The results show that our proposed algorithm can effectively reduce the total mapping cost, the VNF mapping cost, and the link mapping cost and can effectively solve network congestion issues caused by live online services. When the percentages of live online services are 10%, the total mapping cost, the VNF mapping cost, the link mapping cost and the blocking ratios are reduced by approximately 10%, respectively.

REFERENCES

[1] M.-T. Thai, Y.-D. Lin, and Y.-C. Lai, "A joint network and server load balancing algorithm for chaining virtualized network functions," in *Proc. IEEE ICC*, May 2016, pp. 1–6.

[2] G. Sun, Y. Li, D. Liao, and V. Chang, "Service function chain orchestration across multiple domains: A full mesh aggregation approach," *IEEE Trans. Netw. Service Manage.*, vol. 15, no. 3, pp. 1175–1191, Sep. 2018.

[3] G. Sun, Y. Li, Y. Li, D. Liao, and V. Chang, "Low-latency orchestration for workflow-oriented service function chain in edge computing," *Future Gener. Comput. Syst.*, vol. 85, pp. 116–128, Aug. 2018.

[4] M. Jalalitabar, G. Luo, C. Kong, and X. Cao, "Service function graph design and mapping for NFV with priority dependence," in *Proc. IEEE Globecom*, Dec. 2016, pp. 1–5.

[5] G. Sun, M. Yu, D. Liao, and V. Chang, "Analytical exploration of energy savings for parked vehicles to enhance VANET connectivity," *IEEE Trans. Intell. Transp. Syst.*, Jun. 2018, doi: 10.1109/TITS.2018.2834569.

[6] M. Xia, M. Shirazipour, Y. Zhang, H. Green, and A. Takacs, "Optical service chaining for network function virtualization," *IEEE Commun. Mag.*, vol. 53, no. 4, pp. 152–158, Apr. 2015.

[7] J. Sun, S. Sun, K. Li, D. Liao, A. Sangaiah, and V. Chang, "Efficient algorithm for traffic engineering in Cloud-of-Things and edge computing," *Comput. Elect. Eng.*, vol. 69, pp. 610–627, Jul. 2018.

[8] A. Sandhu, R. Sohal, S. K. Sood, and V. Chang, "A cybersecurity framework to identify malicious edge device in fog computing and cloud-of-things environments," *Comput. Secur.*, vol. 74, pp. 340–354, May 2018.

[9] K. Liang, L. Zhao, X. Chu, and H.-H. Chen, "An integrated architecture for software defined and virtualized radio access networks with fog computing," *IEEE Netw.*, vol. 31, no. 1, pp. 80–87, Jan. 2017.

[10] F. Jalali, K. Hinton, R. Ayre, T. Alpcan, and R. Tucker, "Fog Computing May Help to Save Energy in Cloud Computing," *IEEE J. Sel. Areas Commun.*, vol. 34, no. 5, pp. 1728–1739, May 2016.

[11] M. Chiang and T. Zhang, "Fog and IoT: An overview of research opportunities," *IEEE Internet Things J.*, vol. 3, no. 6, pp. 854–864, Dec. 2016.

[12] X. Hou, Y. Li, M. Chen, D. Wu, D. Jin, and S. Chen, "Vehicular fog computing: A viewpoint of vehicles as the infrastructures," *IEEE Trans. Veh. Technol.*, vol. 65, no. 6, pp. 3860–3873, Jun. 2016.

[13] D. Zeng, L. Gu, S. Guo, Z. Cheng, and S. Yu, "Joint optimization of task scheduling and image placement in fog computing supported software-defined embedded system," *IEEE Trans. Comput.*, vol. 65, no. 12, pp. 3702–3712, Dec. 2016.

[14] X. Li and C. Qian, "Low-complexity multi-resource packet scheduling for network function virtualization," in *Proc. IEEE INFOCOM*, Apr. 2015, pp. 1400–1408.

[15] Y. Li, L. T. X. Phan, and B. T. Loo, "Network functions virtualization with soft real-time guarantees," in *Proc. IEEE INFOCOM*, Apr. 2016, pp. 1–9.

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

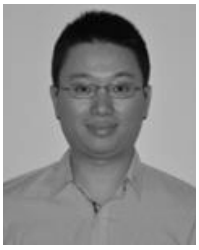
[17] S. Mehraghdam, M. Keller, and H. Karl, "Specifying and placing chains of virtual network functions," in *Proc. IEEE 3rd Int. Conf. Cloud Netw. (CloudNet)*, Oct. 2014, pp. 7–13.

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

[19] Z. Ye, X. Cao, J. Wang, H. Yu, and C. Qiao, "Joint topology design and mapping of service function chains for efficient, scalable, and reliable network functions virtualization," *IEEE Netw.*, vol. 30, no. 3, pp. 81–87, May 2016.

[20] M. Peng, S. Yan, K. Zhang, and C. Wang, "Fog-computing-based radio access networks: Issues and challenges," *IEEE Netw.*, vol. 30, no. 4, pp. 46–53, Jul./Aug. 2016.
[21] S. Yan, M. Peng, and W. Wang, "User access mode selection in fog computing based radio access networks," in *Proc. IEEE ICC*, May 2016, pp. 1–6.
[22] L. M. Vaquero and L. Rodero-Merino, "Finding your way in the fog: Towards a comprehensive definition of fog computing," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 44, no. 5, pp. 27–32, Oct. 2014.
[23] D. Koo and J. Hur, "Privacy-preserving deduplication of encrypted data with dynamic ownership management in fog computing," *Future Gener. Comput. Syst.*, vol. 78, pp. 739–752, Jan. 2018.
[24] R. Deng, R. Lu, C. Lai, and T. H. Luan, "Towards power consumption-delay tradeoff by workload allocation in cloud-fog computing," in *Proc. IEEE ICC*, Jun. 2015, pp. 3909–3914.
[25] D. Bruneo *et al.*, "Stack4Things as a fog computing platform for Smart City applications," in *Proc. IEEE INFOCOM*, Apr. 2016, pp. 848–853.
[26] J. Elias, F. Martignon, S. Paris, and J. Wang, "Efficient orchestration mechanisms for congestion mitigation in NFV: Models and algorithms," *IEEE Trans. Services Comput.*, vol. 10, no. 4, pp. 534–546, Jul. 2017.
[27] C. Pham, N. Tran, S. Ren, W. Saad, and C. Hong, "Traffic-aware and energy-efficient vNF placement for service chaining: Joint sampling and matching approach," *IEEE Trans. Services Comput.*, to be published.
[28] P.-W. Chi, Y.-C. Huang, and C.-L. Lei, "Efficient NFV deployment in data center networks," in *Proc. IEEE ICC*, Jun. 2015, pp. 5290–5295.
[29] J. Liu, Z. Jiang, N. Kato, O. Akashi, and A. Takahara, "Reliability evaluation for NFV deployment of future mobile broadband networks," *IEEE Wireless Commun.*, vol. 23, no. 3, pp. 90–96, Jun. 2016.
[30] A. Shameli-Sendi, Y. Jarraya, M. Pourzandi, and M. Cheriet, "Efficient provisioning of security service function chaining using network security defense patterns," *IEEE Trans. Services Comput.*, Oct. 2016, doi: 10.1109/TSC.2016.2616867.
[31] L. Qu, C. Assi, and K. Shaban, "Delay-aware scheduling and resource optimization with network function virtualization," *IEEE Trans. Commun.*, vol. 64, no. 9, pp. 3746–3758, Sep. 2016.
[32] S. Khebbache, M. Hadji, and D. Zeghlache, "Virtualized network functions chaining and routing algorithms," *Comput. Netw.*, vol. 114, pp. 95–110, Feb. 2017.
[33] D. Bhamare, M. Samaka, A. Erbad, R. Jain, L. Gupta, and H. A. Chan, "Optimal virtual network function placement in multi-cloud service function chaining architecture," *Comput. Commun.*, vol. 102, pp. 1–16, Apr. 2017.

DONGCHENG ZHAO is currently pursuing the Ph.D. degree in communication and information system with the University of Electronic Science and Technology of China. His research interests include network function virtualization, cloud computing, fog computing, and 5G mobile networks.



DAN LIAO received the B.S. degree in electrical engineering from the University of Electronic Science and Technology of China (UESTC) in 2001 and the Ph.D. degree in communication and information engineering from UESTC. His research interests are in the area of wired and wireless computer communication networks and protocols, next generation network. He is currently a Professor with UESTC.

GANG SUN received the Ph.D. degree in communication and information engineering from the University of Electronic Science and Technology of China (UESTC) in 2012. He is currently an Associate Professor with UESTC. His research interests are in the area of network security, datacenter networking, and cloud computing.

SHIZHONG XU received the B.S., M.S., and Ph.D. degrees in electrical engineering from the University of Electronic Science and Technology of China (UESTC), Chengdu, China, in 1994, 1997, and 2000, respectively. He is currently a Professor with UESTC. His research interests include broadband networks, all-optical network, and next-generation networks.