Received February 8, 2018, accepted March 9, 2018, date of publication March 13, 2018, date of current version April 23, 2018.

Digital Object Identifier 10.1109/ACCESS.2018.2815614

A Reliability-Aware Approach for Resource Efficient Virtual Network Function Deployment

JIAN SUN[1], GUANGYANG ZHU[1], GANG SUN[1], (Member, IEEE), DAN LIAO[1], YAO LI[1], ARUN KUMAR SANGAIAH[2], MUTHU RAMACHANDRAN[3], AND VICTOR CHANG[4]

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

[2]School of Computing Science and Engineering, Vellore Institute of Technology, Vellore 632014, India

[3]Leeds Beckett University, Leeds LS1 3HE, U.K.

[4]Xi'an Jiaotong-Liverpool University, Suzhou 215123, China

Corresponding author: Dan Liao (dliao.uestc@gmail.com)

This work was supported in part by the National Natural Science Foundation of China under Grant 61571098, in part by the Fundamental Research Funds for the Central Universities under Grant ZYGX2016J217, and in part by the 111 Project under Grant B14039.

ABSTRACT Network function virtualization (NFV) is a promising technique aimed at reducing capital expenditures (CAPEX) and operating expenditures (OPEX), and improving the flexibility and scalability of an entire network. In contrast to traditional dispatching, NFV can separate network functions from proprietary infrastructure and gather these functions into a resource pool that can efficiently modify and adjust service function chains (SFCs). However, this emerging technique has some challenges. A major problem is reliability, which involves ensuring the availability of deployed SFCs, namely, the probability of successfully chaining a series of virtual network functions while considering both the feasibility and the specific requirements of clients, because the substrate network remains vulnerable to earthquakes, floods, and other natural disasters. Based on the premise of users' demands for SFC requirements, we present an ensure reliability cost saving algorithm to reduce the CAPEX and OPEX of telecommunication service providers by reducing the reliability of the SFC deployments. The results of extensive experiments indicate that the proposed algorithms perform efficiently in terms of the blocking ratio, resource consumption, time consumption, and the first block.

INDEX TERMS Network function virtualization, service function chains, reliability, economical networking.

I. INTRODUCTION

Telecommunication service providers (TSPs) desire flexible and cost-efficient methods for dispatching network services as market demands increase. Network function virtualization (NFV) provides an opportunity to efficiently and dynamically deploy service function chains (SFCs) [1]–[6] without modifying dedicated infrastructure, which is costly and has become complex over time. Due to advances in NFV, network operators can implement SFCs to guarantee services that are both elastic and agile. Thus, reconfiguring the network topology when necessary is more convenient and less expensive. The basic idea behind NFV is to decouple these network functions (e.g., firewall, WAN optimizers, intrusion prevention systems, switches, and proxies) from the underlying customized devices and accomplish equivalent network functions via software-based functions running in virtual machines (VMs) deployed on commercial off-the-shelf (COTS) devices. As shown in Fig. 1, one software based virtual machine can perform several network functions. Traditionally, TSPs use middleware—usually based on dedicated hardware devices or software—to deploy network functions. Although TSPs offer valuable advantages in terms of function provision, such offers consume a non-negligible fraction of network operators' capital expenditures (CAPEX) and operating expenditures (OPEX) [7]–[9], [13], [16]. Thus, using NFV technology, telecom operators can not only deploy network services using a cost-efficient approach but also satisfy users' various requirements, which are typically referred to as service level agreements (SLAs) for networking.

Virtualization began in the 1970s; since then, it has attracted significant attention for network domains [10]–[19]. Many problems derive from the concept of virtualization such

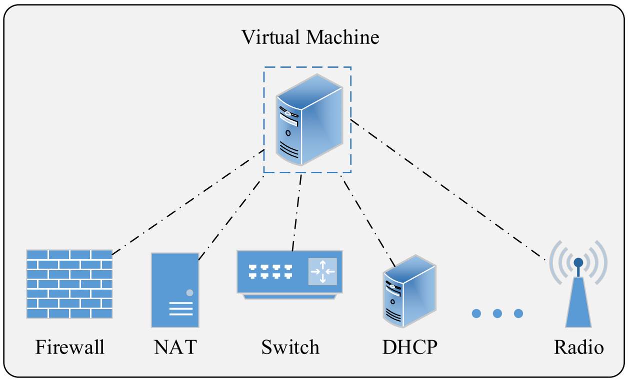

FIGURE 1. Functions that one virtual machine can accomplish.

as the virtual network mapping problem detailed in [10]–[13], and the migration of VMs described in [15]–[19]. NFV enables network providers to implement scalable network services in an agile manner, meaning that TSPs are not inconvenienced by having to add or remove network services in the physical layer. Instead, they can simply implement new functions or delete redundant functions in a virtualized environment (which is in the virtualization layer). Thus, this topic has been extensively investigated by industry and academia as the potential future of networking [20]–[25]. Many studies of virtual network function (VNF) placement have been performed to better serve clients and reduce expenditures [25]–[30]. Some research challenges exist, including NFV management, performance, and orchestration for networking [20], [24]. These challenges provide valuable opportunities because resolving such issues helps NFV to become more mature and applicable.

Since the emergence of NFV, standard descriptions have been developed by the European Telecommunications Standards Institute (ETSI) and some studies have investigated the architecture of NFV [35]–[39]. A simple architecture of NFV is depicted in Figure 2. The virtualization layer that contains all the virtual machines and the physical layer that contains all the substrate nodes have compute, storage and network resources to serve clients NFV environments. The network function virtualization infrastructure (NFVI) is a network service that has been referred to as a service function chain and consists of a series of VNFs. One VNF represents one real network function, as depicted in Fig. 1.

Because the reliability of NFV is critical and is a prerequisite for successfully executing SFCs and satisfying SLAs, improving reliability while reducing the cost of network providers is a research objective in academic and industrial arenas. Thus, the more network services that are mapped onto the substrate network, the greater the revenue of TSPs. Similarly, the high-performance demands of users will influence the cost of TSPs.

In this paper, we investigate how to improve the reliability demand for users by mapping users' requests onto the substrate network. We propose an ER algorithm to solve this problem. We consider that high request reliability is not

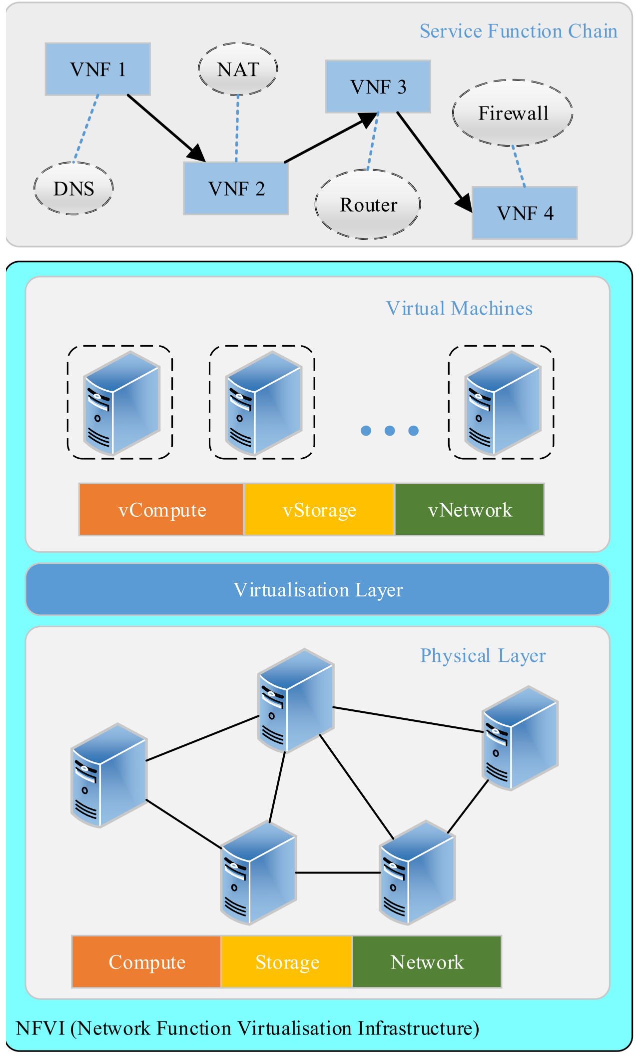

FIGURE 2. Abstract architecture of NFV.

always needed for TSPs. High reliability requires TSPs to increase CAPEX and OPEX. If we can properly reduce the reliability, we can also reduce CAPEX and OPEX. We first propose the algorithm ER_CS (based on ER) that works in conjunction with the load balancing of the substrate network. However, by analyzing the deployment scheme in ER_CS, we discover that it does not appear to be the best scheme. Therefore, we further propose the ER_CS_ADJ algorithm to adjust the deployment scheme by minimizing SFC resource consumption in the physical network. We conduct massive simulations on arbitrary topologies to verify the effectiveness of these algorithms. From the simulations and results, we determine that our network algorithms are profitable in terms of resource cost, block ratio and deployment time. The main contributions in this paper are as follows:

- The primary contribution of this paper is the development of the ER_CS algorithm, which reduces the cost of resources (both computing resources and bandwidth resources), lightening the load on the substrate network.

Using these uncomplicated operations, we can help TSPs reduce user costs and energy consumption. Simultaneously, service prices can decrease due to sharing and analysis of network intelligence, forming an economical strategy and trade-off for both TSPs and users.

- While restricting access to computing resources and bandwidth resources and relaxing reliability requirements for users, we can describe the reliability-aware VNF deployment problem as a mathematical optimization problem. Decreasing the reliability of SFC appropriately during deployment is the essence of our work.
- We propose an algorithm called ER to ensure the reliability of the deployment scheme, through which we can satisfy users' demands. We deploy VNF nodes in the SFC one by one, deploying one VNF on one substrate node. Then, the algorithm finds another unused substrate node that has the maximum reliability to the prior node and deploys the next VNF node on this substrate.
- We adjust the ER_CS algorithm to efficiently decrease the resource allocation for the substrate network in NFV environments.

The remainder of this paper is organized as follows. In Section 2, we analyze related studies. In Section 3, we describe the problem in this research with some formulations. In Section 4, we propose our heuristic algorithm and provide line-by-line details. A performance evaluation of our proposed algorithm is presented in Section 5, and Section 6 concludes this work.

II. RELATED WORK

To satisfy various requests from users, service providers are eager to seek a flexible, scalable, agile, effective, resource-efficient and energy-efficient scheme for placing VNFs. Ensuring service reliability while finding an economical and resource-efficient solution to the problem of VNF deployment is the goal of this work.

Numerous studies are relevant to NFV, including how to determine and place network functions. In [2], Li *et al.* proposed an -based algorithm to provide an efficient method for solving the VNF placement problem. However, this work only simulates the performance of the convergence time and the performance of the acceptance rate for the proposed algorithm and two other provided algorithms and does not consider the resource consumption and transmission delay of the request. Li *et al.* [3] presented a set of affinity and anti-affinity constraints that can be used by TSPs to define placement constraints. They proposed a semantic conflict mechanism to evaluate SFC requests that filters invalid mechanisms to reduce the mapping time. Liu *et al.* [32] designed a heuristic NFV deployment algorithm to allocate, place, and dispatch the traffic for VNFs. They highlight the relationship between the number of VNFs and east-west traffic growth, which they claim is at the root of the VNF placement problem.

Some researchers have considered the problem of improving NFV performance, for example, by optimizing the stringent delay constraints. In [5], the VNF deployment problem was solved by considering the optimization of inter-cloud traffic and response time in a multi-cloud network in NFV environments. The response time includes both link delay and compute delay. In [10], Chowdhury *et al.* focused on the VNF scheduling and resource allocation problems as well as on transmission and processing delays. They aimed to minimize the total network function scheduling latency with strict delay constraints by developing a network algorithm. In [14], Li *et al.* considered that current NFV platforms preclude operating at the network edge. They proposed the Glasgow Network Function, which is a platform based on container VNFs that runs and orchestrates lightweight container VNFs, reduces core network utilization and provides lower latency. Oljira *et al.* [33] conducted experiments to study the impact of virtualization on network delay; their simulations show that end-to-end latency will increase in a virtualized environment.

The performance of NFVs with regard to resource allocation or consumption and the acceptance ratio when mapping VNFs has been investigated for years. A comprehensive resource allocation survey was conducted in [21]. In [28], Fan *et al.* studied the VNF placement and scheduling problem in the radio access network (RAN) domain. They formulated this problem as an integer linear programming (ILP) problem and proposed a heuristic algorithm to solve it. They demonstrated that their algorithm performed better regarding the acceptance ratio, the cost of deployment, and the utilization of the nodes and links. Li *et al.* [29] proposed a genetic algorithm to optimize resource allocation. They demonstrated its efficiency in optimizing resource allocation via three network function centers (NFCs) proposed by the authors.

Some applications have addressed optical networks [8], [9], [22], [30]. The authors studied how to jointly optimize the VNF placement and spectrum assignment, which is a controversial topic. In [8] and [9], the common goal was to cost-effectively realize VNF placements. As previously stated, finding economical schemes for VNF placement has become a common objective for both TSPs and users. Li *et al.* [16] recognized that reducing CAPEX/OPEX was the main goal. In addition to the resource-efficient VNF placement problem, power or energy-efficient service request placement is a controversial research topic [18].

Other research projects have focused on issues such as the availability of NFV. Due to potential failures (such as node or link failures) that can be caused by earthquakes, floods, or malfunctions such as power outages, many researchers have expressed interest in the field of high availability (HA) to protect data or network functions. Unlike some schemes, which aim to solve general VN mapping problems for unicast services (which includes two procedures: virtual node and link mapping) such as [10] and [11], Gao *et al.* proposed the MILP model in [13] to maximize the availability using max-min fairness for multicast VN mapping services. Wang *et al.* [34] proposed an efficient framework for evaluating the reliability of NFV deployments; however, they did not investigate how to adjust NFV deployments based on their framework. The proposed framework can be used only to evaluate deployment schemes but was not intended to improve the schemes based on its results. Trajano *et al.* [35] proposed a novel approach for improving the robustness of the substrate equipment by employing channel coding to improve the robustness of the physical devices in NFV architecture.

Although numerous studies have considered the reliability of deployed SFCs, few studies have considered the needs of users while also considering the TSP revenues. In other words, few studies have focused on building an economical network environment. Therefore, we propose the ER-CS algorithm to reduce reliability under the premise of guaranteeing users' demands while also considering economical VNF deployments.

III. PROBLEM STATEMENT AND FORMULATION

A. NETWORK MODEL

1) SUBSTRATE NETWORK

A substrate network consists of the underlying nodes that are directly connected via physical links between the nodes. Each physical node has a set of service functions with resource attributes, and every physical link has a corresponding bandwidth capacity. We represent the underlying network as the graph $G_P = (V_P, E_P)$, where $V_P = \{v_1, v_2 \ldots v_{|VP|}\}$ is the set of substrate nodes, $|V_P|$ represents the number of physical nodes, $E_P = \{e_1, e_2, \ldots, e_{|EP|}\}$ is the set of edges, and $|E_P|$ denotes the number of physical links.

2) SFC REQUEST

An SFC request typically consists of multiple virtual nodes interconnected by virtual links. These virtual nodes have specific network functions. Different SFCs may have the same function and are likely to share the same underlying physical nodes, which reduces network resource usage. This paper does consider the functionality of VNF in SFC, assuming that a virtual machine can be mapped to different network functions as long as the conditions imposed by the underlying resources are satisfied. A virtual machine corresponds to a node in the underlying layer. Here, we use $SR = (N_S, L_S, s, t)$ as the SFC request. $N_S = \{f_1, f_2 \ldots f_{|NS|}\}$ is a collection of network functions, and $|N_S|$ represents the number of functions of the request. $L_S = \{l_1, l_2 \ldots l_{|LS|}\}$ denotes the set of SFC links, and $|L_S|$ is the number of service links involved in the request. The symbols "s" and "t" in SR respectively denote the source and destination nodes of the request and represent two nodes in the underlying network.

3) SFC MAPPING

The process of mapping SFC requests to physical networks is called SFC mapping. The resources and functions of the assigned underlying nodes must meet the needs of the virtual nodes. The bandwidth capacity of allocated physical links should be no less than the required bandwidth capacity of the virtual links. In this paper, the achieved SFC deployment scheme can be represented as $P^S = (V_N^S, E_L^S) \cdot V_N^S = V_t^S + V_f^S$ represents the collection of all underlying nodes involved in the deployment scheme, which consists of two parts: V_t^S represents the deployed SFC's forwarding node set, and V_f^S represents the function node set. E_L^S is the set of deployed paths for each service link.

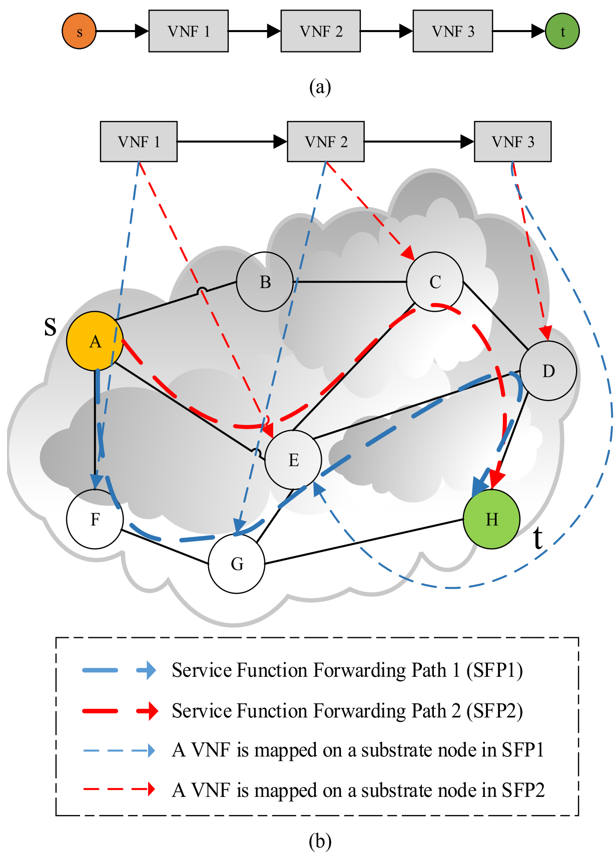

FIGURE 3. Example of mapped VNFs. (a) Service Function Chain request. (b) Mapped SFC on the substrate network.

B. PROBLEM STATEMENT

As described in Figure 3, an SFC request consists of several VNFs, a source node s and a destination node t. Each of these VNFs represents a network function, as described above. The thick blue dashed line represents another scheme whose reliability is 0.94 and resource consumption is 202, called service function forwarding path 1 (SFP1). The thick red dotted line represents one deployment scheme for the request whose reliability is 0.97 and resource consumption is 232, called service function forwarding path 2 (SFP2). We assume that the demand reliability of users is 0.90. The thin blue dashed line, which represents a VNF in SFC, is deployed on a substrate network in SFP1. The red line will yield the best experience for the users, whereas the blue line will generate a better balance for the network providers because the network can hold more requests, which allows greater potential profits for TSPs. The goal of this paper is to find a deployment scheme that both satisfies users' reliability demands and minimizes resource consumption to reduce costs (i.e., resource consumption and load balancing).

This paper focuses on solving the reliability-aware problem, in which SFCs are mapped to the substrate network in a NFV scenario. The high reliability requirements of users usually demand expensive and high-performance physical equipment provided by operators, which significantly increases the cost for TSPs and prevents users from enjoying high-quality network services at low prices. To achieve effective and reliable network services while deploying SFC requests, we need to deploy VNFs to more reliable nodes and attempt to maximize the total availability of the deployment of SFC. This goal can be notated as follows:

$$\max \left\{ R^S = \prod_{v_p \in V_N^S} r_{v_p} \times \prod_{e_p \in E_L^S} r_{e_p} \right\}$$
$$\forall v_p \in V_P, \quad 0 < r_{v_p} < 1.0$$
$$\forall e_p \in E_P, \quad 0 < \mathrm{r}_{e_p} < 1.0 \tag{1}$$

where r_{vp} and r_{ep} represent the reliability of the nodes and links deployed for SFC requests, respectively, v_p denotes any node in the underlying network, and e_p denotes any link in the underlying network. The reliability of each node and link in the underlying network is denoted by a positive number less than 1 according to the constraint behind the optimization objective. This paper estimates the total reliability of SFC by calculating the product of the reliability of each substrate node and link involved in a SFC deployment scheme.

Due to limited resources, considering only the reliability of SFC may cause enormous resource consumption and reduce the mapping success rate. Therefore, the paper aims to solve the contradiction between the reliability and the bandwidth consumption, maintaining a balance between resource consumption and service reliability to ensure the effective use of resources.

The problem involves designing algorithms to obtain the optimal SFC deployment scheme to satisfy users' high reliability requirements while effectively reducing resource consumption. In this paper, we address three specific problems:

Problem 1: A specific number of SFC requests, physical nodes and links with certain reliability, computing resources and bandwidth, and the source and destination nodes of each SFC are given. The objective is to find the optimum scheme for SFC mapping P^S that maximizes the total availability of every SFC. In this scheme, each physical node is matched to only one function for each SFC but it can be regarded as a switch node while calculating P^S.

Problem 2: The SFC requests are the same as those described in Problem 1. To guarantee a certain degree of reliability, the objective is to achieve an ideal scheme of SFC mapping P^S using a load balancing method. Each node is matched to only one function in each SFC.

Problem 3: Based on *Problem 2*, we consider resource consumption. When given the optimal scheme provided by *Problem 2*, the objective is to find one feasible strategy to improve this scheme in terms of reducing resource consumption.

C. VARIABLE DEFINITIONS AND CONSTRAINTS

1) VARIABLE DEFINITIONS

We define the variables and parameters in this paper as follows:

- $RS = \{SR_1, SR_2, \ldots, SR_n\}$: the request set;
- G_P: the topology graph $G_P = (V_P, E_P)$ represents the physical network;
- R^U: the reliability request of a user;
- $v_t \in V_t^s$: the deployed SFC's forwarding node;
- $v_f \in V_f^s$: the node onto which the VNF is deployed;
- $w_{v_i}^r$: The remaining computing resources of v_i, $v_i \in V_P$;
- $m_{e_{v_i}^{o}}^r$: the remaining bandwidth resource of the out-degree edge of vertex v_i;
- $e_{l_i}^s$: the physical edge in the SFC deployment path of the link l_i in the physical network, $l_i \in L_S$;
- $\lambda_{l_i}^s$: the SFC deployment path of the link l_i in the physical network;
- V_{remain}: the set of remaining vertexes that are not deployed as VNFs in the physical graph, $V_{remain} \in V_P$;
- $r_v^{v_{so}}$: the total reliability from node v to the source node, $\forall v \in V_P$;
- $r_v^{v_{si}}$: the total reliability from node v to the destination node;
- v_∞: a node that does not exist in the substrate graph;
- v_{so}^e: the source node of edge e in the substrate graph;
- v_{si}^e: the destination node of edge e in the substrate graph.

2) NETWORK RESOURCE CONSTRAINTS

Different virtual links may be mapped onto the same underlying physical path and share the underlying physical resources. However, they are independent, and the same bandwidth resources cannot be simultaneously employed by different virtual links.

3) NODE OR LINK CAPACITY CONSTRAINTS

$$w_{v_{n_i}^s}^r \geq w_{n_i}, v_{n_i}^s \in V_t^S, n_i \in N_S \tag{2}$$
$$\sum_{r \in RS} \sum_{n_i \in V_f^r} w_{v_{n_i}^s}^r \leq w_v^{total}, \quad \forall v \in V_P \tag{3}$$
$$m_{e_{l_i}^s}^r \geq m_{l_i}, e_{l_i}^s \in \lambda_{l_i}^s \in E_L^S \tag{4}$$
$$\sum_{r \in RS} \sum_{l_i \in E_L^r} m_{e_{l_i}^s}^r \leq m_e^{total}, \quad \forall e \in E_P \tag{5}$$
$$\zeta_{n_f}^v = \begin{cases} 1, & \text{if VNF } n_f \text{ is deployed on node } v \\ 0, & \text{otherwise} \end{cases} \tag{6}$$
$$\zeta_{n_t}^v = \begin{cases} 1, & \text{if node } v \text{ is forwarding node of the request} \\ 0, & \text{otherwise} \end{cases} \tag{7}$$
$$0 \leq \sum_{v \in V_P} \zeta_{n_f}^v \leq 1, \quad \forall n_f \in N_S \tag{8}$$
$$0 \leq \zeta_{n_t}^v + \zeta_{n_f}^v \leq 1 \tag{9}$$
$$0 \leq \zeta_{n_i}^v + \zeta_{n_j}^v \leq 1, \quad i \neq j, \quad \forall (i,j) \in N_S, \quad \forall v \in V_P \tag{10}$$

The constraints (2) and (3) ensure the computing resources of the substrate node. Constraint (2) indicates that the remaining computing capacity of the physical node which the VNF is deployed onto must be greater than the required computing resources of the VNF node. For all substrate nodes, constraint (3) ensures that the sum of the computing resources required by all the VNF instances from various SFC requests deployed on it does not exceed its availability resource. Constraints (4) and (5) represent bandwidth resource constraints. Constraint (4) denotes that the remaining bandwidth resource of the physical edge e_{li} satisfies the bandwidth demand of the virtual link l_i in the SFC. For all substrate end-to-end paths, Constraint (5) guarantees that the sum of the bandwidths required by all the virtual links deployed to it does not exceed its available capacity. A virtual node or link can be successfully mapped to a physical node or link of the underlying network only when both the computing capacity and bandwidth capacity conditions are satisfied. When a SFC request arrives, the physical network must allocate the corresponding nodes or links that satisfy the node and link resource requirements. When the physical network resources are insufficient, the SFC request should be rejected or delayed.

Formulas (6) and (7) mathematically describe the VNF nodes, forwarding nodes and substrate nodes. If a VNF node is mapped onto a substrate node, the value of the variable in (6) is one. If a substrate node is a forwarding node, the value of this variable in (7) is one. Constraint (8) ensures that any VNF node can be deployed on only one or no nodes in the physical network. Constraint (9) indicates that the nodes in the physical network can be deployed only as either function nodes or forwarding nodes. The underlying nodes cannot be both function nodes and forwarding nodes. In (10), no two different VNF nodes in a SFC request can be deployed on the same physical node.

IV. ALGORITHM DESIGN

In this section, we describe our proposed algorithms for the reliability-aware SFC mapping problem. We present three main algorithms: the heuristic algorithm ER, based on reliability guarantee; the heuristic algorithm ER-CS, which is based on load balancing while ensuring reliability and reducing the cost of TSPs; and the bandwidth-optimizing algorithm ER_CS_ADJ. We assume that the reliability of all the substrate nodes and links are known and can be used to compute the reliability of the complete mapping path.

A. RELIABILITY-GUARANTEED ALGORITHM ER

In a NFV environment, many virtual networks share one substrate network; consequently, the failure of one substrate link or substrate node may cause massive failures in virtual networks, have a large-scale impact, and reduce network stability. Therefore, we propose a heuristic algorithm referred to as ER based on the reliability-aware SFC mapping problem.

The ER algorithm aims to improve the reliability of an SFC mapping scheme. In the ER algorithm, we map the VNFs in an SFC one by one. One VNF mapped to one substrate vertex and the virtual link between two VNFs may be a either a single substrate link or a path composed of several links.

Algorithm 1 Ensure reliability (ER)

Input: 1. Substrate network $G_P = (V_P, E_P)$;
  2. SFC request $SR = (N_S, L_S, s, t)$.
Output: SFC deployment scheme P^S, $v_{so} = s$, $v_{si} = t$.
1: Initialization: **let** $V_{remain} = V_P$;
2: **for** all VNF n_f in SR, **do**
3: **if** $n_f$is not the last VNF of SFC, **then**
4: initiateAllVertex() and **let** $r_{v_{so}}^{v_{so}} = r_{v_{so}}$;
5: Call URSO procedure 1 to update the information;
6: **let** $\kappa_r = -\infty$ and $v_{temp} = v_\infty$;
7: **for** each vertex v in V_P, **do**
8: **if** $v \neq v_{si}$ and $w_v^r \geq w_{n_f}$ and $\kappa_r < r_v^{v_{so}}$, **then**
9: $\kappa_r = r_v^{v_{so}}$, $v_{temp} = v$;
10: **end if**
11: **end for**
12: **if** $\kappa_r = -\infty$, **then**
13: **return** null;
14: **end if**
15: generateScheme(κ_r, n_f)
16: **else**
17: repeat the process in line 4 and 5, $r_{v_{si}}^{v_{si}} = r_{v_{si}}$
18: call URSI procedure 2 to update the information;
19: **for** each vertex v in V_P, **do**
20: **if** $w_v^r \geq w_{n_f}$ and $\kappa_r < r_v^{v_{si}} \times r_v^{v_{so}}/r_v$, **then**
21: $\kappa_r = r_v^{v_{si}} \times r_v^{v_{so}}/r_v$, $v_{temp} = v$;
22: **end if**
23: **end for**
24: repeat the process in line 12 to line 15;
25: **end for**

When one VNF is deployed, we choose the substrate vertex that enables the entire scheme to achieve maximum reliability based on the premise that the vertex has sufficient computing resources and bandwidth resources relative to the last VNF mapping vertex to satisfy the SFC demand. The pseudo-code is presented in Algorithm 1.

When receiving an SFC request, including its source and destination, the ER algorithm deploys the VNFs one by one and simultaneously maps the related virtual links. The initial source in this algorithm is the source vertex of one SFC. When one VNF is deployed, the source is set to the mapping vertex of this VNF to become the source of the next VNF. When mapping VNFs, the mapping method for the last VNF of an SFC request differs from the mapping method for previous VNFs in the ER algorithm.

For all the VNFs other than the last one, the ER algorithm initializes the reliability of all vertexes to the source to be negative infinity and the reliability of the source vertex to be its vertex's reliability. Then, it initializes their prior vertex on the path to the source to be an inaccessible node (i.e., a node not in this network). Next, it calls procedure 1—update all reliability to source (URSO)—to update the reliabilities of all nodes to the SFC source, based on the premise that the bandwidth of each link on the path from the source to these nodes satisfies the SFC request. In lines 6 to 11, we initialize

Procedure 1 Update all reliability to source (URSO) values

Input: 1. Vertex set V_{remain};
2. Source vertex v_{so}; (i-1)th SFC link l_{i-1}
Output: Updated G_P
1: **let** $v_{so}^{temp} = v_{so}$;
2: **while** $V_{remain} \neq \emptyset$
3: **for** all out-degree edges e_i of v_{so}^{temp}, **do**
4: **if** $m^r_{e_i} \geq m_{l_i}$ and $v^{e_i}_{si} \in V_{remain}$ **then**
5: **if** $(t = r^{v_{so}}_{v^{temp}_{so}} \times r_{e_i} \times r_{v^{e_i}_{s_i}}) > r^{v_{so}}_{v^{e_i}_{s_i}}$ and $t > R^U$, **then**
6: **let** $r^{v_{so}}_{v^{e_i}_{s_i}} = t$, and v^{temp}_{so} be the prior vertex of $v^{e_i}_{si}$ on whose path to the source;
7: **end if**
8: **end if**
9: **end for**
10: ind the maximal reliability (to the source vertex) of vertex v_{max} in V_{remain};
11: $v^{temp}_{so} = v_{max}$;
12: delete vertex v in V_{remain};
13: **end while**

the maximum reliability variable and the substrate node that has the maximum reliability to map the VNF, and traverse all the nodes in the network topology graph to find the variable defined in line 6, which cannot be the sink vertex, and has sufficient computing resources to satisfy the SFC demand. We generate the mapping scheme and map the VNF onto the vertex v_{temp} with the reliability calculated in the previous procedure. Then, information about the path from the source to the v_{temp} is recorded in URSO. If the reliability variable remains negative infinity, we are unable to find a mapping vertex that satisfies the demands for mapping this VNF.

To map the last VNF in an SFC we must not only consider the mapping vertex's reliability to the previous VNF mapping vertex but also its accessibility and reliability at the destination node of the SFC. Similar to the previously described algorithm, we update the reliabilities of all nodes to the SFC's destination after updating the reliabilities to the SFC's source. When computing the reliability of the mapping vertex of the last VNF, the computational formula is expressed as follows:

$$\kappa_r = r^{v_{si}}_{v} \times r^{v_{so}}_{v} / r_v, \tag{11}$$

where the first symbol to the right of equation (11) denotes the reliability to the sink node of the SFC, the second symbol denotes the reliability to the previous VNF's mapping vertex in the substrate network, and the last symbol denotes the mapping vertex's own reliability.

Next, we present the pseudo-code for Procedure 1.

Procedure 1 updates the reliabilities of all nodes to the source node (i.e., the substrate node mapped by the previously mapped VNF). Similar to [40], we create the vertex set V_{remain}, which initially added all vertexes V_P of the network graph G_P in Algorithm 1. We set v^{temp}_{so} to be the mapping vertex of the previously deployed VNF (initially, it is the source node of SFC). While V_{remain} is not empty, we traverse all the out-degree edges of v^{temp}_{so} to determine whether the edge satisfies the SFC's bandwidth demand and the user's reliability request. The formula in line 4 indicates that the remaining bandwidth resource of edge e_i satisfies the bandwidth demand of l_i in the SFC. The subscript of the last symbol in (12) denotes the destination vertex of e_i. The formula for computing the reliability of the node in line 4 to the source vertex of SFC is expressed as follows:

$$t = r^{v_{so}}_{v^{temp}_{so}} \times r_{e_i} \times r_{v^{e_i}_{s_i}}, \tag{12}$$

where the first symbol on the right-hand side of equation (12) represents the reliability of v^{temp}_{so} to the source of the SFC, the second symbol denotes the reliability of edge e_i, and the third symbol represents the reliability of the node in line 4. In lines 4–8, we estimate whether the edge's remaining bandwidth resource satisfies the demand of link l_i, and whether its destination vertex is in V_{remain}. If the requirement is satisfied, we continue to compute t and determine whether t satisfies the user's reliability request R^U. Then, we update the variable described in line 6 and record the prior vertex on its path to the source. After traversing all the out-degree edges of v^{temp}_{so}, we assign v_{max}, which has the maximal reliability to the source, to v^{temp}_{so}. Finally, we delete vertex v_{max} from V_{remain}. Because we record the prior vertex on its path to the source, we eventually obtain a complete path from the source to the destination from Algorithm 1.

Procedure 2 (i.e., update all reliability to sink (URSI)) is similar to Procedure 1; the only difference is that rather than computing the reliability to the source, it computes the reliability to the destination.

B. RELIABILITY-GUARANTEED ALGORITHM ER-SC BASED ON LOAD BALANCING

To maximize the reliability, SFC functions should be deployed on vertexes with high reliability, which may cause imbalanced loading in the network. Because the network resources are limited and loads characteristically increase suddenly, imbalanced loading can waste resources and cause network congestion and instability, which will reduce TSP profits. Based on the reliability-guarantee algorithm ER, we introduce the idea of load balance and present the reliability-guarantee heuristic algorithm ER-RB, which is based on load balance.

In this thesis, the objective of load balance is to assign service flow transport to links with lighter loads to reduce the possibility of congestion caused by load imbalance. The following mathematical model describes load improvement:

$$\delta = \frac{1}{w^r_{v_i}} + \sum_{e^o_{v_i} \in e^o_i} \frac{1}{m^r_{e^o_{v_i}}} + m^{v_{so}}_{v_i}, \quad \forall v_i \in V_P \tag{13}$$

where the denominator of the first fraction represents the remaining computing resources of v_i, e^o_i denotes the set of the out-degree edge of vertex v_i, the denominator in the second fraction denotes the remaining bandwidth resource of the out-degree edge of vertex v_i, and the last symbol denotes the sum of the bandwidth cost of the path from vertex v_i to the source, v_{so}. The smaller the value of δ (load factor) is, the lighter the network load is. As expressed by the formula, the smaller the load factor is, the larger the vertex's remaining computing resource is, and the larger the remaining bandwidth resource of the out-degree is, the smaller the total bandwidth cost of the vertex to the source is. To achieve load balance, we should prefer the vertexes with smaller load factors for deploying SFC functions.

Therefore, we adjust the ER algorithm to compute the δ of all the vertexes that satisfy the criteria based on satisfying R^U, the node's computing resource demands and the link's bandwidth resource demands. We add a comparison of the values of δ to line 5 in URSO to find the vertexes with smaller δ values to host VNFs. Thus, we obtain a new deployment scheme that considers load balance based on the scheme generated by ER.

1) BANDWIDTH OPTIMIZATION ALGORITHM ER_SC_ADJ

The SFC mapping problem can be divided into two parts: SFC virtual node mapping and SFC virtual link mapping. SFC virtual node mapping requires that the substrate vertexes satisfy the virtual nodes' resource constrains and function demands, whereas SFC virtual link mapping requires that the substrate links of the substrate path satisfy the bandwidth resource demands of the virtual links. One virtual link in SFC can be mapped onto just one substrate link or onto several substrate links (one substrate path): the selection depends on the substrate vertexes onto which the VNF's virtual link connections are deployed. If we map the virtual link with the highest bandwidth demand onto the shortest possible substrate path, the bandwidth cost of this SFC mapping scheme can be reduced considerably.

Therefore, we improve the ER_SC algorithm through bandwidth cost reduction, and we propose the bandwidth optimizing algorithm ER_SC_ADJ. We skillfully adjust the VNFs' mapping position based on the mapping scheme generated by ER_SC to lengthen the mapping paths of virtual links with low bandwidth-demands and shorten the mapping path of the virtual links with high bandwidth demands; consequently, we reduce the bandwidth cost. The pseudocode for the algorithm is shown in Algorithm 2.

The function findMinLink(SFC) finds the virtual link with the minimum bandwidth request in the SFC. The VNFs behind this link are the VNFs that must be moved; we denote these as χ_{move}. When moving these VNFs, if we start at the first VNF, the previous VNFs may not have sufficient options to map, which may cause failure. Thus, we need to traverse the VNFs in reverse order. When we adjust the mapping position of one VNF, we traverse all the forwarding vertexes on the path between this VNF and the updated VNF in reverse order. For example, when moving the last VNF, we traverse forward from the first forwarding vertex prior to the destination of the SFC. When moving the penultimate VNF, the deployment position of the last VNF is determined; thus, we traverse forward from the deployment position of the last VNF. The remaining steps can be performed in the same manner.

Algorithm 2 ER_SC adjust (ER_SC_ADJ)

Input: SFC deployment scheme P^S.
Output: Adjusted SFC deployment scheme P^S.
1: $\chi_{move} = \text{findMinLink(SR)}$;
2: **if** $\chi_{move} = 0$, **then**
3:     **return**;
4: **end if**
5: **while** $\chi_{move} > 0$
6:     **for** all n_f need to be removed, **do**
7:       **for** all forwarding vertex v between two related function vertex, **do**
8:         **if** $w_v^r \geq w_{n_f}$ and $B_{remain}^{\min} \geq B_{request}$, **then**
9:           deploy n_f on vertex v;
10:         **end if**
11:       **end for**
12:     **end for**
13:     χ_{move}–;
14: **end while**

In line 8, while traversing the forwarding vertexes, we need to estimate whether the vertex's remaining computing resource satisfies the VNF's demand and whether the bandwidth resource of the links between this vertex and the two VNFs' deployed immediately before and immediately after it satisfy the request. When we find a forwarding vertex that can satisfy these requirements, we deploy the VNF on this vertex as the new function vertex and deploy the old function vertex (the one on which this VNF was previously deployed) as the forwarding vertex. Finally, we obtain a new SFC deployment scheme.

Note that the ER_SC_ADJ algorithm only adjusts the positions of forwarding vertexes and function vertexes locally based on the existing deployment scheme: the deployment path of the SFC has not changed. The reliability of the new SFC deployment scheme remains the same, which satisfies the user requirements. ER_SC_ADJ simply increases the utilization of bandwidth and reduces costs.

V. SIMULATION RESULTS AND ANALYSIS

This section describes extensive simulation experiments conducted to evaluate the performance of the proposed algorithms. The simulation environment is introduced, and several performance parameters in the simulation are described, including *i)* block rate, *ii*) reliability, *iii*) resource consumption, *iv*) time consumption, and (v) the CDF of the first block. The simulation results are presented and analyzed.

A. SIMULATION ENVIRONMENT

To evaluate the schemes described in Section IV, we implemented an event simulation in Java. To demonstrate the applicability of the algorithm for all circumstances, we employ the Waxman 2 model from the Georgia Tech Internetwork Topology Models (GT-ITM) [41] to randomly generate small and large network instances as substrate networks. The small substrate network includes 20 nodes and the large substrate network contains 100 nodes. The connectivity probability of both the small networks and large networks is 0.7. The diameter of the small network is 6, whereas the diameter of the large network is 30. Considering that the time consumption for deploying a SFC request in the small network is small, we chose to evaluate the approaches in this 20-node network using a machine with an Intel Core 2 CPU and 4 GB of RAM. For the 100-node network, the simulations were solved using an Intel i7 CPU with 9.8 GB of RAM. The computing capacity of every underlying node takes a random integer in the range [5, 10], and the bandwidth capacity of each node is distributed within the range [20, 50]. The bandwidth resources of the virtual links are distributed within the range [5, 20], and the computing capacity of the functional nodes ranges within [1, 2].

B. COMPARISONS WITH OTHER ALGORITHMS

During the simulation process, to compare and evaluate the performance of the three algorithms, we modified Compute followed by Network Load Balance (CNLB) [19] to the Link Mapping First (LMF) algorithm [27] without changing its core concept to be the compared algorithm in this paper. In the LMF approach, the virtual links are selected in descending order in terms of the requested bandwidth. The link with the largest requested bandwidth has priority for being mapped onto the physical links that have the largest amount of remaining computing resources. This approach is referenced in [27], where it is employed as a basic deployment algorithm.

C. SIMULATION RESULTS AND ANALYSIS

Fig. 4 shows the simulation results of the SFC block rate when deploying SFC requests for these four algorithms. We vary the number of functions of each SFC from 3 to 12 and randomly generate 10,000 SFC requests for each number of functions. The block rate denotes the proportion of the failed SFC deployment requests in all 10,000 SFC requests. As shown in the graph, for the three algorithms presented in this paper, especially ER_SC and ER_SC_ADJ, the performance of the block rate is better than that of the LMF algorithm. The comparisons shown in 2(a) and 2(b) indicate that the three algorithms have a distinct advantage in block rate as the network size increases. The ER_SC and ER_SC_ADJ algorithms introduce the load balancing theory, which aims to transfer the service flow to links with light loads and reduce the possibility of congestion caused by unbalanced traffic distribution. Based on the premise that the availability satisfies the R^U, nodes with light loads are more likely to be chosen as function nodes, which prevents the emergence of hot spots in the underlying network. This result reduces the blocking rate and guarantees a high deployment success rate for SFCs.

As the length of the SFC increases, the probability of SFC deployment failure also increases. In Fig. 4(a), as the lengths of the SFC requests increase, the block rate initially remains

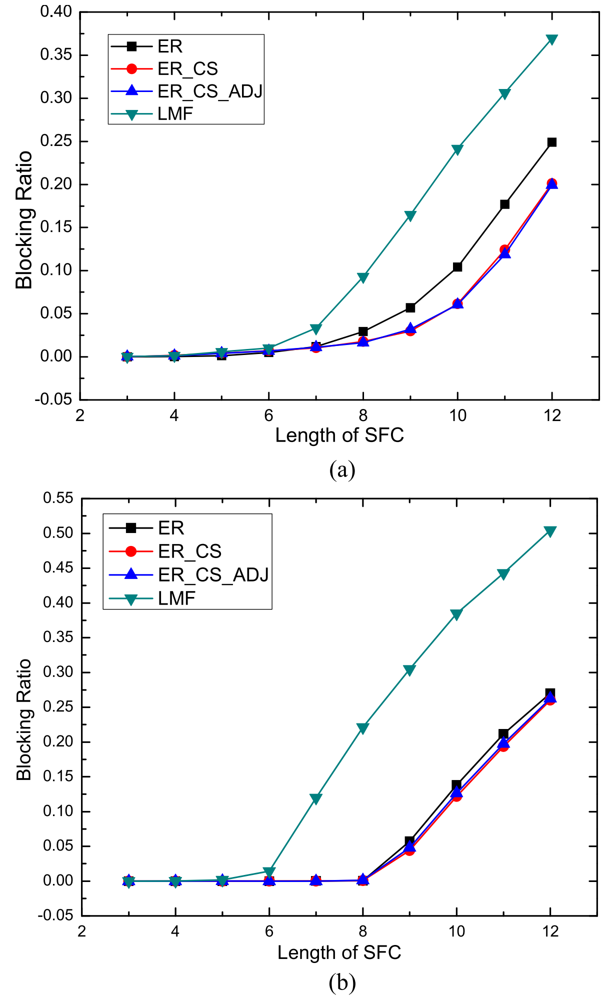

FIGURE 4. Block rates of SFCs in different topology. (a) Small simulation topology. (b) Big simulation topology

stable and then increases for different ranges starting at a length of approximately 6. As shown in Fig. 4(b), prior to a certain point almost all the tested algorithms can successfully deploy the SFC requests. The reliability of both the physical nodes and links is distributed within the range of [0, 1]. The larger the number of physical nodes is onto which an SFC request is deployed and the smaller the availability resulting from the SFC request, the greater the likelihood is that SFC deployment will fail to satisfy R^U and be successful. Thus, the block rates of SFC requests are related to not only the size of the underlying network but also to the length of an SFC request.

Fig. 5 shows the simulation results of the SFC reliability for SFC mapping algorithms, which were achieved by calculating the reliability product of all underlying network nodes and links traversed by the SFC deployment scheme. As shown in Fig. 5, the ER algorithm has the best reliability performance among the four algorithms. However, the total reliability performance of the ER_SC algorithm and the ER_SC_ADJ algorithm is slightly worse than the reliability

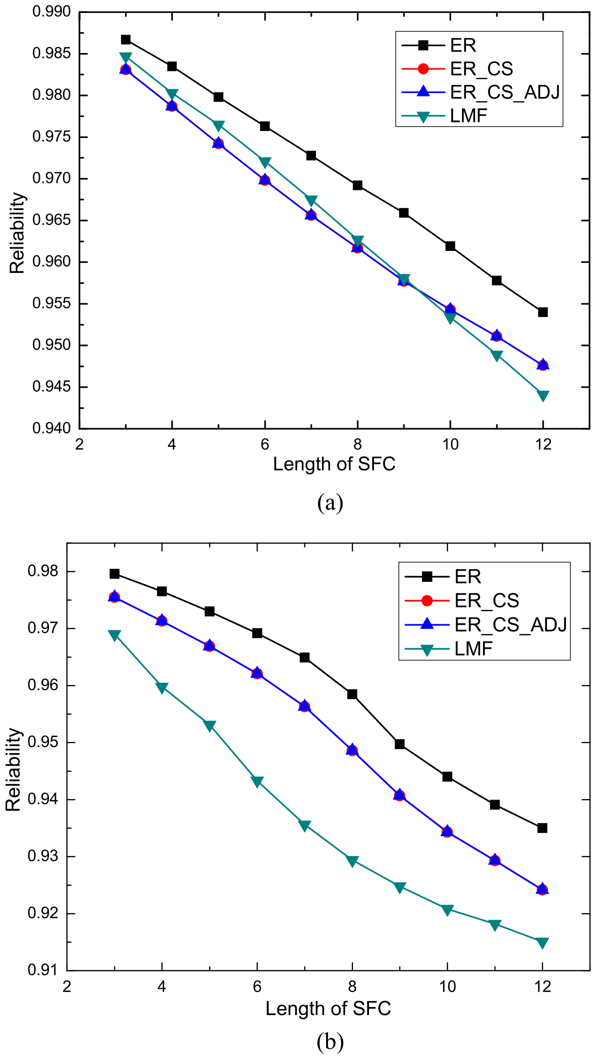

FIGURE 5. **Reliability of SFCs in different topologies. (a) Small simulation topology. (b) Big simulation topology.**

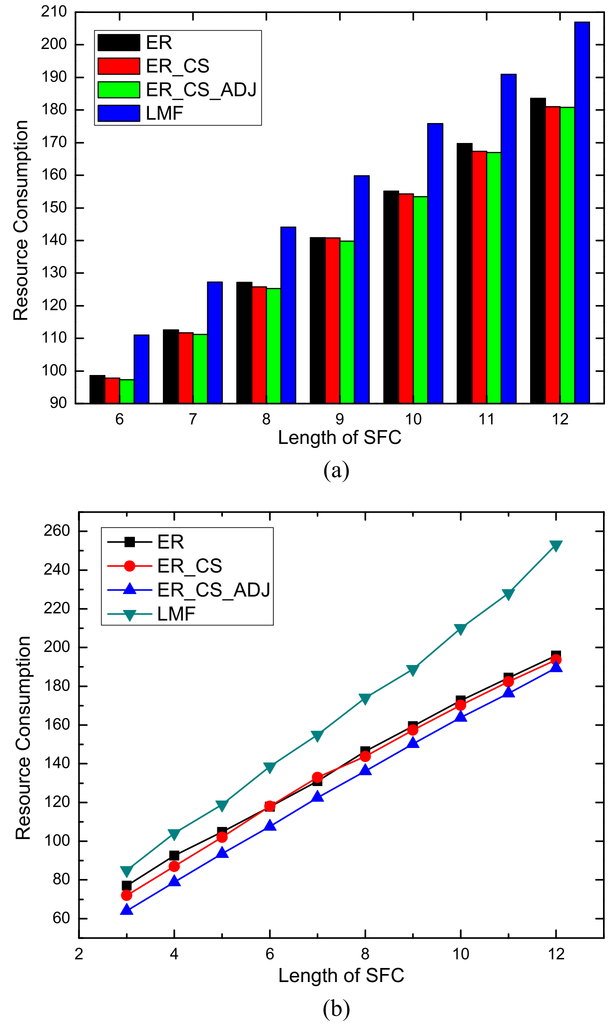

FIGURE 6. **Average resource consumption (i.e., computing resource and bandwidth resource) of SFCs in different topologies. (a) Small simulation topology. (b) Large simulation topology.**

performance of the LMF algorithm. When solving the SFC mapping problem, the ER algorithm deploys service chains with the primary objective of reliability maximization, which yields excellent performance in terms of reliability, while for the ER_SC algorithm and ER_SC_ADJ algorithm, we ensure only the basic R^U.

During the process of load balancing, to enable network functions to be deployed onto nodes with light loads, parts of the virtual links may be mapped to the underlying path using more hops, which decreases the reliability of SFC requests. The two algorithms are adjusted based on ER, which reduces the reliability. Although it may affect the user experience, the total reliability is capable of satisfying user requirements. Consequently, TSP costs will be reduced, which is the purpose of this study.

The results of the bandwidth overhead for SFC requests, shown in Fig. 6, reveal that the three algorithms proposed in this paper have an advantage over the LMF scheme in terms of bandwidth consumption, and that the ER_SC_ADJ algorithm performs the best. Under the same conditions, lower bandwidth overhead improves network resource utilization because the mapping positions of the VNFs for the ER_SC_ADJ algorithm are adjusted to lengthen the mapping paths of virtual links have low bandwidth-demands and shorten the mapping paths of virtual link that have high bandwidth demands while maintaining the total reliability achieved from the stationary ER_SC. In the small network, the difference among the bandwidth cost of three types of algorithms is small due to the relatively small size of the underlying network. Although adjustments were made based on the ER_SC algorithm, these adjustments are slight; thus, the gap is not distinct. However, the results are more distinct in the 100-node network.

Because the three algorithms proposed in this paper are similar in terms of resource consumption, especially in the small simulation topology, they are almost parallel. From a large number of tests, we determine that a line graph cannot

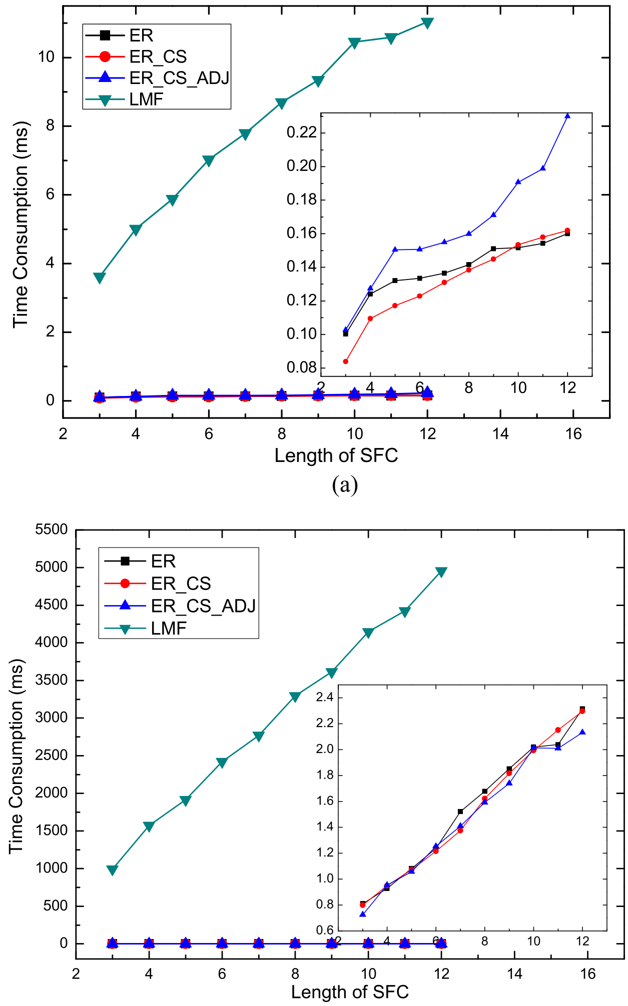

FIGURE 7. Average time consumed when SFCs are deployed. (a) Small simulation topology. (b) Large simulation topology.

adequately highlight the distinctions among the four algorithms. Thus, we separately employ a bar chart to show these data. If we start the X-axis at three, the histogram will be very dense and affect the sharpness of the data. Consequently, we employ a different method to display the data, as shown in Fig. 6(a).

The time consumption of each SFC mapping algorithm was evaluated by gradually increasing the number of service function chain requests, as shown in Fig. 7. The average time overhead of the SFC requests deployed by the three algorithms proposed in this paper is substantially lower than the average time overhead of the LMF algorithm. As the size of the underlying network increases, the difference in time consumption between the proposed schemes and the LMF algorithm increases from two orders of magnitude to a minimum of four orders of magnitude; thus, the performance advantages of the three types of algorithms become more distinct. This occurs because they are heuristic algorithms: all we need to do is evaluate a suitable path node from a certain node to the destination node. Conversely, in the

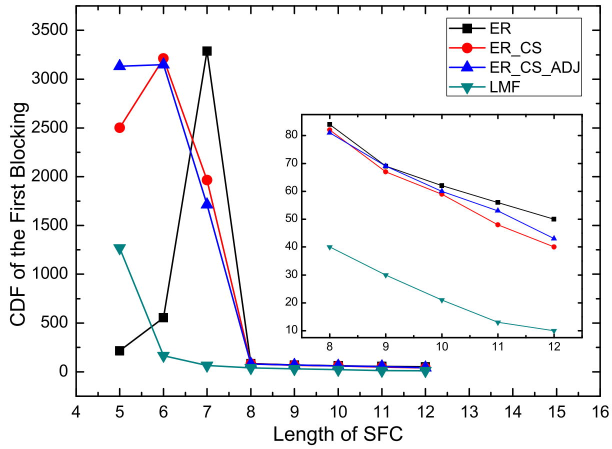

FIGURE 8. CDF of first block over the different lengths of SFCs.

LMF algorithm, each deployment of a virtual link requires traversing all the physical nodes to find node pairs that satisfy the requirements of node computing capacity and the shortest path between the node pairs given the bandwidth capacity conditions. Thus, the search efficiency of the LMF algorithm is relatively low, and the algorithm running time is relatively long.

We executed the algorithms for 10,000 trials for each SFC length and show the cumulative distribution function (CDF) of the first block, as discussed in [19]. Because computing this indicator on the big topology requires a vast amount of time—and considering that the small topology can also describe this problem—we chose only the small topology for this simulation. As shown in Fig. 8, the Y-axis denotes the number of SFCs that deployed successfully without any failure. The lower the number is, the worse the performance is. When the length of the SFC ranges from 5 to 8, the four algorithms appear to fluctuate. When the length is greater, the LMF algorithm shows a greater disadvantage. These results can be inferred from the prior indicators. With a poor success rate and higher resource consumption, LMF will fail at a faster rate.

VI. CONCLUSIONS AND FUTURE WORK

In this paper, we identified a problem: the high reliability requests of users reduce the CAPEX and OPEX of TSPs. Thus, we proposed ER to guarantee the basic reliability needs of users. However, considering the revenue of the TSPs, we discover that network imbalances will influence the request success rate and the resource utilization rate. Therefore, we proposed ER_SC, which is based on ER and considers the load balance factor. Although this algorithm achieved substantial progress, we discovered that the scheme used for ER_SC can be improved. Thus, we proposed ER_SC_ADJ. The simulation results indicate that ER_SC_ADJ achieves the objectives of this study. We demonstrated that our network algorithms can successfully work in a range of test environments and satisfy user demands. Our future work will include integrating our approach with data and network security to ensure that our services are robust and resilient [42]–[45].

We will integrate data fusion with MapReduce and advanced frameworks to accelerate network virtualization, data processing and analysis in big data networked environments [46]. We plan to develop business intelligence as a service to enable scientists and users to track changes in real time and understand all the interpretations within a few seconds [47], including the use of advanced big data network algorithms and services [48].

REFERENCES

[1] H. Moens and F. D. Turck, "Customizable function chains: Managing service chain variability in hybrid NFV networks," *IEEE Trans. Netw. Serv. Manage.*, vol. 13, no. 4, pp. 711–724, Apr. 2016.

[2] J. Li, Y. Zhang, X. Chen, and Y. Xiang, "Secure attribute-based data sharing for resource-limited users in cloud computing," *Comput. Secur.*, vol. 72, pp. 1–12, Jan. 2018.

[3] P. Li, J. Li, Z. Huang, C.-Z. Gao, W.-B. Chen, and K. Chen, "Privacy-preserving outsourced classification in cloud computing," *Cluster Comput.*, pp. 1–10, 2017.

[4] J. Li, J. Li, X. Chen, C. Jia, and W. Lou, "Identity-based encryption with outsourced revocation in cloud computing," *IEEE Trans. Comput.*, vol. 64, no. 2, pp. 425–437, Feb. 2015.

[5] P. Li *et al.*, "Multi-key privacy-preserving deep learning in cloud computing," *Future Generat. Comput. Syst.*, vol. 74, pp. 76–85, Sep. 2017.

[6] M. T. Beck and J. F. Botero, "Scalable and coordinated allocation of service function chains," *Comput. Commun.*, vol. 102, pp. 78–88, Apr. 2017.

[7] J. Wu, Z. Zhang, Y. Hong, and Y. Wen, "Cloud radio access network (C-RAN): A primer," *IEEE Netw.*, vol. 29, no. 1, pp. 35–41, Jan. 2015.

[8] J. Li, Z. Liu, X. Chen, X. Tan, F. Xhafa, and D. S. Wong, "L-EncDB: A lightweight framework for privacy-preserving data queries in cloud computing," *Knowl.-Based Syst.*, vol. 79, pp. 18–26, May 2015.

[9] Y. Zhang, X. Chen, J. Li, S. Duncan Wong, H. Li, and I. You, "Ensuring attribute privacy protection and fast decryption for outsourced data security in mobile cloud computing," *Inf. Sci.*, vol. 379, pp. 42–61, Feb. 2017.

[10] N. M. M. K. Chowdhury, M. R. Rahman, and R. Boutaba, "Virtual network embedding with coordinated node and link mapping," in *Proc. IEEE Int. Conf. Comput. Commun. (INFOCOM)*, Apr. 2009, pp. 783–791.

[11] J. Li, J. Li, D. Xie, and Z. Cai, "Secure auditing and deduplicating data in cloud," *IEEE Trans. Comput.*, vol. 65, no. 8, pp. 2386–2396, Aug. 2016.

[12] Z. Huang, S. Liu, X. Mao, K. Chen, and J. Li, "Insight of the protection for data security under selective opening attacks," *Inf. Sci.*, vols. 412–413, pp. 223–241, Oct. 2017.

[13] X. Gao *et al.*, "Virtual network mapping for multicast services with max-min fairness of reliability," *IEEE/OSA J. Opt. Commun. Netw.*, vol. 7, no. 9, pp. 942–951, Sep. 2015.

[14] J. Li, X. Chen, M. Li, J. Li, P. P. C. Lee, and W. Lou, "Secure deduplication with efficient and reliable convergent key management," *IEEE Trans. Parallel Distrib. Syst.*, vol. 25, no. 6, pp. 1615–1625, Jun. 2014.

[15] G. Sun, D. Liao, D. Zhao, Z. Xu, and H. Yu, "Live migration for multiple correlated virtual machines in cloud-based data centers," *IEEE Trans. Serv. Comput.*, to be published, doi: 10.1109/TSC.2015.2477825.

[16] J. Li, Y. Li, X. Chen, P. Lee, and W. Lou, "A hybrid cloud approach for secure authorized deduplication," *IEEE Trans. Parallel Distrib. Syst.*, vol. 26, no. 5, pp. 1206–1216, May 2015.

[17] G. Sun, D. Liao, V. Anand, D. Zhao, and H. Yu, "A new technique for efficient live migration of multiple virtual machines," *Future Generat. Comput. Syst.*, vol. 55, pp. 74–86, Feb. 2016.

[18] G. Sun, V. Anand, D. Liao, C. Lu, X. Zhang, and N.-H. Bao, "Power-efficient provisioning for online virtual network requests in cloud-based datacenters," *IEEE Syst. J.*, vol. 9, no. 2, pp. 427–441, Feb. 2015.

[19] J. Li *et al.*, "Secure distributed deduplication systems with improved reliability," *IEEE Trans. Comput.*, vol. 64, no. 12, pp. 3569–3579, Dec. 2015.

[20] ETSI Industry Specification Group (ISG) NFV. (Dec. 2014). *ETSI GS NFV-MAN 001: Network Functions Virtualization (NFV): Management and Orchestration*. [Online]. Available: http://www.etsi.org/deliver/etsi_gs/NFV-MAN/001_099/001/01.01.01_60/gs_NFV-MAN001v010101p.pdf

[21] J. Li, X. Huang, J. Li, X. Chen, and Y. Xiang, "Securely outsourcing attribute-based encryption with checkability," *IEEE Trans. Parallel Distrib. Syst.*, vol. 25, no. 8, pp. 2201–2210, Aug. 2014.

[22] X. Chen, J. Li, J. Weng, J. Ma, and W. Lou, "Verifiable computation over large database with incremental updates," *IEEE Trans. Comput.*, vol. 65, no. 10, pp. 3184–3195, Oct. 2016.

[23] W. Lin, Z. Wu, L. Lin, A. Wen, and J. Li, "An ensemble random forest algorithm for insurance big data analysis," *IEEE Access*, vol. 5, pp. 16568–16575, 2017.

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

[25] J. Li, L. Sun, Q. Yan, Z. Li, W. Srisa-an, and H. Ye, "Significant permission identification for machine learning based android malware detection," *IEEE Trans. Ind. Informat.*, to be published, doi: 10.1109/TII.2017.2789219.

[26] Y. Li, G. Wang, W. Tan, L. Nie, and Q. Wang, "Distance metric optimization driven convolutional neural network for age invariant face recognition," *Pattern Recognit.*, vol. 75, pp. 51–62, Mar. 2018.

[27] Z. Ye *et al.*, "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, Mar. 2016.

[28] L. Fan, X. Lei, N. Yang, T. Q. Duong, and G. K. Karagiannidis, "Secure multiple amplify-and-forward relaying with cochannel interference," *IEEE J. Sel. Topics Signal Process.*, vol. 10, no. 8, pp. 1494–1505, Dec. 2016.

[29] J. Li, X. Chen, F. Xhafa, and L. Barolli, "Secure deduplication storage systems supporting keyword search," *J. Comput. Syst. Sci.*, vol. 81, no. 8, pp. 1532–1541, 2015.

[30] W. Fang, M. Zeng, X. Liu, W. Lu, and Z. Zhu, "Joint spectrum and IT resource allocation for efficient vNF service chaining in inter-datacenter elastic optical networks," *IEEE Commun. Lett.*, vol. 20, no. 8, pp. 1539–1542, Aug. 2016.

[31] M. C. Luizelli *et al.*, "A fix-and-optimize approach for efficient and large scale virtual network function placement and chaining," *Comput. Commun.*, vol. 102, pp. 67–77, Apr. 2017.

[32] Z. Liu, Y. Huang, J. Li, X. Cheng, and C. Shen, "DivORAM: Towards a practical oblivious RAM with variable block size," *Inf. Sci.*, vol. 447, pp. 1–11, Jun. 2018.

[33] D. B. Oljira, A. Brunstrom, J. Taheri, and K.-J. Grinnemo, "Analysis of network latency in virtualized environments," in *Proc. IEEE Global Commun. Conf. (GLOBECOM)*, Dec. 2016, pp. 1–6.

[34] Y.-G. Wang, G. Zhu, and Y.-Q. Shi, "Transportation spherical watermarking," *IEEE Trans. Image Process.*, vol. 27, no. 4, pp. 2063–2077, Apr. 2018.

[35] A. F. R. Trajano and M. P. Fernandez, "Two-phase load balancing of in-memory key-value storages using network functions virtualization (NFV)," *J. Netw. Comput. Appl.*, vol. 69, pp. 1–13, Jul. 2016.

[36] Q. Lin, H. Yan, Z. Huang, W. Chen, J. Shen, and Y. Tang, "An ID-based linearly homomorphic signature scheme and its application in blockchain," *IEEE Access*, to be published, doi: 10.1109/ACCESS.2018.2809426.

[37] M. Mechtri *et al.*, "NFV orchestration framework addressing SFC challenges," *IEEE Commun. Mag.*, vol. 55, no. 6, pp. 16–23, Jun. 2017.

[38] H. Kim *et al.*, "Service platform and monitoring architecture for network function virtualization (NFV)," *Cluster Comput.*, vol. 19, pp. 1–7, Jan. 2016.

[39] C. Wang, O. Spatscheck, V. Gopalakrishnan, Y. Xu, and D. Applegate, "Toward high-performance and scalable network functions virtualization," *IEEE Internet Comput.*, vol. 20, no. 6, pp. 10–20, Dec. 2016.

[40] D. E. Knuth, "A generalization of Dijkstra's algorithm," *Inf. Process. Lett.*, vol. 6, no. 1, pp. 1–5, 1977.

[41] L. Kenneth Calvert and E. Zegura, *GT-ITM: Georgia Tech Internetwork Topology Models (Software)*. [Online]. Available: http://www.cc.gatech.edu/fac/Ellen.Zegura/gt-itm/gt-itm.tar.gz

[42] A. S. Aljawarneh, A. Alawneh, and R. Jaradat, "Cloud security engineering: Early stages of SDLC," *Future Generat. Comput. Syst.*, vol. 74, pp. 385–392, Sep. 2017.

[43] S. Aljawarneh, M. B. Yassein, and W. A. Talafha, "A resource-efficient encryption algorithm for multimedia big data," *Multimedia Tools Appl.*, vol. 76, no. 21, pp. 22703–22724, 2017.

[44] S. A. Aljawarneh, R. A. Moftah, and A. M. Maatuk, "Investigations of automatic methods for detecting the polymorphic worms signatures," *Future Generat. Comput. Syst.*, vol. 60, pp. 67–77, Jul. 2016.

[45] V. Chang and M. Ramachandran, "Towards achieving data security with the cloud computing adoption framework," *IEEE Trans. Services Comput.*, vol. 9, no. 1, pp. 138–151, Jan. 2016.

[46] G. Sun *et al.*, "The cost-efficient deployment of replica servers in virtual content distribution networks for data fusion," *Inf. Sci.*, vol. 432, pp. 495–515, Mar. 2018.
[47] V. Chang, "The business intelligence as a service in the cloud," *Future Generat. Comput. Syst.*, vol. 37, pp. 512–534, Jul. 2014.
[48] V. Chang and G. Wills, "A model to compare cloud and non-cloud storage of big data," *Future Generat. Comput. Syst.*, vol. 57, pp. 56–76, Apr. 2016.

JIAN SUN is currently a Senior Engineer with the University of Electronic Science and Technology of China. His research interests are in the area of next generation network and computer communication networks and protocols.

GUANGYANG ZHU is currently pursuing the master's degree in communication and information system with the University of Electronic Science and Technology of China. His research interests include network virtualization and security.

GANG SUN (M'13) is currently an Associate Professor of computer science with the University of Electronic Science and Technology of China. He has co-authored 80 technical publications including paper in refereed journals and conferences, invited papers and presentations, and book chapters. His research interests include network virtualization, cloud computing, high performance computing, parallel and distributed systems, ubiquitous/pervasive computing, and intelligence and cyber security. He was a Visiting Fellow with The Australian National University from 2015 to 2016. He is a member of the IEEE Computer Society. He has also edited special issues at top journals, such as *Future Generation Computer Systems* and *Multimedia Tool and Applications*. He has served as a Reviewers of the IEEE TRANSACTIONS ON INDUSTRIAL INFORMATICS, the IEEE COMMUNICATIONS LETTERS, *Information Fusion*, *Future Generation Computer Systems*, the *Journal of Network and Computer Applications*, *The Journal of Supercomputing*, the *Journal of Parallel and Distributed Computing*, *KSII Transactions on Internet and Information Systems*, *Computers and Electrical Engineering*, and the *Chinese Journal of Electronics*.

DAN LIAO is currently a Professor with the University of Electronic Science and Technology of China. His research interests are in the area of next generation network and wired and wireless computer communication networks and protocols.

YAO LI is currently pursuing the master's degree in communication and information system with the University of Electronic Science and Technology of China. His research interests include network virtualization and security.

ARUN KUMAR SANGAIAH received the Ph.D. degree in computer science and engineering from Vellore Institute of Technology, Vellore, India. He is currently an Associate Professor with the Vellore Institute of Technology. He has authored over 150 publications in different journals and conference of national and international repute. His current research work includes global software development, wireless ad hoc, sensor networks, and machine learning.

MUTHU RAMACHANDRAN is currently a Principal Lecturer with Leeds Beckett University. He has extensive research coupled with teaching experiences on software and systems engineering methods and lifecycle, software development, Agile software engineering, project management, process improvement, internet technology, mobile, networks, SOA, IT systems, cloud computing, and distributed computing. He also co-edited and wrote several books.

VICTOR CHANG is currently an Associate Professor with Xi'an Jiaotong-Liverpool University. He is also a leading expert on Big Data/Cloud/security, a Visiting Scholar/Ph.D. Examiner with several universities, an Editor-in-Chief of IJOCI and OJBD journals, an Editor of FGCS, a Founding Chair of two international workshops, and founding Conference Chair of IoTBD 2016 and COMPLEXIS 2016.