Service Recovery in NFV-Enabled Networks: Algorithm Design and Analysis

Dung H. P. Nguyen, Chih-Chieh Lin, Tu N. Nguyen, *Senior Member, IEEE*, Shao-I Chu, *Member, IEEE*, Bing-Hong Liu

***Abstract*—Network function virtualization (NFV), a novel network architecture, promises to offer a lot of convenience in network design, deployment, and management. This paradigm, although flexible, suffers from many risks engendering interruption of services, such as node and link failures. Thus, resiliency is one of the requirements in NFV-enabled network design for recovering network services once occurring failures. Therefore, in addition to a primary chain of virtual network functions (VNFs) for a service, one typically allocates the corresponding backup VNFs to satisfy the resiliency requirement. Nevertheless, this approach consumes network resources that can be inherently employed to deploy more services. Moreover, one can hardly recover all interrupted services due to the limitation of network backup resources. In this context, the importance of the services is one of the factors employed to judge the recovery priority. In this paper, we first assign each service a weight expressing its importance, then seek to retrieve interrupted services such that the total weight of the recovered services is maximum. Hence, we also call this issue the VNF restoration for recovering weighted services (VRRWS) problem. We next demonstrate the difficulty of the VRRWS problem is NP-hard and propose an effective technique, termed online recovery algorithm (ORA), to address the problem without necessitating the backup resources. Eventually, we conduct extensive simulations to evaluate the performance of the proposed algorithm as well as the factors affecting the recovery. The experiment shows that the available VNFs should be migrated to appropriate nodes during the recovery process to achieve better results.**

***Index Terms*—network function virtualization (NFV), node resource, recovery, virtual network function (VNF), weighted service.**

I. Introduction

The introduction of network functions virtualization (NFV), an advanced network architecture, has received widespread advocation from network operators. With this technique, they can significantly reduce many sorts of costs, such as capital expenditures (CAPEX) and operating expenditures (OPEX) [1], [2]. In addition, by means of replacing network functions running on proprietary hardware, such as firewalls, instruction detection systems, load balancers, and spam protection, with virtual network functions (VNFs) implemented by software running on general-purpose hardware, such as industry standard high volume servers, switches, and storage, the NFV facilitates network deployment, upgrading, and maintenance because installing a software apparently is simpler than deploying dedicated hardware. This character of NFV also contributes to shortening the time to market of new services. By consolidating hardware for processing network functions, NFV can assist with saving space for placing network appliances and power consumption [3], [4]. These benefits are ever meaningful in the context that the requirement for network services is ever increasing along with the appearance of 5G, the next mobile network generation, and the proliferation of mobile devices.

Despite presenting multiple advantages, the NFV has to cope with many risks that can significantly affect the network performance, one of which is the failure of network nodes [5], [6]. There are many reasons inducing this problem. Firstly, the servers employed to install and perform VNFs may suffer interruption due to several reasons, such as hardware errors and overload. Secondly, software implementing network functions may contain bugs leading to unexpected behavior beyond the VNF specification [7]. As a result, in the NFV environment, they are also called hardware and software failures, respectively. Both hardware and software failures are reasons causing the failure of a VNF, which will lead to the failure of the service requiring that VNF because a network service is a sequence of multiple VNFs [8]–[10]. The situation will be worse if a hardware failure occurs at a node hosting many VNFs of different services. In such a case, many services will be unavailable due to the failure of only one node [11].

Because hardware failure is typically more serious than software failure, we consider node failure in this paper as hardware failure. In practice, the probability of failures is very low but not equal to zero, which means failures may happen at any time, although rare. Hence, we need to preserve solutions to guarantee that the services will not be interrupted if there are some failures arising in the substrate network. Because continuity in the delivery of services is crucial, especially for critical applications, the solutions to recover the services from failures are significant. Therefore, resilience is one of the strict requirements of an NFV-enabled network [7].

The intuitive technique for recovering services from node failure to guarantee the resilience of networks is to deploy backup VNF instances for primary instances. If the failure happens at a node, the primary VNF instances at the failed

We thank the anonymous reviewers for their suggestions and feedback. This work was supported in part by the Ministry of Science and Technology, Taiwan, under Grant MOST 111-2221-E-992-079, and in part by U.S. NSF grants CNS-2103405 and AMPS-2229073.

Dung H. P. Nguyen is with the Faculty of Engineering and Technology, Pham Van Dong University, Quang Ngai 570000, Vietnam (e-mail: nphdung@pdu.edu.vn).

Chih-Chieh Lin, Shao-I Chu, and Bing-Hong Liu are with the Department of Electronic Engineering, National Kaohsiung University of Science and Technology, Kaohsiung 80778, Taiwan (e-mail: F108152156@nkust.edu.tw, erwinchu@nkust.edu.tw, and bhliu@nkust.edu.tw).

Tu N. Nguyen is with the Department of Computer Science, Kennesaw State University, Marietta, GA 30060, USA (e-mail: tu.nguyen@kennesaw.edu).

Corresponding author: Bing-Hong Liu

node will be replaced by the corresponding backup VNF instances [12]–[16]. This solution is efficient for large-scale physical networks with substantial redundant resources after deploying services. Nevertheless, this condition is difficult to hold because the number of network services is ever-increasing while the network resources are limited. Installing backup VNF instances for too many services will consume a lot of network resources, which results in confining the deployment of new services. Furthermore, we cannot ensure that nodes hosting the backup VNF instances never fail. Services are still interrupted in the circumstance that the primary and backup VNF instances fail simultaneously due to the failure of the corresponding nodes [17]. In another approach, one can predict the failures before they actually occur, then makes appropriate decisions for both primary and backup VNF instances [18]. However, the prediction inherently is not always accurate and timely. Therefore, this method is also efficient in several particular conditions only.

Quality of Service (QoS) requirement is also a significant factor for a network shared by multiple services. The QoS indicates some criteria the network operators need to satisfy for a service, including service recovery priority levels once failure arises. The customers who need to deploy their services over NFV-enabled networks and the network operators will consolidate the QoS requirement over the Service Level Agreement (SLA) [7], [19]. Different services require different QoS, in which the services with higher QoS, such as emergency telecommunication services, network control services, and real-time services, are typically more critical than ones with lower QoS, such as e-mail and web surfing. In terms of the resilience requirement, the more critical the services, the higher the recovery priority. Network operators apparently need to guarantee the QoS for different services according to the SLA. Nonetheless, because network resources are limited and shared between multiple services, it is challenging to recover all failed services while still ensuring the operation of non-failure services. In this paper, we employ weights to present the critical levels of services and aim to recover failed services such that the total weight of the recovered services is maximum.

Constructing NFV-enabled networks with resilience ability is an issue attracting the attention of researchers. However, the aspect of the service importance has not been investigated thoroughly. Furthermore, the method of deploying backup VNF instances has been shown to consume a particular amount of network resources and still be able to fail in some situations. In this paper, we propose a novel technique for recovering interrupted services without needing the backup VNF instances and take the importance of services into consideration concurrently. The following are the significant contributions of the paper.

- We propose the VNF restoration for recovering weighted services (VRRWS) problem for the first time. The urgency of services is one of the crucial factors in the service level agreement (SLA) between customers and network operators; it needs to be considered thoroughly during both the deployment and recovery of services, particularly when network resources are limited. In this paper, we take the importance into consideration to recover interrupted services. In addition, We also demonstrate that the VRRWS problem is NP-hard.
- Different from prevalent methods, we propose an online approach to recover interrupted services engendered by hardware failure at network nodes without necessitating backup resources. This approach can assist to bridge the challenges of resource limitation and backup node failure. We first suggest the method for recovering services with a single failed VNF. On that basis, we then propose the approach to recover services with multiple failed VNFs.
- We conduct extensive simulations to evaluate the performance of the proposed algorithm and the factors influencing the recovery. According to the experiments, migrating available VNFs hosted by non-failure nodes is necessary to accomplish a better result of the recovery.

The rest of the paper is organized as follows. Section II presents related works. Section III illustrates the system model, formally states, and demonstrates the hardness of the VRRWS problem. The algorithm is proposed in Section IV and analyzed in Section V. Section VI displays the experiment results. Eventually, Section VII is the conclusion.

II. Related Work

NFV is an advanced technique yielding a lot of convenience in deploying, operating, managing, and maintaining network services. Therefore, since its emergence, this technique has attracted the attention of many researchers. Due to the flexibility in installing VNFs, one out of the issues they are first interested in is optimizing the deployment of services over the physical network from many perspectives. Study [20] considers the aspect of cost and network utilization. Specifically, the authors propose an ILP model to minimize operating expenditures and maximize network utilization. The authors in [21] investigate the VNF placement in the joint problem of resource assignment and traffic routing. They first introduce a queuing-based model matching all the characters of 5G networks, then propose a heuristic algorithm, called MaxZ, to solve the problem. In addition, studies [22] and [23] present remedies to minimize bandwidth consumed by services. Study [24] seeks to minimize the computational resource and the risk of bottlenecks on network links. Work [25] attempts to minimize resource utilization while satisfying the quality of services.

Furthermore, due to the diversity of services running over the network, the latency requirement is also a significant issue that needs to be satisfied. Work [26] investigates the issue of network service deployment towards minimizing resource consumption while guaranteeing the latency of services. Accordingly, the authors suggest a two-stage solution to tackle the problem. In the first step, they determine deployment paths using a depth-first search-based algorithm. Afterward, they seek to reuse as many VNFs as possible for assigning VNFs by applying a greedy-based algorithm. Study [27] attempts to tackle the placement and routing of service function chains while ensuring the deadline of services. The authors propose four heuristic algorithms to solve the problem based on a

virtual layer graph constructed from the physical network and the constituent of services.

Network services always suffer from interruption with a specific probability due to many reasons, such as software errors, hardware malfunction, and link failure. Therefore, sustaining the continuity of services is one of the crucial issues in NFV-enabled networks [17], [28]. Accordingly, many approaches were proposed to protect network services from interruption. The prevalent technique is to leverage redundant resources of the network to place the backup VNF instances. In [29], by means of sharing backup instances, the authors seek to minimize resource consumption while guaranteeing a particular availability of services. Study [30] proposes survivable virtual networks for single facility node failure in which the backup paths are shared on physical links to reduce the backup resources.

Moreover, many plans are proposed for backup. Study [17] takes both backup computing and transmission resources into consideration. Accordingly, the authors seek to diminish backup computing capacity once multiple random facility node failures arise by means of sharing backup transmission resources among various services. With this approach, they can guarantee a reasonable probability of unsuccessful protection owing to inadequate backup computing capacity. In [10], the authors propose a backup strategy on top of redundant VNFs in which they leverage the diversity of the redundant VNFs to ensure reliability while satisfying the latency requirement of services. Study [18] employs machine learning algorithms to predict failures that may happen in the network; thereby, one can proactively deploy backup VNFs to guarantee the continuity of services when the failures actually occur. This approach is efficient in some situations, especially when the prediction is correct.

In addition, study [31] considers both software and hardware reliability. The authors believe that in addition to the backup phase, the construction phase also affects the reliability of services. Therefore, they first build the service function graph (SFG) that aggregates all the services based on software reliability. Afterward, they map the SFG to the physical network and evaluate the service reliability again, taking the hardware reliability into account. Finally, backup instances will be deployed for services that do not satisfy the reliability requirements. Hardware and software failure models are also concretely analyzed in [15]. The authors then suggest approaches to minimize the whole latency of all service flows. Coupled with the fact that allocating backup resources consumes a specific amount of network resources, the efficiency of resource utilization will be reduced significantly when a large amount of resources are reserved for the backup.

Although the VNF migration incurs a particular cost and delay, this is inevitable due to some issues, such as server maintenance [32], network device failure [33], energy saving [34], and traffic management [35]. In [32], the authors consider the trade-off between the latency during the maintenance of servers and the migration cost to decide if a VNF installed on the server should be migrated to another active server. In [33], the authors seek to retrieve multicast applications from disruption engendered by multiple node/link failures in which the links cannot be recovered, and hence, the failed VNFs must be migrated to other nodes while guaranteeing the end-to-end delays of the multicast trees. Coupled with the fact that the fewer servers are active, the less energy is consumed, the authors in [34] attempt to consolidate VNF instances in as few servers as possible. They propose a migration policy for the VNF instances in which the trade-off between the energy saving attained from the consolidation and the alleviation of revenue caused by the QoS degradation during the VNF instance migration is considered.

In short, the above studies investigate the service deployment and recovery problems in NFV-enabled networks under many different aspects. Nevertheless, this paper is the first work that takes the importance of services, a crucial term in the service level agreement (SLA) between customers and network operators, into consideration to the best of our knowledge.

III. Preliminaries

In this section, we first describe the system model we are working on, including the physical and virtual networks. We then formally present the problem and show that its difficulty is NP-hard.

A. System Model

In this study, we describe the physical network, the location used to deploy network services, by an undirected graph $G = (N, L)$, where N and L are the sets of nodes and links, respectively. Each link in L is a physical line connecting two nodes in N. Each node in the physical network can be a type of general-purpose hardware on which VNFs can be installed, such as high-volume servers, storage, and switches. Each node also possesses a limited number of computational resources (such as central processing units) representing its capacity. The capacity of node n is denoted by its cap property, namely $n.cap$. At a specific time, several VNFs installed on node n will occupy a particular number of the node resources, and the residual resource of n is denoted by its rr property, namely $n.rr$. Finally, the set of VNFs hosted by node n is described by its V property, namely $n.V$.

For the virtual network, we describe the set of services deployed over the physical network by $S = \{s_1, s_2, ..., s_k\}$. Each service in set S is assigned a weight indicating its critical level according to the QoS requirement stipulated in the SLA agreed between customers and the network operator. Accordingly, the higher the weight, the more critical the service. The weight of service s_i is indicated by its w property, namely $s_i.w$. Moreover, though there are many types of service function chaining in practice, a service is assumed to be a sequence of multiple network functions implemented by VNFs [15], [26], [31], [36]. VNFs are launched in network nodes, and a service is deployed by letting its traffic pass through the nodes hosting its VNFs according to a particular sequence of the VNFs. Therefore, suppose that the j-th VNF of service s_i is $v_{i,j}$, we denote service s_i by an ordered set of VNFs, namely $s_i = \{v_{i,1}, v_{i,2}, ..., v_{i,m}\}$, where m is the number of VNFs required by service s_i (or the length of the service). We suppose that the latency of transmitting data on all physical

TABLE I
NOTATIONS SUMMARY

Notation	Description
G	The physical network, including the sets of nodes and links
N	The set of network nodes
L	The set of network links
S	The set of services deployed over the physical network successfully in the initial state
N_F	The set of failure nodes
P	The set of plans describing the strategy for recovering an interrupted service
C	The set of candidate nodes
H	The table indicating the shortest paths in terms of number of hops between two nodes of the network
$P.V$	The V property of strategy P indicating the set of VNFs that were already considered by P
$n.V$	The V property of node n indicating the set of VNFs hosted by node n
$n.cap$	The cap property of node n indicating the maximum number computational resources that node n possesses
$n.rr$	The rr property of node n indicating the residual resources of node n
$v_{i,j}$	The j-th VNF of service i
$v.res$	The res property of VNF v indicating the resources required by VNF v
$v.n$	The n property of VNF v indicating the node hosting VNF v
$\eta(v_{i,j}, v_{i,j+1})$	The constraint on the number of hops between two successive VNFs $v_{i,j}$ and $v_{i,j+1}$
$s.w$	The w property of service s indicating the weight of service s
p_m^v	The plan to install VNF v on node m
μ	The maximum number of available VNFs that must be migrated to recover an interrupted service

links is identical, and hence, we can express the latency by the number of hops. In this study, as stated in Definition 1, we restrict the number of hops between successive VNFs of a service to a hop constraint for guaranteeing the latency of the service, and use $\eta(v_{i,j}, v_{i,j+1})$ to denote the hop constraint of two adjacent VNFs $v_{i,j}$, $v_{i,j+1}$ of a service function chain. Each VNF installed on a node will consume a number of the node resources indicated by the res property of the VNF, namely $v_{i,j}.res$ for VNF $v_{i,j}$. Furthermore, each VNF also possesses the n property, namely $v_{i,j}.n$ for VNF $v_{i,j}$, which indicates the node hosting the VNF. We summarize crucial notations utilized in this paper in Table I for convenience.

Definition 1. *Two adjacent VNFs $v_{i,j}$, $v_{i,j+1}$ of a service function chain is said to have a hop constraint, denoted by $\eta(v_{i,j}, v_{i,j+1})$, only if the number of hops between the two nodes hosting the VNFs in the substrate network does not exceed $\eta(v_{i,j}, v_{i,j+1})$.*

Finally, the status of the network is monitored by a controller located in the management and orchestration (MANO) layer. Therefore, the controller can detect the failure of physical nodes and make plans to recover interrupted services incurred by the node failure. In addition, the controller can also install VNFs on physical nodes, activate a VNF by assigning computational resources to the VNF instance, and deactivate a VNF by releasing computational resources from the VNF instance.

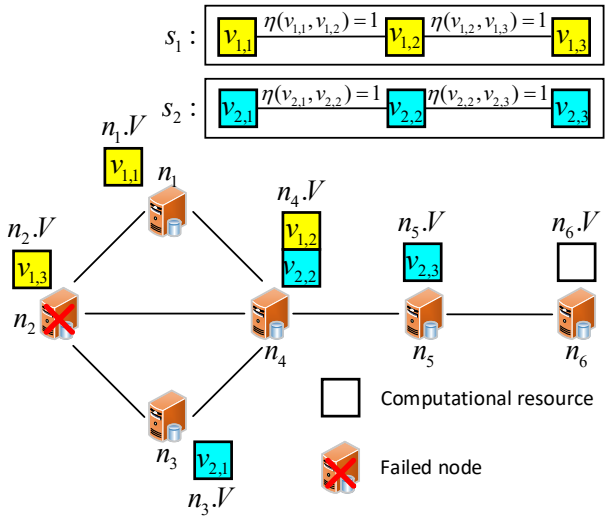


Fig. 1. An example of the system model.

Figure 1 illustrates an example of the system. The physical network comprises 6 nodes, described by set $N = \{n_1, n_2, n_3, n_4, n_5, n_6\}$, and 7 links. Each node is a general-purpose hardware with the ability to host VNFs, such as servers, switches, and storage. The square next to each node denotes its computational resource. Accordingly, all nodes of the physical network possess one unit of the computational resource, except node n_4 with two units; hence, $n_i.cap = 1$ for all $n_i \in N \backslash \{n_4\}$ and $n_4.cap = 2$. The set of services that have been deployed over the network is $S = \{s_1, s_2\}$, where each service is an ordered chain of three VNFs, expressed by $s_1 = \{v_{1,1}, v_{1,2}, v_{1,3}\}$ and $s_2 = \{v_{2,1}, v_{2,2}, v_{2,3}\}$. Each VNF consumes one unit of computational resource for operation, that is, $v_{i,j}.res = 1$ for all $v_{i,j} \in s_i$ and $s_i \in \{s_1, s_2\}$. The hop constraints between two VNFs are all one hop, which means $\eta(v_{i,j}, v_{i,j+1}) = 1$ for all $i, j \in \{1, 2\}$. In the original state, s_1 is deployed over nodes n_1, n_4, and n_2; and s_2 is deployed over nodes n_3, n_4, and n_5. Therefore, the resource of all nodes is occupied, except node n_6, which means $n_i.rr = 0$ for all $n_i \in N \backslash \{n_6\}$ and $n_6.rr = 1$. In addition, the set of VNFs hosted by each node is completely determined, that is, $n_1.V = \{v_{1,1}\}$, $n_2.V = \{v_{1,3}\}$, $n_3.V = \{v_{2,1}\}$, $n_4.V = \{v_{1,2}, v_{2,2}\}$, $n_5.V = \{v_{2,3}\}$, and $n_6.V = \emptyset$.

B. Problem Definition

We suppose that in the original status of the system, the services in set S are deployed successfully over the physical network. However, due to malfunctions of hardware, some network nodes are failed resulting in the interruption of some services. The set of failed nodes is expressed by $N_F \subseteq N$. Based on the elements of set N_F, we can determine the interrupted services. Because backup VNF instances are not deployed in this model, we carry out the recovery by deploying the interrupted services on top of the residual resources of nodes and the current status of the network. The recovery of a service is defined as follows.

Definition 2. *A service is recovered if its failed VNFs can be reinstalled on the non-failure nodes while guaranteeing the*

structure of the service and the availability of other services that are not affected by the failure of network nodes.

According to Definition 2, in addition to the recovery, we need to guarantee the requirements of the structure and the availability of services. Nevertheless, it is hard to satisfy these demands to recover all failed services due to the limitation of the residual resources; hence, we aim to maximize the total weights of the recovered services in this study. Therefore, we call this issue the VNF restoration for recovering weighted services (VRRWS) problem. The VRRWS problem is formally stated as follows.

INSTANCE: Given a physical network described by undirected graph $G = (N, L)$, a set of services deployed successfully on the network, termed S, and a set of failed nodes, termed N_F, where $N_F \subseteq N$.

TASK: Maximize $\sum_{s_i \in S_R} s_i.w$, where $S_R \subseteq S$ is the set of services that can be recovered from the interruption engendered by the failure of nodes in N_F.

Accordingly, we formulate the VRRWS problem as follows.

maximize: $\sum_{s_i \in S_R} s_i.w$

subject to:

$$\sum_{v \in n.V} v.res \leq n.cap \ \forall n \in N \tag{1}$$

$$dist\left(v_{i,j}, v_{i,j+1}\right) \leq \eta\left(v_{i,j}, v_{i,j+1}\right) \ \forall v_{i,j} \in s_i, \ \forall s_i \in (S \backslash S_F) \cup S_R \tag{2}$$

$$S_R \subseteq S_F \tag{3}$$

where $dist\left(v_{i,j}, v_{i,j+1}\right)$ denotes the distance, in terms of the number of hops, between the two physical nodes hosting $v_{i,j}$ and $v_{i,j+1}$; S_F is the set of failed services induced by the failure of nodes in N_F. Constraint (1) means that the total computational resources consumed by the VNFs installed at a node must not exceed the node capacity. Constraint (2) describes the hop constraint between two adjacent VNFs of the running services in set $S \backslash S_F$ and the recovered services in set S_R.

Taking the system in Fig. 1, for example, in which node n_2 is failed due to a malfunction, which indicates $N_F = \{n_2\}$. Because $n_2.V = \{v_{1,3}\}$ and $v_{1,3} \in s_1$, VNF $v_{1,3}$ is failed and only service s_1 is interrupted. Therefore, the task is to find an approach to migrate VNF $v_{1,3}$ and the other VNFs, if required, to the other nodes to recover service s_1 while guaranteeing the availability of service s_2. Figure 2 shows a solution to recover service s_1, where VNFs $v_{2,3}$, $v_{2,2}$, $v_{2,1}$, $v_{1,1}$, and $v_{1,3}$ are migrated to nodes n_6, n_5, n_4, n_3, and n_1, respectively.

After determining plans for recovering interrupted services, the VNFs that need to be migrated will be installed on the new locations (nodes) accordingly. The corresponding VNFs at the old locations will be deactivated by releasing computational resources from the VNF instances, and the new VNFs will be activated by assigning computational resources to the new VNF instances [37] concurrently. Eventually, the traffic of the services affected by the recovery will be orientated to traverse through the new locations. As presented above, the VNF migration is inevitable and apparently suffers from a particular latency. From the perspective of internet service providers, they expect to recover interrupted services as soon as possible, and hence, many researches have concentrated on optimizing the migration latency [32], [35], [38]. The detail of these techniques is beyond the scope of this study.

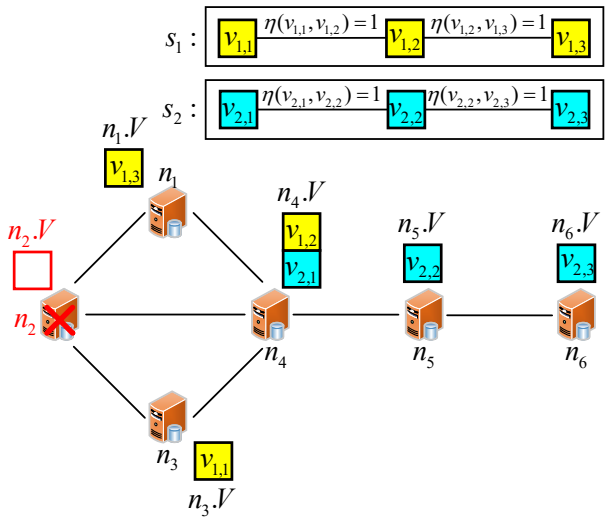


Fig. 2. A solution to recover service s_1.

C. Hardness of the VRRWS Problem

In this subsection, we first present a subproblem of the VRRWS problem, termed reduced VRRWS (RVRRWS) problem. Afterward, we demonstrate that the difficulty of the RVRRWS problem is NP-hard over Lemma 1. Finally, the hardness of the VRRWS problem is shown in Theorem 1. The RVRRWS problem is stated as follows.

INSTANCE: Given a physical network described by undirected graph $G = (N, L)$, where $N = \{n_1, n_2\}$ and $L = \{l\}$, a set of services deployed successfully over the network, termed S, where each service consists of only one VNF, and a set of failed nodes $N_F = \{n_2\}$.

TASK: Maximize $\sum_{s_i \in S_R} s_i.w$, where $S_R \subseteq S$ is the set of services that can be recovered from the interruption engendered by the failure of nodes in N_F.

The Knapsack problem [39] is employed to show the hardness of the RVRRWS problem and is presented as follows.

INSTANCE: Given a set of items described by $\mathcal{I}$ and a knapsack. Each item I_i in $\mathcal{I}$ owns a value and a weight expressed by its v and w properties, that is, $I_i.v$ and $I_i.w$, respectively. The knapsack possesses a specific capacity indicating the maximum total weight of items it can contain, termed W.

TASK: Maximize $\sum_{I_i \in \mathcal{R}} I_i.v$ subject to $\sum_{I_i \in \mathcal{R}} I_i.w \leq W$, where $\mathcal{R} \subseteq \mathcal{I}$

Lemma 1. *The RVRRWS problem is NP-hard.*

Proof. The knapsack in the Knapsack problem can be judged as node n_1 in the RVRRWS problem, in which the capacity W of the knapsack can be treated as the residual resources of node n_1, that is, $n_1.rr$. Suppose that S_F is the set of services that are interrupted due to the failure of node n_2. Each item in set $\mathcal{I}$ of the Knapsack problem can be treated as an interrupted

service in set S_F of the RVRRWS problem, where the value of the item can be treated as the weight of the service, and the weight of the item can be treated as the consumed resources of the VNF of the service. Obviously, the Knapsack problem is also the RVRRWS problem. According to [39], the Knapsack problem is NP-hard, hence is the hardness of the RVRRWS problem. This completes the proof. □

Theorem 1. *The VRRWS problem is NP-hard.*

Proof. Coupled with the fact that the RVRRWS is a sub-problem of the VRRWS, and the difficulty of the RVRRWS problem is NP-hard according to Lemma 1, the hardness of the VRRWS problem is NP-hard. This completes the proof. □

IV. Proposed Algorithm

In this section, we propose an online technique to recover the interrupted services due to the failure of several nodes in the physical network. We seek to restore a failed VNF by migrating it to a non-failure node with a proper residual resource while guaranteeing the structure of the recovered service as well as the availability of the non-failure services. Coupled with the fact that the failure may occur at disparate nodes in the network concurrently, there may be multiple failed VNFs incurring the interruption of the same service. Therefore, we first show the method for restoring a single VNF based on the available VNFs of the same service. On that basis, we suggest a procedure to retrieve an interrupted service by means of restoring all its failed VNFs. Eventually, we propose a technique, dubbed online recovery algorithm (ORA), to recover interrupted services induced by the failure of a set of network nodes.

The rationale behind the proposed method is to leverage the computation resource redundancy of some nodes in the network to relaunch failed VNFs incurred by the failure of nodes while guaranteeing the operation of available services running over the network. Accordingly, a failed VNF will be migrated to an available node. If the computation resource of the node is insufficient to launch the failed VNF, we need to migrate available VNFs being installed at the node to other available nodes, and such VNFs are treated as failed VNFs that need to be retrieved. Nevertheless, different from the original failed VNF, it is imperative that we ensure the successful recovery of these VNFs to sustain the availability of the running services. A failed VNF is recovered successfully if we can find a proper node with enough computation resources to install the last available VNF affected by the migration of the failed VNF. For an interrupted service with multiple failed VNFs, we will retrieve the service by recovering the failed VNFs one by one. Eventually, if there are multiple interrupted services in the network, the ORA considers restoring these services one by one, and services with higher weights are prioritized for recovery first. We next describe the proposed method in detail, including the illustrated example.

A. Restoration of a Single Failed VNF

In this subsection, we propose the RESTORE procedure to restore a single failed VNF that is $v_{i,j}$ in the instance. We first search for the candidate nodes to reinstall the failed VNF over the CANDIDATENODES procedure. Accordingly, the procedure returns a set of nodes (C) that satisfy the hop constraint between the node hosting the failed VNF and the nodes on which the preceding ($v_{i,x}$) and the next ($v_{i,y}$) available VNFs in the corresponding sequence of the service are being installed. In the CANDIDATENODES procedure, H is a table indicating the minimum distance, in terms of the number of hops, between two nodes in the physical network (G). Table H is implemented by applying the Floyd-Warshall algorithm [40]. In the case that the preceding and the next VNFs of the failed VNF are both available ($x = j - 1$ and $y = j+1$), the candidate nodes must fulfill both hop constraints between the failed VNF and the preceding and next VNFs. In the situations that the failed VNF is either the first or the last VNF of the sequence, or two or more successive VNFs are failed, only one out of the preceding and next VNFs is available (either $x = j - 1$ or $y = j + 1$), and hence, the candidate nodes need to satisfy the hop constraint between the failed VNF and the available VNF only. In the cases that the available preceding and the next VNFs are not adjacent to the failed VNF, we need to determine the candidate nodes temporarily.

```
1: procedure CANDIDATENODES(v_{i,j}, G, H, S)
2:     Let v_{i,x} and v_{i,y} be the preceding and the next available VNFs of v_{i,j} in the corresponding chain
3:     C ← ∅
4:     Let A be the set of nodes a such that H(v_{i,x}.n, a) ≤ η(v_{i,x}, v_{i,x+1})
5:     Let B be the set of nodes b such that H(v_{i,y}.n, b) ≤ η(v_{i,y-1}, v_{i,y})
6:     if x = j − 1 and y = j + 1 then
7:         C ← A ∩ B
8:     else if x = j − 1 then
9:         C ← A
10:    else if y = j + 1 then
11:        C ← B
12:    else
13:        if A ≠ ∅ then
14:            C ← A
15:        else
16:            C ← B
17:        end if
18:    end if
19:    return C
20: end procedure
```

Taking Fig. 1, for example, node n_2 is failed, inducing the failure of VNF $v_{1,3}$. Because VNF $v_{1,3}$ is the last VNF of service s_1, only the preceding VNF $v_{1,2}$ being installed on node n_4 is available. Because $\eta(v_{1,2}, v_{1,3}) = 1$, the number of hops between a candidate node for restoring $v_{1,3}$ and node n_4 must not exceed 1. Therefore, the set of the candidate nodes is $C = \{n_1, n_3, n_4, n_5\}$. Because node n_2 is failed, it is not counted as a candidate node.

After finding out the candidate nodes, we next determine a proper strategy to restore the failed VNF with the aim of

recovering the interrupted service. Because an available VNF placed on a non-failure node may be replaced by a failed VNF for the proposed technique, restoring the failed VNFs may impact the availability of other VNFs. Thus, we seek to find the best strategy to deploy the interrupted service again over the physical network by considering each candidate node in set C. In the RESTORE procedure, a plan to install VNF $v_{i,j}$ on node m is indicated by $p_m^{v_{i,j}}$, and the strategy for recovering a service from interruption is determined by a set of plans denoted by P. For each node m in set C, if the residual resource of the node is sufficient to launch the failed VNF ($m.rr \geq v_{i,j}.res$), plan $p_m^{v_{i,j}}$ will be appended to the current strategy, then the best strategy will be updated. Strategy X (P_X) is better than strategy Y (P_Y) if $|P_X| < |P_Y|$, where $|P_X|$ and $|P_Y|$ are the numbers of plans in strategies X and Y, respectively. If $|P_X| = |P_Y|$, we need to consider the first plans of P_X and P_Y, say $p_{n_X}^v$ and $p_{n_Y}^v$, and the node hosting the next VNF of v in the sequence, say q, to decide which strategy is better, that is, if $H(n_X, q) < H(n_Y, q)$, strategy X is better than strategy Y. Accordingly, if the current strategy (P) is better than the best strategy (P_{opt}), the best strategy will be replaced by the current one. Afterward, the residual resource of the current candidate node and the status of the physical network are also updated based on the current best strategy. If the residual resource of the candidate node is not adequate to launch the failed VNF, we can remove one VNF being installed on the node to release a particular amount of the node resource. If the total of the residual resource and the released resource satisfy the requirement of the failed VNF, we can allocate the node to the failed VNF and consider the removed VNF to be a failed VNF that also needs to be restored by a strategy. To this end, we call the RESTORE procedure again, taking the removed VNF as the input. That is the reason the RESTORE procedure is implemented according to a recursive manner. The recursion will end when no more VNF is removed from a candidate node. After finding out the proper node to reinstall the failed VNF, the corresponding plan will be added to the current strategy. Thereby, the number of plans in a strategy also indicates the number of VNFs that need to be installed or migrated from their original nodes to recover an interrupted service. Hence, in other words, strategy A is better than strategy B if strategy A migrates fewer available VNFs than strategy B.

In order to avoid endless loops engendered by continuously removing the same VNF in the recursion, if a VNF is already in $P.V$, which is the set of VNFs in strategy P, it will not be taken into consideration for removing again. In addition, in order to diminish the complexity of the RESTORE procedure, we confine the number of available VNFs that need to be migrated over value μ. While considering the plan of installing the failed VNF, a strategy is infeasible if the number of its plans exceeds $\mu + 1$.

Taking Fig. 1, for example, we consider each node in turn in the set of candidate nodes (C) determined from the previous step to explore the best strategy to recover service s_1 while guaranteeing the availability of s_2. The first node in set C is considered, which is n_1. However, n_1 is hosting $v_{1,1}$, and thus, its residual resource is insufficient to reinstall $v_{1,3}$. To be able to launch $v_{1,3}$ into n_1, $v_{1,1}$ must be pushed out of n_1. Then, node n_1 is allocated to reinstall VNF $v_{1,3}$, and $v_{1,1}$ becomes a failed VNF that needs to be restored. Therefore, the current strategy is $P = \{p_{n_1}^{v_{1,3}}\}$. We next seek to restore the current failed VNF, $v_{1,1}$. Likewise, the set of candidate nodes for reinstalling $v_{1,1}$ is $\{n_1, n_3, n_4, n_5\}$. The first node in the set, n_1, is considered for reinstalling $v_{1,1}$. Nonetheless, n_1 is hosting $v_{1,3}$ that belongs to the current strategy P, we ignore n_1 and deliberate the next node in the candidate node set, n_3. The VNF that is being hosted by n_3, that is $v_{2,1}$, will be pushed out to reinstall $v_{1,1}$. The current strategy becomes $P = \{p_{n_1}^{v_{1,3}}, p_{n_3}^{v_{1,1}}\}$, and $v_{2,1}$ becomes the next failed VNF that needs to be restored. Repeat this process until no more VNF is pushed out, we acquire the first strategy for recovering service s_1, that is, $P_1 = \{p_{n_1}^{v_{1,3}}, p_{n_3}^{v_{1,1}}, p_{n_4}^{v_{2,1}}, p_{n_5}^{v_{2,2}}, p_{n_6}^{v_{2,3}}\}$. Likewise, for candidate node n_3, we obtain the second strategy, $P_2 = \{p_{n_3}^{v_{1,3}}, p_{n_4}^{v_{2,1}}, p_{n_5}^{v_{2,2}}, p_{n_6}^{v_{2,3}}\}$. For candidate nodes n_4 and n_5, we cannot find any feasible strategy. The two feasible strategies are illustrated in Fig. 3. In the figure, a dashed square describes a VNF that is dropped from its original node to

procedure RESTORE($v_{i,j}, G, H, S, t, \mu, P, P_{opt}, G_{opt}$)
  if $t > \mu$ **then**
    return $\{P_{opt}, G_{opt}\}$
  end if
  $C \leftarrow$ CANDIDATENODES($v_{i,j}, G, H, S$)
  for each $m \in C$ **do**
    if $m.rr \geq v_{i,j}.res$ **then**
      Append $p_m^{v_{i,j}}$ to P
      if $P_{opt} = \emptyset$ or P is better than P_{opt} **then**
        $P_{opt} \leftarrow P$
        $G_{opt} \leftarrow G$
        Allocate node m for VNF $v_{i,j}$ and update $m.rr$ and $m.V$ in G_{opt}
      end if
      Remove $p_m^{v_{i,j}}$ from P
    else
      if $P_{opt} \neq \emptyset$ and $|P_{opt}| \leq t$ **then**
        continue
      else
        for each $v \in m.V$ **do**
          if $v \notin P.V$ and $m.rr + v.res \geq v_{i,j}.res$ **then**
            $G_{tmp} \leftarrow G$
            Allocate node m for VNF $v_{i,j}$, remove VNF v from m, and update $m.rr$ and $m.V$ in G_{tmp}
            Append $p_m^{v_{i,j}}$ to P
            $\{P_{opt}, G_{opt}\} \leftarrow$ RESTORE(v, G_{tmp}, H, S, $t+1$, μ, P, P_{opt}, G_{opt})
            Remove $p_m^{v_{i,j}}$ from P
          end if
        end for
      end if
    end if
  end for
  return $\{P_{opt}, G_{opt}\}$
end procedure

procedure RESTOREVNFS(V, G, H, S, μ)
  $G_{opt} \leftarrow G$
  while $V \neq \emptyset$ **do**
    Let v be the first VNF in V
    $P_{opt} \leftarrow \emptyset$; $P \leftarrow \emptyset$
    $\{P_{opt}, G_{opt}\} \leftarrow$ RESTORE(v, G_{opt}, H, S, 0, μ, P, P_{opt}, G_{opt})
    if $P_{opt} = \emptyset$ **then**
      return $\{false, \emptyset\}$
    end if
    Remove v from V
  end while
  return $\{true, G_{opt}\}$
end procedure

reinstall a failed VNF, and a dashed arrow describes a plan to restore a failed VNF. According to the procedure, between the two strategies, P_2 is selected as the final solution because it is better than P_1. In addition, if we confine the length of a strategy by setting the value of μ as 3, P_2 is the unique feasible solution.

B. Restoration of Multiple VNFs

The interruption of a service may be induced by the failure of multiple VNFs. By means of applying the RESTORE procedure iteratively, we can restore the failed VNFs one by one, thereby recovering the interrupted service. The algorithm is concretely described over the RESTOREVNFS procedure, where V is the set of the failed VNFs. For each iteration, the first VNF in V is considered to restore by the RESTORE procedure. If the strategy returned from the RESTORE procedure (P_{opt}) is not empty, it means that the failed VNF can be restored successfully. We then update the current status of the physical network, remove the first VNF in V, and repeat the process for the remaining VNFs in V. If P_{opt} is an empty set, the failed VNF cannot be restored, and hence, the corresponding service also cannot be recovered from the interruption.

C. Service Recovery Algorithm

In this subsection, we propose an algorithm to recover the interrupted services induced by the failure of a set of nodes, expressed by N_F, by applying the above procedures. The algorithm is presented concretely in Algorithm 1. Because there are constraints on the number of hops between two successive VNFs of the same service, we first implement a table indicating the shortest path, in terms of hops, between two nodes in the physical network employing Floyd-Warshall's algorithm [40]. This table assists with rapidly searching the shortest path between two nodes without using Dijkstra's algorithm repeatedly. From the set of failure nodes, we can determine the failed VNFs, thereby inferring the set of interrupted services, denoted by $S_F \subseteq S$. We seek to maximize the total weight of the recovered services by recovering the interrupted services one by one in the descending order of weights. If all the VNFs of a service are failed, we consider deploying the service again by the technique proposed by [41].

For another service with at least one available VNF, we form a corresponding set of failed VNFs with the order according to the service sequence. The recovery is then conducted by restoring the failed VNFs in the set one by one from the first to the last VNF. According to the procedure stated above, finding the candidate nodes for reinstalling a failed VNF requires determining precisely at least one adjacent VNF that is still available. Nevertheless, this requirement cannot be satisfied to restore the first failed VNF if more than one of the first consecutive VNFs in the original sequence of the service are failed. Hence, we divide the set of failed VNFs into two subsets to bridge this situation. The first set, termed V_{front}, involves the first k ($k \geq 1$) VNFs if the first k consecutive VNFs of the service are failed. The set of the remaining failed VNFs is denoted by V_{rear}. By means of applying this method, we can guarantee that the last VNF in V_{front} and the first VNF in V_{rear} are always adjacent to an available VNF. We then restore the VNFs in V_{front} by restoring the last one first. Based on the VNFs that were restored successfully, we restore the remaining VNFs. If all the VNFs in V_{front} are restored successfully, we next restore the VNFs in V_{rear}. It is worth noting that the RESTOREVNFS procedure restores each failed VNF in turn according to a specific order originating from the first VNF in the set of failed VNFs. In order to apply the RESTOREVNFS procedure for restoring VNFs in V_{front} from the last VNF, we reverse V_{front} to obtain set V^r_{front} with the first element being the last VNF in V_{front}. Afterward, the recovery is conducted on V^r_{front} and V_{rear}. If all the VNFs in V^r_{front} and V_{rear} are fixed, the service is successfully recovered.

We take the following example to clarify the algorithm. Suppose that the original chain of a service is $\{v_1, v_2, v_3, v_4, v_5, v_6\}$ in which v_1, v_2, v_3, v_5, and v_6 are failed; thus, sets V_{front}, V^r_{front}, and V_{rear} are $\{v_1, v_2, v_3\}$, $\{v_3, v_2, v_1\}$, and $\{v_5, v_6\}$, respectively. The RESTOREVNFS procedure is applied for V^r_{front} first, and according to the procedure, VNF v_3 is considered restoring first based on the unique available VNF of the service, v_4. If v_3 is fixed successfully, it will be available and become the basis for restoring v_2. The process will be repeated to restore the remaining failed VNFs in V^r_{front} and then V_{rear}.

V. ALGORITHM ANALYSIS

In this section, we first present the time complexity of CANDIDATENODES, RESTORE, and RESTOREVNFS procedures over Lemmas 2, 3, and 4, respectively. On that basis, the time complexity of the ORA is shown in Theorem 2. In addition, we employ κ, ζ, ℓ, and ρ to denote the number of physical network nodes, the maximum resources that a node possesses, the maximum length of services, and the number of services deployed over the physical network, respectively. Moreover, $log\kappa$ stands for $log_2\kappa$.

Lemma 2. *The time complexity of the* CANDIDATENODES *procedure is bounded in* $O(\kappa log\kappa)$.

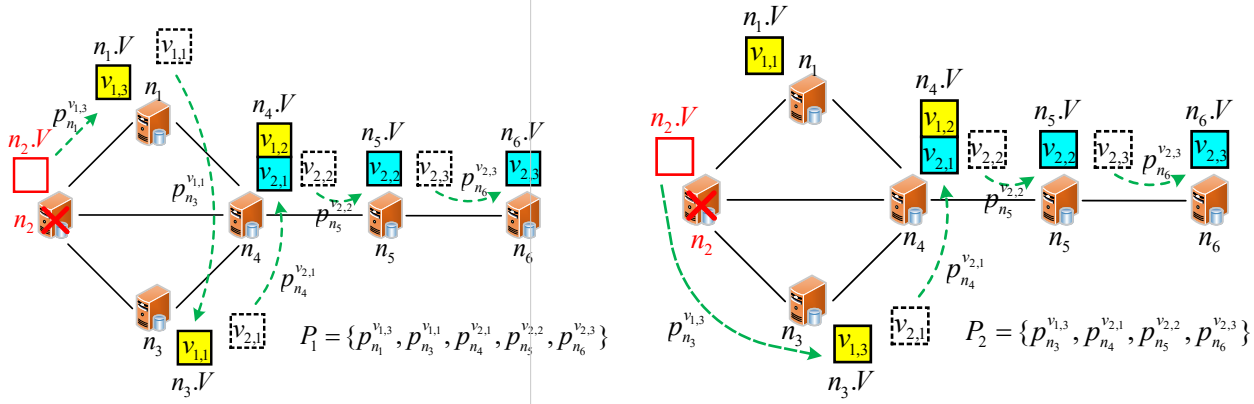


Fig. 3. Feasible strategies for example in Fig. 1 when candidate nodes are n_1 in (a) and n_3 in (b), respectively.

Algorithm 1 ORA (G, N_F, S, μ)

1: Construct table H by applying Floyd-Warshall algorithm to G
2: Let S_F be the set of interrupted services inferred from N_F
3: $\omega_{total} \leftarrow 0$
4: **while** $S_F \neq \emptyset$ **do**
5: Let s_i be the service with the highest weight in S_F
6: $conti \leftarrow true$
7: Let $V_{front} = \{v_{i,1}, v_{i,2}, \ldots, v_{i,k}\}$ be the set of the first k consecutive failed VNFs in s_i and V_{rear} be the set of the remaining failed VNFs in s_i
8: $G_{tmp} \leftarrow G$
9: **if** $V_{front} \neq \emptyset$ **then**
10: **if** $k = |s_i|$ **then**
11: Apply [41] to deploy s_i again based on the current status of the network
12: **else**
13: Let V_{front}^r be the reverse of V_{front}
14: $\{conti, G_{tmp}\} \leftarrow$ RESTOREVNFS(V_{front}^r, G_{tmp}, H, S, μ)
15: **end if**
16: **end if**
17: **if** $conti$ **then**
18: $\{conti, G_{tmp}\} \leftarrow$ RESTOREVNFS(V_{rear}, G_{tmp}, H, S, μ)
19: **end if**
20: **if** $conti$ **then**
21: $\omega_{total} \leftarrow \omega_{total} + s_i.w$
22: $G \leftarrow G_{tmp}$
23: **end if**
24: $S_F \leftarrow S_F \setminus \{s_i\}$
25: **end while**
26: **return** ω_{total}

Proof. In the CANDIDATENODES procedure, because H is inferred from the Floyd-Warshall algorithm [40], H is a table with κ rows and κ columns, where κ is the number of network nodes. Therefore, both Lines 4 and 5 require at most $O(\kappa)$ time to complete. We next sort sets A and B into the same specific order, which takes at most $O(\kappa log\kappa)$ time for each set, where $log\kappa$ stands for $log_2\kappa$. In doing so, the intersection of A and B (Line 7) requires at most $O(\kappa + \kappa) = O(\kappa)$ time to complete [42]. Eventually, the time complexity of the CANDIDATENODES procedure is bounded in $O(\kappa) + O(\kappa) + O(\kappa log\kappa) + O(\kappa log\kappa) + O(\kappa) = O(\kappa log\kappa)$. This completes the proof. □

Lemma 3. *The time complexity of the* RESTORE *procedure is bounded in* $O((\kappa\zeta)^\mu(\kappa log\kappa + \kappa\zeta))$.

Proof. It is worth noting that the RESTORE procedure is called for the first time by the RESTOREVNFS procedure, where the initial value of instance t is always equal to 0. Thus, according to the condition at Line 2, the RESTORE procedure will call itself at most μ times. We employ $T(\mu)$ to denote the time complexity of the RESTORE procedure when it calls itself μ times. It is apparent that $T(0) = O(\kappa log\kappa + \kappa\zeta)$, where κ is the number of network nodes, and ζ is the maximum resources that a node can own, because Line 5 needs at most $O(\kappa log\kappa)$ to complete according to Lemma 2; the loop between Lines 6 and 30 repeats at most κ times, and the loop between Lines 19 and 27 repeats at most ζ times. We will prove that

$$T(\mu) = O((\kappa\zeta)^\mu(\kappa\zeta + \kappa log\kappa)) \tag{4}$$

holds by means of using induction.

For the case that $\mu = 1$, the RESTORE procedure calls itself once. Likewise, Line 5 needs at most $O(\kappa log\kappa)$ time according to Lemma 2, and the loop between Lines 6 and 30 repeats at most $O(\kappa\zeta T(0))$ times. Therefore, $T(1) = O(\kappa log\kappa) + O(\kappa\zeta T(0)) = O(\kappa log\kappa + \kappa\zeta(\kappa log\kappa + \kappa\zeta)) = O(\kappa\zeta(\kappa\zeta + \kappa log\kappa))$. Hence, (4) holds for $\mu = 1$.

Suppose that $T(\mu - 1) = O((\kappa\zeta)^{\mu-1}(\kappa\zeta + \kappa log\kappa))$ holds. Line 5 requires at most $O(\kappa log\kappa)$ time according to Lemma 2, the loop between Lines 6 and 30 repeats at most $O(\kappa\zeta T(\mu - 1))$ times. Thus, $T(\mu) = O(\kappa log\kappa) + O(\kappa\zeta T(\mu - 1)) = O(\kappa log\kappa + (\kappa\zeta)^\mu(\kappa\zeta + \kappa log\kappa)) = O((\kappa\zeta)^\mu(\kappa\zeta + \kappa log\kappa))$. Hence, (4) also holds for μ. This completes the proof. □

Lemma 4. *The time complexity of the* RESTOREVNFS *procedure is bounded in* $O(\ell(\kappa\zeta)^\mu(\kappa log\kappa + \kappa\zeta))$.

Proof. In the RESTOREVNFS procedure, the loop between Lines 3 and 11 repeats at most ℓ times, where ℓ is the maximum length of services. Moreover, the time complexity of the RESTORE procedure inside the loop is $O((\kappa\zeta)^{\mu}(\kappa\zeta+\kappa log\kappa))$ according to Lemma 3. Therefore, the time complexity of the RESTOREVNFS procedure is $O(\ell(\kappa\zeta)^{\mu}(\kappa\zeta+\kappa log\kappa))$. This completes the proof. □

Theorem 2. *The time complexity of the ORA is bounded in* $O(\kappa^3+\rho(\rho+\ell(\kappa\zeta)^{\mu}(\kappa\zeta+\kappa log\kappa)))$.

Proof. In Algorithm 1, the time complexity of the Floyd-Warshall algorithm is κ^3 [40]. The loop between Lines 4 and 25 iterates at most ρ times. Inside the loop, determining the service with the highest weight (Line 5) needs at most $O(\rho)$ time. The time complexity of the RESTOREVNFS procedure is $O(\ell(\kappa\zeta)^{\mu}(\kappa log\kappa+\kappa\zeta))$ according to Lemma 4. Therefore, the time complexity of the ORA is $O(\kappa^3+\rho(\rho+\ell(\kappa\zeta)^{\mu}(\kappa\zeta+\kappa log\kappa)))$. This completes the proof. □

VI. PERFORMANCE EVALUATION

In this section, we conduct extensive simulations to evaluate the performance of the proposed algorithm, ORA, and factors affecting the resiliency of the physical network.

A. Simulation Setting

In this subsection, we set the default values for parameters employed in the experiments. If these parameters are not further mentioned in the simulation scenarios, they will get the default values.

1) Network Topology: The physical networks are generated randomly employing the Erdos-Renyi model [43]. We set the probability of generating network edges to $ln|N|/|N|$ to ensure the network is connected, where $|N|$ is the number of nodes, and $ln|N|$ is the natural logarithm of $|N|$. The maximum available resource of a node is restricted to 10 units. The failure at nodes arises randomly with the number indicated by $\alpha\times|N|$. By default, N and α are set to 500 and 0.3, respectively. In addition, to demonstrate the applicability of the proposed method for different network models, we also conduct simulations using the Barabasi-Albert (BA) [44], another popular network model.

2) Network Services: In this study, a service consists of multiple VNFs forming a chain with a particular order. It is worth noting that the services are deployed successfully over the physical network before the failure of nodes engenders the interruption. The number of deployed services is set to $\beta\times|N|$. The number of VNFs (or the length) of services varies from 1 to λ. In addition, the resource required by each VNF alters from 1 to γ units. The hop constraint between two adjacent VNFs varies from 1 to σ. Finally, the weight of services varies from 1 to δ. The values of β, λ, γ, σ, and δ are set to 0.5, 8, 5, 2, and 10 by default, respectively.

3) Comparison: From the detail of the proposed algorithm, μ is a significant parameter indicating the influence of the recovery of interrupted services on the available services. Therefore, in experiments, we evaluate the performance of the ORAs with different values of μ, which are 1, 2, 3, and 4. In addition, we also implement a greedy-based algorithm, hence named GBA, for the comparison. In the GBA, we seek to recover the interrupted services without affecting the available services. In more detail, if the residual resource of a candidate node is not adequate for reinstalling a failed VNF, we will consider the other candidate nodes without removing any VNF. By means of using the GBA, the location of the available services will not be changed. In other words, the GBA is also the ORA with $\mu=0$.

4) Performance Metrics: We assess the performance of the algorithms with respect to two metrics, the number of restored VNFs and the total weight of the recovered services, in which the latter expresses the resiliency of the network. Only when a service is successfully recovered, the failed VNFs are considered to be restored successfully. A higher value of these metrics means a better performance of the algorithm. In the experiments, we investigate the dependence of the two metrics on the number of network nodes ($|N|$), the number of failed nodes (α), the number of deployed services (β), the length of services (λ), the resource required by VNFs (γ), the hop constraint between two adjacent VNFs (σ), and the weight of services (δ).

B. Experiment Results

Main observation: In all scenarios, the ORA, though with any μ, is always better than the GBA. This is because the ORA allows adjusting the location of VNFs of the available services. Therefore, there are multiple options for recovering the interrupted services. This is also the reason why the ORA with a higher μ is better than the one with a smaller μ.

1) Impact of Network Scale: The network scale in this scenario is indicated by the number of nodes. In this experiment, we vary the number of nodes from 100 to 1000 to study the effect of the network scale on the total weight of the recovered services. From the simulation results, shown in Fig. 4, both the number of the recovered VNFs and the corresponding total weight increase almost linearly with the number of physical network nodes. This is because the increment of the number of nodes results in the increment of network resources, that is, the number of physical links and the node resource. Therefore, there will be more options to reinstall failed VNFs and recover interrupted services. In addition, Fig. 5 shows the simulation results when α is set to 0.4. From the figures, the results nearly cannot be improved when the number of nodes exceeds 600. This is because though the computing resources are sufficient to reinstall failed nodes, the hop constraint cannot be satisfied. Moreover, the number of restored VNFs and the total weight of recovered services are also decreased significantly compared with the case of $\alpha=0.3$. This is because the number of failed nodes is increased while the number of services is kept constant.

Among the algorithms, the performance of the ORAs is better than that of the GBA. This is shown more clearly in large-scale networks. Among the ORAs, the difference in the results is not substantial, particularly between $\mu=2$, $\mu=3$, and $\mu=4$. This indicates that with large-scale networks, we can consider employing a proper value of μ for the recovery

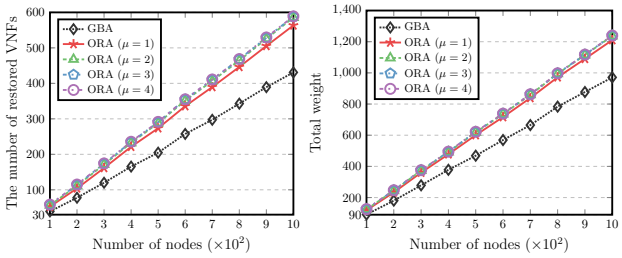


Fig. 4. Dependence of (a) the number of restored VNFs and (b) the total weight of the recovered services on the network scale when $\alpha = 0.3$.

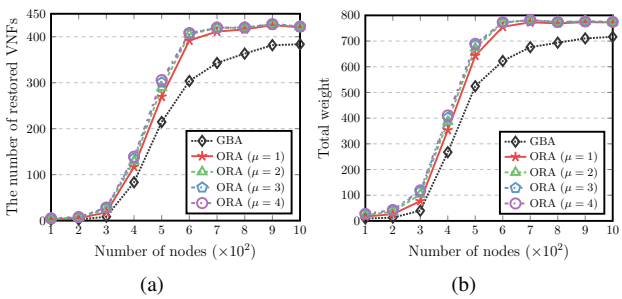


Fig. 5. Dependence of (a) the number of restored VNFs and (b) the total weight of the recovered services on the network scale when $\alpha = 0.4$.

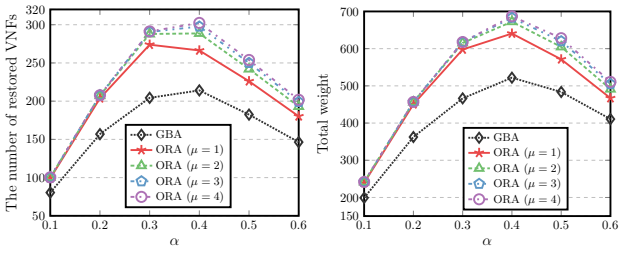


Fig. 6. Dependence of (a) the number of restored VNFs and (b) the total weight of the recovered services on the number of failed nodes.

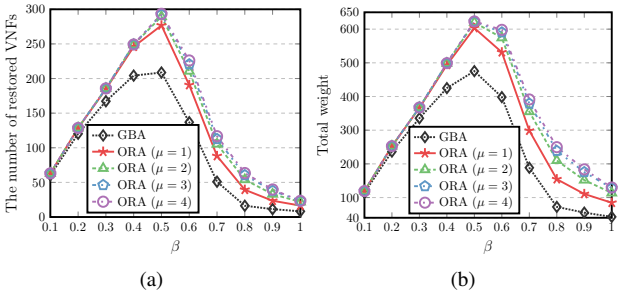


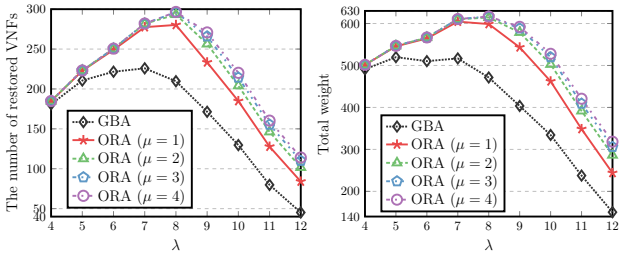
Fig. 7. Dependence of (a) the number of restored VNFs and (b) the total weight of the recovered services on the number of services deployed over the network.

to reduce the time complexity while obtaining an acceptable result.

2) Impact of Failed Nodes: In this set of simulations, we vary the number of failed nodes by changing the value of α from 0.1 to 0.6 while keeping the other parameters in order to evaluate the influence of the failed nodes on the recovery. Because the number of nodes is 500, the number of failed nodes varies from 50 to 300. The simulation results are displayed in Fig. 6. From the figure, we can observe that when α increases from 0.1 to 0.4, the number of repaired VNFs and the total weight tend to increase but decrease once α exceeds 0.4. This is because the increment of the failed nodes engenders an increase in the failed VNFs. This renders the number of interrupted services not only increased but also complicated to recover. In addition, it is worth noting that the number of network nodes is held unchanged. Hence, the total available resource of nodes significantly decreases when the number of failed nodes increases. As a consequence, when the number of failed nodes is low, the network has sufficient resources, and the interrupted services are straightforward to recover, which improves the total weight as well as the number of repaired VNFs. Nevertheless, when the number of failed nodes is too large, both the values are not only not enhanced but also degraded.

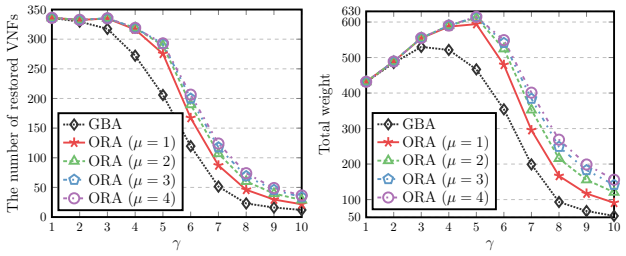
3) Impact of Services: In this experiment, we tailor the number of services deployed over the network before the failure by altering β from 0.1 to 1. This indicates the number of services varies from 50 to 500. Fig. 7 shows how the number of repaired VNFs and the total weight of the recovered services change with the number of services deployed over the network. There are two definitive trends, the upward trend when β varies from 0.1 to 0.5 and the downward trend when β changes from 0.5 to 1. We can interpret this point as follows. With the low values of β, the services are sparsely deployed over the network and occupy fewer node resources. Therefore, the available node resources are still large enough to recover the interrupted services. Moreover, the influence of the node failure on the services is not critical, which renders the interrupted services easy to be recovered. Hence, the ORA curve with $\mu = 1$ nearly coincides with the ORA curves with $\mu = 2$, $\mu = 3$, and $\mu = 4$. This also explains why the total weight of the recovered services is relatively low with the low values of β. When β increases, the density of the services is more crowded; thus, the failure of nodes renders the interruption more serious. Meanwhile, the available node resources are ever-decreasing due to the increase in the resources occupied by services. As a result, when β increases to a particular threshold, the resiliency of the network will decrease, which degrades the number of repaired VNFs and the total weight of the recovered services.

4) Impact of the Length of Services: To investigate the influence of the length of services on the total weight of the recovered services, we change λ from 4 to 12. The simulation results are shown in Fig. 8. The performance of the ORAs is demonstrated clearly in this experiment. While the GBA cannot improve the total weight of the recovered services, the ORAs can enhance the total weight until λ reaches 7 ($\mu = 1$) or 8 ($\mu = 2, 3, 4$). Because the number of failure nodes is kept constant and any failure of the VNFs will induce the interruption of services, the longer the services, the higher possibility of failure. Therefore, the number of interrupted services will increase with the length of the services. That is the reason we can improve the total weight of the recovered services by means of applying ORAs. Nevertheless, the services with more



(a) (b)

Fig. 8. Dependence of (a) the number of restored VNFs and (b) the total weight of the recovered services on the service length.



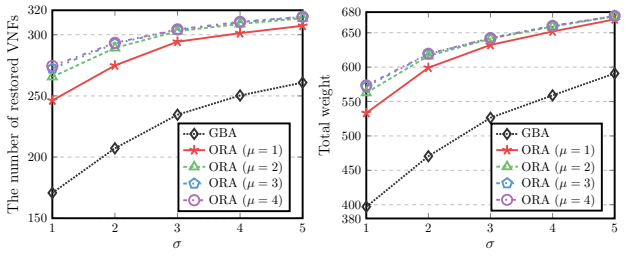
(a) (b)

Fig. 9. Dependence of (a) the number of restored VNFs and (b) the total weight of the recovered services on the resource requirement of VNFs.

VNFs also indicate that the node resources occupied by the services increase. Therefore, when λ increases to a threshold, the available node resources are insufficient to recover all the interrupted services, which engenders the degradation of the total weight of the recovered services.

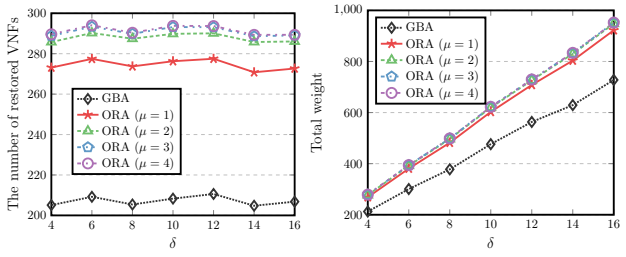
5) Impact of the Resource Requirement of VNFs: To evaluate the resiliency of the physical network according to the resource requirement of VNFs, we vary the value of γ from 1 to 10. This indicates that the average resource of nodes required for installing VNFs increases with γ. From the simulation results, shown in Fig. 9, we can observe different tendencies of the two performance metrics once increasing γ. While the number of repaired VNFs continuously decreases, particularly with a faster pace when γ alters from 4 to 8, the total weight of the recovered services can be improved until γ reaches 3 (with GBA) or 5 (with ORAs) and then definitively decreases. The results can be interpreted as follows. Because the resources of network nodes are fixed and limited to 10, the increment of the average resource required by VNFs renders the number of VNFs hosted by each node decreased, which means the VNFs will be broadly placed over the network. Therefore, when γ increases, the failure of nodes will increase the possibility of interrupted services. Moreover, with the low values of γ, the node resources are still sufficient to recover almost the interrupted services. Hence, the total weight of the recovered services will be enhanced. Nevertheless, when the average resources claimed by VNFs increase beyond a specific threshold, the deployment of services causes the node resources to exhaust rapidly. As a result, the available resources are inadequate to recover all the interrupted services, which reduces both the number of recovered services and the total weight. It is worth noting that a high value of γ renders the failed VNFs not trivial to repair due to the restriction of node resources. Thus, coupled with the fact that the interrupted services will increase significantly once increasing γ, we can still improve the total weight of the recovered services though the number of repaired VNFs is not enhanced.

6) Impact of the Number of Hops between Two VNFs: In this set of simulations, we evaluate the resiliency of the network according to the restriction on the number of hops between two VNFs by altering σ from 1 to 5. The experiment results in Fig. 10 show that both the number of repaired VNFs and the total weight of the recovered services are improved once increasing σ. It is worth noting that a larger σ renders the restriction on the number of hops between two VNFs looser,



(a) (b)

Fig. 10. Dependence of (a) the number of restored VNFs and (b) the total weight of the recovered services on the number of hops between two VNFs.



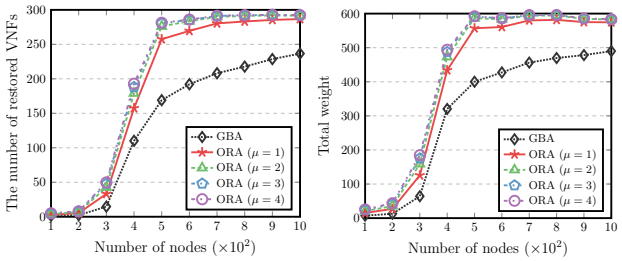
(a) (b)

Fig. 11. Dependence of (a) the number of restored VNFs and (b) the total weight of the recovered services on the service weight.

which assists in increasing the number of candidate nodes for failed VNFs. This provides more options to repair failed VNFs when increasing σ, hence improving the number of repaired VNFs and the total weight of the recovered services.

7) Impact of the Service Weight: In this experiment, we vary δ from 4 to 16, which means increasing the average weight of services. It is worth noting that the weights of services do not significantly affect the fixing process of failed VNFs. Therefore, the number of repaired VNFs is nearly kept unchanged, while the total weight of the recovered services is substantially improved when increasing the average weight of services. The results of the experiment are displayed clearly in Fig. 11.

8) Experiment with the Barabasi-Albert (BA) network model: In this experiment, the network employed to conduct the simulation is generated randomly using the BA model, in which the number of edges that are preferentially attached to existing nodes with high degree is set to 2. Similar to section VI-B1, we vary the number of nodes from 100 to 1000 to evaluate the impact of network scale on the total weight



(a) (b)

Fig. 12. Dependence of (a) the number of restored VNFs and (b) the total weight of the recovered services on the network scale using the Barabasi-Albert (BA) network model.

of recovered services. The other simulation parameters get the default values. It is worth noting that the BA network model has power-law degree distribution that causes some nodes to get more connections than the rest of the nodes in the network. As a result, such nodes are typically selected to launch VNFs for many services, and hence, if these nodes are failed, it is not trivial to recover the services due to two key reasons. The first one is that many services will be interrupted because these nodes are hosting many VNFs. The second one is that the remaining available nodes, although, possess enough capacity to reinstall failed VNFs, the distances in terms of hops between them are very long. These reasons render that the total weight cannot be improved as shown in Fig. 12. They also engender that the total weight of the recovered services decreases considerably compared with the Erdos-Renyi network model.

VII. Conclusion

In this paper, the VNF restoration for recovering weighted services (VRRWS) issue is proposed for the first time, in which we consider retrieving interrupted services that are induced by node failures via restoring failed VNFs without necessitating backup resources. The objective of the problem is to maximize the total weight of the recovered services. We first demonstrate that the hardness of the problem is NP-hard, then propose a heuristic algorithm, named online recovery algorithm (ORA), to address the issue. We then conduct extensive simulations to assess the performance of the proposed algorithm and the factors affecting the recovery. The experiment indicates that migrating available VNFs is essential to enhance the recovery result.

There are many directions for developing the problem in the future. Firstly, the solution to the VRRWS problem with more complicated service function chain models, such as directed acyclic graphs [45], will be investigated. Secondly, during service recovery, in addition to an efficient strategy for restoring services from interruption, minimizing the downtime of services is also a significant problem. Many studies have proposed useful techniques for optimizing VNF migration latency, as stated in the previous section. Therefore, future research will consider the joint optimization problem of maximizing the total weight of recovered services and minimizing the maximum migration latency concurrently by combining our method with these proposed techniques.

References

[1] E. Hernandez-Valencia, S. Izzo, and B. Polonsky, "How will NFV/SDN transform service provider opex?" *IEEE Network*, vol. 29, no. 3, pp. 60–67, 2015.

[2] L. Qu, C. Assi, and K. Shaban, "Delay-aware scheduling and resource optimization with network function virtualization," *IEEE Transactions on Communications*, vol. 64, no. 9, pp. 3746–3758, 2016.

[3] J. Gil Herrera and J. F. Botero, "Resource allocation in NFV: A comprehensive survey," *IEEE Transactions on Network and Service Management*, vol. 13, no. 3, pp. 518–532, 2016.

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

[5] G. Aceto, A. Botta, P. Marchetta, V. Persico, and A. Pescapé, "A comprehensive survey on internet outages," *Journal of Network and Computer Applications*, vol. 113, pp. 36–63, 2018.

[6] R. Kang, F. He, and E. Oki, "Resilient virtual network function allocation with diversity and fault tolerance considering dynamic requests," in *NOMS 2022-2022 IEEE/IFIP Network Operations and Management Symposium*, 2022, pp. 1–9.

[7] N. ISG, "Network function virtualisation (NFV)-resiliency requirements," *ETSI GS NFV-REL*, vol. 1, p. v1, 2014.

[8] L. Qu, C. Assi, K. Shaban, and M. J. Khabbaz, "A reliability-aware network service chain provisioning with delay guarantees in NFV-enabled enterprise datacenter networks," *IEEE Transactions on Network and Service Management*, vol. 14, no. 3, pp. 554–568, 2017.

[9] J. Fan, Z. Ye, C. Guan, X. Gao, K. Ren, and C. Qiao, "GREP: Guaranteeing reliability with enhanced protection in NFV," in *Proceedings of the 2015 ACM SIGCOMM Workshop on Hot Topics in Middleboxes and Network Function Virtualization*, ser. HotMiddlebox '15. New York, NY, USA: Association for Computing Machinery, 2015, p. 13–18.

[10] L. Qu, C. Assi, M. J. Khabbaz, and Y. Ye, "Reliability-aware service function chaining with function decomposition and multipath routing," *IEEE Transactions on Network and Service Management*, vol. 17, no. 2, pp. 835–848, 2020.

[11] P. M. Mohan and M. Gurusamy, "Resilient VNF placement for service chain embedding in diversified 5G network slices," in *2019 IEEE Global Communications Conference (GLOBECOM)*, 2019, pp. 1–6.

[12] Y. Kanizo, O. Rottenstreich, I. Segall, and J. Yallouz, "Optimizing virtual backup allocation for middleboxes," *IEEE/ACM Transactions on Networking*, vol. 25, no. 5, pp. 2759–2772, 2017.

[13] M. Zhu, F. He, and E. Oki, "Optimization model for multiple backup resource allocation with workload-dependent failure probability," *IEEE Transactions on Network and Service Management*, vol. 18, no. 3, pp. 3733–3752, 2021.

[14] F. He, T. Sato, B. C. Chatterjee, T. Kurimoto, U. Shigeo, and E. Oki, "Robust optimization model for primary and backup resource allocation in cloud providers," *IEEE Transactions on Cloud Computing*, pp. 1–1, 2021.

[15] Y. Chen and J. Wu, "Latency-efficient vnf deployment and path routing for reliable service chain," *IEEE Transactions on Network Science and Engineering*, vol. 8, no. 1, pp. 651–661, 2021.

[16] R. Xia, H. Dai, J. Zheng, R. Gu, X. Wang, and G. Chen, "SAFE: Service availability via failure elimination through VNF scaling," in *Proceedings of the 48th International Conference on Parallel Processing*, ser. ICPP 2019. New York, NY, USA: Association for Computing Machinery, 2019.

[17] F. He and E. Oki, "Backup allocation model with probabilistic protection for virtual networks against multiple facility node failures," *IEEE Transactions on Network and Service Management*, vol. 18, no. 3, pp. 2943–2959, 2021.

[18] H. Huang and S. Guo, "Proactive failure recovery for NFV in distributed edge computing," *IEEE Communications Magazine*, vol. 57, no. 5, pp. 131–137, 2019.

[19] J. Son and R. Buyya, "Priority-aware VM allocation and network bandwidth provisioning in Software-Defined Networking (SDN)-enabled clouds," *IEEE Transactions on Sustainable Computing*, vol. 4, no. 1, pp. 17–28, 2019.

[20] F. Bari, S. R. Chowdhury, R. Ahmed, R. Boutaba, and O. C. M. B. Duarte, "Orchestrating virtualized network functions," *IEEE Transactions on Network and Service Management*, vol. 13, no. 4, pp. 725–739, 2016.

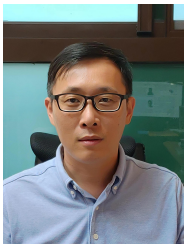
[21] S. Agarwal, F. Malandrino, C. F. Chiasserini, and S. De, "VNF placement and resource allocation for the support of vertical services in 5G

networks," *IEEE/ACM Transactions on Networking*, vol. 27, no. 1, pp. 433–446, 2019.
[22] A. Gupta, M. F. Habib, P. Chowdhury, M. Tornatore, and B. Mukherjee, "Joint virtual network function placement and routing of traffic in operator networks," *UC Davis, Davis, CA, USA, Tech. Rep*, 2015.
[23] R. Riggio, A. Bradai, D. Harutyunyan, T. Rasheed, and T. Ahmed, "Scheduling wireless virtual networks functions," *IEEE Transactions on Network and Service Management*, vol. 13, no. 2, pp. 240–252, 2016.
[24] B. Addis, D. Belabed, M. Bouet, and S. Secci, "Virtual network functions placement and routing optimization," in *2015 IEEE 4th International Conference on Cloud Networking (CloudNet)*, 2015, pp. 171–177.
[25] L. Liu, S. Guo, G. Liu, and Y. Yang, "Joint dynamical VNF placement and SFC routing in NFV-enabled SDNs," *IEEE Transactions on Network and Service Management*, vol. 18, no. 4, pp. 4263–4276, 2021.
[26] P. Jin, X. Fei, Q. Zhang, F. Liu, and B. Li, "Latency-aware VNF chain deployment with efficient resource reuse at network edge," in *IEEE INFOCOM 2020 - IEEE Conference on Computer Communications*, 2020, pp. 267–276.
[27] Y. Wang, C.-K. Huang, S.-H. Shen, and G.-M. Chiu, "Adaptive placement and routing for service function chains with service deadlines," *IEEE Transactions on Network and Service Management*, vol. 18, no. 3, pp. 3021–3036, 2021.
[28] A. Hmaity, M. Savi, F. Musumeci, M. Tornatore, and A. Pattavina, "Protection strategies for virtual network functions placement and service chains provisioning," *Networks*, vol. 70, no. 4, pp. 373–387, 2017.
[29] J. Fan, C. Guan, Y. Zhao, and C. Qiao, "Availability-aware mapping of service function chains," in *IEEE INFOCOM 2017 - IEEE Conference on Computer Communications*, 2017, pp. 1–9.
[30] S. Ayoubi, Y. Chen, and C. Assi, "Towards promoting backup-sharing in survivable virtual network design," *IEEE/ACM Transactions on Networking*, vol. 24, no. 5, pp. 3218–3231, 2016.
[31] Y. Wang, L. Zhang, P. Yu, K. Chen, X. Qiu, L. Meng, M. Kadoch, and M. Cheriet, "Reliability-oriented and resource-efficient service function chain construction and backup," *IEEE Transactions on Network and Service Management*, vol. 18, no. 1, pp. 240–257, 2021.
[32] X. Wang, X. Chen, C. Yuen, W. Wu, M. Zhang, and C. Zhan, "Delay-cost tradeoff for virtual machine migration in cloud data centers," *Journal of Network and Computer Applications*, vol. 78, pp. 62–72, 2017.
[33] A. M. Ghaleb, T. Khalifa, S. Ayoubi, K. B. Shaban, and C. Assi, "Surviving multiple failures in multicast virtual networks with virtual machines migration," *IEEE Transactions on Network and Service Management*, vol. 13, no. 4, pp. 899–912, 2016.
[34] V. Eramo, E. Miucci, M. Ammar, and F. G. Lavacca, "An approach for service function chain routing and virtual function network instance migration in network function virtualization architectures," *IEEE/ACM Transactions on Networking*, vol. 25, no. 4, pp. 2008–2025, 2017.
[35] F. Zhang, G. Liu, X. Fu, and R. Yahyapour, "A survey on virtual machine migration: Challenges, techniques, and open issues," *IEEE Communications Surveys & Tutorials*, vol. 20, no. 2, pp. 1206–1243, 2018.
[36] P. K. Thiruvasagam, V. J. Kotagi, and C. S. R. Murthy, "A reliability-aware, delay guaranteed, and resource efficient placement of service function chains in softwarized 5G networks," *IEEE Transactions on Cloud Computing*, vol. 10, no. 3, pp. 1515–1531, 2022.
[37] L. Askari, A. Hmaity, F. Musumeci, and M. Tornatore, "Virtual-network-function placement for dynamic service chaining in metro-area networks," in *2018 International Conference on Optical Network Design and Modeling (ONDM)*, 2018, pp. 136–141.
[38] J. Zhang, D. Zeng, L. Gu, H. Yao, and M. Xiong, "Joint optimization of virtual function migration and rule update in software defined NFV networks," in *GLOBECOM 2017 - 2017 IEEE Global Communications Conference*, 2017, pp. 1–5.
[39] K. Bernhard and J. Vygen, "Combinatorial optimization: Theory and algorithms," *Springer, Third Edition, 2005.*, 2008.
[40] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein, *Introduction to algorithms*. MIT press, 2022.
[41] 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," *IEEE/ACM Transactions on Networking*, vol. 26, no. 4, pp. 1562–1576, 2018.
[42] H. Inoue, M. Ohara, and K. Taura, "Faster set intersection with SIMD instructions by reducing branch mispredictions," *Proc. VLDB Endow.*, vol. 8, no. 3, p. 293–304, Nov 2014.
[43] E. N. Gilbert, "Random graphs," *The Annals of Mathematical Statistics*, vol. 30, no. 4, pp. 1141–1144, 1959.
[44] A.-L. Barabási and R. Albert, "Emergence of scaling in random networks," *Science*, vol. 286, no. 5439, pp. 509–512, 1999. [Online]. Available: https://www.science.org/doi/abs/10.1126/science.286.5439.509
[45] X. Lin, D. Guo, Y. Shen, G. Tang, and B. Ren, "DAG-SFC: Minimize the embedding cost of sfc with parallel vnfs," in *Proceedings of the 47th International Conference on Parallel Processing*, ser. ICPP '18. New York, NY, USA: Association for Computing Machinery, 2018. [Online]. Available: https://doi.org/10.1145/3225058.3225111

Dung H. P. Nguyen received the B.Sc. degree in electrical and electronic engineering, the M.Sc. degree in electronic engineering from the Ho Chi Minh City University of Technology, in 2008 and 2013, respectively, and the Ph.D. degree in electronic engineering from the National Kaohsiung University of Science and Technology, Taiwan, in 2023. He is currently a lecturer with the Faculty of Engineering and Technology, Pham Van Dong University, Quang Ngai, Vietnam. His research interests include wireless sensor networks, network function virtualization, and quantum networking.

Chih-Chieh Lin received the M.Sc. degree in electronic engineering from National Kaohsiung University of Science and Technology in 2021. His research interests include design and analysis of algorithms, wireless sensor networks, and network function virtualization.

Tu N. Nguyen (Senior Member, IEEE) received the Ph.D. degree in electronic engineering from the National Kaohsiung University of Science and Technology (formerly, National Kaohsiung University of Applied Sciences), Kaohsiung, Taiwan, in 2016. He is currently an Assistant Professor with the Department of Computer Science, Kennesaw State University (KSU), Kennesaw, GA, USA. In August 2019, he was an Assistant Professor of computer science with Purdue University FortWayne, FortWayne, IN, USA. Since 2021, he has been with KSU. In 2017, he was a Postdoctoral Associate with the Department of Computer Science and Engineering, University of Minnesota Twin Cities, Minneapolis, MN, USA. In 2016, prior to joining the University of Minnesota, he joined the Missouri University of Science and Technology as a Postdoctoral Researcher with the Intelligent Systems Center. His research interests include design and analysis of algorithms, network science, cyberphysical systems, and cybersecurity. Dr. Nguyen is an Associate Editor for IEEE ACCESS (since 2019) and EURASIP Journal on Wireless Communications and Networking (since 2017). He is also on the Editorial Board of the Cybersecurity Journal, Internet Technology Letters (since 2017), International Journal of Vehicle Information and Communication Systems (since 2017), International Journal of Intelligent Systems Design and Computing (since 2017), and IET Wireless Sensor Systems (since 2017). He was a TPC Chair for the NICS 2019, SoftCOM (25th), and ICCASA 2017, a Publicity Chair for iCAST 2017 and BigDataSecurity 2017, and a Track Chair for ACT 2017. He was a Technical Program Committee Member for more than 70 premium conferences in the areas of network and communication, such as INFOCOM, Globecom, ICC, and RFID.



Shao-I Chu (Member, IEEE) received the B.Sc. degree in industrial engineering from National Tsing Hua University, Hsinchu, Taiwan, in 1998, and the M.Sc. and Ph.D. degrees in electrical engineering from National Taiwan University, Taipei, Taiwan, in 2000 and 2007, respectively. From 2008-2012, he was with the Department of Information Engineering, I-Shou University, Kaohsiung, Taiwan. He is currently a professor with the Department of Electronic Engineering, National Kaohsiung University of Science and Technology, Kaohsiung, Taiwan. His current research interests include stochastic computing, digital IC design, signal processing, and computer networks. He is a member of the Phi Tau Phi Scholastic Honor Society.

Bing-Hong Liu received the B.Sc. and Ph.D. degrees in computer science from National Tsing Hua University in 2001 and 2008, respectively. He is currently a Professor with the Department of Electronic Engineering, National Kaohsiung University of Science and Technology, Kaohsiung, Taiwan. His research interests include mobile computing, distributed computing, mobile ad hoc networks, and wireless sensor networks.