Received: 24 April 2018 | Revised: 5 August 2018 | Accepted: 5 September 2018

DOI: 10.4218/etrij.2018-0156

ORIGINAL ARTICLE

WILEY ETRI Journal

Reliability-aware service chaining mapping in NFV-enabled networks

Yicen Liu | **Yu Lu** | **Wenxin Qiao** | **Xingkai Chen**

Shijiazhuang Campus, Army Engineering University, Hebei, China.

Correspondence
Yu Lu, Shijiazhuang Campus, Army Engineering University, Hebei, China.
Email: oecluyu@sina.com

Funding information
This research was supported by the National Natural Science Foundation of China under Grants 51377170, 61271152, and the National Youth Science Foundation of China under Grant 61602505.

Network function virtualization can significantly improve the flexibility and effectiveness of network appliances via a mapping process called service function chaining. However, the failure of any single virtualized network function causes the breakdown of the entire chain, which results in resource wastage, delays, and significant data loss. Redundancy can be used to protect network appliances; however, when failures occur, it may significantly degrade network efficiency. In addition, it is difficult to efficiently map the primary and backups to optimize the management cost and service reliability without violating the capacity, delay, and reliability constraints, which is referred to as the reliability-aware service chaining mapping problem. In this paper, a mixed integer linear programming formulation is provided to address this problem along with a novel online algorithm that adopts the joint protection redundancy model and novel backup selection scheme. The results show that the proposed algorithm can significantly improve the request acceptance ratio and reduce the consumption of physical resources compared to existing backup algorithms.

KEYWORDS
network function virtualization, Q-learning algorithm, redundancy, service function chaining

1 | INTRODUCTION

In recent years, the demand for Internet services has grown rapidly due to the explosive adoption of mobile devices and the emergence of novel networking paradigms, such as the Internet of Things (IoTs) [1]. The networks of today include a multiplicity of vertically integrated proprietary middleboxes, such as firewalls, intrusion detection systems (IDSs), and network address translation (NAT) appliances. However, the deployment of statically configured middleboxes results in two main problems. The first is that both a high capital expenditure (CAPEX) and operating expenditure (OPEX) is required, and second, it is impossible to add new functionality to an existing middlebox. These problems make it difficult for network operators to deploy new services. Consequently, research into new dynamic service models has grown in importance [2,3].

Both of these problems can be solved by software-defined networking (SDN) [4] and network function virtualization (NFV) [5]. In SDN, the control plane is decoupled from the data plane. The controller in these networks is the central administrator of the network and programmatically configures the forwarding flow tables in the switches, thereby enabling virtualized network function (VNF) orchestration. NFV shifts packet processing functions from hardware middleboxes toward software implementations, thereby enabling network optimization and cost reduction. Dynamic service chaining is an enabler of the SDN/NFV networking paradigm and provides a flexible and economic alternative to the static network environment of today for application service providers and Internet service providers.

At present, dynamic service chaining technology is still in its infancy and software-based middleboxes pose unique reliability challenges for future networks [6–8]. Managing

the network reliability is critically important as the failure of any VNF in an arbitrary service chain, be it hardware, such as the failure of the virtual machine providing that VNF, or software, such as the misconfiguration of the VNF itself, will disrupt the entire chain [9]. Sherry et al. [10] proposed a middlebox architecture with high fault tolerance that reduces the probability of VNF failure by rolling back the processing state. However, as this mechanism is only optimized inside the hardware-based middlebox, this can increase physical resource consumption and has certain limitations for the improvement ratio. Long et al. [11] focused on the reliability-aware service chaining mapping (SCM) problem in order to minimize the network-wide communication bandwidth required for service chaining under service reliability constraints. However, they did not consider failures at the VM level in their reliability functions and their focus was on specific issues related to chaining, such as routing and VNF ordering, that constrain VNF-to-host placement. Furthermore, their model did not include CPU capacity constraints. Long et al. [12] further proposed a reliability-aware joint VNF chaining placement and flow routing optimization method, which can be easily extended to support resource sharing between adjacent backup VNFs. Fan et al. [13] proposed a joint protection (JP) redundancy model to effectively save resources, such as CPU and bandwidth, compared to two other redundancy models, namely, the dedicated protection (DP) redundancy model and the shared protection (SP) redundancy model. However, a mathematical proof of their model was not provided. Based on their proposed JP redundancy model, Fan et al. [14] further proposed a highly reliable service chaining backup method that used a greedy strategy to select the two least reliable VNFs for redundancy, which made it easy to obtain the local optimal solution. Carpio and Jukan [15] proposed a migration-based reliability service chaining deployment method that ensured service continuity via a resource migration strategy; however, this approach resulted in a longer recovery time. Gill et al. [16] and Potharaju et al. [17] proposed the typical 1 + 1 standby redundancy model; however, this approach was later abandoned as it wasted a large number of backup resources. Hmaity et al. [18] proposed different approaches to provision service chainings with resiliency against single-link and single-node failures, then proposed three integer linear programming (ILP) models to jointly solve the problem of VNF placement and traffic routing while guaranteeing resiliency against single-link and/or single-node failures. Scholler et al [19] presented a solution for the resilient deployment of network functions based on OpenStack for the design and implementation of the proposed service orchestrator mechanism. Liu et al. [20] evaluated the reliability criterion within a probabilistic model and proposed a linear programming (LP) model to address the joint optimization of the chain composition problem.

The above studies on reliability-aware SCM focused on aspects of migration and redundancy. However, few studies have focused on the design of an online reliability-aware SCM strategy that optimizes the conflicting objectives of management cost and service reliability while respecting the resource capacity, delay, and reliability constraints. The focus of the current study was to develop an optimal mapping strategy in terms of the improvement ratio and management cost. Here, as redundancy is introduced in a virtualization layer, it is therefore straightforward to dynamically create VNFs in dynamic NFV-enabled scenarios. A JP model is also adopted [13], which requires a backup VNF to reserve sufficient CPU resources for all primary VNFs it protects. This allows service chaining to continue as normal even if all primary VNFs fail simultaneously. In addition, in the shared protection model, one backup VNF can take over the traffic when any one of the primary VNFs it protects fails by allocating the maximum amount of resources required among all primary VNFs. This JP approach provides higher reliability while consuming fewer resources compared to the shared protection model. A detailed proof can be found in Appendix A.

To address reliability-aware SCM in the dynamic NFV-enabled scenario, a novel online algorithm is proposed that adopts the JP redundancy model and a novel backup selection scheme. To the best of our knowledge, no existing works have addressed this problem. Two key contributions of the current work are as follows:

The reliability-aware SCM problem is theoretically formulated based on the mixed integer linear programming model.

A novel online algorithm is proposed based on the JP redundancy model and a novel backup selection scheme, the performance of which is evaluated in MATLAB.

The remainder of this paper is organized as follows. First, the mathematical model used for the proposed system is introduced and the reliability-aware SCM problem is formally defined (Section 2). Then, a mixed integer linear programming (MILP) formulation is presented (Section 3). Next, a novel online algorithm is proposed to obtain the near-optimal solution (Section 4). Finally, the proposed algorithm is validated via MATLAB (Section 5) and the work is concluded by outlining several promising future directions (Section 6).

2 | NETWORK MODEL AND PROBLEM DESCRIPTION

In this section, the mathematical model for the proposed system is introduced (Sections 2.1 and 2.2) and the reliability-aware SFC mapping problem is formally introduced (Section 2.3).

2.1 | Physical networks

In this model, the physical network is defined as an undirected graph $G^s = (N^s, E^s, M_N, M_L)$, where N^s represents a set of physical nodes where the VNFs can be deployed, E^s represents a set of physical links connecting $n^s \in N^s$, M_N represents the remaining CPU resources on $n^s \in N^s$, and M_L represents the remaining bandwidth on $e^s \in E^s$. Each physical node $n^s \in N^s$ is associated with a set of x types of resources $S_n^x = \{s_n^i | i \in [1, x]\}$, where s_n^i represents the capacity of resources of type i, namely CPU, memory, or bandwidth. Given the set of resources available at a physical node $n^s \in N^s$, each VNF requires a set of resources to perform one single network function denoted $f_n \in F$, where $F = \cup_{i=1}^{N^s} f_i$ represents the set of all VNFs. In addition, each physical node $n^s \in N^s$ is associated with a reliability r_n.

2.2 | SFC request

It is assumed there is a set of m SFC requests represented by $\Gamma = (t_1, t_2, \ldots, t_m)$ and each request can be described as $t = (s_t, d_t, F_t, \delta_t, \Theta_{\text{req}}^t)$, where s_t and d_t represent the fixed ingress and egress of t, respectively, and F_t represents the set of VNFs of t. Each SFC request requires that the expected propagation delay from ingress to egress is within δ_t and the reliability is above Θ_{req}^t, as specified in the corresponding service level agreements (SLAs).

2.3 | Reliability-aware SCM problem description

Next, a dynamic NFV-enabled scenario is considered where an operating network serves a set of online requests Γ. In this network, a set of VNFs has already been placed and routing paths for the traffic in Γ have also been provisioned. Here, the operating network receives new online requests and seeks to provision the required VNFs and routing paths. When mapping a service chain, it is possible to map a VNF to a physical node (PN) by reserving an appropriate type and amount of CPU resources in a chosen PN to perform the function requested by that VNF or to map a logical link by allocating an appropriate amount of bandwidth along the chosen service path to carry the traffic flow from one VNF to another.

Example: As shown in Figure 1, the VNF requiring a Media Resource Function (MRF) on SFC 1 can only be mapped onto PN4, which is the only PN providing this function. The traffic from the MRF to HSS can flow directly from PN4 to PN6 or be redirected via any other path beginning with PN4 and ending with PN6.

A service chain is considered to be available at a given time if all associated VNFs are functioning normally. In other words, all VNFs have heterogeneous reliability.

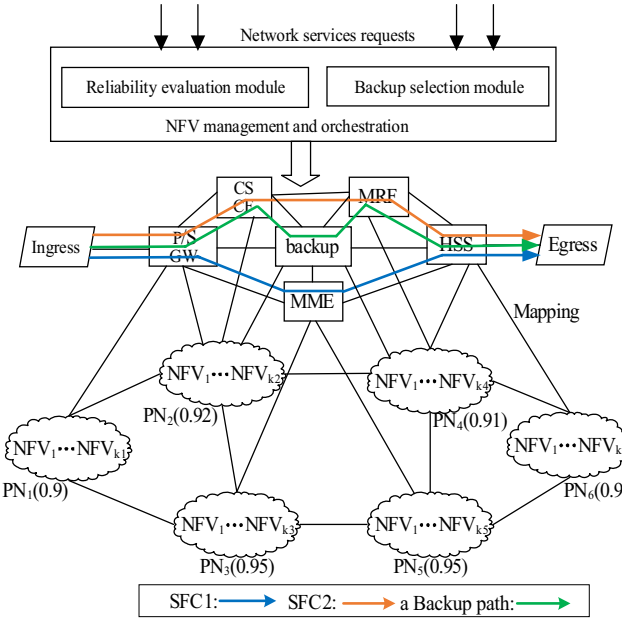

FIGURE 1 Service chain mapping for an network function virtualization with off-site redundancy

However, the presence of only high-reliability VNFs is not enough to satisfy the requirements of service chain reliability. In addition, the failure of any one of the VNFs in a service chain results in a breakdown of the entire chain, which results in resource wastage, delays, and significant data loss.

Example: Figure 1 illustrates two service chains, namely, SFC1 and SFC2. SFC1 includes three primary VNFs, SGW, MME, and HSS, for which the reliability requirement is 0.82. The reliability of each VNF is 0.9, 0.95, and 0.98, respectively. The reliability of this chaining is 0.84, which is sufficient to satisfy the requirements of the application. Thus, there is no need to provide backup functions. SFC2 includes four primary VNFs, namely, SGW, CSCF, MRF, and HSS for which the reliability requirement is also 0.82, and the reliability of each VNF is 0.9, 0.92, 0.91, and 0.98, respectively. Thus, the reliability of this SFC does not satisfy the requirements of the application. To mask failures in such configurations, off-site redundancy protection strategies can be employed to ensure sufficient reliability.

In this paper, a JP model is adopted for VNF failures that provides end-to-end protection to all connected VNFs. The amount of CPU resources and bandwidth reserved at the backup will be sufficient for all primary VNFs and interconnected logical links, respectively.

At this stage, it should be noted that if a failure occurs, it is assumed that an SDN controller will configure intermediate switches along the path of the service chain to reconstruct the chain by activating switch ports connected to backup VNFs to replace the one that failed and blocking

ports connected to the failed VNF. This ensures a seamless switchover of the traffic in the reconfigured backup path and enables the SFC request to still function normally even if all primary VNFs failed simultaneously.

Example: For simplicity, it is assumed that both CSCF and MRF can be backed up, in other words, $n_b = n_{b1} + n_{b2}$, and sufficient amount of backup resources will be reserved at backup n_b for CSCF n_1 and MRF n_2, in other words, $s_b = s_1 + s_2$. In addition, sufficient bandwidth will be reserved for backup links $e^b \in E^b$ to connect each backup VNF in sequence to form a backup path, as shown in Figure 1. Here, $e^b = (n_1, n_{b1}) \bigcup (n_{b1}, n_{b2}) \bigcup (n_{b2}, n_1)$ represents the requested backup logical links, where (n_{b1}, n_{b2}) represents the virtual link between backup CSCF n_{b1} and backup MRF n_{b2}, (n_1, n_{b1}) represents the virtual link between failed CSCF n_1 and backup CSCF n_{b1}, and (n_{b1}, n_2) represents the virtual link between failed MRF n_2 and backup MRF n_{b2}.

As shown in Figure 1, the mapping procedure is as follows. First, an online service chain request arrives with a specific reliability requirement at the centralized controller, such as NFV management & orchestration (MANO), and is mapped via all primary VNFs to the datacenters along the shortest path. Second, the controller executes the reliability evaluation module, which periodically evaluates reliability of network and collects information related to the state of the network. Third, based on the results of the reliability evaluation module, the controller runs the backup selection module to determine the optimal number of backup VNFs. Then, the results of the backup selection module are incorporated into the configuration policy and the backup links are instructed to connect these selected backup VNF instances based on the JP model to form the backup topology. Finally, the backup topology is reviewed to ensure it efficiently connects the datacenter sites and that all logical links are mapped to physical links, thus optimizing the conflicting objectives of management cost and service reliability while simultaneously satisfying the capacity, reliability, delay, and connectivity constraints (discussed in Section 3.3).

In summary, reliability-aware service chain mapping can be decomposed into two phases: primary mapping and backup mapping. In primary mapping, all primary VNFs and the associated logical links are mapped to the physical network. In backup mapping, the constraints to select backup VNFs should be considered when forming backup paths based on the JP model. The reliability-aware SCM can be abstractly described as follows.

2.3.1 | Primary mapping

Given a virtual network $G^v = (N^v, E^v)$, where N^v represents a set of primary VNFs, and E^v represents a set of primary logical links, and given a physical network $G^{\text{sub}} = (N^{\text{sub}}, E^{\text{sub}})$, where N^{sub} represents a set of physical nodes and E^{sub} represents a set of physical links, the primary topology mapping process f can be described as:

$$f:\ G^v(N^v, E^v) \mapsto G^{\text{sub}}(N^{\text{sub}}, E^{\text{sub}}). \tag{1}$$

Primary mapping must consider the site capacity, link capacity, and location constraints. That is:

$$\begin{cases} M_N(n^s, n^v) \geq c(n^v), & \forall n^s \in N^{\text{sub}},\ n^v \in N^v \\ n^s \in Loc_i(n^v) & \\ M_L(e^s, e^v) \geq bw(e^v), & \forall e^s \in E^{\text{sub}},\ e^v \in E^v \end{cases}, \tag{2}$$

where G^{sub} represents a subgraph of G^s, $N^{\text{sub}} \subset N^s$, and $E^{\text{sub}} \subset E^s$. The primary mapping f can be decomposed into the primary VNF mapping f_N: $N^v \mapsto N^{\text{sub}}$ and the primary logical link mapping f_E: $E^v \mapsto E^{\text{sub}}$, $M_N(n^s, n^v)$ represents the total load of the CPU resource provided to the primary VNF $n^v \in N^v$ at each physical node $n^s \in N^{\text{sub}}$, $Loc_i(n^v)$ represents the orchestration order constraint of service chain t, $M_L(e^s, e^v)$ represents the total bandwidth provided to primary logical link $e^v \in E^v$ at each physical link $e^s \in E^{\text{sub}}$, $c(n^v)$ represents the amount of requested CPU resources for the primary VNF $n^v \in N^v$, and $bw(e^v)$ represents the amount of requested bandwidth for the primary logical link $e^v \in E^v$.

2.3.2 | Backup mapping

Given a virtual network $G_b^v = (N^b, E^b)$, where N^b represents a set of backup VNFs and E^b represents a set of backup logical links, and a physical network $G_b^s = (B_N, B_E)$, where B_N represents a set of backup physical nodes and B_E represents a set of backup physical links, the backup topology mapping process f_b can be described as:

$$f_b:\ G_b^v(N^b, E^b) \mapsto G_b^s(B_N, B_E). \tag{3}$$

In backup mapping, the site capacity and link capacity constraint must be considered. That is:

$$\begin{cases} M_N(n^s, n^b) \geq c(n^b), & \forall n^s \in B_N,\ n^b \in N^b \\ M_L(e^s, e^b) \geq bw(e^b), & \forall e^s \in B_E,\ e^b \in E^b \end{cases}, \tag{4}$$

where G_b^s represents a subgraph of G^s, $B_N \subset N^s$, and $B_E \subset E^s$. The backup mapping f can be decomposed into the backup VNF mapping f_b^E: $N^b \mapsto B_N$ and the backup logical link mapping f_b^E: $N^b \mapsto B_N$. Here, $M_N(n^s, n^b)$ represents the total load of the CPU resource provided to backup VNF $n^b \in N^b$ at each physical node $n^s \in B_N$, $M_L(e^s, e^b)$ represents the total bandwidth provided to the backup logical link $e^b \in E^b$ at each physical link $e^s \in B_E$, $c(n^b)$ represents the amount of requested CPU resources for the backup VNF $n^b \in N^b$, and $bw(e^b)$ represents the bandwidth requested for the backup logical link $e^b \in E^b$.

3 | MODELS FOR RELIABILITY-AWARE SERVICE CHAIN MAPPING

In this section, the reliability (Section 3.1) and cost models (Section 3.2) considered in this work are described, then a mixed integer linear programming (MILP) formulation is provided (Section 3.3). The symbols used for the model are listed in Appendix B.

3.1 | Reliability model

The reliability model is based on the following definitions:

Definition 1. *Reliability of a single component*. A service chain can be decomposed into its constituent components, of which the reliability can be obtained via the network management system, such as the SDN. Here, the reliability of a component is based on the time the component is available relative to the total time the network was operational. Therefore, the reliability of a VNF can be expressed based on the *Mean Time Between Failure* (MTBF) and *Mean Time To Repair* (MTTF). In general, the reliability of a VNF can be characterized as:

$$r_{\mathrm{VNF}} = \mathrm{MTBF}_{\mathrm{VNF}} \times \left(\mathrm{MTBF}_{\mathrm{VNF}} + \mathrm{MTTR}_{\mathrm{VNF}}\right)^{-1}. \quad (5)$$

Definition 2. *Reliability of the composed system*. In general, each service chain is composed of a number of VNFs and ideally, each constituent component must operate normally. In this paper, for simplicity, only VNF failures are considered. To increase the reliability of a service chain, each traffic node can be only provisioned onto exactly one VNF to ensure the failure probabilities are independent between VNFs. When no-backup VNF is provisioned, the corresponding reliability of an arbitrary service chain is given by:

$$R_t = \prod_{f_n \in [1,|F|]} r_{f_n},\ \forall t \in [1, |\Gamma|]. \quad (6)$$

If the reliability Rt of t satisfies the requirement Θ^t_{req}, in other words, $\Theta^t_{\mathrm{req}} < R_t$, then there is no need to provide backup VNFs. Otherwise, the addition of backup VNFs will increase the robustness of t. Each primary VNF $f_n \in F$ has at least one redundant function provisioned in case any primary VNFs along the service chain are blocked. Here, it is assumed that F^b represents the set of all redundant functions of t, namely, the primary VNFs. The reliability after the backup VNF is added can be expressed as follows:

$$R'_t = \prod_{f_n \in [1,|F|]} \left[1 - \prod_{f_b \in [1,|F^b|]} \sum_{k \in [1,|N^s|]} \left(1 - r_{f_b} x_k^{f_b}\right)\right],\ \forall t \in [1, |\Gamma|]. \quad (7)$$

Definition 3. *Improvement ratio*. The improvement ratio is the ratio of the improvement on the reliability of the composed system to the reliability before backups are added. The improvement ratio is defined as follows:

$$U_t = \frac{R'_t - R_t}{R_t},\ \forall t \in [1, |\Gamma|]. \quad (8)$$

3.2 | Cost model

It is assumed that the mapping of a reliability-aware service chain comes with a fixed management overhead. The novel concept of *resource stress factor* (RSF) is defined to ensure critical resources, such as CPU and bandwidth, are not overused without incurring significant management cost. In this context, the following terms are defined:

Definition 4. *Node importance factor*. In G^s, it is assumed the degree of n^s is d_i, then NIF = ($1/d_1$, $1/d_2$, …, $1/d_n$) represents the node importance factor of adjacent points.

Definition 5. *Node connectivity factor*. The node connectivity factor is defined as the degree of impact the failing service node has on the remaining network topology. If $A(G^s)$ represents the adjacency matrix of G^s, then let NCF = NIF × $A(G^s)$ be the degree of impact of each service node. Thus, the *node connection factor* (NCF) can be expressed as follows:

$$\mathrm{NCF}(n^s) = \begin{cases} \dfrac{\mathrm{NCF}_{(n^s)} - \min(\mathrm{NCF})}{\max(\mathrm{NCF}) - \min(\mathrm{NCF})}, & \max(\mathrm{NCF}) - \min(\mathrm{NCF}) \neq 0 \\ 1, & \max(\mathrm{NCF}) - \min(\mathrm{NCF}) = 0 \end{cases}. \quad (9)$$

Definition 6. *Edge connectivity factor*. The edge connectivity factor is defined as the degree of impact of the failing edge on the remaining network topology and $\mathrm{ECF}_{ij} = (1/d_i + 1/d_j)/2$ represents the edge connectivity factor.

Definition 7. *Saturation factor*. The *saturation factor* (SF) is defined as the degree of impact when a service node or edge fails. Here, the *saturation factor* is expressed as follows:

$$\mathrm{SF}(x) = \begin{cases} 0, & k = 0 \\ (\mathrm{PM}_x)^{1/y}, & x \in [1, |N^s|], \\ (\mathrm{PM}_x)^{1/y}, & x \in [1, |E^s|] \end{cases} \quad (10)$$

where SF(x) represents the saturation factor of a resource of type x, y represents the number of times a resource of type x can instantiate a VNF or a logical link, and PM_x represents the percentage of resources allocated by a resource of type x.

Definition 8. *Resource stress factor*. Considering the above three definitions, the *resource stress factor* can be represented as follows:

$$\psi(x) = \begin{cases} \alpha \times \mathrm{NCF}(x) + \beta \times \mathrm{SF}(x), & x \in [1, |N^s|] \\ \alpha \times \mathrm{ECF}(x) + \beta \times \mathrm{SF}(x), & x \in [1, |E^s|] \end{cases}, \quad (11)$$

where α and β are weighting factors that are used to adjust the relative importance between the *connectivity factor* and the *saturation factor*, and $\alpha + \beta = 1$. Here, the management cost can be expressed as follows:

$$C = \sum_{n^s \in [1,|N^s|]} \psi(n^s) C(n^s) + \sum_{e^s \in [1,|E^s|]} \psi(e^s) C(e^s), \quad (12)$$

where $C(e^s)$ and $C(n^s)$ represent the initial resource consumption of the service node and edge, respectively. Note that the higher the RSF value, the higher the cost of the mapping. Once the mapping has been defined, the total management cost can be decomposed into two parts: the primary cost and the backup cost. The total mapping cost can then be computed as follows:

$$\begin{aligned} C_t = {} & \mu_1 \sum_{f_n \in [1,|F|]} C(f_n) x_k^{f_n} + \mu_2 \sum_{m \in [1,|E^{\mathrm{sub}}|]} C(m) y_{ij}^m \\ & + \mu_3 \sum_{f^b \in [1,|F^b|]} C(f_b) x_k^{f_b} + \mu_4 \sum_{m' \in [1,|E^b|]} C(m') y_{f_b}^{m'}, \quad (13) \\ & \forall k \in [1, |N^s|], i \in [1, |S|], j \in [1, |F|], \end{aligned}$$

where μ_1, μ_2, μ_3, and μ_4 are weighting factors that are used to adjust the relative importance of these cost components.

3.3 | MILP formulation

Next, the problem of reliability-aware SFC mapping in a dynamic NFV-enabled scenario is considered and $x_k^{f_n}$ ($f_n \in [1, |F|]$, $k \in [1, |N^s|]$) is introduced to indicate which of the VNFs in the set f_n can be provisioned on a physical node k. That is:

$$x_k^{f_n} = \begin{cases} 1 & \text{a primary VNF } f_n \in [1, |F|] \text{is provisioned on} \\ & k \in \left[1, \left|N^{\mathrm{sub}}\right|\right] \\ 0 & \text{otherwise.} \end{cases} \quad (14)$$

A binary variable y_{ij}^m ($i \in [1, |S|]$, $j \in [1, |F|]$, $m \in [1, |E^{\mathrm{sub}}|]$) is now introduced to formulate the routing of the primary link.

$$y_{ij}^m = \begin{cases} 1 & \text{a primary link } m \in \left[1, \left|E^{\mathrm{sub}}\right|\right] \\ & \text{is selected by function } j \text{ of service } i \\ 0 & \text{otherwise.} \end{cases} \quad (15)$$

Also, a binary variable $x_k^{f_b}$ ($f_b \in [1, |F^b|]$, $k \in [1, |N^s|]$) is defined to represent the backup VNF hosted on the physical node $k \in B_N$. That is:

$$x_k^{f_b} = \begin{cases} 1 & \text{a backup VNF } f_b \in \left[1, \left|F^b\right|\right] \\ & \text{is hosted on } k \in \left[1, \left|B_N\right|\right] \\ 0 & \text{otherwise.} \end{cases} \quad (16)$$

A binary variable $y_{f_b}^{m'}$ ($f_b \in [1, |F^b|]$, $m' \in [1, |E^b|]$) is introduced to formulate the routing of the backup link.

$$y_{f_b}^{m'} = \begin{cases} 1 & \text{a backup link } m \in \left[1, \left|E^b\right|\right] \\ & \text{is selected by } f_b \in \left[1, \left|F^b\right|\right] \\ 0 & \text{otherwise.} \end{cases} \quad (17)$$

Now, define F_{ij}^b to be the number of instances of the jth VNF of network service i that have been provisioned to protect the corresponding chain (the calculation of F_{ij}^b is detailed in the next section). These VNF instances can be instantiated and mapped along the service path. A binary variable h_{ij} ($i \in [1, |S|]$, $j \in [1, |F|]$) is introduced to distinguish between no-backup and backup placements. That is:

$$h_{ij} = \begin{cases} 1 & j\text{th function of service } i \text{ is protected by backups} \\ 0 & \text{otherwise.} \end{cases} \quad (18)$$

When an online service chaining request arrives with a specific reliability requirement, the objective of the system operator is to derive an optimal resource assignment strategy in the dynamic NFV-enabled scenario that minimizes the cost while maximizing the reliability. Note that these two criteria are in conflict: the more the backup VNFs, the less the risk of service unreliability; however, the higher the cost of the mapping. Because it is hard to optimize for both criteria simultaneously, a scalarization method is applied to transform this problem into a single-objective one. In this case, the objective can be mathematically described as:

$$\text{maximize } (\gamma_U U_t - \gamma_C C_t). \quad (19)$$

Note that the intent of the normalization introduced in the above expression is to minimize the management cost while maximizing the improvement ratio. The relative importance of the two criteria is dictated by a specific policy as encoded in a pair of weights γ_U and γ_C that represent the reliability and cost, respectively. In addition, the terms U_t and C_t are transformed into non-dimensional forms.

The constraints of the optimization model are as follows.

3.3.1 | Site capacity constraint

The total load of the CPU resource across all service chaining requests and primary/backup VNFs at each physical node $k \in N^s$ should be less than or equal to its CPU capacity. This can be expressed as follows:

$$\sum_{f_n \in [1,|F|]} c(f_n)x_k^{f_n} + \sum_{f^b \in [1,|F^b|]} c(f_b)x_k^{f_b} \le M_N(k), \forall k \in [1,|N^s|]. \tag{20}$$

This enforces a limit on the CPU resource capacity of the node, where the sum of $c(f_n)x_k^{f_n}$ represents the total CPU usage by all primary VNFs and the sum of $c(f_b)x_k^{f_b}$ represents the total CPU usage by all backup VNFs.

3.3.2 | Link capacity constraint

The total required bandwidth across all service chain requests and primary/backup logical links at each physical link $m \in E^s$ should be less than or equal to its bandwidth capacity. This can be represented as follows:

$$\sum_{m \in [1,|E^{\text{sub}}|]} bw(m)y_{ij}^m + \sum_{m' \in [1,|E^b|]} bw(m')y_{f_b}^{m'} \le M_L(m), \quad \forall k \in [1,|N^s|], i \in [1,|S|], j \in [1,|F|], f_b \in \left[1,\left|F^b\right|\right]. \tag{21}$$

This enforces a limit on the bandwidth capacity of a link, where the sum of $bw(m)y_{ij}^m$ represents the total bandwidth used by the primary logical links and the sum of $bw(m')y_{f_b}^{m'}$ represents the total bandwidth used by the backup logical links.

3.3.3 | Reliability constraint

Note that the maximum reliability that a provisioned VNF chain can achieve is determined by the number of redundant VNFs F_{ij}^b. Each primary function $f_n \in F$ may have one or more redundant functions (of the same type) provisioned to restore the service (of the chain) when the primary function fails. In order to obtain a feasible solution with reliability guarantee Θ_{req}^t, it is necessary to determine the optimal number of F_{ij}^b. aaaTherefore, a reinforcement learning model that exploits VNF redundancy is adopted and employed hereinafter. The optimal number of redundant VNFs can be obtained via an iterative method. To achieve the reliability requirements of a network service, the following constraint is introduced:

$$\sum_{k \in [1,|N^s|]} y_k^{f_n} \ge F_{ij}^b, \quad \forall i \in [1,|S|], j \in [1,|F|]. \tag{22}$$

This ensures there are sufficient backup VNFs F_{ij}^b to achieve the required reliability of t. An iterative method to determine the value of F_{ij}^b is presented in Section 4.2.

3.3.4 | Connectivity constraint

As it is important to enforce the right order for traversing the routing between primary and backup VNFs, a connectivity constraint based on the flow balance criteria is defined to ensure the incoming flow is equivalent to the outgoing flow at physical nodes, except the ingress and egress. This can be expressed as follows:

$$\sum_{m.\text{node}=k_1} y_{ij}^m - \sum_{m'.\text{node}=k_2} y_{ij}^{m'}(1-h_{ij}) - \sum_{m''.\text{node}=k_3} y_{f_b}^{m''}h_{ij} = 0 \quad \forall k_1,k_2,k_3 \in [1,|N^s|], i \in [1,|S|], j \in [1,|F|], \forall m,m' \in \left[1,\left|E^b\right|\right], m'' \in \left[1,\left|E^{\text{sub}}\right|\right], \tag{23}$$

$$\sum_{m \in [1,|E^{\text{sub}}|]} \sum_{m' \in [1,|E^b|]} y_{ij}^m y_{f_b}^{m'} = 0, \quad \forall i \in [1,|S|], j \in [1,|F|], f_b \in \left[1,\left|F^b\right|\right]. \tag{24}$$

The first ensures that, for an arbitrary physical node, the flow balance must be satisfied if the jth VNF of service i is protected by the service path, while the second ensures the primary and backup paths do not coincide with each other.

3.3.5 | Delay constraint

The overall end-to-end delay is composed of two components: (i) the processing and transmission delay resulting from VNF mapping without backups instantiated, in other words, $h_{ij} = 0$ and $D_{ij}^s > 0$; and (ii) the processing and transmission delay resulting from VNF mapping with backups instantiated, in other words, $h_{ij} = 1$ and $D_{ij}^b > 0$. This can be represented as follows:

$$\sum_{j \in [1,|F|]} \left[(1-h_{ij})D_{ij}^s + h_{ij}D_{ij}^b\right] \le \delta_t, \ \forall i \in [1,|S|], t \in [1,|\Gamma|], \tag{25}$$

$$D_{ij}^b = \sum_{m \in [1,|E^{\text{sub}}|]} y_{ij}^m \mu_m + \sum_{m' \in [1,|E^b|\ ]} \sum_{f_b \in [1,|F^b|]} y_{f_b}^{m'} x_k^{f_b} \mu_{m'} - \sum_{m'' \in [1,|E^{\text{sub}}|]} \sum_{f_n \in [1,|F|]} y_{f_b}^{m''} x_k^{f_n} \mu_{m''}, \quad \forall k \in [1,|N^s|], \ \forall i \in [1,|S|], j \in [1,|F|], \tag{26}$$

$$D_{ij}^s = \sum_{m \in [1,|E^{\text{sub}}|]} y_{ij}^m \mu_m, \ \forall i \in [1,|S|], j \in [1,|F|], \tag{27}$$

where (25) ensures the end-to-end delay of each service chain should be less than or equal to δ_t. There are two parts to (26): the delay experienced by the backup VNF instances and traffic steering through the rest of the service chain. In (26), the sum of $y_{ij}^m \mu_m$ and $y_{f_b}^{m'} x_k^{f_b} \mu_{m'}$ indicates the total processing and transmission delay along the service path. However, this value includes any overlapping delay from the shared backup VNFs. Thus, the corresponding delay of the shared instance of every backup VNF is subtracted. The processing and transmission delay without backups instantiated are represented in (27).

3.3.6 | Placement constraint

The primary/backup VNF placement constraints can be formulated as follows:

$$x_k^{f_n} \le \sum_{m.\text{node}=k} y_{ij}^{m}, \tag{28}$$

$$\begin{aligned} &x_k^{f_b} \le \sum_{m'.\text{node}=k} y_{ij}^{m'}, \\ &\forall i \in [1, |S|], j \in [1, |F|], k \in [1, |N^s|], f_b \in \left[1, \left|F^b\right|\right], \\ &m \in \left[1, \left|E^b\right|\right], m' \in \left[1, \left|E^{\text{sub}}\right|\right] \end{aligned} \tag{29}$$

where (28) and (29) represent the relationship between the primary/backup VNF placement and the routing variables.

4 | ALGORITHM DESIGN

In this section, a novel online algorithm is proposed to provide off-site redundancy for reliability-aware wide area service chaining in a dynamic NFV-enabled scenario. The objective is to determine an efficient mapping strategy for each service chain request while satisfying the constraints. The metric of interest is the service chaining acceptance ratio, which is defined by the number of accepted service chaining requests over the total number of service chaining requests, and the redundancy model is the JP model. Based on the complexity of the above-presented MILP, the problem of finding the optimal mapping plan for one service chaining request is NP-Hard [21]. To address this, a novel online algorithm called *Balancing between Cost and Reliability* (BCR) is proposed that employs a JP model and a novel backup selection mechanism to maximum the reliability of each client while simultaneously minimizing the amount of resources allocated for each online request. There are two main steps in the BCR algorithm:

i. For primary mapping, find K-shortest paths between ingress and egress and sort these paths in descending order based on the communication delay. Then, map primary VNFs to the sites along the selected path.
ii. For backup mapping, model the backup selection process as a *Markov Decision Process* (MDP). For each primary VNF encountered along a particular service chaining, determine whether to provision that VNF with backups and allocate the appropriate amount of backup resources for each primary VNF. Then, find the optimal number of backup VNFs by running the improved Q-learning algorithm. Finally, find sites at which to locate each of the redundant VNF instances.

As the primary function of the BCR algorithm is to determine an optimal backup plan, the second step in the BCR algorithm is discussed in more detail below. First, the MDP is applied to model the backup selection process (Section 4.1). Then, an improved Q-learning algorithm is proposed to obtain the optimal solution for the MDP model. (Section 4.2). Finally, a formulation of the BCR algorithm is introduced (Section 4.3).

4.1 | Backup selection modeling

Han et al. [22] reported that the MDP model achieves good performance in making optimal decisions in a dynamic environment. Therefore, the backup VNFs selection process is considered as the MDP model. Here, it is assumed that the backup selection process can be defined as a five-tuple $\{S, A, r, P, J\}$, where S represents the finite state space (the basic events in state space S are shown in (30)) and $X(t)$ represents the primary VNF placement state at time t. The binary variable X_{ij} ($i \in [1, m]$, $j \in [1, n]$) is introduced to formulate the site assignment of the primary VNFs ($X_{ij} = 1$ indicates that n_i^s is the location of f_j; otherwise n_i^s is not the location of f_j), A represents the finite action space, and each action vector $a \in F_{ij}^b = \{f_b^1, f_b^2, \ldots, f_b^n\}$ represents the number of backup instances of the jth VNF of network service i that has been provisioned to protect the corresponding chaining. To ensure synchronization between the primary and backup VNFs, it is necessary for $F \bigcap F_{ij}^b = F_{ij}^b$ and $F \bigcup F_{ij}^b = F$. In other words, the backup VNFs F_{ij}^b must be a subset of the primary VNFs F and r represents the set of rewards corresponding to each vector in the action space. If each pair (s, a) satisfies the constraints of the MILP model described above, the instant reward $r(s, a) = \gamma_U U_t - \gamma_C C_t$ can be obtained from action a executed on state s; otherwise, the instant reward can be defined as the penalty factor $r = -1/\xi$, where ξ represents an infinitesimal positive real number. Here, P represents the state transition probability and J represents the total revenue donated by $\sum_{t=0}^{T} r(t)$. The objective is to select the optimal action in the MDP that yields the maximum revenue in the processing time.

$$X(t) = \begin{pmatrix} X_{11} & \cdots & X_{1n} \\ & \ddots & \\ \vdots & X_{ij} & \vdots \\ & & \ddots \\ X_{m1} & \cdots & X_{mn} \end{pmatrix}, \quad X(t) \in S. \tag{30}$$

4.2 | Find the near-optimal solution

Wang et al. [23] and Zoph et al. [24] demonstrated that the Q-learning algorithm is an effective way to solve the MDP model. As was mentioned earlier, the BCR algorithm was

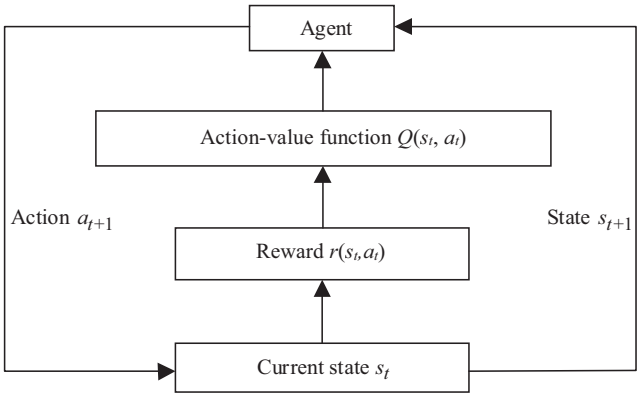

FIGURE 2 Reinforcement learning model

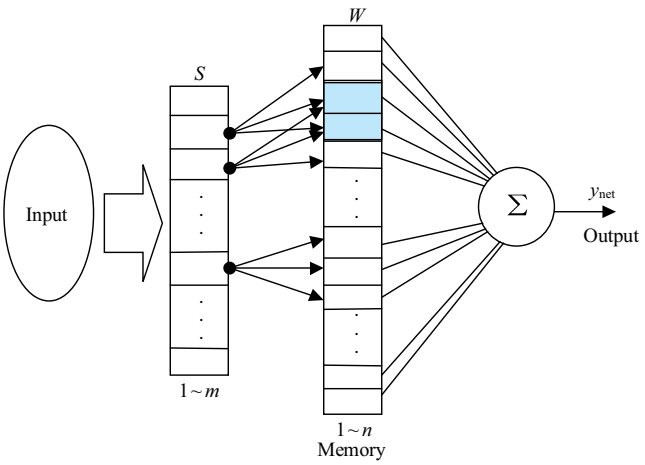

FIGURE 3 Structure of a cerebellar model articulation controller neural network

proposed based on the MDP model to obtain the optimal mapping solution.

During operation, the Q-learning algorithm interacts with the surrounding environment. At first, the control system selects an action, and then the surrounding environment gives proper feedback to the agent in the form of rewards or punishments. The instant reward is used to evaluate the pros and cons of the selected action, until the maximum reward has been achieved. Finally, the agent arrives at the optimal solution via trial and error. The Q-learning model is a closed-loop feedback control system, as depicted in Figure 2.

At time t, the agent reaches the state s_{t+1} after executing action a_t on state s_t. The term $r(s_t, a_t)$ can be calculated from the improvement ratio and management cost, then $Q(s, a)$ can be updated accordingly. At time $t + 1$, the above process is repeatedly iterated to obtain both the optimal action-value $Q^*(s_t, a_t)$ and the optimal backup plan $\pi^*(s_t) = \text{argmax } Q^*(s_t, a_t)$. In each iteration, the gradient descent method is applied to estimate $Q(s, a)$. Thus, the action-value updating rule can be computed as shown in (31), as long as the optimal action-value $Q^*(s_t, a_t)$ satisfy the Bellman's equation, as shown in (32).

$$Q(s_t, a_t) = Q(s_t, a_t) + [\alpha_Q r(s_t, a_t) + \gamma \max_{a_{t+1}} Q(s_{t+1}, a_{t+1}) - Q(s_t, a_t)], \quad (31)$$

$$Q^*(s_t, a_t) = E[r(s_t, a_t) + \max Q^*(s_{t+1}, a_{t+1})], \quad (32)$$

where $\alpha_Q \in (0, 1)$ represents the learning factor and $\gamma \in (0, 1)$ represents the penalty factor.

As the current network is composed of hundreds of physical servers, the Q-learning algorithm will likely encounter the "Curse of Dimensionality" problem due to the complicated datacenter network topology [25]. This problem arises as follows. In each iteration, the size of $Q(s, a)$ is $|S| \cdot |A|$, and as the learning cycle increases, $Q(s, a)$ comes to occupy a large number of storage resources, which prevents the learning process from finishing. To solve this problem, a cerebellar model articulation controller (CMAC) neural network is applied to achieve fast convergence of $Q(s, a)$.

Note that in the backup selection process, there are an insufficient number of training samples and limited online trial results. A benefit of the CMAC neural network is that it can be easily embedded into the controller module, which allows it to accelerate the generalization of the online learning. The basic structure of a CMAC neural network is shown in Figure 3. This type of neural network consists of three parts, namely, the input, memory, and output, where the input space S represents m-discrete state spaces, the memory layer W represents the storage weight value of n-discrete memory addresses, and y_{net} represents the output of the network. The actual state can be mapped onto the state space S and each discrete state s_i can be mapped onto the weights stored by multiple physical addresses in the memory layer W. If the different states are in close proximity, the physical addresses overlap (represented by the shadow in Figure 3); otherwise, the physical address do not overlap.

Harmon et al. [26] reported that a CMAC neural network achieves good performance in both the convergence speed and computational complexity. To solve the "Curse of Dimensionality" problem, the basic Q-learning algorithm is extended by the addition of a CMAC neural network, as shown in Figure 4. At time t, the CMAC-Q agent obtains the instant reward $r(s_t, a_t)$ after executing action a_t on state s_t. The CMAC neural network can fit $Q(s, a)$ via a table query based on both the instant reward $r(s_t, a_t)$ and the input (s_t, a_t), and then adjusts the weight vectors. With this process, the learning system is no longer required to store and update the estimation table of size $|S| \cdot |A|$, instead, it only needs to store the weights of the local neurons in the network. The CMAC neural network is employed for estimating the $Q(s, a)$ of each action. When $s_t = s_i$, the

estimated action-value function that executes the kth action is Q_k. The gradient descent method is also applied to update the local weights and obtain the minimum value of the output error. The update rule for the improved Q-learning system is shown in (33):

$$\begin{cases} Q_k = W_k^T F_{ki} = \sum_{j=1}^{n} \omega_{ki} f_{ij} \\ W_{k(t+1)} = W_{kt} + \alpha_W [r(s_t, a_t) + \gamma \max_{a_{t+1}} Q_{NN}(s_{t+1}, a_{t+1}) \\ \qquad - Q_{NN}(s_t, a_t)] \cdot F_{ki}, \end{cases} \tag{33}$$

where W_{kt} represents the weight, F_{ki} represents the activation state, and α_W represents the weight learning rate.

4.3 | Formulation of the BCR algorithm

The proposed availability-aware SCM algorithm, or the BCR algorithm is shown in Table 1.

The intent of this algorithm is to overcome the complexity of the formulated MILP. Here, it is assumed that a network supports a constant number of network services. Let the number of physical nodes and physical edges be m and n, respectively. In the first part of the BCR algorithm, the primary mapping is based on the *K-Dijkstra* algorithm [27], and it is well-known that this algorithm has a certain complexity of order $O(k(m\log m + n))$. The second part of the proposed algorithm is based on the improved Q-learning algorithm, the complexity of which is primarily caused by updating the weight vectors rather than storing samples. For an arbitrary sample, the corresponding weight vector must be updated as per (28). Here, it is assumed that the number of discrete actions is M and the backup selection has a complexity of $O(M)$. In the process of finding the maximum $Q^*(s_t, a_t)$, the required number of updates is N and there are kT samples for each primary service path, where T represents the maximum learning step size. This

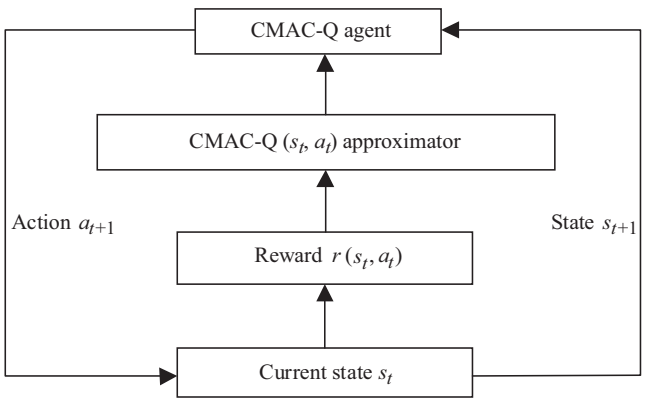

FIGURE 4 Cerebellar model articulation controller-Q-learning model

TABLE 1 The process flow of the balancing between cost and reliability algorithm

Input: Service Request t and Physical Network G^s
Output: Primary Mapping f and Backup Mapping f_b
for each pair of ingress $n_i \in N^s$ and egress $n_e \in N^s$ **do**
  The *K-Dijkstra* algorithm is used to calculate the k-shortest path sets from n_i to n_e, denoted $P = \{p_1, p_2, \ldots, p_k\}$
  Sort P in descending order based on their respective delays, denoted E
end for
for each service chaining request t **do**
  for each service path p_i **do**
    Map all VNFs $f_i \in F$ to the data centers along the path $e \in E^{\text{sub}}$ subject to the function constraints
  if primary mapping succeeds **then**
    Calculate R_t based on (7)
    if $R_t \geq \Theta_{\text{req}}^t$ **then**
    There is no need to provide backup resources
    else
    Initialize state space S, behavioral space A, and transition probability P according to the primary mapping
    Ensure the synchronization between primary VNFs and backup VNFs, $F \bigcap F_{ij}^b = F_{ij}^b$, $F \bigcup F_{ij}^b = F$
    $t \leftarrow 0$, $s \leftarrow s_0$
    while ($|Q_{NN}(s_t, a_t) - \max Q(s_t, a_t)| \geq \xi \&\& t \leq Max\_t$) **do**
      for each action $a \in F_{ij}^b$ in state space S **do**
        $step \leftarrow 0$
        Calculate $Q_{NN}(s_t, a_t)$
        while ($step$!= Max_t) **do**
          $dataset \leftarrow \{s_t, a_t, Q_{NN}(s_t, a_t)\}$
          Execute action a_t to obtain $r(s_t, a_t)$, and enter state s_{t+1}
          $step \leftarrow step + 1$
          if $rand < 1 - \xi$ **then**
            $a^* = \text{argmax}\ Q_{NN}(s_{t+1}, a_{t+1})$
            $\delta = r(s_t, a_t) + Q_{NN}(s_{t+1}, a_{t+1}) - Q_{NN}(s_t, a_t)$
          else
            Select a_t based on π
            $Q(s_t, a_t) = Q_{NN}(s_{t+1}, a_{t+1}) + \alpha_W \delta$
            Update $dataset$
          end if
          CMAC neural network starts training
          Update weight vector W
        end while
        $t \leftarrow t + 1$
        Update learning factor α_W

(Continues)

35 **end for**
36 **end while**
37 **end for**
38 **end for**
39 **if** $F_{ij}^{b} \neq \{\varnothing\}$ **then**
40 Determine the backup plan based on the learning output results
$B_N \leftarrow F_{ij}^{b}$, select the optimal number of backup VNFs based on $\pi^*(s_t) = \text{argmax}\ Q^*(s_t, a_t)$
$B_E \leftarrow m$, calculate the backup logical links based on $\max Q^{\pi}(s_t, a_t)$
41 Use the JP protection model for backup, and map the backup VNFs and links subject to proposed constraints (15) to (21), thus forming the backup path from the ingress $n_i \in N^s$ to the egress $n_e \in N^s$ that minimizes the cost while maximizing the reliability
42 Update the topology
43 **end if**

process is executed for k cycles, which runs in $O(Mk^2TN)$ time. In summary, the complexity is $O(k(m\log m + n) + Mk^2TN)$ when processing a single online service request.

5 | PERFORMANCE EVALUATION

The performance evaluation focused on the following aspects: (i) compare the results of the BCR algorithm to the results of the MILP based solution via the CPLEX tool; (ii) analyze the performance of the BCR algorithm on the Abilene backbone network topology as per the SNDlib library. Compare the performance of the proposed solution to that of the READ and GREP based optimal solutions, which have already been studied in the literature for reliability-aware SCM problems. The proposed algorithm is evaluated in terms of the service chaining request acceptance ratio, the backup resources consumed by the requests, and the algorithm run time. Furthermore, all of the simulations were executed on a personal computer with a 3.6 GHz dual core Intel® CoreTM i7 processor and 8 GB RAM. In order to make the results more accurate, each simulation was run 100 times and the results were averaged.

First, the results of the proposed algorithm are compared to the results of the MILP based solution via the CPLEX tool used to solve the MILP model. In the simulation, three typical physical networks were generated: (i) a small network consisting of 10 nodes and five hosted network services, (ii) a medium network consisting of 20 nodes and 10 hosted network services, and (iii) a large network consisting of 40 nodes and 20 hosted services. The detailed results of this comparison are presented in Table 2.

TABLE 2 MILP vs. BCR

Physical topology	Solution	Objective value	Execution time
10 nodes, 5 services	MILP	0.25	3.2 s
	BCR	0.20	1.6 s
20 nodes, 10 services	MILP	0.39	13,279 s
	BCR	0.31	1.8 s
40 nodes, 20 services	MILP	N/A	∞
	BCR	0.48	2.5 s

BCR, balancing between cost and reliability; MILP, mixed integer linear programming.

As indicated in the table, the MILP approach obtained an objective value when the BCR algorithm could not. This was because the typical CPLEX tool used to solve the MILP provided an exact solution, albeit at the expense of execution time. As the number of physical node increased, the MILP algorithm fails to obtain the objective value. Therefore, this algorithm had limited applicability due to its computational complexity, especially in the cases with a medium or large network. In contrast, the proposed BCR algorithm iteratively approached a near-optimal solution within a polynomial time and therefore satisfied the execution time requirement in the large network.

To better evaluate the performance of the BCR algorithm, it was compared to that of the READ and GREP based optimal solutions, which were previously studied in the literature for the reliability-aware SCM problem. The GT-ITM tool [28] was used to generate different network topologies and the BCR algorithm was implemented in MATLAB. The Abilene backbone network topology provided by the SNDlib library [29] was used for the topology dataset (12 nodes, 15 links). As shown in Figure 5, Node 12 was used for egress and the remaining 11 nodes were used for ingress. In the physical network, it was assumed that each node can be used as a service-providing node.

The selected simulation parameters and the corresponding distribution were motivated by simulations and evaluations of a well-known related reliability-aware SCM problem [14]. For the physical network, the delay between the ingress/egress and their associated datacenter site was assumed to be randomly in the range of (0, 1] ms. Each node of the network represents one single data center with a physical resource capacity, such as CPU, memory, and storage, between [1,500, 2,500]. If there are 10 types of functions in the network, each physical node can provide one to five functions. The mapping cost of each node and link was fixed to 10 and 5, respectively. The reliability of each mapped VNF was randomly distributed within [0.9, 0.99]. Each service chaining request consists of different types of functions, the number of which was randomly

distributed in range [2, 5]. The instantiated resources required by each VNF were randomly distributed within [20, 40] and the bandwidth required by the logical links was randomly distributed within [5, 10]. The reliability requirement was selected among {95%, 96%, 97%, 98%, 99%, and 99.9%}, which is similar to those used by Google Apps [30]. The survival time of each request followed the exponential distribution and the mean number of life cycles was 500. In the Q-learning algorithm, the penalty factor ξ was fixed to 0.01, the maximum learning training period *Max_t* was fixed to 1,500, the maximum learning step size *Max_step* was fixed to 1,000, and the initial value of the learning factor α_0 was fixed to 0.8.

To analyze the convergence speed of the improved Q-learning algorithm, the dynamic exploration strategy was compared to the static exploration strategy. In this test, the controller system must be continuously trained online to improve the distribution of backup resources and the action selection strategy was adopted based on the Boltzmann distribution, in other words, a balance between greedy search and free search. Thus, the mathematical description of the action selection strategy π is:

$$p(a_t|s_t) = \frac{e^{Q(s_t,a_t)/T_{\text{temp}}}}{\sum_{a\in A} e^{Q(s_t,a)/T_{\text{temp}}}}, \tag{34}$$

where $p(a_t|s_t)$ represents the transition probability of action a_t executed on the state s_t, $A = \{a_k|k = 1, 2, \ldots\}$ represents the optional action set and T_{temp} represents the exploration factor. The improved Q-learning algorithm gradually reduces the exploration factor and ultimately tends toward a greedy search.

The relationship between the cycles and steps in each iteration are shown in Figure 6, where it can be seen that the curve of the static exploration strategy fluctuates, thus providing larger errors and output weights to the CMAC neural network. However, the dynamic exploration strategy is smoother. This is because the learning system gradually reduces the exploration factor, which ensures a higher exploration rate in the early stages until finally tends toward a greedy search. In this way, the algorithm avoids falling into a local optimal solution and changes the inertia thinking mode of the agent, thus improving the convergence and generalization ability of the Q-learning system.

A synthetic policy was employed to evaluate the proposed algorithm in terms of the service chaining request acceptance ratio, the backup resources consumed by requests, and the running time. The proposed algorithm was compared to three other mapping algorithms that have been widely used in the literature. The notations and descriptions of the different algorithms are listed in Table 3. The results of these simulations are shown in Figures 7–9. The results are presented in the following sections.

5.1 | Metric 1: Run time

This is the time required to find primary and backup topology mapping for a given traffic batch and network topology.

The run time with service arrivals for different backup algorithms is shown in Figure 7. Toward the end of the simulation (at about 100 service requests), the run time of the No-backup algorithm was 212.1 ms, the running time of the READ algorithm was 837.1 ms, the running time of the GREP algorithm was 678.5 ms, and the running time of the BCR algorithm was 411.5 ms. The results show that the run time of READ was about four times longer than that of the No-backup. This is because the No-backup only executes the primary topology mapping, and the run time is therefore minimal according to the order of $O(m\lambda + m^2)$ (for details, refer to Bari et al. [31]), where λ represents the length of the service chain. However, the three other backup algorithms implement both the primary and backup topology mappings, thus requiring more time compared to

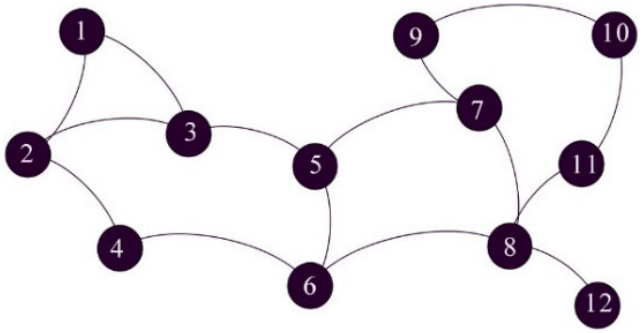

FIGURE 5 Abilene backbone network topology

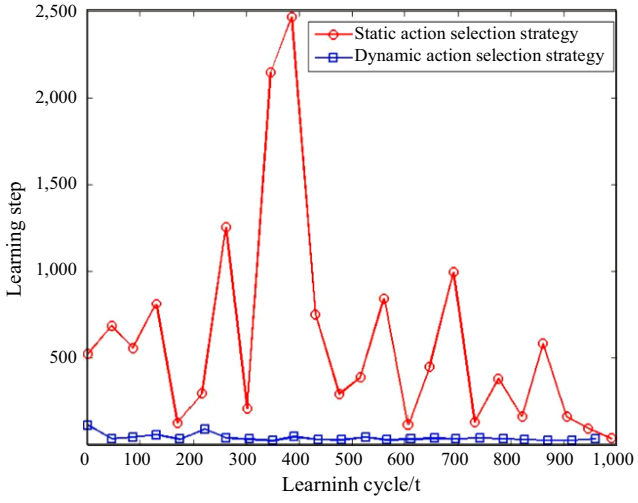

FIGURE 6 Comparison of the static and dynamic action selection strategies

TABLE 3 Comparison of algorithms

Algorithm	Description
READ [11]	Reliability-aware and delay constrained optimization framework
GREP [13]	Guaranteeing reliability with enhanced protection in network function virtualization
No-backup [31]	Reliability-agnostic SFC mapping scheme.
BCR	Balancing between cost and reliability

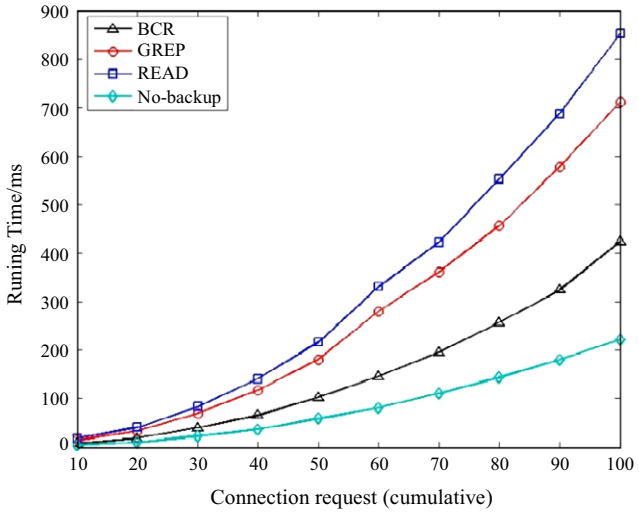

FIGURE 7 Comparison of running time

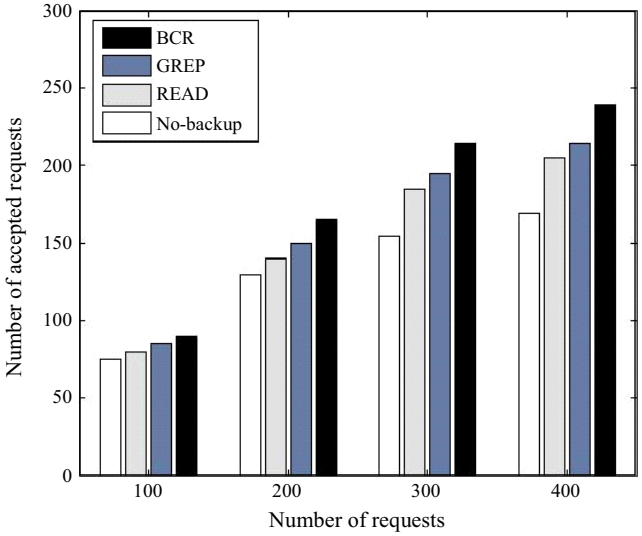

FIGURE 8 Number of accepted requests (reliability = 0.95)

the No-backup algorithm. Among these, GREP selects two primary VNF nodes with the lowest reliability in each iteration, and the primary VNF that prioritizes the backup

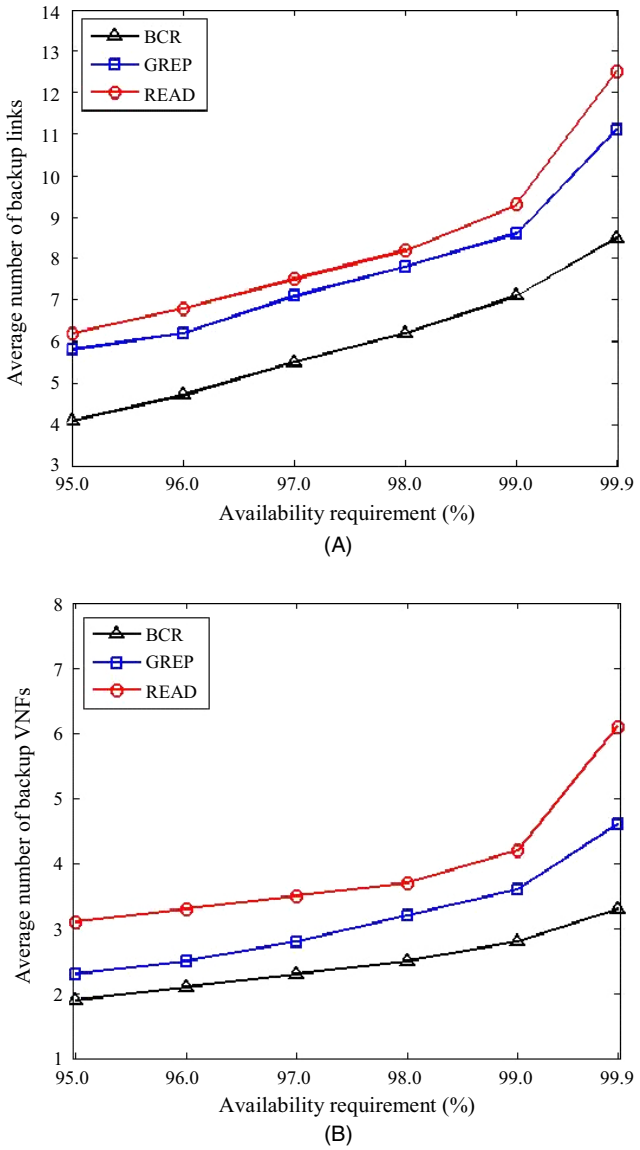

FIGURE 9 (A) Average number of backup links and (B) average number of backup virtualized network functions

must wait for the next primary VNF to complete the backup operation in the cache queue. Therefore, the complexity of GREP is $O(m\lambda)$ (for details, refer to Fan et al. [13]). READ can be divided into two steps, namely, the generation of an initial set of paths and the expansion of single-path routes to multi-path routes together with their corresponding VNF placement. Thus, the complexity of READ is $O(kn(m\log m + n))$ (for details, refer to Long et al. [11]). Finally, BCR searches for the optimal backup plan via updating $Q(s, a)$ to the direction of the total revenue J, thus avoiding the need to blindly traverse the entire state space. The dynamic exploration strategy was also adopted to select the action, thus speeding up the convergence of $Q(s, a)$. Combined with the analysis presented in Section 4.3, the curve of each algorithm in Figure 7 is consistent with those of the theoretical analysis.

5.2 | Metric 2: Request acceptance ratio

This is the ratio of the number of service chains successfully deployed on the physical network over the total number of service chaining requests.

Figure 8 shows the variation of the request acceptance ratio with service arrivals and Table 4 shows the mean rejected number of requests that were blocked due to reliability and capacity constraints, respectively. As the number of arriving requests increases, the amount of available resources gradually decreases, which may cause the rejection of more requests. The mean request acceptance ratio of the No-backup algorithm was 58.56%, the mean request acceptance ratio of the READ algorithm was 65.56%, the mean request acceptance ratio of the GREP algorithm was 69.71%, and the request acceptance ratio of the BCR algorithm was 75.25%. In the experiment, a request can be accepted if and only if there are enough resources and the reliability requirement can be satisfied. No-backup does not consume backup resources and the number of remaining physical resources, such as CPU and bandwidth, with this algorithm is higher than three other algorithms. However, it is hard to satisfy the reliability constraints with the No-backup algorithm, which could possibly lead to the rejection of service requests. READ investigates the feasibility of adequately embedding each of these VNFs and selectively provisioning some of them with an adequate number of backup VNFs that will allow those services to achieve their respective reliability requirements $R_t = 0.95$. However, READ does not include CPU capacity constraints, thus resulting in more backup resources being wasted and a lower request acceptance ratio. The performance of GREP was only slightly worse than BCR, which is because GREP uses a greedy strategy to select the two least reliable VNFs for redundancy in each iteration, and can thus satisfy the reliability requirements while consuming fewer resources, thereby improving the request acceptance ratio compared to READ. However, GREP is based on the greedy algorithm, which easily obtains a local optimal solution. Finally, it can be seen that BCR has the highest request acceptance ratio, which is because BCR has a chance to iteratively obtain the optimal number of backup VNFs and considers the potential remaining resources released after the deployed request completes. This makes the best use of global resources and can lead to receiving more SFC requests compared to READ and GREP. Based on the above discussion, the results shown in Table 4 are consistent the theoretical analysis.

TABLE 4 Mean rejected number of requests

Algorithm	Reliability	Capacity
BCR	10	52
GREP	15	61
READ	21	65
No-backup	92	12

5.3 | Metric 3: Backup resource consumption

This is the number of backup VNFs and backup logical links. Here, backup links refer to the links connecting the backup and their associated primary VNFs.

The number of requests in this experiment was 400 and only accepted requests were considered. The average number of backup links and backup VNFs used for each reliability-aware SCM are shown in Figure 9A,B, where it can be seen that BCR used 32.1% and 22.7% fewer logical links compared to READ and GREP, respectively, when the reliability requirement was "three-nines" or 99.9%. Similar conclusions can be drawn when comparing the number of backup VNFs, as shown in Figure 9B. It was found that the BCR algorithm required 46.7% fewer backup VNFs compared to READ. As the reliability requirements increased, BCR saved more backup resources. This is because the BCR algorithm has advantages in the redundancy model and backup selection method as it adopts the JP model and determines the optimal number of backup VNFs from a global perspective. Therefore, fewer backup VNFs result in fewer logical links. It was also found that GREP requires fewer backup VNFs than READ because READ adopts the shared protection model while GREP adopts the JP model. Finally, GREP had worse performance than BCR because GREP is based on the greedy search, which can easily obtain the local optimal solution. Thus, it consumes more resources than BCR. When combined with the proof in Appendix A, the curve of each algorithm in Figure 9 is consistent with the theoretical analysis.

6 | CONCLUSION

Network function virtualization employs virtualization technologies to offer network-as-a-service capability through connected VNFs. Since telecom networks must be online at all times, it is essential to provide efficient and effective protection and resource allocation schemes to ensure the reliability of service chaining. In this paper, a novel online learning algorithm for reliability-aware SCM is provided that optimizes the conflicting objectives of management cost and service reliability without violating the capacity, delay, and reliability constraints. In addition, the Q-learning algorithm was improved in order to select the backup VNFs. The proposed BCR algorithm was evaluated by way of extensive simulation and the significant performance improvement was demonstrated in terms of the service

chaining request ratio, the backup resources consumed by requests, and the run time. In future work, the proposed mapping algorithm will be used to intelligently construct service chains.

ORCID

Yicen Liu https://orcid.org/0000-0002-7720-6854

REFERENCES

1. Cisco, *Cisco visual networking index: forecast and methodology, 2016-2021*, Sept. 2017.
2. D. Bhamare, R. Jain, and M. Samaka, *A survey on service function chaining*, J. Netw. Comput. Applicat. **75** (2016), no. 3, 138–155.
3. M. Mechtri, C. Ghribi, and D. Zeghlache, *A scalable algorithm for the placement of service function chains*, IEEE Trans. Netw. Service Manag. **13** (2016), no. 3, 533–546.
4. N. McKeown, T. Anderson, and H. Balakrishnan, *Openflow: enabling innovation in campus networks*, ACM SIGCOMM Comput. Commun. Rev. **38** (2008), no. 2, 69–75.
5. R. Mijumbi, J. Serrat, and J. L. Gorricho, *Management and orchestration challenges in network functions virtualization*, IEEE Commun. Mag. **54** (2016), no. 1, 98–105.
6. Y. Li and M. Chen, *Software-defined network function virtualization: a survey*, IEEE Access **3** (2017), 2542–2553.
7. S. Herker et al., Data-center architecture impacts on virtualized network functions service chain embedding with high availability requirements, *IEEE GLOBECOM Workshops*, San Diego, CA, Dec. 6–10, 2016, pp. 1–7.
8. S. Ayoubi, Y. Zhang, and C. Assi, RAS: reliable auto-scaling of virtual machines in multi-tenant cloud networks, *IEEE Int. Conf. Cloud Netw.*, Niagara Falls, Canada, Oct. 5–7, 2015, pp. 1–6.
9. R. Guerzoni et al., Modeling reliability requirements in coordinated node and link mapping, *IEEE Int. Symp. Reliable Distrib. Syst.*, Nara, Japan, Oct. 6–9, 2014, pp. 321–330.
10. J. Sherry et al., *Rollback-recovery for middleboxes*, ACM SIGCOMM Comput. Commun. Rev. **45** (2015), no. 4, 227–240.
11. Q. Long et al., Reliability-aware service provisioning in NFV-enabled enterprise datacenter networks, *Int. Conf. Netw. Service Manag.*, Montreal, Canada, Oct. 31-Nov. 4, 2016, pp. 153–159.
12. Q. Long, M. Khabbaz, and C. Assi, *Reliability-aware service chaining in carrier-grade softwarized networks*, IEEE J. Sel. Areas Commun. **36** (2018), no. 3, 558–579.
13. J. Fan et al., GREP: guaranteeing reliability with enhanced protection in NFV, *ACM SIGCOMM Workshop Hot Topics Middleboxes Netw. Function Virtualization*, London, UK, Aug. 21, 2015, pp. 13–18.
14. J. Fan et al., Availability-aware mapping of service function chains, *IEEE INFOCOM 2017 - IEEE Conf. Comput. Commun.*, Atlanta, GA, May 1–4, 2017, pp. 1–9.
15. F. Carpio and A. Jukan, *Improving reliability of service function chains with combined VNF migrations and replications*, arXiv:1711.08965, 2017.
16. P. Gill, N. Jain, and N. Nagappan, *Understanding network failures in data centers: measurement, analysis, and implications*, ACM SIGCOMM Comput. Commun. Rev. **41** (2011), no. 4, 350–361.
17. J. Liu et al., *Reliability evaluation for NFV deployment of future mobile broadband networks*, IEEE Wireless Commun. **23** (2016), no. 3, 90–96.
18. R. Potharaju and N. Jain, Demystifying the dark side of the middle: a field study of middlebox failures in datacenters, *Conf. Int. Measurement Conf.*, Barcelona, Spain, Oct. 23–25, 2013, pp. 9–22.
19. A. Hmaity et al., *Protection strategies for virtual network functions placement and service chains provisioning*, Netw. **70** (2017), no. 4, 373–387.
20. M. Scholler et al., Resilient deployment of virtual network functions, *Int. Congress Ultra Modern Telecommun. Contr. Syst. Workshops*, Almaty, Kazahstan, Sept. 10–13, 2013, pp. 208–214.
21. J. Kwisthout, *Most probable explanations in Bayesian networks: complexity and tractability*, Int. J. Approx. Reason. **52** (2011), no. 9, pp. 1452–1469.
22. Z. Han et al., Dynamic virtual machine management via approximate Markov decision process, *IEEE INFOCOM 2016 - IEEE Int. Conf. Comput. Commun.*, San Francisco, CA, Apr. 10–14, 2016, pp. 1–9.
23. S. Wang et al., *Deep reinforcement learning for dynamic multichannel access in wireless networks*, IEEE Trans. Cognitive Commun. Netw. **4** (2018), no. 2, 257–265.
24. B. Zoph and Q. V. Le, *Neural architecture search with reinforcement learning*, arXiv:1611.01578, 2016.
25. M. L. Puterman, *Markov decision processes: discrete stochastic dynamic programming*, John Wiley & Sons, Inc, New York, NY, 2005.
26. F. G. Harmon, A. A. Frank, and S. S. Joshi, *The control of a parallel hybrid-electric propulsion system for a small unmanned aerial vehicle using a CMAC neural network*, Neural Netw. **18** (2005), no. 5, 772–780.
27. M. Yu, Y. Yi, and J. Rexford, *Rethinking virtual network embedding: substrate support for path splitting and migration*, ACM SIGCOMM Comput. Commun. Rev. **38** (2008), no. 2, 17–29.
28. E. W. Zegura, K. L. Calvert, and S. Bhattacharjee, How to model an internetwork, *Proc. IEEE Infocom, Conf. Comput. Commun.*, San Francisco, CA, Mar. 24–28, 1996, pp. 589–594.
29. S. Orlowski et al., *SNDlib 1.0—survivable network design library*, Netw. **55** (2010), no. 3, pp. 276–286.
30. Google, Google apps service level agreement, Nov. 3, 2016. Available at http://www.google.com/apps/intl/en/terms/sla/html
31. M. F. Bari et al., On orchestrating virtual network functions, *Int. Conf. Netw. Service Manag.*, Barcelona, Spain, Nov. 9–13, 2015, pp. 50–56.

AUTHOR BIOGRAPHIES

Yicen Liu received his BS degree in communication and information systems from Ordnance Engineering College, Shijiazhuang, China, in 2016 and his MS degree in communication and information systems from Army Engineering University, Shijiazhuang, China, in 2018. He is currently pursuing his PhD in communication and information systems at Army Engineering University, Shijiazhuang, China. His current interest is software-defined service and network function virtualization.

Yu Lu received his PhD in information and communication engineering from the Beijing University of Aeronautics and Astronautics, Beijing, China, in 2011. He is currently a full professor of information and communication engineering. His research primarily focuses on future network architectures.

Wenxin Qiao received her BS degree in software engineering from the Beijing University of Posts and Telecommunications, Beijing, China, in 2014 and her MS degree in software engineering from Ordnance Engineering College, Shijiazhuang, China, in 2016. She is currently pursuing her PhD in information and communication engineering at Army Engineering University, Shijiazhuang, China. Her current interest is network virtualization technology.

Xingkai Chen received his PhD in communication and information systems from Army Engineering University, Shijiazhuang, China, in 2017. He is currently a lecturer at Army Engineering University, Shijiazhuang, China. His current interest is software-defined networks and network function virtualization.

APPENDIX A

In this appendix, it is proven that the JP method mentioned in Fan et al. [13] can provide higher reliability while consuming fewer resources compared to the shared protection model [11]. For simplicity, assume that one backup can provide at most two primaries. In the traditional shared protection model shown in Figure A1A, this approach saves resources by allocating $s_b = \max\{s_i, s_j\}$ resources on backup VNF_b to protect either VNF_i or VNF_j by connecting the backup to them. However, the network may break down when both primary VNFs protected by one backup fail. Therefore, the reliability of service chaining after adding the backup VNF can be characterized as:

$$R_{\text{SP}} = r_i r_j + r_b((1 - r_j)r_i + (1 - r_i)r_j). \tag{A1}$$

The JP model requires a backup VNF to reserve resources that are sufficient for all primary VNFs it protects. As shown in Figure A1B, the amount of backup resources reserved at VNF_b will not be sufficient for either VNF_i or $\text{VNF}_{j,}$ in other words, $s_b = s_i + s_j$. However, the service chain can still function even though both VNF_i and VNF_j fail simultaneously. Therefore, the reliability of service chaining after adding the backup VNF can be characterized as:

$$R_{\text{JP}} = 1 - (1 - r_b)(1 - r_i r_j). \tag{A2}$$

To compare the JP model to the shared protection model, ΔR can be defined as follows:

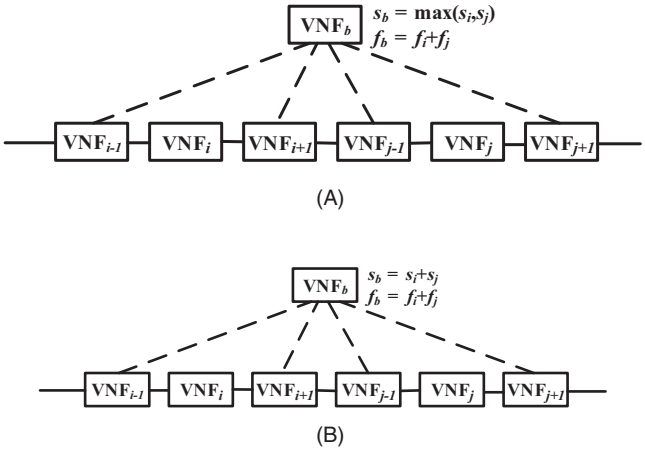

FIGURE A1 (A) Shared protection model and (B) Joint protection model

$$\Delta R = R_{\mathrm{JP}} - R_{\mathrm{SP}} = r_b(1 - r_i)(1 - r_j).$$

Then find the partial derivative of ΔR. This provides:

$$\frac{\partial \Delta R}{\partial r_i} = r_b(r_j - 1), \tag{A3}$$

$$\frac{\partial \Delta R}{\partial r_j} = r_b(r_i - 1). \tag{A4}$$

To obtain the minimum value of ΔR when $r_i = r_j = 1$, let the (A3) and (A4) be equal to zero. Thus:

$$\Delta R = R_{\mathrm{JP}} - R_{\mathrm{SP}} = r_b(1 - r_i)(1 - r_j) > 0. \tag{A5}$$

As shown, JP can provide higher reliability while consuming fewer resources compared to the shared protection model.

APPENDIX B

The symbols and corresponding definitions used in this paper are listed in Table B1.

TABLE B1 List of symbols and corresponding definitions

Symbol	Definition
S	The set of network services
F	The set of VNF types
N^s	The set of physical nodes
E^s	The set of physical links
N^v	The set of primary VNFs
E^v	The set of primary logical link
N^b	The set of backup VNF
E^b	The set of backup links
G^{sub}	The subgraph of G^s
$M_N(n^s)$	The available CPU resources of physical node n^s
$M_L(e^s)$	The available bandwidth of physical link e^s
$c(n^v)$	The requested CPU resources for VNF n^v
$bw(e^v)$	The requested bandwidth for link e^v
μ_m	The experienced hop delay of link m
r_j	The availability of a VNF
R_t	The achieved reliability of an SFC t
d_i	The importance factor of node n^s
$NCF(n^s)$	The connection factor of physical node n^s
$SF(x)$	The saturation factor of resource of type x
$\psi(x)$	The stress factor of a resource of type x
C_t	The management cost of constructing the service path
m.node	The physical node along link m
F_{ij}^b	The set of all redundant instances of the jth VNF of s_i
$x_k^{f_n}$	A binary variable that equals 1 if and only if the primary VNF f_n is located on k
y_{ij}^m	A binary variable that equals 1 if and only if link m is selected by function j of service i
$x_k^{f_b}$	A binary variable that equals 1 if and only if the backup VNF f_b is located on k
$y_{f_b}^{m'}$	A binary variable that equals 1 if and only if link m' is selected by backup f_b
h_{ij}	A binary variable that equals 1 if and only if the jth VNF of s_i is protected by a backup