Virtual Network Function Allocation in Service Function Chains Using Backups With Availability Schedule

Rui Kang, *Graduate Student Member, IEEE*, Fujun He, *Member, IEEE*, and Eiji Oki, *Fellow, IEEE*

***Abstract*—A suitable virtual network function (VNF) placement considering a node availability schedule extends service continuous serviceable time by suppressing service interruptions caused by function reallocation and node unavailabilities. However, function placement cannot avoid service interruptions caused by node unavailabilities. This paper proposes a primary and backup VNF placement model to avoid service interruptions caused by node unavailabilities by using backup functions. The considered backup functions have a period of startup time for preparation before they can be used and the number of them is limited. The proposed model is formulated as an integer linear programming problem to place the primary and backup VNFs based on the availability schedule at continuous time slots. We aim to maximize the minimum number of continuously available time slots in all service function chains (SFCs) over the deterministic availability schedule. We obverse that the proposed model considering the limited number of backup functions outperforms baseline models in terms of the minimum number of longest continuous available time slots in all SFCs. We introduce an algorithm to estimate the number of key unavailabilities at each time slot, which can find the unavailable nodes which are the bottlenecks to increase the service continuous available time at each time slot.**

***Index Terms*—Network function virtualization, virtual network function placement, service function chain, availability schedule, integer linear programming, backup.**

I. Introduction

TO INCREASE the network function placement flexibility [1] and reduce the deployment cost [2], [3], network function virtualization (NFV) has been introduced [4] by decoupling network functions from hardware. A network function is virtualized as a virtual network function (VNF) running on a node in the network; each node is equipped with virtual machines (VMs). To satisfy various service requirements from users, a service function chain (SFC) which contains a chain of VNFs is generated so that a service provider (SP) can provide users with suitable services flexibly and efficiently [5].

Manuscript received January 19, 2021; revised April 22, 2021 and July 5, 2021; accepted July 6, 2021. Date of publication July 12, 2021; date of current version December 9, 2021. This work was supported in part by Grants-in-Aid for Scientific Research of Japan Society for the Promotion of Science with KAKENHI Grant Numbers 18H03230, 21H03426 and 21J21552. The associate editor coordinating the review of this article and approving it for publication was J.-F. Botero. *(Corresponding author: Rui Kang.)*

The authors are with the Graduate School of Informatics, Kyoto University, Kyoto 606-8501, Japan (e-mail: kang.rui.74z@st.kyoto-u.ac.jp; fujunhe@i.kyoto-u.ac.jp; oki@i.kyoto-u.ac.jp).

Digital Object Identifier 10.1109/TNSM.2021.3096254

The unavailability of VNF interrupts the SFCs that use the VNFs running on nodes in a network and degrades the user experience. The works in [6]–[8] introduced several placement models to offset the effect of unavailability. The availability of nodes can be estimated in advance by taking advantage of records and maintenance schedules. This paper calls the availability of each node at each time slot *availability schedule*. The length of a time slot is determined by the requirement of network operators, such as the error detection interval. We assume that the availability schedule can be obtained in advance from operation records, maintenance schedules, and so on.

The availability schedules are considered in [9], [10] to obtain the VNF placement that maximizes the continuous available time of SFCs by avoiding service interruptions caused by the same VNF placed on different nodes between adjacent time slots [11], [12] and a VNF placed on an unavailable node under a given availability schedule. In [9], [10], two metrics are defined for evaluating the period of stable service: service continuous available time (SCAT) and the shortest SCAT (SSCAT). The model in [9], [10] places VNFs to maximize SSCAT without considering backup, e.g., cold backup and hot backup. If the model also places backup functions in addition to placing primary functions, the continuous available time slots of SFCs can be extended.

Cold backup only marks each location of backup function at the beginning of the system configuration [13]. A backup function becomes active and starts to consume the computational resource when the corresponding primary function becomes unavailable. The advantage of cold backup is that it does not consume the computational resource of the backup function before it becomes active. The disadvantage is that it needs time to startup and synchronize the state information for backup and generates an interruption. When a VM monitoring function which checks the status of every VM periodically detects VM failures, the latest snapshots of each VM are used to re-establish the failed VMs in the corresponding backup resources in physical machines [14].

Hot backup keeps an online backup function which synchronizes with the corresponding primary function. It can achieve faster recovery than cold backup, but requires the extra computational and memory resources in the system. Hot backup for service composition in a network is applied in [15]. According to the availability and current state of service composition before the services are interrupted, it restores the

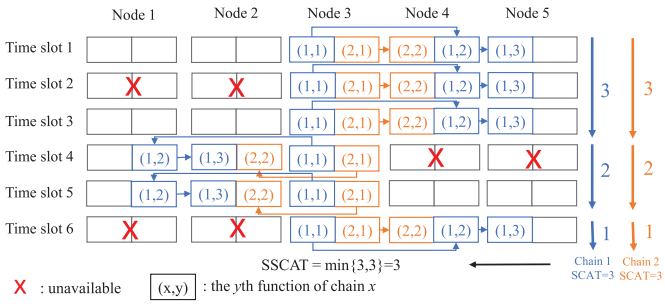
service composition dynamically. Hot backup can avoid service interruptions at the cost of extra resource consumption for backup functions.

In this paper, we consider a kind of hot backup. A backup function needs to be prepared for a period before it is implemented. We call this period *recovery time*. If backup functions are placed and are activated within suitable time slots for preparation instead of all the time slots from the beginning, the interruptions can be suppressed. The extra resource consumption for backup functions can be reduced since we do not need to backup functions from the beginning time slot.

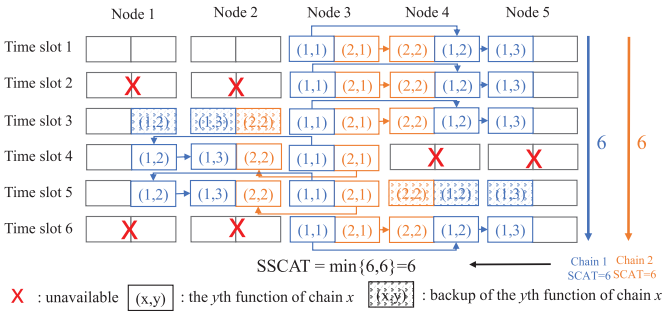
Figure 1 shows the examples for calculating SSCAT and SCATs and the differences between the allocations with and without considering backups under a given availability schedule. We place two function chains 1 and 2 with two and three VNFs, respectively, to a set of five nodes among six time slots. Nodes 1 and 2 are not available at time slots 2 and 6, and nodes 4 and 5 are not available at time slot 4. (x, y) denotes the yth function of chain x. If we place the functions without considering backup as shown in Fig. 1(a), chains 1 and 2 are interrupted between time slots 3 and 4 caused by the sudden replacements of functions (1, 2) and (1, 3), and (2, 2), respectively, and interrupted between time slots 5 and 6 caused by the sudden replacements of functions (1, 2) and (1, 3), and (2, 2), respectively. SCAT of chain 1 is the largest continuous available time, i.e., the largest one among three, two, and one, i.e., three. SCAT of chain 2 is the largest continuous available time, i.e., the largest one among three, two, and one, i.e., three. SSCAT is the smallest value among SCATs of all chains, i.e., three. We assume that the recovery times of all functions are one. Functions (1, 2), (1, 3) and (2, 2) at time slots 3 and 5 are backed up to prevent the interruptions caused by function replacement at time slots 4 and 6, as shown in Fig. 1(b). As a result, the backup can increase SCATs of all the chains from three to six and SSCAT from three to six.

This paper proposes an optimization model to place the primary and backup VNFs in SFCs on nodes, which aims to maximize SSCAT in the network under a given availability schedule against service interruptions. Linear SFCs are considered in this paper. We formulate the problem to maximize SSCAT over the given availability schedule as an integer linear programming (ILP) problem. We analyze the proposed model and derive the upper bound of SSCAT provided by the proposed model. We provide a heuristic algorithm; the problem becomes more difficult to solve in practical time as the size of the problem increases. Numerical results compare the proposed model by using the ILP approach with four baseline models. We also compare the results obtained by the heuristic algorithm and the ILP approach. We give an algorithm to estimate the number of bottlenecks of unavailable nodes at different time slots. The proposed model using our backup strategy improves SSCAT compared with the model without using backup functions in our examined cases.

This paper is an extended version of [16]. We describe the motivation and applicability of the proposed model. We develop a heuristic algorithm to speed up the computation for the case that the problem size increases. We expand the performance evaluations of the proposed model with various



(a) Placement without backup.



(b) Placement with backup. The recovery time of each VNF is one time slot.

Fig. 1. Examples of service continuous available time of a chain by comparing the allocations of all VNFs in a chain in different time slots with and without backups.

additions. We extend the proposed model with considering a routing problem. We compare the results obtained by the heuristic algorithm with those of the ILP approach. We extensively survey existing related works. We provide the discussion on dealing with multiple replicas of a function and their backups.

The remainder of the paper is organized as follows. Section II introduces the motivation and several applicabilities of the proposed model. Section III presents the proposed model and gives the upper bound of SSACT provided by the proposed model. Section IV gives a heuristic algorithm for solving the considered problem. Section V compares the performance of the proposed model with those of baseline models in different cases. Section VI describes the related works. Section VII summarizes the paper.

II. Motivation and Applicability

SPs perform maintenance and system updates regularly. Specific VMs waiting for maintenance and updating are marked as being unavailable at a specific time by SPs. SPs can reduce the number of service interruptions by knowing where and when service interruptions occur and controlling the positions and time of service interruptions referring to the maintenance schedule known forehead. In terms of the unscheduled and sudden failures, we can rely on protection mechanisms [17]–[19] in the unavailable periods of nodes. This paper does not concern the details of the mechanisms. We assume that the estimated availability schedules are provided by administrators. If a node is marked as available, we consider that it is available. The availability is ensured by the other protection systems and techniques, e.g., duplication.

If we do not care about the exact period or the starting time that a physical machine becomes unavailable, the availability and reliability are often evaluated by other metrics including probability, mean time to repair, and mean time between failures in existing studies [20], [21]. The above evaluation is widely used since the exact time of the unavailability is not usually obtainable. Once we obtain the exact period of unavailability from the maintenance schedules, the available period of a service which is evaluated by time, i.e., SSCAT in this paper, is more intuitionistic and applicable compared with the mean times or probability. Our model is based on the motivation that the available and unavailable time of a physical node is given and the normal running time of services is the metric to measure. The model is not designed for the environment where we do not care about the running time and the total availability evaluated by the probability is the objective. If the available time of physical nodes cannot be estimated, the proposed model cannot be applied, either. The schedule is assumed to be obtained in advance.

The models introduced in [9], [10] provided VNF allocation models to suppress the interruptions from the allocations on unavailable nodes and the reallocations. However, it is a passive way to avoid assigning to unavailable nodes and has a limitation decided by the availability schedules. The proposed model actively uses backups to avoid the interruptions caused by the reallocation for improving the SSCAT compared with the model without considering backup VNFs.

The proposed model is designed for the initial allocation of VNFs in SFCs with given deterministic availability schedules. We provide two directions for the deployment of the proposed model: deployment in a container network and deployment in a software-defined network (SDN). The proposed model can be deployed as a part of the scheduler, which decides the location of Pods in an orchestration engine such as Kubernetes [22]. The proposed model works as a scoring mechanism in a scheduler [23] in the engine. The placement is decided by the score provided by the model. Based on the information provided by administrators including availability schedules and the information collected by the system including the nodes, the proposed model calculates the suitable locations for each VNF and gives the highest score for the location.

In an SDN compatible environment, the proposed model provides the allocation result to the controller according to the relative information including the availability schedules, VNFs, services, and network devices stored in the database by SPs. A computation element in the network, which is an independent node or a node with VNFs, calculates the allocation with the information. The allocation result is sent to each corresponding node. The nodes run the corresponding functions or the containers in the nodes by pulling the corresponding images from repositories. A demonstration of the application of allocation models was introduced in [24].

III. Model Design

A. Problem Formulation

This paper considers a primary and backup VNF placement model in directed graph $G(N, L)$; N and L denote a set of nodes and a set of directed links connecting two nodes, respectively. We divide the continuous time into several discrete time slots. We use T to indicate the set of time slots. $T_i = [1, |T| - i + 1] \subseteq T, i \in [1, |T|]$, denotes a set of time slots from 1 to $|T|-i+1$. Let C denote the set of capacities of all nodes at all time slots. The capacity of node $n \in N$ at time slot $t \in T$ is $c_t^n \in C$. Here the capacity is an abstract concept regardless of the specific resource type. We consider the placement of VNFs in different SFCs. We assume that one request corresponds to one SFC. We use R to indicate the set of SFCs waiting for placement. There are K_r ordered VNFs in SFC $r \in R$. There may be several requests sharing the same SFC with different requests of package processing abilities. We give an extension of the proposed model to separate the requests from the SFCs in Appendix C-A.

Availability schedule E is given in this paper. Binary parameter $e_t^n \in E$ expresses the availability of node $n \in N$ at time slot $t \in T$. If node n at time slot t is unavailable, $e_t^n = 1$; otherwise, 0.

Binary decision variable x_{tn}^{rk} represents the placement of the kth function of SFC r. If it is placed on node n at time slot t $x_{tn}^{rk} = 1$; 0 otherwise.

$g_r^k \in T, r \in R, k \in K_r$, is a given parameter which represents the recovery time of the kth function of request r. We assume that $0 < g_r^k < |T|$. Binary decision variable u_{tn}^{rk} represents the placement of the backup for the kth function of SFC r. If the $k \in K_r$th function of request $r \in R$ is backed up on node $n \in N$ at time slot $t \in T$, $u_{tn}^{rk} = 1$; otherwise, 0. A backup function also consumes computational resources. If n can be continuously occupied for backup from time slot $t - g_r^k$ to time slot $t - 1$, that is, $u_{t-g_r^k,n}^{rk} = u_{t-g_r^k+1,n}^{rk} = \cdots = u_{t-1,n}^{rk} = 1$, and this function is placed on node n at time slot t, that is, $x_{tn}^{rk} = 1$, this function can be prevented from the interruption caused by unavailability $e_t^{n'}$.

The notations frequently used in the proposed model are summarized in Table I.

$\alpha_{tn}^{rk}, t \in T, k \in K_r, r \in R, n \in N$, denotes a binary variable which is set to 1 if the kth function of request r is prevented from the interruption happening on node n at time slot t; 0 otherwise. $\forall r \in R, k \in K_r, t \in T, n \in N$, α_{tn}^{rk} is expressed by:

$$\alpha_{tn}^{rk} = \begin{cases} \left\{ (\vee_{n' \in N \setminus \{n\}} e_t^{n'} x_{t-1,n'}^{rk}) \wedge \left(\left(\prod_{t' \in [t-g_r^k, t-1]} u_{t'n}^{rk} \right) \wedge x_{tn}^{rk} \right) \right\} \vee \left\{ (e_t^n x_{t-1,n}^{rk}) \wedge \left(\vee_{n' \in N \setminus \{n\}} \left(\prod_{t' \in [t-g_r^k, t-1]} u_{t'n'}^{rk} \right) \wedge x_{tn'}^{rk} \right) \right\}, & t \in T \setminus \left[1, g_r^k\right] \\ 0, & t \in \left[2, g_r^k\right], \end{cases} \tag{1}$$

where $\vee$ expresses a binary *or* operator and $\wedge$ expresses a binary *and* operator. The right hand side of (1) consists of two components with $\vee$. The first component indicates if a backup function on node n can avoid the interruption that happens on node n' at time slot t for the kth function of request r. $(\vee_{n' \in N \setminus \{n\}} e_t^{n'} x_{t-1,n'}^{rk}) = 1$ is set if any unavailable node n' at time t, i.e., $e_t^{n'} = 1$, interrupts the placement of the kth

TABLE I
NOTATIONS USED IN SECTION III

Parameter	Description
$G(N, L)$	Virtual network
N	Set of virtual nodes
T	Set of time slots
T_i	Set of time slots from 1 to $\|T\| - i + 1$
R	Set of requests
L	Set of virtual links
S	Set of types of resources
K_r	Length of request $r \in R$
c_t^n	Capacity of node $n \in N$ at time slot $t \in T$
q_s^{rk}	Requirement of resource $s \in S$ for the $k \in K_r$th function of request $r \in R$
e_t^n	An availability schedule; if node $n \in N$ is unavailable at time slot $t \in T$, $e_t^n = 1$, and otherwise 0
d_r	Transmission resources demanded by request $r \in R$
b_{ij}	Available transmission resource between nodes $i \in N$ and $j \in N$
g_r^k	Preparation time of the $k \in K_r$th function of request $r \in R$
l_{ij}	End-to-end latency between nodes $i \in N$ and $j \in N$
Decision variable	**Description**
x_{tn}^{rk}	If the kth function of request r is assigned to node n at time slot t, $x_{tn}^{rk} = 1$; otherwise, 0
u_{tn}^{rk}	If the kth function of request $r$is backed up on node n at time slot t, $u_{tn}^{rk} = 1$; otherwise, 0
α_{tn}^{rk}	If the kth function of request r is prevented from the interruption caused by unavailability $e_t^{n'}$, $\alpha_{tn}^{rk} = 1$; otherwise, 0
o_t^r	If the allocations of at least one function in request r is changed between time slot $t-1$ and time slot t or any VNF of request r at time slot t or $t-1$ is allocated to an unavailable node, is set to 0, and otherwise 1
z_i^{jr}	If allocations of request r from time slot i to time slot $i+j-1$ are consecutively unchanged, or j consecutive o_t^r are all 1 from $t = i$ to $t = i + j - 1$, z_i^{jr} is set to 1, and otherwise 0
y_j^r	If the allocations are consecutively unchanged during j time slots existing in T, y_j^r is set to 1, and otherwise 0
β_r	SCAT of request $r \in R$
λ	SSCAT

function of request r on node n' between time slots t and $t-1$; 0 otherwise. To ensure the backup of this interrupted function work, we need to keep the maintenance of this backup from $t - g_r^k$ to $t-1$, i.e., $\prod_{t' \in [t-g_r^k, t-1]} u_{t'n}^{rk} = 1$. Finally, the backup is activated, i.e., $x_{tn}^{rk} = 1$. The second component indicates if a backup function on node n' can avoid the interruption that happens on node n at time slot t for the kth function of request r. $e_t^n x_{t-1,n}^{rk} = 1$ is set if e_t^n interrupts the placement of the kth function of request r on node n' between time slots t and $t-1$; 0 otherwise. To ensure the backup of this interrupted function work, we need to keep the maintenance of the backup for the kth function of request r on node n from $t - g_r^k$ to $t-1$. $x_{tn'}^{rk} = 1$ is set if the backup on node n' is activated; 0 otherwise.

We cannot back up the function on an unavailable node, which is expressed by:

$$e_t^n u_{tn}^{rk} = 0, \forall r \in R, k \in K_r, n \in N, t \in T. \tag{2}$$

There are two metrics in this model. The integer variable $\beta_r \in [1, |T|]$ denotes the maximum number of continuous available time slots in request $r \in R$. We refer to it as SCAT hereafter. $\lambda \in [1, |T|]$ is the other metric, which denotes the smallest β_r among R. We refer to it as SSCAT hereafter. λ is constrained by:

$$\lambda \leq \beta_r, \forall r \in R. \tag{3}$$

To define β_r, we introduce three binary decision variables: $o_t^r, t \in T, r \in R$, $z_i^{jr}, i \in T, j \in T_i, r \in R$, and $y_j^r, j \in T, r \in R$. o_t^r indicates whether there is an invalid placement. If the placement of request r is changed between time slot $t-1$ and time slot t and this function is not backed up, or any VNF of request r at time slot t or $t-1$ is placed on an unavailable node, $o_t^r = 0$; otherwise, 1. If the placements of all VNFs belonging to SFC r are consecutively unchanged from time slot i to $i + j - 1$, i.e., o_t^j is all 1 from $t = i$ to $t = i + j - 1$, $z_i^{jr} = 1$; otherwise, 0. If there is a period whose length is j and the placement of SFC $r \in R$ is unchanged during this period, $y_j^r = 1$; otherwise, 0. The above variables are constrained by:

$$o_t^r = \prod_{k \in K_r} \prod_{n \in N} \Big(\Big(x_{tn}^{rk} \odot x_{t-1,n}^{rk} \vee \alpha_{tn}^{rk} \Big) \wedge \Big(1 - e_t^n x_{tn}^{rk} \Big) \wedge \Big(1 - e_{t-1}^n x_{t-1,n}^{rk} \Big) \Big), \forall r \in R, t \in T \backslash \{1\}, \tag{4}$$

$$\left(\sum_{t=i}^{i+j-1} o_t^r \right) - j + 1 \leq z_i^{jr}, \forall i \in T, j \in T_i, r \in R, \tag{5}$$

$$z_i^{jr} \leq \frac{\sum_{t=i}^{i+j-1} o_t^r}{j}, \forall i \in T, j \in T_i, r \in R, \tag{6}$$

$$z_i^{jr} \leq y_j^r, \forall i \in T, j \in T_i, r \in R, \tag{7}$$

$$y_j^r \leq \sum_{t=1}^{|T|-j+1} z_t^{jr}, \forall j \in T, r \in R. \tag{8}$$

$$\beta_r = \max_{j \in T} \{ j y_j^r \} + 1, \forall r \in R \tag{9}$$

where $\odot$ expresses an exclusive *nor* operation between two binary variables.

If the computational resources of backup functions are limited to a constant value R_{B}, u_{tn}^{rk} is constrained by:

$$\sum_{r \in R} \sum_{k \in K_r} \sum_{t \in T} \sum_{n \in N} u_{tn}^{rk} \leq R_{\mathrm{B}}. \tag{10}$$

R_{B} has a range from 0 to $\sum_{r \in R} \sum_{k \in K_r} g_r^k$.

The objective function of the proposed model is given by:

$$\max \ \lambda + \epsilon_1 \sum_{r \in R} \beta_r. \tag{11}$$

In this problem, the solution that maximizes the sum of continuous available time slots for all requests, $\sum_{r \in R} \beta_r$, is chosen when there are multiple solutions that maximize λ. Therefore, a sufficiently small positive number, ϵ_1, is multiplied to the second term to prioritize the first term over the second term. ϵ_1 is given by $\frac{1}{|R| \cdot |T|}$.

The node capacity constraint is given by:

$$\sum_{r \in R} \sum_{k \in K_r} \Big(x_{tn}^{rk} + u_{tn}^{rk} \Big) \leq c_t^n, \forall n \in N, t \in T. \tag{12}$$

Each node can carry only a limited number of functions, because of the computational resource limitation.

Equation (12) ensures that each node's computational resources must not exceed its capacity during placement.

The assignment constraint is given by:

$$\sum_{k\in K_r}\left(u_{tn}^{rk}+x_{tn}^{rk}\right)\leq 1,\forall r\in R,n\in N,t\in T, \quad (13)$$

$$\sum_{n\in N}x_{tn}^{rk}=1,\forall r\in R,k\in K_r,t\in T, \quad (14)$$

$$\sum_{n\in N}u_{tn}^{rk}\leq 1,\forall r\in R,k\in K_r,t\in T. \quad (15)$$

Equation (13) assumes that one service chain does not allocate multiple VNFs in this chain on one VM considering the influence of the reallocation of VMs [25] and that the primary VNF and the backup VNF of the same function cannot be allocated to the same nodes at one time slot. Equation (14) ensures that all VNFs are placed once in the network. If any requested SFCs cannot be accepted, there is no feasible solution. To discuss the case that too many SFC are requested to accept, we relax the constraint in (14) and give an extension of the proposed model to maximize the number of accepted SFCs as the objective in Appendix C-C. Equation (15) assumes that each VNF has at most one backup at one time slot. Equations (14) and (15) impose that the proposed model only uses at most two replicas, one for primary VNF and one for backup VNF. If we want to prevent service interruptions from the failures of functions and load balancing by using VNF replicas in an SFC, a possible extension is introduced in Appendix C-B to support multiple replicas for primary and backup VNFs.

According to the linearization process in [9], (1) is linearized to (24a)-(24q), (9) is linearized to (25a)-(25f), and (4) is linearized to (26a)-(26s) with some auxiliary variables in Appendix A.

In summary, we give the following model:

$$\max\ \lambda+\epsilon_1\sum_{r\in R}\beta_r \quad (16a)$$

$$\text{s.t. } (2)-(3),\ (5)-(8),\ (10),\ (12)-(15),\ (24a)-(26r), \quad (16b)$$

$$z_i^{jr}\in\{0,1\},\forall r\in R,i\in T,j\in T_i. \quad (16c)$$

When there are multiple solutions that maximize $\lambda+\epsilon_1\sum_{r\in R}\beta_r$, the solution with a small number of backup functions may be preferable. For this purpose, we can use the following objective function:

$$\max\ \lambda+\epsilon_1\sum_{r\in R}\beta_r-\epsilon_2\sum_{r\in R}\sum_{k\in K_r}\sum_{t\in T}\sum_{n\in N}u_{tn}^{rk}. \quad (17)$$

The solution that minimizes the number of backup functions $\sum_{r\in R}\sum_{k\in K_r}\sum_{t\in T}\sum_{n\in N}u_{tn}^{rk}$ is chosen when there are multiple solutions that maximize $\lambda+\epsilon_1\sum_{r\in R}\beta_r$. Therefore, a sufficiently small positive number, ϵ_2, is multiplied to the third term to prioritize the second term over the third term. ϵ_2 is given by $\frac{1}{\sum_{r\in R}\sum_{k\in K_r}g_r^k}$.

B. Analysis of Proposed Model

This subsection derives the upper bound of SSCAT provided by the proposed model.

Algorithm 1 Calculate the Number of Nodes on Which Functions Are Placed at Time Slot t

Input: Set of nodes N, a time slot $t\in T$, set of SFCs R, set of ordered VNFs in SFC $r\in R$ K_r, set of the capacities of all nodes at time slot t C_t
Output: Set of numbers of key unavailabilities at each time slot S

```
function m_CALCULATION(N, t, R, K_r, C_t)
    Sort requests in R in non-increasing order of |K_r|, r ∈ R
    Sort node in N in non-increasing order of c_t^n ∈ C_t
    for r ∈ R do
        for k ∈ K_r do
            for n ∈ N do
                if used capacity of n is less than c_t^n and any other functions
in request r were not placed on node n then
                    Place the kth function in SFC r on n
                    Break
                else
                    Continue
                end if
            end for
        end for
    end for
    return the number of nodes on which functions are placed.
end function
```

Lemma 1: The m_CALCULATION function in Algorithm 1 gives the minimum number of required nodes on which all functions in R are placed at time slot $t\in T$, m_t.

Proof: We prove this lemma by contradiction. If the result provided by m_CALCULATION is not the minimum number of required nodes at time slot $t\in T$, m_t, there is a set of $m_t'<m_t$ nodes that can accommodate all functions limited by the constraints. To get m_t', we need to move all the functions placed on a node that belongs to a set of m_t nodes obtained by function m_CALCULATION to other nodes that have been assigned functions. However, the reallocation causes a situation that at least one node accommodates more functions than it can accommodate, or two functions from the same SFC are placed on the same node according to the placement procedure at lines 35-40 of Algorithm 2. There is no valid m_t' such that it is less than m_t. ■

Let E_t be a set of unavailable nodes at time slot $t\in T$. Let $\Xi_t^{t'}, t\in T\backslash\{|T|\}, t'\in T\backslash\{1\}, t'>t$, denote a binary variable, which is set to 1 if $|N|-|E_t\cup E_{t'}|\geq m_{t'}$; 0 otherwise.

Lemma 2: The upper bound of SSCAT provided by the proposed model without considering backup functions is $\Delta\equiv\max_{t\in T\backslash\{|T|\},t'\in T\backslash\{1\},t'>t:\Xi_t^{t+1}=\Xi_t^{t+2}=\cdots=\Xi_t^{t'}=1}\{t'-t\}$.

Proof: m_t is the minimum number of required nodes at time slot $t\in T$ by using Lemma 1. If $|N|-E_t<m_t$, there are so many unavailable nodes that no enough space to place all functions in R at time slot t exists. If $|N|-|E_t\cup E_{t'}|<m_{t'}$, some functions migrate to avoid being placed in an unavailable node or there is no enough space for function placement. The longest consecutive duration for which all functions in R can be placed is the largest difference between t and t', where $\Xi_t^{t''}=1,\forall t''\in[t+1,t']$. ■

Let g denote the smallest $g_r^k,\forall r\in R,k\in K_r$. Let E' denote a set of $(n,t),n\in N,t\in T$, where each (n,t) has $e_t^n=1,e_t^n\in E$, and there is any node $n'\in N\backslash\{n\}$ that satisfies the following conditions: $t>g$, $e_t^{n'}=e_{t-1}^{n'}=$

Algorithm 2 Calculate the Number of Key Unavailabilities

Input: Set of nodes *N*, set of time slots *T*, set of SFCs *R*, set of ordered VNFs in SFC $r \in R$ K_r, set of the capacities of all nodes at all time slots *C*, availability schedule *E*
Output: Set of numbers of key unavailabilities at each time slot *S*

Set $U \leftarrow \emptyset$
Define u_t for storing the number of key unavailabilities at time slot $t \in T$ and set $u_t \leftarrow 0$.
for $t = 1 \rightarrow |T|$ **do**
  $m \leftarrow m_$CALCULATION(*N*, *T*, *R*, K_r, C_t).
  if $m > |N|$ **then**
    return No solutions.
  end if
  Check the nearest time slot $t' < t$, which has a different set of unavailabilities from those in *t*.
  if t' exists **then**
    Define N' as a set of $n \in N$ where $e_n^t = 1$ or $e_n^{t'} = 1$.
    if $|N| - |N'| < m$ **then**
      $u_t \leftarrow m - |N| + |N'|$. Add u_t to *U*.
    end if
  else
    if $|N| - \sum_{n \in N} e_n^t < m$ **then**
      $u_t \leftarrow m - |N| + \sum_{n \in N} e_n^t$. Add u_t to *U*.
    end if
  end if
end for
A_k is a set of all combinations of $k \in [1, |U|]$ elements in *U*, A_k^a is the *a*th element of A_k, or $A_k^a \in A_k, a \in [1, |A_k|]$.
$S \leftarrow \emptyset, s \leftarrow 0$
for $k = 1 \rightarrow |U|$ **do**
  for $a = 1 \rightarrow |A_k|$ **do**
    $v \leftarrow$ the objective value computed by assuming that $u_t \in A_k^a$ unavailable nodes become available at time slot *t*. u_t unavailable nodes are chosen in the following sets of unavailable nodes at time slot *t* in order they are found: new unavailable nodes at *t*; unavailable nodes chosen before *t*. In each set, any nodes are chosen.
    if $v >= s$ **then**
      if $v = s$ and $|A_k^a| > |S|$ **then**
        Continue
      end if
      $S \leftarrow A_k^a, s \leftarrow v$
    end if
  end for
end for

$\cdots = e_{t-g}^{n'} = 0$, and $c_t^{n'}, c_{t-1}^{n'}, \ldots, c_{t-g}^{n'} \geq 1$. We choose $\theta \in [1, |E'|]$ different elements and store each possible combination as a set to set E'_θ. Let E'' denote a set of sets, $E'' = \{\emptyset \cup E'_1 \cup E'_2 \cup \cdots \cup E'_{|E'|}\}$. Let $E'''_a, a \in [1, |E''|]$, denote a set of $e_t^n, t \in T, n \in N$, in which e_t^n is set to 0 if (n, t) is in E''_a and other e_t^n has the same value with the corresponding $e_t^n \in E$. Lemma 1 obtains m_t by using *E*. We calculate $\Delta_a, a \in [1, |E''|]$, with $m_t, \forall t \in T$, and E'''_a, by using Lemma 2.

Theorem 1: The upper bound of SSCAT provided by the proposed model with considering backup functions is $\max_{a \in [1, |E''|]} \Delta_a$.

Proof: We assume that the recovery time is the smallest one among all recovery times *g* in this proof. We assume that R_{B} is $\sum_{r \in R} \sum_{k \in K_r} g_r^k$. We relax the constraint on the required capacities of backup functions. E' stores the unavailable locations (n, t) that can be avoided by using backup functions limited by recovery time *g*. By calculating different combinations of elements in E', we can know all the possible backup plans and store them into E''. We form new availability schedule $E'''_a \in E''$ by assuming that the unavailable locations in E'''_a are recovered. We regard each modified availability schedule $E'''_a, a \in [1, |E''|]$, as an availability schedule in the VNF placement model. We calculate Δ_a by using $E'''_a, a \in [1, |E''|]$. The largest Δ_a is the upper bound of SSCAT in the proposed model. ■

C. Network-Aware Model

Section III-A does not consider the routing among VNFs in an SFC and the recovery path between primary and backup VNFs. Some paths cannot be chosen because of the limitation of the characteristic of links, such as the link capacity. The consideration of networks while deciding the VNF allocations can avoid problems such as the turn-back traffic and the long-distance traffic, which lead to inefficient resource consumption and increased latencies [26]. This subsection gives an extension on how to handle the routing problems by considering the networks.

We use d_r to express the required transmission resources of request $r \in R$. For link $(i, j) \in L$, we use b_{ij} and l_{ij} to express the available transmission resources and the end-to-end latency from node *i* to node *j*. The end-to-end latency contains propagation delay, packetization delay, and queueing delay it represents the required time for data forwarding between two nodes. We assume that the end-to-end latency can be estimated [27], which is related to the network congestion and physical distance. For the estimation of end-to-end latency, some network measurement tools, e.g., ping, netperf, and TCPing for TCP transmission, are used. The flow constraint to compute the routes of all SFCs is given by:

$$\sum_{(i,j) \in L} a_{w,ij} p_{rt}^{k,ij} = \begin{cases} -1, & \text{if } x_{tw}^{rk} = 1 \quad (18a) \\ 1, & \text{if } x_{tw}^{r,k+1} = 1 \quad (18b) \\ 0, & \text{if } x_{tw}^{rk} = x_{tw}^{r,k+1} = 0. \quad (18c) \end{cases}$$

For each request, there are three types of nodes: source node (to which the first function of a request is allocated), destination node (to which the last function of a request is allocated) and others. We define indicator $a_{w,ij}, w \in N, (i, j) \in L$, to represent the adjacency of nodes on directed graph *G*, where $a_{w,ij} = 1$ if node *w* is the tail of the directed link (i, j), i.e., $w = j$; $a_{w,ij} = -1$ if node *w* is the head of the directed link (i, j), i.e., $w = i$; $a_{w,ij} = 0$ otherwise. We use binary variable $p_{rt}^{k,ij}$ to express the routes in an SFC. If link (i, j) is a segment link between the *k*th node and the $(k+1)$th node of request $r \in R$ at time slot $t \in T$, $p_{rt}^{k,ij} = 1$; otherwise, 0. According to (13), x_{tw}^{rk} and $x_{tw}^{r,k+1}$ cannot be 1 at the same time. Thus (18) can be simplified to:

$$\sum_{(i,j) \in L} a_{w,ij} p_{rt}^{k,ij} = -x_{tw}^{rk} + x_{tw}^{r,k+1}, \forall k \in K_r \backslash \{|K_r|\}, r \in R, t \in T, w \in N. \quad (19)$$

The flow constraint to compute the routes for primary and backup synchronizations is given by:

$$\sum_{(i,j) \in L} a_{w,ij} q_{rt}^{k,ij} = \begin{cases} -1, & \text{if } x_{tw}^{rk} = \sum_{w' \in N} u_{tw'}^{rk} = 1 \quad (20a) \\ 1, & \text{if } u_{tw}^{rk} = \sum_{w' \in N} x_{tw'}^{rk} = 1 \quad (20b) \\ 0, & \text{otherwise.} \quad (20c) \end{cases}$$

We use binary variable $q_{rt}^{k,ij}$ to express the routes between the primary VNF and the backup VNF of a function if there exists any backup. If link (i,j) is a segment link between the locations of the primary node and the backup node of the kth VNF in request $r \in R$ at time slot $t \in T$, $q_{rt}^{k,ij} = 1$, and otherwise 0. According to (13), x_{tw}^{rk} and u_{tw}^{rk} cannot be 1 at the same time. Thus (20) can be simplified to:

$$\sum_{(i,j)\in L} a_{w,ij}\, q_{rt}^{k,ij} = -x_{tw}^{rk} \sum_{w'\in N} u_{tw'}^{rk} + u_{tw}^{rk} \sum_{w'\in N} x_{tw'}^{rk}, \quad \forall k \in K_r, r \in R, t \in T, w \in N. \quad (21)$$

The link capacity constraint is given by:

$$\sum_{r\in R} \sum_{k\in K_r} \left(p_{rt}^{k,ij} + q_{rt}^{k,ij}\right) d_r \le b_{ij}, \forall (i,j) \in L, t \in T. \quad (22)$$

Equation (22) ensures that each link's transmission resource is not overused.

We take the connections between nodes into consideration. The network-aware model is given by:

$$\max \ \lambda + \epsilon_1 \sum_{r\in R} \beta_r - \epsilon_3 \sum_{r\in R} \sum_{t\in T} \sum_{k\in K_r} \sum_{(i,j)\in L} l_{ij} p_{rt}^{k,ij} \quad (23a)$$

$$\text{s.t.} \ \ (2)-(3), (5)-(8), \ (10), \ (12)-(15), \ (19)-(22), \ (24a)-(27g), \quad (23b)$$

$$z_i^{jr} \in \{0,1\}, \forall r \in R, i \in T, j \in T_i. \quad (23c)$$

The solution that minimizes the sum of end-to-end latencies for all routes, $\sum_{r\in R}\sum_{t\in T}\sum_{k\in K_r}\sum_{(i,j)\in L} l_{ij} p_{rt}^{k,ij}$, is chosen when there are multiple solutions that maximize $\lambda + \epsilon_1 \sum_{r\in R} \beta_r$. Therefore, a sufficiently small positive number, ϵ_1, is multiplied to the second term of objective function to prioritize the first term over the second term; a sufficiently small positive number, ϵ_3, is multiplied to the third term to prioritize the second term over the third term. ϵ_3 is given by $\frac{1}{|R|\cdot|Kr|\cdot|T|\cdot|L|}$.

The model in Section III-A is a subset of the model in this subsection. Routing is given and the minimum of end-to-end latencies is considered in this subsection. If the SPs are only responsible for function allocations and not for routing and forwarding between different VNFs in the SFCs, the model in Section III-A is more suitable and faster than that in this subsection since the model in Section III-A has less decision variables and items in the objective function.

IV. Heuristic Algorithm

As the size of the ILP problem presented in Section III-A increases, the problem becomes more difficult to solve in practical time. Genetic algorithms have been used for VNF resource allocation problems in previous works, e.g., [28] and [29]. We introduce a genetic-algorithm-based heuristic algorithm in this section based on that introduced in [10].

A. Overall Structure

The running parameters for the heuristic algorithm are shown in Table II, which are set before the running of the algorithm by experience. The procedure of the heuristic algorithm is shown in Fig 2. At first, we generate a group of initial feasible solutions whose size is *IP* by using the approach in Section IV-B. For each solution in the group, the algorithm uses three mutation approaches, *internal crossover*, *external crossover*, and *mutation*, to generate new solutions based on the selected solution. Each approach has a probability which decides if the approach is used for generating new solutions. The three approaches are introduced in Section IV-C. Newly generated solutions are added to the group of solutions. The algorithm calculates the fitness score for each solution by using the method in Section IV-D and updates the best solutions. If the number of solutions exceeds the limitation *UP*, *UP* solutions are selected from the group for the next process by using the roulette gambler approach. The unselected solutions are deleted. The probability of a solution being selected is related to its fitness score; the higher the fitness score is, the higher the probability of the solution being selected is. The above steps are repeated *MG* times, and then the algorithm outputs the best solution explored in the procedures. The details and pseudocodes of the functions in the heuristic algorithm are listed in Appendix B.

TABLE II
Parameters in Algorithm

Parameter	Description
MG	Maximum generation
IP	Initial population
UP	Upper limitation of population
IC	Internal crossover probability
EC	External crossover probability
MP	Mutation probability
BP	Probability of applying the backup strategy

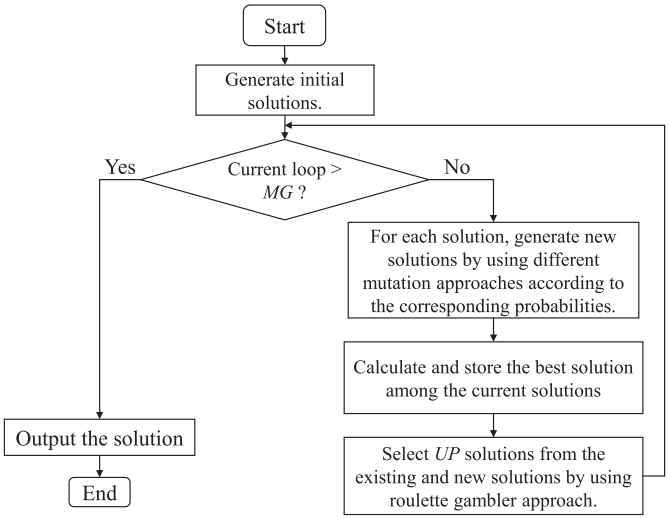


Fig. 2. Procedure of heuristic algorithm.

B. Initial Solution Generation

The heuristic algorithm generates a group of initial feasible solutions for primary VNF allocations by using the function described in this subsection. Based on this group, more feasible solutions can be generated by the mutation approaches in Section IV-C.

In the heuristic algorithm, each solution is a three-dimensional matrix. The first dimension represents time slots,

the second one represents requests, and the third one represents functions. The value of an element whose location is (t, r, k) is the allocation of the kth function of request r at time slot t, which belongs to N.

We reorder the set of chains in non-increasing order of the number of VNFs in this chain. We try to allocate each function to each node and ensure that the capacity of the node does not exceed. If all the functions of all chains among all time slots can be allocated, an initial solution is generated and stored. Since one initial solution is not enough for providing variations of the solutions, we duplicate the generated initial solution to a given parameter, initial population (*IP*). A relatively large population can speed up the convergence of the algorithm and provide more variations of the solutions.

C. Mutation

The heuristic algorithm provides three approaches for generating new solutions based on the current solutions so that better solutions may appear. One of the approaches for generating new solutions for primary VNF allocations is *mutation* based on an existing solution. The other two approaches are *internal crossover* and *external crossover*, which generate the solutions based on the length of each SFC. On the other hand, *mutation* generates a new solution based on the SCAT of each SFC.

We calculate the SCAT for all SFCs in the input solution. We sort chains and nodes in weighted randomized order: the larger the SCAT is, the higher the order of the corresponding chain is; the larger the time from the first time slot to a time slot where a node becomes unavailable is, the higher the order of the corresponding node is. We generate a new solution with the above orders and the same method as the generation of the initial solution.

D. Calculation of Fitness

The algorithm computes the fitness score for each solution by using (11) and the possible backup function allocations. We find the noncontinuous allocation of each chain. We calculate possible backup function allocations to extend the SCAT of each chain. There is a probability of applying the backup strategy (BP). Finally, we returns the calculated fitness score defined in (1), (3)-(9), and (11), and the corresponding backup function allocations.

V. Numerical Evaluations

A. Analysis of Allocation Model

We compare the proposed model solving by the ILP approach with the placement models presented in [9], [10] and with three baseline models introduced in [9], [10]: a persistent placement model, a single-slot placement model, and a double-slot placement model. The persistent placement model does not consider the interruptions caused by unavailability. The single-slot placement model considers the interruptions caused by unavailabilities at each time slot, regardless of reallocations between all adjacent time slots. The double-slot placement model considers the interruptions caused by unavailabilities at each time slot and the reallocation between the current and the last time slots. We compare SSCATs provided by the above five models.

TABLE III
Evaluation Conditions

Case	Nodes	Cap.	Req.	Func.	Time slots	Max. no. of unavailable nodes in a time slot
1a	8	2	4	3, 2, 2, 4	6	2
1b	8	2	4	3, 2, 2, 4	6	6
1c	16	2	8	6, 3, 2, 2, 4, 4, 3, 2	6	2

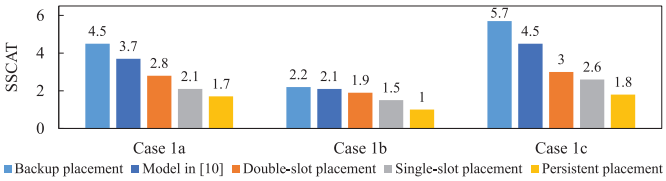


Fig. 3. SSCAT provided by proposed model yielded by four baseline models.

The persistent placement model is implemented by Python 3.7, using Intel Core i7-7700 3.60 GHz 4-core CPU, 32 GB memory. The ILP problems of the other models are solved by the IBM ILOG CPLEX Interactive Optimizer with version 12.10.0 [30] and run on the same hardware.

There are two tests for comparison. In test 1, we set $R_{\mathrm{B}} = \sum_{r \in R} \sum_{k \in K_r} g_r^k$ in the proposed model and compare the result obtained by the proposed model with those of other models. Three cases are examined in test 1. The conditions of these cases are shown in Table III.

In all cases, we obtain the VNF placement of the proposed model by solving the ILP problem in Section III. In cases 1a, 1b, and 1c, we randomly generate ten different availability schedules. We set unavailable nodes at each time slot so that the number of unavailable nodes is uniformly distributed in [1, maximum number of unavailable nodes in a time slot]. In each case, we compute the VNF placement in the same network. This yields the computed SSCAT values, which are then averaged.

Figure 3 shows SSCATs provided by the proposed model and provided by the four baseline models in all the examined cases. Compared with case 1b, SSCAT provided by the proposed mode increases more obviously than that of other baseline models in case 1a since the maximum number of unavailable nodes at each time slot in case 1a is smaller than that of case 1b. The average computation time of case 1a is 71 seconds. The average computation time of case 1b is 53 seconds. The average computation time of case 1c is 1920 seconds.

In test 2, we compare the results with different values of R_{B}. There are three cases in the second test. In case 2a, we place four SFCs in an eight-node network at six time slots. The lengths of these SFCs are four, three, two, and two, respectively. The unavailabilities are: $e_2^1, e_2^2, e_4^7, e_4^8, e_6^1, e_6^2$ are 1, and other e_t^n are 0. The recovery time is one. In case 2b, we reduce the number of SFCs from four to three compared with case 2a. The lengths of these SFCs are four, three, and two, respectively. In case 2c, we increase the length of the third SFC in case 2a from two to three.

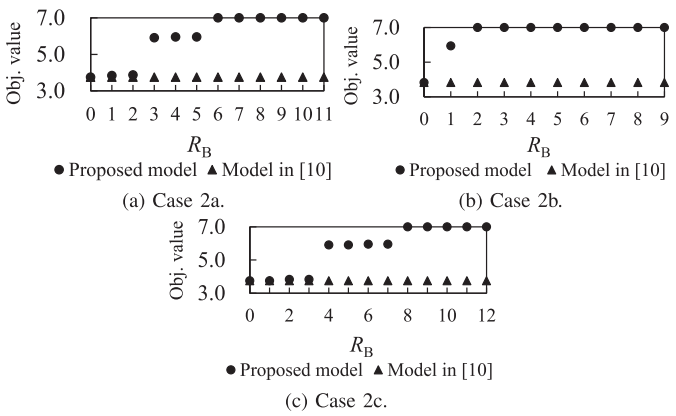


Fig. 4. Dependency of objective value on R_{B}.

Figure 4 shows the dependency of objective value on R_{B}. We observe that, with increase of R_B, there are three obvious increments of objective values in Figs. 4(a), (b), and (c). There is a set of unavailabilities in the availability schedule that are bottlenecks to reach higher SSCAT, which causes an obvious increment if R_B increases. We call the unavailabilities, which are the bottlenecks of reaching the largest objective value based on the given availability schedule, *key unavailabilities*.

Algorithm 2, which we develop, estimates the number of key unavailabilities in each time slot. We assume that the capacities of all nodes are the same. With this estimate, SPs can find the unavailable nodes which are the bottlenecks to increase SSCAT at each time slot. These unavailabilities are required to be eliminated priorly so that SPs can increase the service continuous available time with the least cost of failure recovery.

B. Impact of Different Types of Availability Schedules

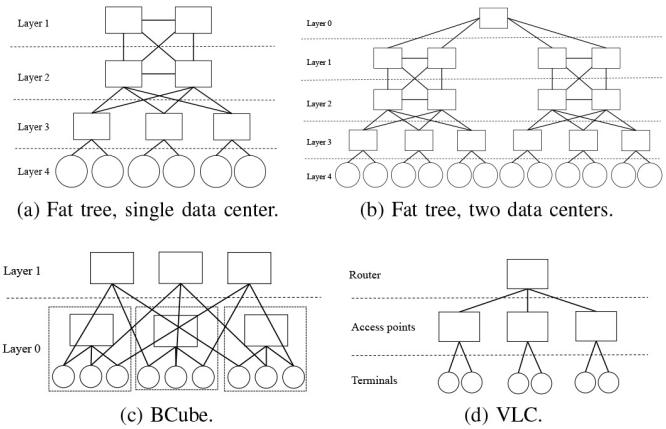
The distributions of unavailabilities in the availability schedules impact the SSCAT values obtained by the proposed model. In addition, the effect of backup functions may be different in different types of availability schedules.

To demonstrate this impact, we randomly generate five different types of availability schedules with different positions of the unavailable nodes in each time slot and calculate the SSCAT values with these availability schedules under the same condition. We consider an eight-node network, where the capacity of each node is set to two. We consider four requests with lengths of three, two, two, and four, respectively. We consider six time slots in this demonstration. In each time slot, there are two node unavailabilities. From types 1 to 5, the positions of unavailabilities become more random and scattered. The unavailabilities in the five types of schedules appear on two, three, four, five, and six nodes, respectively. We randomly generate ten availability schedules for each type and calculate the SSCATs and the sums of SCATs under all availability schedules. The average values of ten schedules belonging to the same type are shown in Table IV.

Table IV shows the SSCAT values for the five types of availability schedules given by the proposed model and the model in [10]. We observe that the proposed model provides a lower

TABLE IV
COMPARISON BETWEEN PROPOSED MODEL AND MODEL IN [10] WITH DIFFERENT AVAILABILITY SCHEDULES

Model	Type	SSCAT	Sum of SCATs
[10]	1	6.0	24.0
	2	3.0	18.9
	3	3.0	18.0
	4	2.9	15.8
	5	2.8	15.4
Proposed model	1	6.0	24.0
	2	3.0	19.3
	3	3.0	18.3
	4	3.0	18.0
	5	3.0	18.1



Note: □: switch or router. ◯: node to which the VNFs can be allocated.

Fig. 5. Two graphs for tests of network-aware allocation model.

improvement on SSCATs if the positions of unavailabilities are more compact, such as type 1, and provides a higher improvement on SSCATs if the positions of unavailabilities are more sparse, such as type 5.

C. Analysis of Network-Aware Allocation Model

This subsection evaluates the performance of the proposed network-aware model in Section III-C compared with the model in Section III-A which does not consider routing and the model in [10], which considers routing without backup VNFs,

We use four types of networks in this analysis: single data center of fat tree topology, multiple data centers dispersed geographically of fat tree topology [26], BCube topology [31], and visible light communication (VLC) networking topology [32]. The networks are shown in Fig. 5. □ represents the network devices, e.g., switches and routers. ◯ represents the nodes to which the VNFs can be allocated. Both two types of elements are considered as nodes in the model proposed in Section III-C. To distinguish different nodes, we set different capacities for them since VNFs cannot be allocated to switches and routers. The capacities of □ and ◯ are set to 0 and 2, respectively. We consider the allocations among six time slots. In Figs. 5(a) and (b), the available transmission resources of links between layers 0 and 1, 1 and 2, 2 and 3, and 3 and 4 are set to 16, 8, 4, and 2 units, respectively.

Time slot \ Node	1	2	3	4	5	6
1	U			U		
2		U				
3						U
4					U	
5		U		U		
6			U			

(a) Availability schedules for the tests using single data center and VLC networks.

Time slot \ Node	1	2	3	4	5	6	7	8	9	10	11	12
1	U		U				U					
2		U						U				
3						U						U
4					U						U	
5		U						U		U		
6			U						U			

(b) Availability schedules for the tests using double data center network.

Time slot \ Node	1	2	3	4	5	6	7	8	9
1	U			U			U		
2		U						U	
3						U			
4					U				
5		U						U	
6			U						U

(c) Availability schedules for the tests using BCube network.

Fig. 6. Availability schedules for tests of network-aware allocation model.

The end-to-end latencies of links between layers 0 and 1, 1 and 2, 2 and 3, and 3 and 4 are set to 8, 4, 2, and 1 units, respectively. In Fig. 5(c), the available transmission resources of links in layer 0 and those between layers 0 and 1 are set to 8 and 4 units, respectively. The end-to-end latencies of links in layer 0 and those between layers 0 and 1 are set to 1 and 4 units, respectively. In Fig. 5(d), the available transmission resources of links between the router and access points, and those between access points and terminals are set to 8 and 4 units, respectively. The end-to-end latencies of links between the router and access points, and those between access points and terminals are set to 1 and 8 units, respectively. In Figs. 5(a) and (d), we consider three requests with lengths of three, two, and two, respectively. The availability schedule used in these graphs is shown in Fig. 6(a). In Figs. 5(b) and (c), we consider four requests with lengths of four, three, two, and two. The availability schedules used in these graphs are shown in Figs. 6(b) and (c), respectively.

We observe from Table V that the network-aware proposed model reduces the latencies of services with keeping the same values of SCATs of services compared with the non-network-aware proposed model. Compared with the model in [10], the network-aware proposed model improves the value of SSCAT but the latencies of services also increase.

D. Analysis of Heuristic Algorithm

This subsection evaluates the performance of the proposed model with the heuristic algorithm introduced in Section IV. We compare the performances between the ILP approach and the heuristic algorithm in a 16-node network as the first test. We compare the performances among the proposed model with the heuristic algorithm and the five baseline models in a 100-node network during 365 time slots based on the Microsoft Azure maintenance records in [33].

The heuristic algorithm is designed to reduce the computation time within an acceptable performance degradation. We evaluate the performance of the proposed model with the heuristic algorithm in terms of the objective value calculated by (11), SSCAT, the sum of SCATs, and the computation time.

TABLE V
COMPARISON AMONG PROPOSED MODEL WITH AND WITHOUT NETWORK-AWARE ALLOCATION AND MODEL IN [10]

Model	Type†	SSCAT	Sum of SCATs	Sum of latencies
[10]	S	3	9	84
	D	4	16	244
	B	3	17	198
	V	3	9	402
Network-aware proposed model	S	4	15	96
	D	5	23	176
	B	5	22	216
	V	4	15	402
Non-network-aware proposed model	S	4	15	156
	D	5	23	1640
	B	5	22	580
	V	4	15	410

†Type S: fat tree, single database network. Type D: fat tree, double database network. Type B: BCube network. Type V: VLC network.

TABLE VI
PARAMETER SETTING IN HEURISTIC ALGORITHM

Parameter	Setting
MG	100
IP	12
UP	60
IC	0.6
EC	0.4
MP	0.3
BP	0.9

We consider a 16-node network, where the capacity of each node is set to five. We consider seven requests with lengths of three, two, two, three, two, two, and four, respectively. The recovery time of each function is set to one time slot. We consider six time slots in this analysis. In each time slot, there are five node unavailabilities. We randomly generate ten different availability schedules with different positions of the unavailable nodes in each time slot and calculate the values with these availability schedules under the same condition. We compare the results obtained by the proposed model with the heuristic algorithm, the ILP approach of the proposed model, and the ILP approach of the model in [10].

The heuristic algorithm is implemented by C++, compiled by Microsoft Visual Studio Community 2019 with version 16.8.3, using Intel Core i7-7700 3.60 GHz 4-core CPU, 32 GB memory.

Table VI shows the parameter settings used in this evaluation. Table VII shows the results in ten tests and Table VIII shows the average differences between the ILP approach and the heuristic algorithm in terms of objective values and computation times among ten tests. We observe that the heuristic algorithm reduces 66.75% computation time with a 1.57% performance degradation in terms of the objective value on average in the examined test cases. In test 10, the allocation result obtained by the heuristic algorithm has a lower objective value than that obtained by the ILP approach, but the objective value of the allocation obtained by the heuristic algorithm is still larger than that of the allocation obtained by [10].

TABLE VII
COMPARISONS AMONG ILP APPROACH OF [10] AND ILP APPROACH AND HEURISTIC ALGORITHM OF PROPOSED MODEL

Test	ILP approach of [10]				ILP approach of proposed model				Heuristic algorithm of proposed model			
	SSCAT	$\sum_{r\in R}\beta_r$	Obj. value	Time [s]	SSCAT	$\sum_{r\in R}\beta_r$	Obj. value	Time [s]	SSCAT	$\sum_{r\in R}\beta_r$	Obj. value	Time [s]
1	5	39	5.93	8.93	6	42	7.00	3.80	6	42	7.00	1.35
2	5	35	5.83	6.28	6	42	7.00	3.34	6	42	7.00	1.31
3	5	38	5.90	8.17	6	42	7.00	3.43	6	42	7.00	1.22
4	5	39	5.93	11.95	6	42	7.00	3.72	6	42	7.00	1.13
5	5	39	5.93	9.20	6	42	7.00	3.40	6	42	7.00	1.32
6	3	27	3.64	9.84	6	42	7.00	5.62	6	42	7.00	1.76
7	5	38	5.90	13.06	6	42	7.00	3.69	6	42	7.00	1.54
8	5	38	5.90	10.13	6	42	7.00	4.33	6	42	7.00	1.17
9	5	39	5.93	9.80	6	42	7.00	3.37	6	42	7.00	1.19
10	4	34	4.81	26.71	6	42	7.00	5.03	5	37	5.88	1.19

TABLE VIII
COMPARISON OF AVERAGE COMPUTATION TIMES AND OBJECTIVE VALUES OBTAINED BY ILP APPROACH AND HEURISTIC ALGORITHM

Method	Computation times [s]	Objective values
ILP approach	3.97	7.00
Heuristic algorithm	1.32	6.89
Difference (%)	66.75	1.57

TABLE IX
COMPARISON OF AVERAGE COMPUTATION TIMES AND OBJECTIVE VALUES OBTAINED BY PROPOSED MODEL WITH HEURISTIC ALGORITHM AND FOUR BASELINE MODELS

Method	Computation times [s]	Objective values
Proposed model	22248.00	14.05
Model in [10]	22076.40	13.06
Double-slot placement	2298.609	9.034
Single-slot placement	517.6713	2.010
Persistent placement	1.345946	6.032

We need to decide the allocations on a 100-node network during 365 time slots in the second test. We consider 40 requests. The lengths of three requests are seven. The lengths of 10 requests are five. The lengths of 20 requests are four. The lengths of five requests are three. The lengths of two requests are two. The recovery time of each function is set to one time slot. The capacity of each node is two. The heuristic algorithms of the proposed model, the model in [10] and the persistent placement model are implemented by C++, compiled by Microsoft Visual Studio Community 2019 with version 16.8.3, using Intel Core i7-7700 3.60 GHz 4-core CPU, 32 GB memory. The other baseline models are solved by the IBM(R) ILOG(R) CPLEX(R) Interactive Optimizer with version 12.10.0 and run on the same hardware.

Table IX shows the comparison results. We can observe that the proposed model provides the maximum objective value and has the longest computation time among the five examined models in this case. The single-slot placement outperforms the persistent placement since the unavailabilities in the given availability schedule are loose. The effect of the unavailable nodes on the persistent placement is weaker than that of the reallocations between different time slots on the single-slot placement. If we consider both effects by using the double-slot placement, the double-slot placement outperforms both of them in terms of the objective value with the cost of longer computation time. The proposed model outperforms the model in [10] with less cost on computation time. It is valuable to take backup measures by using the proposed model in the examined case compared with the model in [10].

VI. RELATED WORKS

The VNF allocation problem considered by the proposed model is impacted by the service interruptions. There are also some existing studies caring about the interruptions. The work in [34] concerned the low reliability of softwarized networks caused by service interruptions. The model presented in [25] minimizes the cost of migration which is evaluated by the number of migrations. The work in [35] introduced a model of the adaptive and dynamic VNF allocation problem considering the interruptions caused by VNF migration. The work in [36] introduced a model for dynamic VNF placement under changing traffic load. A static placement decreases the operation cost. Reconfiguration causes service interruptions and the operation cost increases. Compared with the above existing models, the proposed model considers a sequence of time slots at one time so that the model only calculates the placement at the beginning of service deployments instead of at each time slot. In addition, the proposed model uses backup VNFs to prevent the service interruptions caused by VNF reallocation.

Several studies considered the backups in SFC. The works in [37], [38] studied the placement methods for active VNFs and backup VNFs with flow and SFC parallelism. The models split a large flow into multiple parallel smaller sub-flows and any SFC is replicated into multiple sub-SFCs. The models aimed to increase service reliable probability while fewer backup resources are required. The work in [39] designed a redundancy mechanism to protect the service from interruptions by introducing a node-ranking algorithm. The mechanism reduces the consumption of backup resources with respect to a higher acceptance ratio of SFC requests. The work in [40] introduced a backup model that combines path backup and VNF backup in a joint way. The backup model aimed to reduce resource consumption for backups. The work in [41] introduced a method to allocate SFCs aiming to maximize the number of SFC requests that can be served while meeting their heterogeneous availability requirements, which includes an

ILP model for one SFC and a heuristic algorithm for mapping multiple SFC requests. The work also introduced a backup pooling mechanism to further improve the efficiency of backup resource usage. The work in [42] presented a framework to provide availability of SFC requests with the objective of minimizing resource usage including an optimization problem of SFC mapping and backup estimation. The work in [43] introduced a coordinated protection mechanism that adopts both backup path protection in the network and VNF replicas at nodes to guarantee an SFC's availability aiming to reduce the SFC blocking and the cost of computational resources.

Compared with the existing studies, the objective of our proposed model is different. The existing studies care about end-to-end reliability, which is evaluated by the metrics such as an available probability or a mean time between failures. Our model cares about time sensitive services which are evaluated by the continuous servable time. In addition, the existing models only consider one possible error pattern which may lead to service interruptions at one time. The proposed model considers the error patterns in a sequence of time slots and the reallocations between different allocations in the adjacent time slots. Moreover, the proposed model gives an initial allocation for VNFs in SFCs. It focuses on the optimal allocation at the beginning of function deployment instead of after a disturbance.

VII. Conclusion

This paper proposed a primary and backup VNF placement model for improving the continuous available time of service function chains by avoiding the interruptions caused by unavailable nodes and function reallocations. We formulated the proposed model as an ILP problem that maximizes the minimum number of the longest continuous available time slots in each SFC by considering deterministic availability schedules. We extended the proposed model to a network-aware one with considering a routing problem. Evaluation results showed that the proposed model improves the continuous available time of SFCs, compared with the baseline models in the examined cases. The network-aware proposed model reduces the latencies of services with keeping the same values of SCATs of services compared with the non-network-aware proposed model. We introduced an algorithm that estimates the number of key unavailabilities at each time slot, which indicates the number of unavailable nodes that are required to be eliminated priorly. The number of key unavailabilities helps SPs to increase the service continuous available time with the least cost of failure recovery. We analyzed the impact of different types of availability schedules on the proposed model. In the examined test cases, the proposed model provides a lower improvement on SSCATs if the positions of unavailabilities are more compact, and provides a higher improvement on SSCATs if the positions of unavailabilities are more sparse. We developed and analyzed a heuristic algorithm to speed up the computation for the case that the problem size increases. The heuristic algorithm reduces 66.75% computation time with a 1.57% performance degradation in terms of the objective value on average in the examined test cases. We provided the discussion on dealing with multiple replicas of a function and their backups.

Appendix A
Linearization of Proposed Model

For the sake of brevity and readability, let $(r,k,n,t) \in \Phi$ and $(r,k,n,t) \in \Phi'$ denote $r \in R, k \in K_r, n \in N, t \in T\backslash[1, g_r^k]$ and $r \in R, k \in K_r, n \in N, t \in T\backslash\{1\}$, respectively.

Equation (1) is linearized to (24a)-(24q), (9) is linearized to (25a)-(25f), (4) is linearized to (26a)-(26s), and (21) is linearized to (27a)-(27h) with some auxiliary variables as follows:

$$\gamma_{tn}^{rk} \le \sum_{n' \in N\backslash\{n\}} e_t^{n'} x_{t-1,n'}^{rk}, \forall (r,k,n,t) \in \Phi, \tag{24a}$$

$$\gamma_{tn}^{rk} \ge \frac{1}{|N|-1} \sum_{n' \in N\backslash\{n\}} e_t^{n'} x_{t-1,n'}^{rk}, \quad \forall (r,k,n,t) \in \Phi, \tag{24b}$$

$$\mu_{tn}^{rk} \le u_{t'n}^{rk}, \forall r \in R, k \in K_r, n \in N, t \in T\backslash\left[1, g_r^k\right], \quad t' \in T\backslash\left[t - g_r^k, t-1\right], \tag{24c}$$

$$\mu_{tn}^{rk} \le x_{tn}^{rk}, \forall (r,k,n,t) \in \Phi, \tag{24d}$$

$$\mu_{tn}^{rk} \ge \sum_{t' \in T\backslash[t-g_r^k, t-1]} u_{t'n}^{rk} - (|T| - g_r^k) + x_{tn}^{rk}, \forall (r,k,n,t) \in \Phi, \tag{24e}$$

$$\tau_{tn}^{rk} \le \sum_{n' \in N\backslash\{n\}} \mu_{tn'}^{rk}, \forall (r,k,n,t) \in \Phi, \tag{24f}$$

$$\tau_{tn}^{rk} \ge \frac{1}{|N|-1} \sum_{n' \in N\backslash\{n\}} \mu_{tn'}^{rk}, \forall (r,k,n,t) \in \Phi, \tag{24g}$$

$$\xi_{tn}^{rk} \le e_t^n x_{t-1,n}^{rk}, \forall (r,k,n,t) \in \Phi, \tag{24h}$$

$$\xi_{tn}^{rk} \le \tau_{tn}^{rk}, \forall (r,k,n,t) \in \Phi, \tag{24i}$$

$$\xi_{tn}^{rk} \ge \tau_{tn}^{rk} + e_t^n x_{t-1,n}^{rk} - 1, \forall (r,k,n,t) \in \Phi, \tag{24j}$$

$$\psi_{tn}^{rk} \le \mu_{tn}^{rk}, \forall (r,k,n,t) \in \Phi, \tag{24k}$$

$$\psi_{tn}^{rk} \le \gamma_{tn}^{rk}, \forall (r,k,n,t) \in \Phi, \tag{24l}$$

$$\psi_{tn}^{rk} \ge \mu_{tn}^{rk} + \gamma_{tn}^{rk} - 1, \forall (r,k,n,t) \in \Phi, \tag{24m}$$

$$\alpha_{tn}^{rk} \le \psi_{tn}^{rk} + \xi_{tn}^{rk}, \forall (r,k,n,t) \in \Phi, \tag{24n}$$

$$\alpha_{tn}^{rk} \ge 0.5(\psi_{tn}^{rk} + \xi_{tn}^{rk}), \forall (r,k,n,t) \in \Phi, \tag{24o}$$

$$\alpha_{tn}^{rk} \in \{0,1\}, \forall r \in R, k \in K_r, t \in \left[2, g_r^k\right], n \in N, \tag{24p}$$

$$\gamma_{tn}^{rk}, \mu_{tn}^{rk}, \psi_{tn}^{rk}, \xi_{tn}^{rk}, \tau_{tn}^{rk} \in \{0,1\}, \forall (r,k,n,t) \in \Phi, \tag{24q}$$

$$\beta_r - 1 \le j y_j^r + \left(1 - \delta_j^r\right) \cdot B, \forall j \in T, r \in R, \tag{25a}$$

$$\beta_r - 1 \ge j y_j^r - \left(1 - \delta_j^r\right) \cdot B, \forall j \in T, r \in R, \tag{25b}$$

$$j y_j^r \ge \left(\delta_j^r - 1\right) \cdot B + j' y_{j'}^r, \forall j \in T, j' \in T\backslash\{j\}, r \in R, \tag{25c}$$

$$\sum_{j \in T} \delta_j^r = 1, \forall r \in R, \tag{25d}$$

$$\beta_r - 1 \ge j y_j^r, \forall j \in T, r \in R, \tag{25e}$$

$$\delta_j^r, y_j^r \in \{0,1\}, \forall j \in T, r \in R, \tag{25f}$$

$$\phi_{tn}^{rk} = 1 - x_{tn}^{rk} - x_{t-1,n}^{rk} + 2 \cdot h_{tn}^{rk}, \forall (r,k,n,t) \in \Phi', \quad (26a)$$
$$h_{tn}^{rk} \leq x_{tn}^{rk}, \forall (r,k,n,t) \in \Phi', \quad (26b)$$
$$h_{tn}^{rk} \leq x_{t-1,n}^{rk}, \forall (r,k,n,t) \in \Phi', \quad (26c)$$
$$h_{tn}^{rk} \geq x_{tn}^{rk} + x_{t-1,n}^{rk} - 1, \forall (r,k,n,t) \in \Phi', \quad (26d)$$
$$\rho_{tn}^{rk} \leq \phi_{tn}^{rk} + \alpha_{tn}^{rk}, \forall (r,k,n,t) \in \Phi', \quad (26e)$$
$$\rho_{tn}^{rk} \geq \frac{1}{2}(\phi_{tn}^{rk} + \alpha_{tn}^{rk}), \forall (r,k,n,t) \in \Phi', \quad (26f)$$
$$\pi_{tn}^{rk} \leq \rho_{tn}^{rk}, \forall (r,k,n,t) \in \Phi', \quad (26g)$$
$$\pi_{tn}^{rk} \leq 1 - e_t^n x_{tn}^{rk}, \forall (r,k,n,t) \in \Phi', \quad (26h)$$
$$\pi_{tn}^{rk} \leq 1 - e_{t-1}^n x_{t-1,n}^{rk}, \forall (r,k,n,t) \in \Phi', \quad (26i)$$
$$\pi_{tn}^{rk} \geq \rho_{tn}^{rk} - e_t^n x_{t,n}^{rk} - e_{t-1}^n x_{t-1,n}^{rk}, \forall (r,k,n,t) \in \Phi', \quad (26j)$$
$$w_t^{rk} \leq \pi_{tn}^{rk}, \forall (r,k,n,t) \in \Phi', \quad (26k)$$
$$w_t^{rk} \geq \sum_{n \in N} \pi_{tn}^{rk} - |N| + 1, \forall r \in R, t \in T \backslash \{1\}, k \in K_r, \quad (26l)$$
$$o_t^r \leq w_t^{rk}, \forall r \in R, t \in T \backslash \{1\}, k \in K_r, \quad (26m)$$
$$o_t^r \geq \sum_{k \in K_r} w_t^{rk} - |K_r| + 1, \forall r \in R, t \in T \backslash \{1\}, \quad (26n)$$
$$o_1^r = 1, \forall r \in R, \quad (26o)$$
$$o_t^r \in \{0,1\}, \forall r \in R, t \in T, \quad (26p)$$
$$w_t^{rk} \in \{0,1\}, \forall r \in R, k \in K_r, t \in T \backslash \{1\}, \quad (26q)$$
$$x_{tn}^{rk}, u_{tn}^{rk} \in \{0,1\}, \forall r \in R, k \in K_r, t \in T, n \in N, \quad (26r)$$
$$\alpha_{tn}^{rk}, \phi_{tn}^{rk}, h_{tn}^{rk}, \pi_{tn}^{rk}, \rho_{tn}^{rk} \in \{0,1\}, \forall (r,k,n,t) \in \Phi'. \quad (26s)$$

$$\sum_{(i,j) \in L} a_{w,ij} q_{rt}^{k,ij} = -\zeta_{tw}^{rk} + \iota_{tw}^{rk}, \forall k \in K_r, r \in R, t \in T, w \in N, \quad (27a)$$
$$\zeta_{tw}^{rk} \leq x_{tw}^{rk}, \forall k \in K_r, r \in R, t \in T, w \in N, \quad (27b)$$
$$\zeta_{tw}^{rk} \leq \sum_{w' \in N} u_{tw'}^{rk}, \forall k \in K_r, r \in R, t \in T, w \in N, \quad (27c)$$
$$\zeta_{tw}^{rk} \geq x_{tw}^{rk} + \sum_{w' \in N} u_{tw'}^{rk} - 1, \forall k \in K_r, r \in R, t \in T, w \in N, \quad (27d)$$
$$\iota_{tw}^{rk} \leq u_{tw}^{rk}, \forall k \in K_r, r \in R, t \in T, w \in N, \quad (27e)$$
$$\iota_{tw}^{rk} \leq \sum_{w' \in N} x_{tw'}^{rk}, \forall k \in K_r, r \in R, t \in T, w \in N, \quad (27f)$$
$$\iota_{tw}^{rk} \geq u_{tw}^{rk} + \sum_{w' \in N} x_{tw'}^{rk} - 1, \forall k \in K_r, r \in R, t \in T, w \in N, \quad (27g)$$
$$\zeta_{tw}^{rk}, \iota_{tw}^{rk} \in \{0,1\}, \forall k \in K_r, r \in R, t \in T, w \in N. \quad (27h)$$

In the above equations, B is a given constant value that satisfies $B > jy_j^r, \forall j \in T, r \in R$; the minimum of B can be taken to be $|T| + 1$.

APPENDIX B
PSEUDOCODES USED IN SECTION IV

A. Main Function

The main function of the heuristic algorithm is shown in Algorithm 3. The algorithm calculates the allocations of primary and backup VNFs in given requests.

Algorithm 3 Main Function

Input: *N, T, R,* K_r, $c_{nt}^s \in C$, *IP, MG, EC, IC, MP, UP, BP*
Output: primary and backup function allocations
1: $I \leftarrow$ Generate set of initial feasible solutions by using function *init_chromos* in Algorithm 6
2: **if** $I = \emptyset$ **then**
3: **return** No feasible solution
4: **end if**
5: **for** $step = 1 \rightarrow$*MG* **do**
6: Define I_n as the new feasible solution set
7: **for** each solution in *I* **do**
8: **if** a random number in [0, 1] $>$ 1 - *IC* **then**
9: $I_n \leftarrow$Generate a non-redundant and mutant solution by using function *cross_in* in Algorithm 4 whose inputs are the selected solution in *I* and random time slot *t*
10: **end if**
11: **end for**
12: **for** each solution in *I* except for the first one **do**
13: **if** a random number [0, 1] $>$ 1 - *EC* **then**
14: $I_n \leftarrow$Generate a non-redundant and mutant solution by using function *cross_out* in Algorithm 4 whose inputs are the selected solution and its previous solution in *I*
15: **end if**
16: **end for**
17: **for** each solution in *I* **do**
18: **if** a random number [0, 1] $>$ 1 - *MP* **then**
19: $I_n \leftarrow$Generate a non-redundant and mutant solution by using function *mutation* in Algorithm 7 whose input is the selected solution in *I*
20: **end if**
21: **end for**
22: Integrate I_n into *I*
23: Calculate the fitness score of the solutions in *I* by using function *calc_fin_ness* in Algorithm 8
24: Store the solution with the highest fitness score
25: **if** size of $I >$*UP* **then**
26: Reduce the size of the set to *UP* by using function *roulette_gambler* in Algorithm 5
27: **end if**
28: **end for**

Algorithm 4 Crossover

1: **function** CROSS_IN($s \leftarrow$ the selected solution , $t \leftarrow$ the random time)
2: $s[t] \leftarrow s[t+1]$
3: **return** s
4: **end function**
5: **function** CROSS_OUT(s_1, s_2)
6: $location \leftarrow$ a random integer in $[1, |T|]$
7: $s_c \leftarrow s_1$
8: $s_c[location] \leftarrow s_2[location]$
9: **return** s_c
10: **end function**

In line 1, Algorithm 6 introduced in Appendix B-D gives a set of initial feasible solutions for primary function allocation whose size is the initial population (*IP*). In lines 2-4, if Algorithm 6 cannot provide a feasible solution, the heuristic algorithm reports that no feasible solution is found. In lines 5-28, the heuristic algorithm enters a loop with maximum generation (*MG*) cycles. In each cycle, the heuristic algorithm generates new feasible solutions by performing internal crossover (see function CROSS IN in Algorithm 4 of Appendix B-B), external crossover (see function CROSS OUT in Algorithm 4 of Appendix B-B), and mutation (see Algorithm 7 of Appendix B-E) according to three probabilities: internal crossover probability (*IC*); external crossover probability (*EC*); and mutation probability (*MP*), respectively. Newly generated solutions in lines 9, 14, and 19 are stored in

Algorithm 5 Choice

```
1: function ROULETTE_GAMBLER(fit_pros, chroms)
2:     pick ← a random number in [0, 1]
3:     for j = 1 → |chroms| do
4:         pick ← pick − fit_pros[j]/sum(fit_pros)
5:         if pick ≤ 0 then
6:             return j
7:         end if
8:     end for
9:     return |chroms| − 1
10: end function
11: function CHOICE(chroms, fit_pros)
12:     choice_gens ← φ
13:     for i = 1 → min(|chroms|,UP) do
14:         j ← ROULETTE_GAMBLER(fit_pros, chroms)
15:         append chroms[j] to choice_gens
16:     end for
17:     return choice_gens
18: end function
```

Algorithm 6 Initial Solution

```
1: function INIT_CHROMOS(E)
2:     Set of initial solutions s ← φ
3:     Sort requests in R in non-increasing order of K_r
4:     for each time slot in T do
5:         Sort nodes in N in non-increasing order of time from time slot t to
       a time slot in which a node becomes unavailable. If the above values are
       the same for some n, sort them in non-increasing order of time from time
       slot t to a time slot in which a node becomes unavailable secondly. If the
       above values are the same for some n, sort them in non-increasing order
       of time from time slot t to the last time slot in which a node becomes
       unavailable.
6:         for r ∈ R do
7:             for f = 1 → K_r do
8:                 for n ∈ N do
9:                     if used capacity of n is less than c_n^t AND any other
       functions in r were not allocated to n then
10:                        Allocate the fth function in SFC r to n
11:                        Break
12:                    else
13:                        Continue
14:                    end if
15:                end for
16:            end for
17:        end for
18:        Store the allocation to s
19:    end for
20:    Duplicate a solution iteratively until the number of solutions in s
       becomes IP
21:    return s
22: end function
```

Algorithm 7 Mutation

```
1: function MUTATION(s ← the selected solution, E)
2:     Calculate the SCAT for all SFCs in solution s
3:     Sort requests in R in a weighted randomized order. The larger the
       SCAT, the higher the order of the corresponding r.
4:     Sort nodes in N in a weighted randomized order. The larger the time
       from the first time slot to a time slot where a node becomes unavailable,
       the higher the order of the corresponding n.
5:     for r ∈ R do
6:         for f = 1 → K_r of request r do
7:             for n ∈ N do
8:                 if used capacity of n is less than c_n^t AND any other
       functions in r were not allocated to n then
9:                     Allocate the fth function in SFC r to n
10:                    Break
11:                else
12:                    Continue
13:                end if
14:            end for
15:        end for
16:    end for
17:    return new solution
18: end function
```

the new feasible solution set. They are added to the feasible solution set at a time in line 22. In line 23, the heuristic algorithm calculates the fitness for each solution and the backup function allocations for each function, and then returns the fitness score and the backup allocations (see Algorithm 8 in Appendix B-F). In line 24, the genetic algorithm finds the solution with the highest fitness score and stores the primary and backup function allocation. In lines 25-27, if the size of the feasible solution set exceeds the upper limitation of population (*UP*), the heuristic algorithm chooses *UP* feasible solutions as a new set of feasible solutions according to roulette gambler (see Algorithm 5 in Appendix B-C).

B. Crossover

The internal crossover, function *cross_in* in Algorithm 4, crosses adjacent time slots in the same solution. The aim of *cross_in* is to suppress the reallocations of VNFs between adjacent time slots.

The external crossover, function *cross_out* in Algorithm 4, crosses the same time slot between two solutions in the feasible solution set. A new solution is generated by modifying the VNF allocation in a randomly selected time slot of one solution based on that of another solution.

C. Choice of Solutions

The heuristic algorithm uses roulette wheel selection to create a new feasible solution set by choosing *UP* solutions from the feasible solution set.

In the *roulette_gambler* and *choice* functions in Algorithm 5, input *chroms* is the set of solutions and *fit_pros* is the set of the fitness scores of the corresponding solutions in *chroms*.

D. Initial Solution Generation

Algorithm 6 reorders set *R* according to the corresponding K_r from long to short first (line 3). Then, it performs function allocation one by one (lines 4-19). At each time slot, the heuristic algorithm reorders set *N* according to the occurrence of unavailabilities from late to early (line 5). Then the heuristic algorithm allocates the functions to physical nodes according to these new orders (lines 9-13). Finally, the heuristic algorithm duplicates a solution iteratively until the number of solutions becomes initial population (IP) (line 20).

E. Mutation

See Algorithm 7.

F. Calculation of Fitness

See Algorithm 8.

APPENDIX C
EXTENSIONS FOR PROPOSED MODEL IN SECTION III

A. Separate SFCs From Requests

We assume that one request corresponds to one SFC in Section III. There may be several requests sharing the same

Algorithm 8 Fitness Calculation

function CALC_FIN_NESS($s \leftarrow$ the selected solution, E)
  Calculate the available resources of each node in solution s at each time slot
  for $r \in R$ **do**
    for $t \in T \backslash \{1\}$ **do**
      if Allocation of functions in request r at time slot t is different from that at time slot $t-1$ **then**
        for $f = 1 \rightarrow K_r$ of request r **do**
          if the allocations of the kth function in request r at time slot t and $t-1$ are different AND t is larger than g_r^k **then**
            $n \leftarrow$ the allocation of the kth function in request r at time slot t
            for $t2 \leftarrow t - g_r^k 1, \ldots, t-1$ **do**
              if n has no enough resources at time slot $t2$ OR any other function of request r is not allocated to n at time slot $t2$ **then**
                Request r is not continuous between time slots $t-1$ and t
                Break
              end if
              diff $\leftarrow (t, r, k, n)$
            end for
          end if
        end for
        if Continuity of request r between time slots $t-1$ and t has not been decided AND *diff* is not empty AND there are enough resources for allocations stored in *diff* AND a random number [0, 1] > 1 - *BP* **then**
          Request r is continuous between time slots $t-1$ and t
          Merge set *diff* to backup function allocations and update the available resources of each node
        else
          Request r is not continuous between time slots $t-1$ and t
        end if
      else if Allocate any function in request r on an unavailable node at time slot t or $t-1$ **then**
        Request r is not continuous between time slots $t-1$ and t
      else
        Request r is continuous between time slots $t-1$ and t
      end if
    end for
  end for
  Calculate the SCAT for all SFCs in solution s
  return $\min(SCAT)$+sum$(SCAT)/(|T| \times |R|)$, backup allocations
end function

SFC with different requests of package processing abilities. We give an extension of the proposed model to separate the requests from the SFCs. We redefine the requests, SFCs, VNFs, and recovery time to replace the definitions in paragraphs 1 and 4 of Section III-A. The definitions of decision variables x_{tn}^{rk} and u_{tn}^{rk} in paragraphs 3 and 4 of Section III-A are changed as follows.

R represents the set of requests from the users. *C* represents the set of SFCs waiting for provisions. Each SFC is an ordered set of VNFs. *F* is the set of functions. $F^c \subseteq F$ is the ordered set of VNFs used in SFC $c \in C$. Binary given parameter ψ_{fc} is set to 1 if function $f \in F$ is used in SFC $c \in C$; 0 otherwise. Binary given parameter γ_{cr}, $r \in R$, $c \in C$, is set to 1 if request r requests SFC c; 0 otherwise. Given parameter q_s^f represents the amount of resource $s \in S$ which function $f \in F$ requires. g_f is a given parameter which represents the recovery time of function $f \in F$.

We use binary decision variable x_{tn}^f to represent the allocation of primary VNF; x_{tn}^f is set to 1 if function $f \in F$ is assigned to node $n \in N$ at time slot $t \in T$; 0 otherwise. We use binary decision variable u_{tn}^f to represent the allocation of backup VNF; u_{tn}^f is set to 1 if function $f \in F$ is assigned to node $n \in N$ at time slot $t \in T$; 0 otherwise.

α_{tn}^{rk} in paragraph 6 of Section III-A is redefined to α_{tn}^f, which denotes a binary variable which is set to 1 if function $f \in F$ is prevented from the interruption happening on node $n \in N$ at time slot $t \in T$; 0 otherwise, $\forall f \in F, n \in N$, α_{tn}^f is expressed by:

$$\alpha_{tn}^{rk} = \begin{cases} \left\{ \left(\vee_{n' \in N \backslash \{n\}} e_t^{n'} x_{t-1,n'}^f \right) \wedge \left(\left(\prod_{t' \in [t-g_f,\ t-1]} u_{t'n}^f \right) \wedge x_{tn}^f \right) \right\} \vee \left\{ \left(e_t^n x_{t-1,n}^f \right) \wedge \left(\vee_{n' \in N \backslash \{n\}} \left(\prod_{t' \in [t-g_f,\ t-1]} u_{t'n'}^f \right) \wedge x_{tn'}^f \right) \right\}, & t \in T \backslash [1, g_f] \\ 0, & t \in [2, g_f], \end{cases} \tag{28}$$

According to the above redefinitions, we give a new version of the related parameters and constraints as follows:

$$o_t^r = \prod_{n \in N} \left\{ \left(\prod_{c \in C} \prod_{f \in F^c} \psi_{fc} \gamma_{cr} (x_{tn}^f \odot x_{t-1,n}^f) \right) \wedge \left(1 - \left(\prod_{c \in C} \prod_{f \in F^c} \psi_{fc} \gamma_{cr} x_{tn}^f \right) \wedge e_t^n \right), \wedge \left(1 - \left(\prod_{c \in C} \prod_{f \in F^c} \psi_{fc} \gamma_{cr} x_{t-1,n}^f \right) \wedge e_{t-1}^n \right) \right\}, \quad \forall r \in R, t \in T \backslash \{1\}, \tag{29a}$$

$$\sum_{f \in F} (x_{tn}^f + u_{tn}^f) q_s^f \leq c_t^n, \forall n \in N, t \in T, \tag{29b}$$

$$\sum_{f \in F^c} (x_{tn}^f + u_{tn}^f) \leq 1, \forall c \in C, n \in N, t \in T, \tag{29c}$$

$$\sum_{n \in N} x_{tn}^f = 1, \forall f \in F, t \in T, \tag{29d}$$

$$\sum_{n \in N} u_{tn}^f \leq 1, \forall f \in F, t \in T. \tag{29e}$$

Equation (29a) replaces x_{tn}^{rk} in (4) with $\prod_{c \in C} \prod_{f \in F^c} \psi_{fc} \gamma_{cr} x_{tn}^f$. Equation (29b) replaces (12) and ensures that each node's computational resources must not exceed its capacity during allocation. Equation (29c) replaces (13) and assumes that one service chain does not allocate multiple VNFs in this chain on one VM to avoid the influence of the reallocation of VMs [25]. Equation (29d) replaces (14) and ensures that all functions are allocated in the network. Equation (29e) replaces (15) and assumes that each VNF has at most one backup at one time slot.

B. Replicas for Primary and Backup VNFs

VNF replicas in an SFC are used to prevent service interruptions from the failures of functions and load balancing. In Section III, the proposed model only uses at most two replicas, one for primary VNF and one for backup VNF, which are limited by (14) and (15). A possible extension for the proposed

model is to support multiple replicas for primary and backup VNFs.

Let ρ_t^{rk} be an non-negative integer given parameter, which represents that ρ_t^{rk} active replicas are necessary at time slot $t \in T$ for the $k \in K_r$th function in SFC $r \in R$. Equations (14) and (15) are replaced by:

$$\sum_{n \in N} x_{tn}^{rk} = \rho_t^{rk} \forall r \in R, k \in K_r, t \in T. \tag{30}$$

C. Considering Numbers of Accepted SFCs

The models in Section III do not consider maximizing the acceptance ratio. There is no feasible solution if any requested SFCs are not accepted. Sometimes the system may receive too many SFCs and not all of them can be accepted and allocated. We can consider maximizing the number of accepted SFCs as the objective of the proposed model. We give the following model:

$$\max \sum_{r \in R} \prod_{t \in T} \prod_{k \in K_r} \sum_{n \in N} x_{tn}^{rk} \tag{31a}$$

$$\text{s.t. } (2)-(3),\ (5)-(8),\ (10),\ (12)-(13),\ (15),\ (24a)-(26s), \tag{31b}$$

$$\lambda \geq \lambda_{\min}, \tag{31c}$$

$$\sum_{n \in N} x_{tn}^{rk} \leq 1, \forall r \in R, k \in K_r, t \in T, \tag{31d}$$

$$\beta_r, \lambda \in [1, |T|], \forall r \in R, i \in T, j \in T_i, \tag{31e}$$

$$z_i^{jr} \in \{0,1\}, \forall r \in R, i \in T, j \in T_i. \tag{31f}$$

Equation (14) is replaced by (31d), which ensures that there is at most one primary function for each VNF. $\sum_{r \in R} \prod_{t \in T} \prod_{k \in K_r} \sum_{n \in N} x_{tn}^{rk}$ represents the number of accepted SFCs. $\lambda_{\min}$ is a given parameter, which represents the specified minimum SSCAT that the SFCs should satisfy.

REFERENCES

[1] A. Gember, P. Prabhu, Z. Ghadiyali, and A. Akella, "Toward software-defined middlebox networking," in *Proc. 11th ACM Workshop Hot Topics Netw.*, 2012, pp. 7–12.

[2] A. Lombardo, A. Manzalini, V. Riccobene, and G. Schembra, "An analytical tool for performance evaluation of software defined networking services," in *Proc. IEEE Netw. Oper. Manage. Symp. (NOMS)*, 2014, pp. 1–7.

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

[4] J. G. Herrera and J. F. Botero, "Resource allocation in NFV: A comprehensive survey," *IEEE Trans. Netw. Service Manag.*, vol. 13, no. 3, pp. 518–532, Sep. 2016.

[5] E. J. Halpern and E. C. Pignataro, "Service function chaining (SFC) architecture," Internet Eng. Task Force, Fremont, CA, USA, RFC 7665, Oct. 2015.

[6] F. He, T. Sato, and E. Oki, "Backup resource allocation model for virtual networks with probabilistic protection against multiple facility node failures," in *Proc. 15th Int. Conf. Design Rel. Commun. Netw. (DRCN)*, 2019, pp. 37–42.

[7] M. Zhu, F. He, and E. Oki, "Optimal multiple backup resource allocation with workload-dependent failure probability," in *Proc. IEEE Globecom*, 2020, pp. 1–6.

[8] R. Wen *et al.*, "On robustness of network slicing for next-generation mobile networks," *IEEE Trans. Commun.*, vol. 67, no. 1, pp. 430–444, Jan. 2019.

[9] R. Kang, F. He, T. Sato, and E. Oki, "Virtual network function allocation to maximize continuous available time of service function chains," in *Proc. IEEE 8th Int. Conf. Cloud Netw. (CloudNet)*, Nov. 2019, pp. 1–6.

[10] R. Kang, F. He, T. Sato, and E. Oki, "Virtual network function allocation to maximize continuous available time of service function chains with availability schedule," *IEEE Trans. Netw. Service Manag.*, vol. 18, no. 2, pp. 1556–1570, Jun. 2021.

[11] H. Abid and N. Samaan, "A novel scheme for node failure recovery in virtualized networks," in *Proc. IFIP/IEEE Int. Symp. Integr. Netw. Manage. (IM)*, 2013, pp. 1154–1160.

[12] M. Raza, V. Samineni, and W. Robertson, "Physical and logical topology slicing through SDN," in *Proc. IEEE Can. Conf. Electr. Comput. Eng. (CCECE)*, 2016, pp. 1–4.

[13] M. Wessler, *Oracle DBA on Unix and Linux*. Indianapolis, IN, USA: SAMS, 2001.

[14] T. Sato, F. He, E. Oki, T. Kurimoto, and S. Urushidani, "Implementation and testing of failure recovery based on backup resource sharing model for distributed cloud computing system," in *Proc. IEEE 7th Int. Conf. Cloud Netw. (CloudNet)*, 2018, pp. 1–3.

[15] L. Sun, J. An, Y. Yang, and M. Zeng, "Recovery strategies for service composition in dynamic network," in *Proc. Int. Conf. Cloud Service Comput.*, 2011, pp. 60–64.

[16] R. Kang, F. He, and E. Oki, "Optimal virtual network function placement in chains using backups with availability schedule," in *Proc. IEEE 9th Int. Conf. Cloud Netw. (CloudNet)*, Nov. 2020, pp. 1–6.

[17] S. Herker, X. An, W. Kiess, S. Beker, and A. Kirstaedter, "Data-center architecture impacts on virtualized network functions service chain embedding with high availability requirements," in *Proc. IEEE Globecom Workshops (GC Wkshps)*, 2015, pp. 1–7.

[18] M. Pióro, M. Mycek, and A. Tomaszewski, "Network protection against node attacks based on probabilistic availability measures," *IEEE Trans. Netw. Service Manag.*, early access, Mar. 22, 2021, doi: 10.1109/TNSM.2021.3067775

[19] T. Kimura *et al.*, "Spatio-temporal factorization of log data for understanding network events," in *Proc. IEEE INFOCOM Conf. Comput. Commun.*, 2014, pp. 610–618.

[20] S. Yang, F. Li, R. Yahyapour, and X. Fu, "Delay-sensitive and availability-aware virtual network function scheduling for NFV," *IEEE Trans. Services Comput.*, early access, Jul. 9, 2019, doi: 10.1109/TSC.2019.2927339.

[21] D. Li, P. Hong, K. Xue, and J. Pei, "Availability aware VNF deployment in datacenter through shared redundancy and multi-tenancy," *IEEE Trans. Netw. Service Manag.*, vol. 16, no. 4, pp. 1651–1664, Dec. 2019.

[22] The Kubernetes Authors. *Kubernetes*. Accessed: Nov. 10, 2020. [Online]. Available: https://kubernetes.io/

[23] The Kubernetes Authors. *Scheduling Framework*. Accessed: Nov. 8, 2020. [Online]. Available: https://kubernetes.io/docs/concepts/scheduling-eviction/scheduling-framework/

[24] R. Kang, F. He, T. Sato, and E. Oki, "Demonstration of network service header based service function chain application with function allocation model," in *Proc. IEEE/IFIP Netw. Oper. Manage. Symp.*, Apr. 2020, pp. 1–2.

[25] F. Carpio, A. Jukan, and R. Pries, "Balancing the migration of virtual network functions with replications in data centers," in *Proc. IEEE/IFIP Netw. Oper. Manage. Symp.*, 2018, pp. 1–8.

[26] E. S. Kumar, E. M. Tufail, E. S. Majee, E. C. Captari, and S. Homma. (Feb. 2017). *Service Function Chaining Use Cases in Data Centers*. [Online]. Available: https://tools.ietf.org/html/draft-ietf-sfc-dc-use-cases-06

[27] L. Deng, H. Zheng, X.-Y. Liu, X. Feng, and Z. D. Chen, "Network latency estimation with leverage sampling for personal devices: An adaptive tensor completion approach," *IEEE/ACM Trans. Netw.*, vol. 28, no. 6, pp. 2797–2808, Dec. 2020.

[28] W. Rankothge, F. Le, A. Russo, and J. Lobo, "Optimizing resource allocation for virtualized network functions in a cloud center using genetic algorithms," *IEEE Trans. Netw. Service Manag.*, vol. 14, no. 2, pp. 343–356, Jun. 2017.

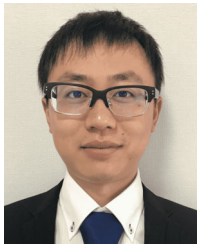
[29] W. Rankothge, F. Le, A. Russo, and J. Lobo, "Experimental results on the use of genetic algorithms for scaling virtualized network functions," in *Proc. IEEE Conf. Netw. Funct. Virtualization Softw. Defined Netw. (NFV-SDN)*, 2015, pp. 47–53.

[30] IBM Knowledge Center. *Introducing IBM ILOG CPLEX Optimization Studio V12.10.0.0*. Accessed: Jan 8, 2021. [Online]. Available: https://www.ibm.com/support/knowledgecenter/SSSA5P_12.10.0/ilog.odms.studio.help/Optimization_Studio/topics/COS_home.html

[31] C. Guo *et al.*, "BCube: A high performance, server-centric network architecture for modular data centers," *SIGCOMM Comput. Commun. Rev.*, vol. 39, no. 4, pp. 63–74, Aug. 2009. [Online]. Available: https://doi.org/10.1145/1594977.1592577

[32] J. Zhao, D. Han, X. Jiang, T. Li, M. Zhang, and J. He, "Design and implementation of a topology discovery mechanism for bidirectional VLC networking," in *Proc. 12th Int. Symp. Commun. Syst. Netw. Digit. Signal Process. (CSNDSP)*, 2020, pp. 1–6.
[33] Microsoft. *Azure AI Notebooks for Predictive Maintenance*. Accessed: Apr. 7, 2021. [Online]. Available: https://azuremlsampleexperiments.blob.core.windows.net/datasets/PdM_maint.csv
[34] L. Qu, M. Khabbaz, and C. Assi, "Reliability-aware service chaining in carrier-grade softwarized networks," *IEEE J. Sel. Areas Commun.*, vol. 36, no. 3, pp. 558–573, Mar. 2018.
[35] P. T. A. Quang, A. Bradai, K. D. Singh, G. Picard, and R. Riggio, "Single and multi-domain adaptive allocation algorithms for VNF forwarding graph embedding," *IEEE Trans. Netw. Service Manag.*, vol. 6, no. 1, pp. 98–112, Mar. 2019.
[36] K. A. Noghani, A. Kassler, and J. Taheri, "On the cost-optimality trade-off for service function chain reconfiguration," in *Proc. IEEE 8th Int. Conf. Cloud Netw. (CloudNet)*, Nov. 2019, pp. 1–6.
[37] A. Engelmann and A. Jukan, "A reliability study of parallelized VNF chaining," in *Proc. IEEE Int. Conf. Commun. (ICC)*, 2018, pp. 1–6.
[38] A. Engelmann, A. Jukan, and R. Pries, "On coding for reliable VNF chaining in DCNs," in *Proc. 15th Int. Conf. Design Rel. Commun. Netw. (DRCN)*, 2019, pp. 83–90.
[39] L. Zhang, Y. Wang, X. Qiu, and H. Guo, "Redundancy mechanism of service function chain with node-ranking algorithm," in *Proc. IFIP/IEEE Symp. Integr. Netw. Service Manage. (IM)*, 2019, pp. 586–589.
[40] M. Wang, B. Cheng, and J. Chen, "Joint availability guarantee and resource optimization of virtual network function placement in data center networks," *IEEE Trans. Netw. Service Manag.*, vol. 17, no. 2, pp. 821–834, Jun. 2020.
[41] J. Fan, M. Jiang, and C. Qiao, "Carrier-grade availability-aware mapping of service function chains with on-site backups," in *Proc. IEEE/ACM 25th Int. Symp. Qual. Service (IWQoS)*, 2017, pp. 1–10.
[42] J. Fan *et al.*, "A framework for provisioning availability of NFV in data center networks," *IEEE J. Sel. Areas Commun.*, vol. 36, no. 10, pp. 2246–2259, Oct. 2018.
[43] J. Kong *et al.*, "Guaranteed-availability network function virtualization with network protection and VNF replication," in *Proc. IEEE Global Commun. Conf.*, 2017, pp. 1–6.

Rui Kang (Graduate Student Member, IEEE) received the B.E. degree from the University of Electronic Science and Technology of China, Chengdu, China, in 2018, and the M.E. degree from Kyoto University, Kyoto, Japan, in 2020, where he is currently pursuing the Ph.D. degree. He was an exchange student with The University of Electro-Communications, Tokyo, Japan, from 2017 to 2018. His research interests include virtual network resource allocation, network virtualization, and software-defined network.



Fujun He (Member, IEEE) received the B.E. and M.E. degrees from the University of Electronic Science and Technology of China, Chengdu, China, in 2014 and 2017, respectively, and the Ph.D. degree from Kyoto University, Kyoto, Japan, in 2020, where he is a Program-Specific Researcher. His research interests include modeling, algorithm, optimization, resource allocation, survivability, and optical networks.

Eiji Oki (Fellow, IEEE) is a Professor with Kyoto University, Kyoto, Japan. He was with Nippon Telegraph and Telephone Corporation Laboratories, Tokyo, from 1993 to 2008, and The University of Electro-Communications, Tokyo, from 2008 to 2017. From 2000 to 2001, he was a Visiting Scholar with Polytechnic University, Brooklyn, NY, USA. His research interests include routing, switching, protocols, optimization, and traffic engineering in communication and information networks.