Robust Virtual Network Function Allocation in Service Function Chains With Uncertain Availability Schedule

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

***Abstract*—The availability schedule provides information on whether each network node is available at each time slot. The service interruptions caused by node unavailability marked in availability schedule can be suppressed if the functions are allocated according to the availability schedule. However, the given availability schedule may have gaps with the actual one and influence the VNF allocation. This paper proposes a robust optimization model to allocate virtual network functions (VNFs) in service function chains (SFCs) for time slots in sequence aiming to maximize the continuous available time of SFCs in a network with uncertain availability schedules by suppressing the interruptions caused by node unavailability marked in availability schedule and function reallocation. We formulate the problem as a mixed integer linear programming (MILP) problem over the given uncertainty set of the start time slot and period of unavailability on each node in the availability schedule. For solving the model in a practical time in a relative large size of network, we develop a heuristic algorithm. The numerical results show that the proposed model outperforms the baseline models under different levels of robustness in terms of the worst-case minimum number of the longest continuous available time slot in each SFC. The heuristic algorithm reduces the computation time with limited performance loss compared with the MILP approach. In the discussion, we introduce a constraint condition for the maintenance ability, which reduces the size of uncertainty set, and an extension for supporting more than one unavailability periods in the availability schedule on each node.**

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

I. Introduction

IN TRADITIONAL networks, each network function depends on specific hardware, which makes function deployment and management difficult and costly. Network function virtualization (NFV) has been introduced to decouple the functions from specific hardware [1]. The functions are virtualized to virtual network functions (VNFs). The functions form a service function chain (SFC) in a specific order [2] to provide a service. Service providers (SPs) generate an SFC to connect a user with the desired VNFs when the user requests a service.

The flexible allocation of VNFs is an advantage of NFV, which deploys the network functions rapidly and improves network performance. The works in [3], [4] considered VNF allocation model to minimize the consumption of network resources. The works in [5], [6] addressed allocation models for minimizing service latency to ensure real-time business.

A node in the network can be sometimes unavailable because of failure or maintenance. The unavailability interrupts the SFCs that use the VNFs running on the node, so a VNF allocation model that prevents service interruptions is required. Existing studies in [7], [8] presented several allocation models to offset the effect of unavailability.

By using the operation records of node availabilities, SPs can mark some specific nodes as unavailabilities during a period for system update and regular maintenance, which identify unavailable nodes at each time slot. We refer to the information that presents the availability of each network node at each time slot in future as the *availability schedule* hereafter. The duration of a time slot can be determined according to conditions of network operation, such as the interval of data sampling for fault detection. The number of uninterrupted time slots can be a metric that expresses the service quality of SFCs.

An availability schedule has the positions and time of node availability. Allocations based on the availability schedule can minimize service interruptions. The work in [9], [10] introduced an optimization model to obtain the VNF allocation that maximizes the continuous available time of the SFCs by avoiding service interruptions caused by different VNF allocations between adjacent time slots [11], [12] and VNF allocations to unavailable nodes 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).

However, in most cases, we cannot know an exact availability schedule whose positions and time of unavailabilities are all deterministic. The actual availability schedule may have some gaps with the maintenance schedule. The starting time of maintenance may be delayed due to the delay of preamble maintenance. The scheduled duration of maintenance may be longer than the real one since conservative estimates are used. The scheduled duration of maintenance may be shorter than the real one since the occurrence of unexpected failures. As a result, availability schedule may mark an availability

Manuscript received August 5, 2020; revised November 20, 2020 and February 17, 2021; accepted April 12, 2021. Date of publication April 29, 2021; date of current version September 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. The associate editor coordinating the review of this article and approving it for publication was B. Cheng. *(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; he.fujun.76z@st.kyoto-u.ac.jp; oki@i.kyoto-u.ac.jp).

Digital Object Identifier 10.1109/TNSM.2021.3076511

as unavailability, and vice versa, which influences the VNF allocation. When the results of the maintenance schedule are not credible, we need a model to obtain the VNF allocation with uncertain availability schedules. We incorporate the uncertainty in the model with deterministic availability schedules in order to make it robust against uncertain availability schedules.

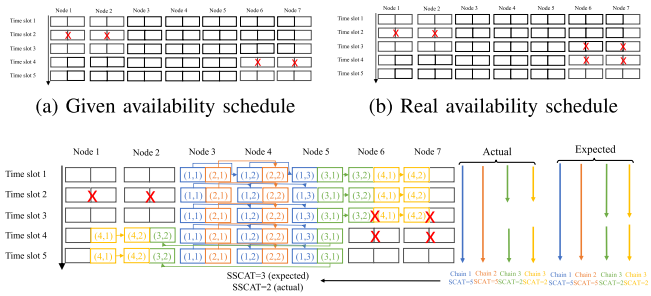
This paper proposes a robust optimization model to obtain the VNF allocation that maximizes SSCAT in the network under an uncertain availability schedule. We consider the uncertain start time slot and period of unavailability on each node in availability schedules. The uncertainty set of availability schedule is given. The robustness of the solution can be controlled. The proposed model obtains the solution by traversing and comparing all possible choices in the uncertainty set limited by the given robustness. We formulate the problem to maximize SSCAT over the given uncertainty set as one mixed integer linear programming (MILP) problem, which provides different levels of robustness against uncertain availability schedule. The model handles the uncertainty set and provides an exact solution under the worst-case condition. Numerical results compare the proposed model with three baseline models. In addition, we provide a heuristic algorithm to speed up the computation process and introduce an extension for supporting parallelization acceleration. Compared with the existing researches [7], [13], [14], the proposed model provides an exact solution of the problem under the uncertainty set.

The remainder of the paper is organized as follows. Section II describes the application scenario and the selection of metrics in the proposed model. Section III describes the proposed model. Section IV introduces a heuristic algorithm for large size networks. Section V presents numerical results that show the performance of the proposed model in different cases. Section VI discusses the influence of limited maintenance ability on the proposed model, and an extension of the proposed model for supporting multiple unavailable time periods on each node. Section VII gives several directions to extend the proposed model. Section VIII introduces some past works related to this paper. Section IX summarizes the key points of this paper.

II. Motivation and Applicability

A. Motivation

An availability schedule from a maintenance schedule [15] has the locations of unavailable VMs in the network during a period of time. The availability in the paper is not a metric. It is a state or event of a node, i.e., the node is able or unable to allocate a VNF. Our model is based on a motivation that the starting and stop times of a machine is given and the normal running time of services is what we value. 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 operation time cannot be estimated, the proposed model cannot be applied, either. The allocation of VNFs before service running can suppress the interruptions caused by unavailabilities and reallocations with



(a) Given availability schedule (b) Real availability schedule

(c) Expected and real SSCATs from an allocation based on given availability schedule

Fig. 1. Demonstration of the impact of uncertain available schedule.

a given availability schedule during the given period of time so that SSCAT can be increased.

The scheduled maintenance may not be sometimes followed and is not usually exact on the time and locations. A source of availability schedules is predictive maintenance. A machine should be maintained after it continuously works [16]. The duration of a maintenance activity may vary according to some schedule-dependent factors, e.g., the starting time [17] and workload [18]. It is a common case that the duration of unavailability is longer than the duration in the plan. The unavailability can start a little earlier or later than the time given by availability schedules; for example, the constraint of starting time of maintenance activity is only a deadline in [17]. We can rely on recovery mechanisms [19] in the unavailable periods of nodes. This paper does not concern the details of the mechanisms. The model fully believes the estimated availability schedules 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. However, the protection system cannot be fully believed in practice, which causes the uncertainty of the availability schedules.

We care about the exact running time of services instead of the mean time or probability of the normal running. As a condition of applying this model, the administrator or system must provide the time information of service starting and stopping.

A demonstration of the impact of availability schedule is shown in Fig. 1. An availability schedule including four unavailable periods is given in Fig. 1(a). Based on this availability schedule, an allocation of four SFCs is given in Fig. 1(c). The expected SSCAT is three. However, the given availability schedule is not exact. The unavailable periods on nodes 6 and 7 are longer than the estimation as shown in Fig. 1(b). so that the actual SSCAT of the calculated allocation decreases from three to two. It is necessary to design a robust model against the influences from these gaps. The challenge is how to design a robust model to maximize SSCAT against the uncertain availability schedules.

B. Applicability

The proposed model is designed for the initial allocation of VNFs in SFCs. SPs store the relative information including the availability schedules, VNFs, services, and network devices in the database. A computation element in the network, which is an independent node or a node with VNFs, calculates the

allocation with these 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 [20], which is not specified to the proposed model. The proposed model is designed to determine the locations of disturbance sensitive functions in the network, where the available schedule is uncertain and the number of available VMs for functions is limited.

In the real deployment of VNFs, the container or VM of the VNF must be placed to a node which is determined by an orchestration engine such as Kubernetes [21]. The proposed model works as a scoring mechanism in scheduler [22] in the engine. The placement is decided by the score provided by the model. Based on the information provided by administers including availability schedules and the information collected by the system including the nodes, the model calculates the suitable locations for each VNF and gives the highest score for the location.

We usually have several replicas of a VNF in deployment for reliability and redundancy. For a VNF, the nodes can be divided into three types according to the state of the replica on the node: the replica is active, the image of the VNF is pulled but not active, and the image does not exist on the node. Without loss of generality, we assume that only one node where the replica is active for each VNF at the same time. We give a direction of the assumption in Section VII. For stateless applications, the interruptions come from the download of VNF images and data redirection from the failed replicas to other replicas. For stateful applications, an additional interruption comes from the status data synchronization. The interruptions periods are estimated by the administrators and turned to the number of time slots used in the availability schedules.

III. Problem Formulation

A. Formulation With Deterministic Unavailability Periods

The model with given deterministic availability schedules was introduced in [9], [10]. It considered each unavailability marked in the availability schedule independently even on the same node.

This paper assumes that the unavailabilities on a node can be continuous from one time slot to another time slot, i.e., form a period that can be one or more time slots, which is called *unavailability period*. We assume that there is at most one unavailability period on each node in the given *T*. We give a future discussion on multiple unavailability periods on each node in Section VI-B.

We use a binary decision variable e_t^n to express elements in the availability schedule; if node $n \in N$ is unavailable at time slot *t*, $e_t^n = 1$, and otherwise 0. For node $n \in N$, the period starts at time slot $s_n \in T$ and lasts for another $f_n \in [0, |T| - s_n]$ continuous time slots, i.e., ends at time slot $s_n + f_n \in T$, as shown in Fig. 2. If there is no unavailability marked in availability schedule at node *n*, f_n is set to 0 and s_n is set to a positive integer value B_{S} which is larger than $|T|$.

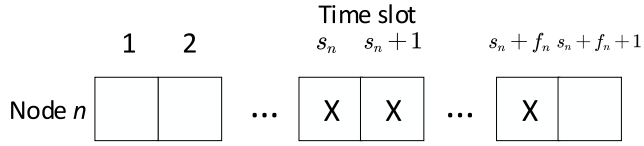


Fig. 2. Unavailability period. "×" means unavailability.

The values of $e_t^n, t \in T, n \in N$, can be obtained from s_n and f_n given by:

$$\textbf{If} \quad s_n \le t \le s_n + f_n \quad \textbf{then}$$
$$e_t^n = 1$$
$$\textbf{Else}$$
$$e_t^n = 0 \tag{1a}$$
$$e_t^n \in \{0, 1\}, \forall n \in N, t \in T. \tag{1b}$$

Based on the unavailability period, we introduce the uncertainties of the beginning time slots and durations of unavailability periods in Section III-B and propose the model considering the uncertain availability schedule. The other definitions and formulations in this subsection are the same with those in [9], [10] except for (2) and (3) because e_t^n is a variable in this paper.

Consider a virtual network as directed graph *G*(*N*, *L*), which consists of a set of virtual nodes, *N*, and a set of directed virtual links connecting these nodes, *L*. We consider a set of different types of resources *S* for each node, such as CPU, memory, and storage. Node $n \in N$ at time slot $t \in T$ has c_{nt}^s units of available resources of resource $s \in S$ in total.

R is the set of SFCs, which represent SPs' requirements. Without loss of generality, we assume that one request corresponds to one SFC. Each SFC contains several VNFs. According to the SPs' requirements, VNFs are allocated to physical nodes and connected to each other in some specified orders. Let K_r be the set of ordered VNFs in chain $r \in R$, each of which is associated with an index in the range of $[1, |K_r|]$. Instances of any VNF type can be deployed in a VM. The VNF instance of the $k \in K_r$th function of request $r \in R$ placed on a VM occupies q_s^{rk} units of resource $s \in S$. Several different functions from different requests can be assigned to the same node, each of which utilizes certain computing.

We divide the continuous time into discrete time slots. A set of time slots is represented by $T = [1, |T|]$. Let $T_i = [1, |T| - i + 1] \subseteq T, i \in [1, |T|]$, be a set of time slots from 1 to $|T| - i + 1$.

The decision variables in the problem are represented as follows. We use binary decision variable x_{tn}^{rk} to represent the allocation; x_{tn}^{rk} is set to 1 if the *k*th function of request *r* is assigned to node *n* at time slot *t*, and 0 otherwise.

An interruption occurs when a function is allocated to an unavailable node or the location of a function is changed between two adjacent time slots. To count SSCAT, we introduce a binary decision variable o_t^r, which indicates whether there is an interruption of request $r \in R$ at time slot $t \in T$. If the allocation of at least one VNF 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,

o_t^r is set to 0, and otherwise 1. If $t = 1, o_t^r = 0$; otherwise, o_t^r can be expressed by:

$$o_t^r = \prod_{k \in K_r} \prod_{n \in N} \Big(\Big(x_{tn}^{rk} \odot x_{t-1,n}^{rk} \Big) \wedge \Big(1 - x_{tn}^{rk} \wedge e_t^n \Big) \wedge \Big(1 - x_{t-1,n}^{rk} \wedge e_{t-1}^n \Big) \Big), \quad \forall r \in R, t \in T \backslash \{1\}. \tag{2}$$

Here $\odot$ expresses exclusive NOR operation between two binary variables whose operations are: $1 \odot 1 = 1, 0 \odot 0 = 1, 1 \odot 0 = 0, 0 \odot 1 = 0$. $\wedge$ expresses the multiplication between two binary variables whose operations are: $1 \wedge 1 = 1, 0 \wedge 0 = 0, 1 \wedge 0 = 0, 0 \wedge 1 = 0$.

To count the continuous available time slots of each request, we introduce two binary decision variables, $z_i^{jr}, i \in T, j \in T_i, r \in R$ and $y_j^r, j \in T, r \in R$. If allocations of request r from i to $i + j - 1$ 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. If the allocations are consecutively unchanged during j time slots existing in T, y_j^r is set to 1, and otherwise 0. z_i^{jr} and y_j^r are constrained as [10, eqs. (2)–(5)].

$\beta_r \in [1, |T|], r \in R$, is an integer variable that represents the SCAT of request r, i.e., the maximum number of continuous available time slots in request r. β_r is given by [10, eq. (6)].

$\lambda \in [1, |T|]$ is an integer variable that represents SSCAT, i.e., the minimum number of longest continuous available time slots among all requests. λ is constrained by [10, eq. (7)].

The objective function under deterministic availability schedules is given in [9], [10], [10, eq. (8)].

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 number, ϵ, is multiplied by the second term so that the first term can be prioritized over the second term. ϵ is given by $\frac{1}{|R| \cdot |T|}$.

The node capacity constraint is given by [10, eq. (9)]. Each node, because of the computational resource limitation, can carry only a limited number of functions. Reference [10, eq. (9)] ensures that each node's computational resources must not exceed its capacity during allocation.

The assignment constraints are given by (10)-(11) in [10]. Reference [10, eq. (10)] 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 [23]. Reference [10, eq. (11)] ensures that all functions are allocated in the network. We assume that there is only one VNF is active for each function in [10, eq. (11)].

If it is necessary to avoid allocating functions to unavailable nodes, we add the following constraint:

$$x_{tn}^{rk} \wedge e_t^n = 0, \ \forall r \in R, t \in T, n \in N, k \in K_r. \tag{3}$$

The notations used in this subsection are summarized in Table I.

According to the linearization process in [9] and Appendix A, (1) is linearized to (29a)-(29h), [10, eq. (6)] is linearized to (30a)-(30f), and (2) is linearized to (31a)-(31w) with some auxiliary variables in Appendix B.

TABLE I
NOTATIONS USED IN SECTION III-A

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_{nt}^s	Capacity of node $n \in N$ at time slot $t \in T$ for resource $s \in S$
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
s_n	Starting time slot of unavailability period
f_n	Duration of unavailability period
Decision variable	**Description**
x_{tn}^{rk}	If the kth function of request r is assigned to node n at time slot t, $x_{tn}^{rk} = 1$, and otherwise 0.
e_t^n	If node $n \in N$ is unavailable at time slot t, $e_t^n = 1$, and 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

In summary, the following model is given for VNF allocation with considering the deterministic availability schedule:

$$\begin{aligned} \max \ & \lambda + \epsilon \sum_{r \in R} \beta_r \\ \text{s.t.} \ & \text{(2)-(5), (7), (9)-(11) in [10], (29a)-(31w)}, \\ & \beta_r, \lambda \in [1, |T|], \ \forall r \in R, i \in T, j \in T_i, \\ & z_i^{jr} \in \{0, 1\}, \ \forall r \in R, i \in T, j \in T_i. \end{aligned}$$

B. Formulation With Uncertain Unavailability Periods

In Section III-A, s_n and f_n are deterministic. This subsection considers the uncertainty caused by an uncertain beginning time slot and an uncertain duration of each node, as shown in Fig. 3. The uncertainty set of s_n is symmetric about $\bar{s_n}$ and that of f_n is symmetric about $\bar{f_n}$, i.e., $s_n \in [\bar{s_n} - \hat{s_n}, \bar{s_n} + \hat{s_n}]$ and $f_n \in [\bar{f_n} - \hat{f_n}, \bar{f_n} + \hat{f_n}]$, respectively, where $s_n, s_n + f_n \in T$. If there is no unavailability marked in availability schedule on node $n \in N$, the corresponding $\bar{s_n}$ is set to B_{S}; $\hat{s_n}$ $\bar{f_n}$, and $\hat{f_n}$ are set to 0.

In (29a)-(29d), s_n and f_n are uncertain. We model an robust optimization problem which can control the degree of robustness against uncertainties and conservation of solutions. We define Γ_n^{S} as the degree of robustness of $s_n, n \in N$ and

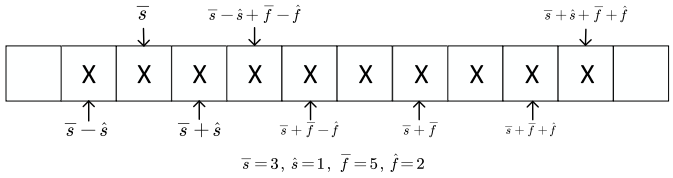


Fig. 3. Type of uncertainty. "×" means unavailability.

$\Gamma_n^{\mathrm{F}} \in [-1,1]$ as the degree of robustness of $f_n, n \in N$. The larger Γ_n^{S} or Γ_n^{F} is, the higher the level of robustness is.

Γ_n^{S} controls the size of uncertainty set of beginning time slots $\mathcal{P}_n$ on node n. The larger Γ_n^{S} is, the more the considered possible beginning time slots of unavailability period on node n are. We choose $\lfloor \Gamma_n^{\mathrm{S}} \cdot (2 \cdot \hat{s_n} + 1) \rfloor$ different beginning time slots from $[\bar{s_n} - \hat{s_n}, \bar{s_n} + \hat{s_n}]$ and forms a set which is an element in $\mathcal{P}_n$. $\mathcal{P}_n$ contains all possible choices. The size of $\mathcal{P}_n$ is $|\mathcal{P}_n| = \binom{2 \cdot \hat{s_n}+1}{\lfloor \Gamma_n^{\mathrm{S}} \cdot (2 \cdot \hat{s_n}+1) \rfloor}$. Since the size of each element in $\mathcal{P}_n$ is at least 1, i.e., $\lfloor \Gamma_n^{\mathrm{S}} \cdot (2 \cdot \hat{s_n}+1) \rfloor \geq 1$, $\Gamma_n^{\mathrm{S}} \in [\frac{1}{2 \cdot \hat{s_n}+1}, 1]$. The uncertainty set for all nodes is denoted by $\mathcal{P} = \{\mathcal{P}_1, \mathcal{P}_2, \ldots, \mathcal{P}_{|N|}\}$. One element in $\mathcal{P}$ is a combination of one element in each $\mathcal{P}_n$. The size of $\mathcal{P}$ is $|\mathcal{P}| = |\mathcal{P}_1| \cdot |\mathcal{P}_2| \cdot \cdots \cdot |\mathcal{P}_{|N|}|$. Let p denote an element in $\mathcal{P}$; p_n denote the selected set from $\mathcal{P}_n$; p_n^q denote the qth element in p_n.

Γ_n^{F} controls the considered length of the duration of unavailability period on node n, which is $\bar{f_n} + \hat{f_n} \cdot \lfloor \Gamma_n^{\mathrm{F}} \rfloor$.

To deal the uncertain unavailability periods, we develop an MILP formulation in terms of Γ_n^{S} and Γ_n^{F}. For the unavailability period on each node n under the uncertainty of $\mathcal{P}$, we consider the worst case. The objective of the considered problem is presented by:

$$\max \min_{p \in \mathcal{P}} \left\{ \lambda^p + \epsilon \sum_{r \in R} \beta_r^p \right\}. \tag{4}$$

Here, λ^p means the value of λ under the uncertainty set $p \in \mathcal{P}$. β_r^p means the value of β_r under the uncertainty set $p \in \mathcal{P}$. The selected set of beginning time slots p_n contains one or more time slot. Equations (29a)-(29h) are transformed to:

$$t - p_n^q + \epsilon_1 \leq \eta_{tn}^q \cdot B,\ \forall t \in T, n \in N, q \in [1, |p_n|], \tag{5a}$$
$$t - p_n^q + \epsilon_1 \geq \left(\eta_{tn}^q - 1\right) \cdot B,\ \forall t \in T, n \in N, q \in [1, |p_n|], \tag{5b}$$
$$p_n^q + \bar{f}_n + \Gamma_n^{\mathrm{F}} \cdot \hat{f}_n - t + \epsilon_1 \leq \rho_{tn}^q \cdot B,\ \forall t \in T, n \in N, q \in [1, |p_n|], \tag{5c}$$
$$p_n^q + \bar{f}_n + \Gamma_n^{\mathrm{F}} \cdot \hat{f}_n - t + \epsilon_1 \geq \left(\rho_{tn}^q - 1\right) \cdot B, \forall t \in T, n \in N, q \in [1, |p_n|], \tag{5d}$$
$$\tau_{tn}^q \leq \eta_{tn}^q,\ \forall t \in T, n \in N, q \in [1, |p_n|], \tag{5e}$$
$$\tau_{tn}^q \leq \rho_{tn}^q,\ \forall t \in T, n \in N, q \in [1, |p_n|], \tag{5f}$$
$$\tau_{tn}^q \geq \eta_{tn}^q + \rho_{tn}^q - 1,\ \forall t \in T, n \in N, q \in [1, |p_n|], \tag{5g}$$
$$\tau_{tn}^q, \eta_{tn}^q, \rho_{tn}^q \in [0,1],\ \forall t \in T, n \in N, q \in [1, |p_n|]. \tag{5h}$$

The above equations use τ_{tn}^q to replace e_t^n in (29a)-(29h). If node n is unavailable at time slot t when the beginning time slot q is considered, τ_{tn}^q is set to 1, and otherwise 0. $e_t^n \in [0,1], t \in T, n \in N$, is expressed by:

$$e_t^n = \vee_{q \in [1, |p_n|]} \tau_{tn}^q,\ \forall t, \in T, n \in N. \tag{6}$$

Here $\vee$ means OR operation between two binary variables whose operations are: $0 \vee 0 = 0, 0 \vee 1 = 1, 1 \vee 0 = 1, 1 \vee 1 = 1$. According to the linearization process in Appendix A, (6) can be linearized to:

$$e_t^n \geq \frac{1}{|p_n|} \cdot \sum_{q \in [1, |p_n|]} \tau_{tn}^q,\ \forall t, \in T, n \in N, \tag{7a}$$
$$e_t^n \leq \sum_{q \in [1, |p_n|]} \tau_{tn}^q,\ \forall t, \in T, n \in N. \tag{7b}$$

According to the linearization process in Appendix A, the objective function (13) can be linearized to:

$$\max\ \Lambda \tag{8a}$$
$$\text{s.t. } \Lambda \leq \lambda^p + \epsilon \sum_{r \in R} \beta_r^p + (1 - \delta_{\mathrm{o}}^p) \cdot B_{\mathrm{O}},\ \forall p \in \mathcal{P}, \tag{8b}$$
$$\Lambda \geq \lambda^p + \epsilon \sum_{r \in R} \beta_r^p - (1 - \delta_{\mathrm{o}}^p) \cdot B_{\mathrm{O}},\ \forall p \in \mathcal{P}, \tag{8c}$$
$$\lambda^p + \epsilon \sum_{r \in R} \beta_r^p \leq (1 - \delta_{\mathrm{o}}^p) \cdot B_{\mathrm{O}} + \lambda^{p'} + \epsilon \sum_{r \in R} \beta_r^{p'}, \quad \forall p \in \mathcal{P}, p' \in \mathcal{P} \backslash p, \tag{8d}$$
$$\sum_{p \in \mathcal{P}} \delta_{\mathrm{o}}^p = 1, \tag{8e}$$
$$\Lambda \leq \lambda^p + \epsilon \sum_{r \in R} \beta_r^p,\ \forall p \in \mathcal{P}, \tag{8f}$$
$$\delta_{\mathrm{o}}^p \in [0,1],\ \forall p \in \mathcal{P}. \tag{8g}$$

Here, B_{O} is a given integer which is larger than $\lambda^p + \epsilon \sum_{r \in R} \beta_r^p$. The minimum of B_{O} can be taken to be $|T| + 2$.

To calculate the objective values in element $p \in \mathcal{P}$, we perform the following transformations. Reference [10, eqs. (2)–(5)] are transformed to (32a)-(32d) by replacing o_t^r, z_i^{jr}, y_j^r to $o_t^{rp}, z_i^{jrp}, y_j^{rp}$, respectively. Reference [10, eq. (7)] is transformed to (33a) by replacing λ, β_r to λ^p, β_r^p, respectively. Reference [10, eqs. (9)–(11)] are transformed to (34a)-(34c) by replacing x_{tn}^{rk} to x_{tn}^{rkp}, respectively. Equations (30a)-(31w) are transformed to (35a)-(36w) by replacing β_r, y_j^r, δ_j^r, ϕ_{tn}^{rk}, x_{tn}^{rk}, h_{tn}^{rk}, α_{tn}^{rk}, θ_{tn}^{rk}, π_{tn}^{rk}, w_t^{rk}, o_t^r to β_r^p, y_j^{rp}, δ_j^{rp}, ϕ_{tn}^{rkp}, x_{tn}^{rkp}, h_{tn}^{rkp}, α_{tn}^{rkp}, θ_{tn}^{rkp}, π_{tn}^{rkp}, w_t^{rkp}, o_t^{rp}, respectively. Equations (5a)-(5h) are transformed to (37a)-(37h) by replacing $e_t^n, \eta_{tn}^q, \rho_{tn}^q, \tau_{tn}^q$ to $e_t^{np}, \eta_{tn}^{qp}, \rho_{tn}^{qp}, \tau_{tn}^{qp}$, respectively. Equations (7a)-(7b) are transformed to (38a)-(38b) by replacing e_t^n to e_t^{np}, respectively. The new objective function and constraints are expressed in Appendix C.

In summary, the robust optimization problem is given by:

$$\max\ \Lambda \tag{9a}$$
$$\text{s.t. } (8\text{b})\text{-}(8\text{g}),\ (32\text{a})\text{-}(32\text{b}), \tag{9b}$$
$$\Lambda \in [1, |T|], \tag{9c}$$
$$\beta_r^p, \lambda^p \in [1, |T|],\ \forall r \in R, i \in T, j \in T_i, p \in \mathcal{P}, \tag{9d}$$
$$z_i^{jrp} \in \{0,1\},\ \forall r \in R, i \in T, j \in T_i, p \in \mathcal{P}. \tag{9e}$$

TABLE II
PARAMETERS IN ALGORITHM

Parameter	Description
N	Set of nodes
T	Set of time slots
R	Set of requests
K_r	Length of request $r \in R$
c_{nt}^s	Capacity for resource $s \in S$ of node $n \in N$ at time slot t
e_t^n	Availability schedule, if node $n \in N$ is unavailable at time slot t, $e_t^n = 1$, and otherwise 0
$\bar{s_n}, \hat{s_n}$	the uncertainty set of s_n
$\bar{f_n}, \hat{f_n}$	the uncertainty set of f_n
Γ_n^{S}	the degree of robustness of s_n
Γ_n^{F}	the degree of robustness of f_n
MG	Maximum generation
IP	Initial population number
UP	Upper limitation of population number
IC	Internal crossover probability
EC	External crossover probability
MP	Mutation probability

IV. HEURISTIC ALGORITHM

As the size of the ILP problem in Section II increases, the problem becomes difficult to solve in practical time. This paper introduces a heuristic algorithm based on the genetic algorithm [24]. To accelerate the computation process, this paper introduces an extension of this algorithm by using CPU and GPU acceleration [25].

A. Framework

The framework of this heuristic algorithm is shown in Algorithm 1. We set some parameters shown in Table II. The set of possible beginning time slots and the length of considered unavailability period are calculated. The availability schedule is given by using (1) under a combination of possible beginning time set p in line 5. A set of initial feasible solutions whose size is *IP* is given by Algorithm 2 in lines 6-7. From line 8 to line 31, the heuristic algorithm enters a loop that has *MG* cycles. In each cycle, the heuristic algorithm explores new feasible solutions by performing internal crossover (see function *cross_in* in Algorithm 3), external crossover (see function $cross\_out$ in Algorithm 3), and mutation (see Algorithm 4) according to three probabilities, *IC*, *EC*, and *MP*, respectively. Newly generated solutions at lines 12, 17, and 22 are stored in the new feasible solution set S_{n} once. They are added to the feasible solution set at a time in line 25. The heuristic algorithm calculates the fitness for each solution (see Algorithm 5). The heuristic algorithm finds the solution with the highest fitness score and stores it. Finally, if the size of the feasible solution set exceeds *UP*, the heuristic algorithm chooses *UP* feasible solutions as a new set of feasible solutions according to the roulette gambler (see Algorithm 6).

B. Initial Solution Generation

The heuristic algorithm generates a set of initial feasible solutions by using Algorithm 2. Based on this set, more feasible solutions can be generated by Algorithms 3 and 4.

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 *k*th function of request *r* at time slot *t*, which belongs to *N*.

Algorithm 1 Framework

Input: N, T, R, K_r, $c_{nt}^s \in C$, $\bar{s_n}, \hat{s_n}, \bar{f_n}, \hat{f_n}$, Γ_n^{S}, Γ_n^{F}, *IP, MG, EC, IC, MP, UP*
Output: allocation for all functions
1: $f_n \leftarrow \bar{f_n} + \Gamma_n^{\mathrm{F}} \cdot \hat{f_n}$
2: Calculate the possible choices of s_n by $\Gamma_n^{\mathrm{S}}, \bar{s_n}, \hat{s_n}$ and stored in $\mathcal{P}_n$
3: Calculate different combinations of $\mathcal{P}_n$ on all nodes and store them in set P
4: **for** $p \in \mathcal{P}$ **do**
5: Calculate the value of e_t^n according to (1) by using p and f_n
6: Define S^p as the feasible solution set for the possible uncertainty set p
7: $S^p \leftarrow$ Generate set of initial feasible solutions by using function *init_chromos* in Algorithm 2
8: **for** $step = 1 \rightarrow MG$ **do**
9: Define S_{n}^p as the new feasible solution set
10: **for** each solution in S^p **do**
11: **if** a random number in $[0, 1]$ >1- *IC* **then**
12: $S_{\mathrm{n}}^p \leftarrow$Generate a non-redundant and mutant solution by using function $cross\_in$ in Algorithm 3 whose inputs are the selected solution in S^p and random time slot *t*
13: **end if**
14: **end for**
15: **for** each solution in S^p except for the first one **do**
16: **if** a random number [0, 1] >1- *EC* **then**
17: $S_{\mathrm{n}}^p \leftarrow$Generate a non-redundant and mutant solution by using function *cross_out* in Algorithm 3 whose inputs are the selected solution and its previous solution in *S*
18: **end if**
19: **end for**
20: **for** each solution in S^p **do**
21: **if** a random number [0, 1] >1- *MP* **then**
22: $S_{\mathrm{n}}^p \leftarrow$Generate a non-redundant and mutant solution by using function *mutation* in Algorithm 4 whose input is the selected solution in S^p
23: **end if**
24: **end for**
25: Integrate S_{n}^p into *S*
26: Calculate the fitness score of the solutions in S^p by using function *calc_fin_ness* in Algorithm 5
27: Store the solution with the highest fitness score
28: **if** size of $S^p > UP$ **then**
29: Reduce the size of the set to *UP* by using function *roulette_gambler* in Algorithm 6
30: **end if**
31: **end for**
32: Output the solution which has the smallest objective value among $S^p, p \in \mathcal{P}$
33: **end for**

Algorithm 2 Initial Solution

1: **function** INIT_CHROMOS(*E*)
2: Set of initial solutions $s \leftarrow \phi$
3: Sort requests in *R* in a non-increasing order of K_r
4: **for** each time slot in *T* **do**
5: Sort nodes in *N* in a 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 a 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 a non-increasing order of time from time slot *t* to the last time slot in which a node becomes unavailable.
6: **for** $r \in R$ **do**
7: **for** $f = 1 \rightarrow K_r$ **do**
8: **for** $n \in N$ **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 *f*th 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 2 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 *IP* (line 20).

C. New Solution Generation

There are three methods for generating new solutions in the heuristic algorithm.

The internal crossover, function *cross_in* in Algorithm 3, 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 3, 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.

Algorithm 3 Crossover

function CROSS_IN($s \leftarrow$ the selected solution , $t \leftarrow$ the random time)
  $s[t] \leftarrow s[t+1]$
  return s
end function
function CROSS_OUT(s_1, s_2)
  $location \leftarrow$ a random integer in $[1, |T|]$
  $s_c \leftarrow s_1$
  $s_c[location] \leftarrow s_2[location]$
  return s_c
end function

Algorithm 4 Mutation

function MUTATION($s \leftarrow$ the selected solution, *E*)
  Calculate the SCAT for all SFCs in solution s
  Sort requests in *R* in a weighted randomized order. The larger the SCAT, the higher the order of the corresponding *r*.
  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*.
  for $r \in R$ **do**
    for $f = 1 \rightarrow K_r$ of request *r* **do**
      for $n \in N$ **do**
        if used capacity of *n* is less than c_n^t AND any other functions in *r* were not allocated to *n* **then**
          Allocate the *f*th function in SFC *r* to *n*
          Break
        else
          Continue
        end if
      end for
    end for
  end for
  return new solution
end function

Algorithm 5 Fitness Calculation

function CALC_FIN_NESS($s \leftarrow$ the selected solution, *E*)
  Calculate the SCAT for all SFCs in solution s
  return $\min(SCAT) + \text{sum}(SCAT)/(|T| \times |R|)$
end function

The other function for generating new solutions is function *mutation* in Algorithm 4. *init_chromos* function in Algorithm 2 generates the initial solution set based on the length of each SFC. On the other hand, *mutation* function generates a new solution based on the SCAT of each SFC.

D. Calculation of Fitness

The algorithm computes the fitness score for each solution by using the objective function.

Algorithm 6 Choice

```
function ROULETTE_GAMBLER(fit_pros, chroms)
    pick ← a random number in [0, 1]
    for j = 1 → |chroms| do
        pick ← pick − fit_pros[j]/sum(fit_pros)
        if pick ≤ 0 then
            return j
        end if
    end for
    return |chroms| − 1
end function
function CHOICE(chroms, fit_pros)
    choice_gens ← φ
    for i = 1 → min(|chroms|,UP) do
        j ← ROULETTE_GAMBLER(fit_pros, chroms)
        append chroms[j] to choice_gens
    end for
    return choice_gens
end function
```

E. 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 6, input *chroms* is the set of solutions and *fit_pros* is the set of the fitness scores of the corresponding solutions in *chroms*.

F. Parallelization Acceleration

In lines 4-33 of Algorithm 1, the algorithm calculates the allocation under different set *P*. These calculations can be parallelized. The algorithm initializes $|P|$ threads and performs the calculations at the same time. $|P|$ results can be obtained. The final result is the smallest one among these results.

In order to further reduce the calculation time, we can also make the parallelization of the functions called in the heuristic algorithm. The loops in lines 10-24 of Algorithm 1 can be parallelized. Each thread contains the cross and mutation of one solution in S^p. The new generated solutions are collected and merged into set *S* in line 25 of Algorithm 1. However, the number of solutions required to be processed in parallel is greater than the number of cores in a CPU. If we apply CPU threads on the parallelization, the time consumed by thread switching slows down the computation time. A method to reduce the overhead time and the overall computation time is to apply GPU acceleration on these functions instead of CPU multi-threading [25].

V. Numerical Evaluations

A. Comparison With Other Models

In this subsection, we compare the proposed model with a persistence allocation model, a single-slot allocation model, and a double-slot allocation model with deterministic and uncertain availability schedules in two tests, respectively. The given conditions $\hat{s_n}$ and $\hat{f_n}$ are set to 0 in the tests considering deterministic availability schedules.

The persistence allocation model does not consider the service interruptions caused by node unavailabilities. This model maximizes SSCAT by suppressing the interruptions caused by reallocations of VNFs. It determines a node to which each VNF is allocated randomly and keeps this allocation from the first time slot to the last one. This model has no ability to avoid unavailable nodes.

The single-slot allocation model considers the service interruptions caused by node unavailabilities at each time slot, regardless of VNF reallocations between all adjacent time slots. This model minimizes the number of VNFs allocated to unavailable nodes at each time slot. This model independently determines VNF allocation of each time slot. This model tries to avoid allocating VNFs to unavailable nodes according to the availability schedule. However, this model does not consider the relationship between VNF allocations at adjacent time slots; different allocations at adjacent time slots cause service interruptions. For each $t \in T$, the optimization problem of single-slot allocation model is formulated as an ILP problem by:

$$\begin{aligned} \min \ & \sum_{r\in R}\sum_{k\in K_r}\sum_{n\in N} x_{tn}^{rk} e_t^n \\ \text{s.t.} \ & \text{(9)-(11) in [10]}. \end{aligned} \tag{10}$$

The double-slot allocation model is an improvement of the single-slot allocation model. This model computes the VNF allocation at time slot t by considering that in the last time slot, $x_{t-1,n}^{rk}, \forall n \in N, r \in R, k \in K_r$. The solution that minimizes the differences between time slots t and $t-1$ is chosen when there are multiple solutions that minimize $\sum_{r\in R}\sum_{k\in K_r}\sum_{n\in N} x_{tn}^{rk} e_t^n$. For each $t \in T$, the optimization problem of double-slot allocation model is formulated by:

$$\begin{aligned} \min \ & \sum_{r\in R}\sum_{k\in K_r}\sum_{n\in N} \left(x_{tn}^{rk} e_t^n + \epsilon_3 x_{tn}^{rk} \odot x_{t-1,n}^{rk} \right) \\ \text{s.t.} \ & \text{(9)-(11) in [10]}. \end{aligned} \tag{11}$$

A small number, ϵ_3, is multiplied to the second term to prioritize the first term over the second term. ϵ_3 is given by $\frac{1}{|N||R|\max_{r\in R}\{K_r\}}$.

The proposed model, the single-slot allocation model, and the double-slot allocation model are solved by the IBM ILOG CPLEX Interactive Optimizer with version 12.7.1 [26], using Intel Core i7-7700 3.60 GHz 4-core CPU, 32 GB memory. The persistence allocation model is implemented by Python 3.7 and runs on the same hardware.

In the first test, we compare the proposed model with the three baseline models considering deterministic availability schedules in terms of the objective value. Seven cases are examined in this situation. The conditions of these cases are shown in Table III. In this table, *Nodes* means the number of nodes; *Capacity* means the capacity of each node at each time slot; *Requests* means the number of functions in each request; *Functions* means the number of functions in each request; *Time slots* means the number of time slots; *Unavailability*

TABLE III
EVALUATION CONDITIONS IN TEST 1

Case	Nodes	Capacity	Request	Functions	Time slots	Unavailability parameters
1	5	2	3	2, 2, 3	6	$s_1 = 2, f_1 = 3$, other $s_n = B_S$ and $f_n = 0$
2	5	2	3	2, 2, 3	6	$s_1 = 2, f_1 = 3, s_2 = 1, f_2 = 2, s_3 = 5, f_3 = 0$, other $s_n = B_S$ and $f_n = 0$
3	5	2	3	2, 2, 3	12	$s_1 = 2, f_1 = 3, s_2 = 1, f_2 = 2, s_3 = 5, f_3 = 0$, other $s_n = B_S$ and $f_n = 0$
4	5	2	3	2, 2, 3	12	$s_1 = 2, f_1 = 3, s_2 = 1, f_2 = 3, s_3 = 6, f_3 = 4$, other $s_n = B_S$ and $f_n = 0$
5	15	2	5	2, 3, 4, 5, 7	6	$s_1 = 2, f_1 = 3, s_2 = 1, f_2 = 3, s_3 = 6, f_3 = 0$, other $s_n = B_S$ and $f_n = 0$
6	15	2	5	2, 3, 4, 5, 7	6	$s_1 = 2, f_1 = 3, s_2 = 1, f_2 = 3, s_3 = 6, f_3 = 0, s_5 = 3,$ $f_5 = 3, s_7 = 4, f_7 = 2, s_9 = 1, f_9 = 2$, other $s_n = B_S$ and $f_n = 0$
7	15	2	5	2, 3, 4, 5, 7	12	$s_1 = 2, f_1 = 3, s_2 = 1, f_2 = 3, s_3 = 6, f_3 = 0, s_5 = 3,$ $f_5 = 3, s_7 = 4, f_7 = 2, s_9 = 1, f_9 = 2$, other $s_n = B_S$ and $f_n = 0$
8	15	2	5	2, 3, 4, 5, 7	12	$s_1 = 2, f_1 = 3, s_2 = 1, f_2 = 3, s_3 = 6, f_3 = 4, s_5 = 3, f_5 = 4,$ $s_7 = 4, f_7 = 8, s_9 = 1, f_9 = 2, s_{10} = 6, f_{10} = 1, s_{12} = 6, f_{12} = 5,$ $s_{14} = 1, f_{14} = 0, s_{15} = 9, f_{15} = 2$, other $s_n = B_S$ and $f_n = 0$

TABLE IV
EVALUATION RESULTS OF TEST 2

(a) Objective values in case 1.

(Γ_2^F, Γ_1^S)	$(-1, \frac{1}{3})$	$(-1, \frac{2}{3})$	$(-1, 1)$	$(0, \frac{1}{3})$	$(0, \frac{2}{3})$	$(0, 1)$	$(1, \frac{1}{3})$	$(1, \frac{2}{3})$	$(1, 1)$
Proposed model	3.75	3.75	3.75	2.67	2.67	2.67	2.67	2.67	2.67
Double-slot allocation model	3.58	3.67	3.67	2.50	2.58	2.58	2.50	2.58	2.58
Single-slot allocation model	2.33	2.33	2.33	2.33	2.42	2.42	1.25	1.33	1.33
Persistent allocation model	2.33	2.33	2.33	2.33	2.33	2.33	1.17	1.17	1.17

(b) Objective values in case 2.

$(\Gamma_1^F, \Gamma_5^F, \Gamma_5^S, \Gamma_6^S)$	$(-1, -1, \frac{1}{3}, \frac{1}{3})$	$(-1, -1, 1, 1)$	$(0, 0, \frac{1}{3}, \frac{1}{3})$	$(0, 0, 1, 1)$	$(1, 1, \frac{1}{3}, \frac{1}{3})$	$(1, 1, 1, 1)$
Proposed model	4.81	4.81	4.76	4.76	4.76	4.76
Double-slot allocation model	2.33	2.38	2.33	2.38	2.33	2.38
Single-slot allocation model	2.29	2.29	1.24	1.19	1.29	1.19
Persistent allocation model	1.14	0.10	1.14	0.10	1.14	0.10

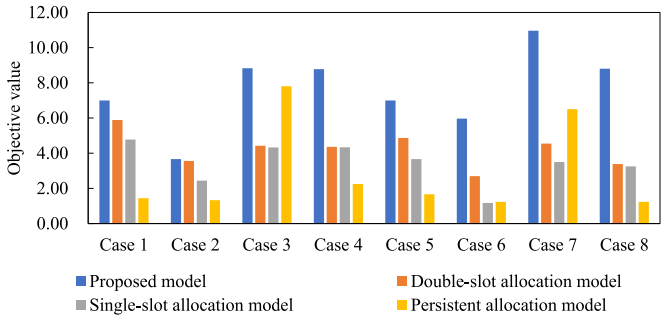


Fig. 4. Evaluation results of test 1.

parameters shows the beginning time slot and duration of each unavailability period. The results are shown in Fig. 4. It shows that the proposed model outperforms the three baseline models when availability schedule is deterministic in terms of the objective value.

In the second test, we compare the proposed model with the three baseline models considering uncertain availability schedule in terms of the objective value. The three baseline models obtain the value of e_t^n by using (5a)-(5h), and (7a)-(7b) and consider the worst case in uncertainty set $\mathcal{P}$. We consider two cases in this text. In the first case, a four-node network is considered. We determine the allocation of functions among six time slots in two requests. The lengths of two requests are two and three, respectively. The uncertainty set is given by: $\bar{s_1} = 2, \hat{s_1} = 1, \bar{f_1} = 1, \bar{s_2} = 2, \bar{f_2} = 2, \hat{f_2} = 1$, other $\bar{s}$ are B_S, and $\hat{s}, \bar{f}$, and $\hat{f}$ are zero. In the second case, a ten-node network is considered. We determine the allocation of functions among seven time slots in three requests. The lengths of three requests are two, three and five, respectively. The uncertainty set is given by: $\bar{s_1} = 3, \bar{f_1} = 1, \hat{f_1} = 1, \bar{s_2} = 2, \bar{f_2} = 2, \bar{s_3} = 6, \bar{f_3} = 1, \bar{s_5} = 4, \hat{s_5} = 1, \bar{f_5} = 1, \hat{f_5} = 1, \bar{s_6} = 2, \hat{s_6} = 1, \bar{s_7} = 5, \bar{f_7} = 1, \bar{s_9} = 1$, other $\bar{s}$ are B_S, and $\hat{s}, \bar{f}$, and $\hat{f}$ are zero. We evaluate the SSCAT values and the sum of SCATs of all requests in different Γ_n^F and Γ_n^S. The result is shown in Table IV.

Table IV shows that the proposed model outperforms the three baseline models under all levels of robustness in terms of the objective value. With the increasing of the level of robustness of the solution, the objective value decreases. By comparing the results between the proposed model and the three baseline models, we observe that the allocation considering availability schedule provides higher SSCAT than the allocation without considering availability schedule and the longer time slots the model considers, the allocation with higher SSCAT the model provides.

B. Evaluation of Different Uncertainty Sets

In this subsection, we evaluate the influences of different uncertainty sets with the same graph and robustness level. Because the durations of unavailability periods $f_n, n \in N$ are

Time slot \ Node	1	2	3	4	5
1	P	P	P	P	P
2	P	P	P	P	P
3	P	P	P	P	P
4	P	P	P	P	P
5	P	P	P	P	P
6	P	P	P	P	P
7	P	P	P	P	P

(a) Availability schedule 1. $\bar{s_n} = 4, \hat{s_n} = 3, \forall n \in N$.

Time slot \ Node	1	2	3	4	5
1	P	P	P	P	P
2	P	P	P	P	P
3	P	P	P	P	P
4	P	P	P	P	P
5	P	P	P	P	P
6					
7					

(b) Availability schedule 2. $\bar{s_n} = 3, \hat{s_n} = 2, \forall n \in N$.

Time slot \ Node	1	2	3	4	5
1	P				
2	P	P			
3	P	P	P		
4		P	P	P	
5			P	P	P
6				P	P
7					P

(c) Availability schedule 3. $\bar{s_n} = n + 1, \hat{s_n} = 1, \forall n \in N$.

Time slot \ Node	1	2	3	4	5
1	P	P			
2	P	P			
3	P	P			
4					
5			P	P	
6			P	P	
7			P	P	

(d) Availability schedule 4. $\bar{s_1} = \bar{s_2} = 2, \bar{s_3} = \bar{s_4} = 6, \hat{s_n} = 1$.

Fig. 5. Possible locations of unavailabilities in four availability schedules. If "P" is marked in a time slot, it is a possible location of an unavailability, and otherwise it is available.

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

(a) Worst case availability schedule 1 and 3. Obj. value is 3.52.

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

(b) Worst case availability schedule 2. Obj. value is 3.57.

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

(c) Worst case availability schedule 4. Obj. value is 4.57.

Fig. 6. Worst case availability schedule in four tests. If "U" is marked in a time slot, it is unavailable, and otherwise available.

determined after the robustness levels Γ_n^{F} are given. We force on the influence of $f_n, n \in N$ on the objective values. The evaluation is performed in a graph which has five nodes whose capacities are two. Three requests need to be allocated in the graph in seven time slots, whose lengths are two, two, three, respectively. There are four tests with different $\bar{s_n}$ and $\hat{s_n}$. The parameters and availability schedules are shown in Fig. 5.

We evaluate the relationship between different shapes of availability schedules and the objective values under the same value of $\Gamma_n^{\mathrm{S}}, n \in N$, which is $\frac{1}{\bar{s_n}}$. The objective values and the availability schedule under the worst case in each test are shown in Fig. 6. We observe that when unavailabilities are concentrated in the middle of the availability schedule, it is most likely to be the availability schedule under the worst case as shown in Fig. 6(a), 6(b), and 6(c). When the unavailabilities are distributed among nodes in different time slots, it is most likely to be the availability schedule under the worst case as shown in Fig. 6(a) and 6(b). By taking advantage of this observation, a number of possible choices, such as the availability schedule with the unavailabilities concentrated in one time slot, can be removed and the computational time can be reduced. The worst case availability schedules in Fig. 5(a) and Fig. 5(c) are the same, which is Fig. 6(a). We can reduce the input parameters of Fig. 5(a) to those of Fig. 5(c) in order to reduce the size of the uncertainty set.

TABLE V
UNCERTAIN CONDITIONS IN TEST 3

Case	Uncertain unavailability parameters
1	$\bar{s_1} = 2, \hat{s_1} = 1, \bar{f}_1 = 1, \bar{s_2} = 2, \bar{f}_2 = 2, \hat{f}_2 = 1$, other $\bar{s}$ are B_{S}, and $\hat{s}$, $\bar{f}$, and $\hat{f}$ are zero.
4	$\bar{s_1} = 2, \hat{s_1} = 1, \bar{f}_1 = 3, \hat{f}_1 = 1, \bar{s_2} = 1, \bar{f}_2 = 3, \hat{f}_2 = 2, \bar{s_3} = 6, \bar{f}_3 = 4$, other $\bar{s}$ are B_{S}, and $\hat{s}$, $\bar{f}$, and $\hat{f}$ are zero.
6	$\bar{s_1} = 2, \bar{f}_1 = 3, \bar{s_2} = 1, \bar{f}_2 = 3, \hat{f}_2 = 2, \bar{s_3} = 6, \hat{s_3} = 3, \bar{s_5} = 3, \bar{f}_5 = 3, \bar{s_9} = 3, \bar{f}_9 = 2, \hat{f}_9 = 1$, other $\bar{s}$ are B_{S}, and $\hat{s}$, $\bar{f}$, and $\hat{f}$ are zero.

TABLE VI
PARAMETER SETTING

Parameter	Setting
MG	250
IP	12
UP	100
IC	0.6
EC	0.3
MP	0.3

C. Effect of Heuristic Algorithm

We evaluate the performance of the heuristic algorithm in terms of the objective value and the computation time, which is called test 3. We use cases 1, 4, and 6 in Table III in this evaluation with uncertain parameters in Table V. In each case, we randomly set several different values of Γ_n^{F} and Γ_n^{S} for each case. We compute the allocations with these different robustnesses by using the heuristic algorithm and the MILP approach. Table VI shows the parameter setting used in this evaluation. The heuristic algorithm is implemented by C++ 15, compiled by Microsoft Visual C++ 2017 v15.9.16, using Intel Core i7-7700 3.60 GHz 4-core CPU, 32 GB memory.

Table VII compares the results of each test obtained by using the heuristic algorithm with those obtained by the MILP approach. We observe that the objective values obtained by the heuristic algorithm have smaller differences with those of the MILP approach, as the size of the network is smaller or the number of unavailabilities is smaller. We note that, when the size of the network is larger or the number of unavailabilities is larger, the difference becomes large; they are 13.79% in rows 7 and 8 of case 4 and 24.41% in row 3 of case 6 in Table VII. The difference in test 3 between the MILP approach and the heuristic algorithm is 3.37% on average.

Table VIII shows the average computation times for the MILP approach, the heuristic algorithm, and the CPU accelerated heuristic algorithm for each case. We observe that the heuristic algorithm without CPU-accumulation reduces the computation time by 99.47% compared with the MILP approach on average; the larger the problem size is, the more the computation time of the heuristic algorithm is reduced compared with that of the MILP approach. The CPU-accumulation reduces the computation time by 71.07% compared with the heuristic algorithm without CPU-accumulation.

D. Large-Scale Evaluation

We compare the proposed model with the baseline models with larger test cases in this subsection. The MILP approach

TABLE VII
COMPARISON OF OBJECTIVE VALUES IN TEST 3

Case	Values of Γ^{F} and Γ^{S}†	Objective value		
		MILP	Heuristic algorithm	Difference
Case 1	$(-1, \frac{1}{3})$	3.83	3.83	0.00%
	$(-1, \frac{2}{3})$	3.83	3.83	0.00%
	$(-1, 1)$	3.83	3.83	0.00%
	$(0, \frac{1}{3})$	3.67	3.67	0.00%
	$(0, \frac{2}{3})$	3.67	3.50	4.63%
	$(0, 1)$	3.67	3.67	0.00%
	$(1, \frac{1}{3})$	3.67	3.67	0.00%
	$(1, \frac{2}{3})$	3.67	3.50	4.63%
	$(1, 1)$	3.67	3.67	0.00%
Case 4	$(-1, -1, \frac{1}{3})$	7.81	7.81	0.00%
	$(-1, -1, 1)$	7.81	7.81	0.00%
	$(-1, 1, \frac{1}{3})$	6.69	6.69	0.00%
	$(-1, 1, 1)$	6.69	6.69	0.00%
	$(1, -1, \frac{1}{3})$	6.67	6.50	2.55%
	$(1, -1, 1)$	6.67	6.50	2.55%
	$(1, 1, \frac{1}{3})$	6.67	5.75	13.79%
	$(1, 1, 1)$	6.67	5.75	13.79%
Case 6	$(-1, -1, \frac{1}{3})$	4.93	4.90	0.61%
	$(-1, -1, 1)$	4.93	4.90	0.61%
	$(1, 1, 1)$	3.80	2.88	24.21%

†$(\Gamma_2^{\mathrm{F}}, \Gamma_1^{\mathrm{S}})$ for Case 1; $(\Gamma_1^{\mathrm{F}}, \Gamma_2^{\mathrm{F}}, \Gamma_1^{\mathrm{S}})$ for Case 4; $(\Gamma_2^{\mathrm{F}}, \Gamma_9^{\mathrm{F}}, \Gamma_3^{\mathrm{S}})$ for Case 6.

TABLE VIII
COMPARISON OF AVERAGE COMPUTATION TIMES IN TEST 3

Case	MILP (s)	Heuristic algorithm (s)	CPU accelerated heuristic algorithm (s)
Case 1	21.43	6.19	2.86
Case 4	249.66	10.69	5.84
Case 6	11310.59	44.89	9.17

spends so much time that it cannot be applied in large-scale tests cases as shown in Table VIII. We compare the results obtained from the heuristic algorithm and three baseline models. The parameters of the algorithm are shown in Table VI. We prepare four test cases with different given conditions as shown in Table IX.

We calculate the allocation of VNFs with given conditions by using the heuristic algorithm and three baseline models. The objective values of the above four methods are shown in Table X. We obverse that allocations obtained from the heuristic algorithm have larger objective values than those from the compared baseline models, especially in cases 9 and 10, where the available space for functions is large and the number of functions is small. Even if there are performance gaps between the results obtained from the heuristic algorithm and those from the proposed model, the results of the heuristic algorithm are still better than those of the baseline models in examined cases.

VI. DISCUSSION

A. Maintenance Ability

Sometimes the maintenance ability of SP is limited. The number of unavailable nodes in one time slot is limited. This section assumes that the maintenance ability on time slot t is limited from M_t^{L} to M_t^{U}, where $M_t^{\mathrm{L}} \leq \sum_{n\in N} e_n^t \leq M_t^{\mathrm{U}}$. By using the known maintenance ability, the size of the uncertainty set $\mathcal{P}_n$ can be reduced to reduce the computation time. The reduction is performed by using the following algorithm which is applied before Algorithm 1.

Algorithm 7 Uncertainty Set Reduction by Using Maintenance Ability

Input: M_t^{L}, M_t^{U}, $\mathcal{P}$, Γ_n^{F}, $\bar{s_n}, \hat{s_n}, \bar{f_n}, \hat{f_n}$, T, N
Output: new uncertainty set
1: **for** $p \in \mathcal{P}$ **do**
2: Calculate the value of e_t^n according to (1) by using p and f_n
3: **for** $t \in T$ **do**
4: **if** $\sum_{n\in N} e_t^n < M_t^{\mathrm{L}}$ or$\sum_{n\in N} e_t^n > M_t^{\mathrm{U}}$ **then**
5: Delete the current p from $\mathcal{P}$
6: **end if**
7: **end for**
8: **end for**

If the MILP approach is applied to solve this model, we add the following equations to reduce the number of possible choices and computational time by using the maintenance ability:

$$\textbf{If} \quad M_t^{\mathrm{L}} \leq \sum_{n\in N} e_t^{np} \leq M_t^{\mathrm{U}} \quad \textbf{then}$$
$$\Delta_t^p = 1$$
$$\textbf{Else}$$
$$\Delta_t^p = 0, \tag{12a}$$
$$\Delta_t^p \in \{0,1\},\ \forall p \in \mathcal{P}, t \in T, \tag{12b}$$

which can be linearized by using Appendix A is changed to:

$$\max \min_{p\in\mathcal{P}} \left\{ \left(\lambda^p + \epsilon \sum_{r\in R} \beta_r^p \right) \cdot \sum_{t\in T} \{\Delta_t^p \cdot B\} \right\}. \tag{13}$$

Here, B is a large constant which is larger than $(|R|+1)\cdot|T|$.

B. More Than One Unavailability Period Per Node

1) Deterministic Unavailability Period: the model in Section III assumes that there is at most one unavailability period on each node. Let us give an extension for supporting more than one unavailability period on each node in this model.

A_n is a set for unavailability periods on node $n \in N$. s_n^a is the starting time slot of the a*th* unavailability period in set A_n. f_n^a is the duration of the a*th* unavailability period in set A_n. Equation (1) is replaced by:

$$\textbf{If} \quad s_n^a \leq t \leq s_n^a + f_n^a \quad \textbf{then}$$
$$e_{tn}^a = 1$$
$$\textbf{Else}$$
$$e_{tn}^a = 0, \tag{14a}$$
$$e_t^n = \vee_{a\in A_n} e_{tn}^a,\ \forall n \in N, t \in T, \tag{14b}$$
$$e_{tn}^a \in \{0,1\},\ \forall n \in N, t \in T, a \in A_n, \tag{14c}$$
$$e_t^n \in \{0,1\},\ \forall n \in N, t \in T. \tag{14d}$$

TABLE IX
EVALUATION CONDITIONS IN SECTION V-D

Case	Nodes	Capacity	Request	Length†	Time slots	Unavailability parameters
9	24	2	12	2:8, 3:2, 4:2	6	$\bar{s_1^1} = \bar{s_2^2} = \bar{s_7^7} = \bar{s_8^8} = \bar{s_{13}^{13}} = \bar{s_{14}^{14}} = \bar{s_{19}^{19}} = \bar{s_{20}^{20}} = 2, \hat{s_1^1} = \hat{s_7^7} = \hat{s_{13}^{13}} = \hat{s_{19}^{19}} = 1, \bar{f_1^1} = \bar{f_7^7} = \bar{f_{13}^{13}} = \bar{f_{19}^{19}} = 1, \bar{f_2^2} = \bar{f_8^8} = \bar{f_{14}^{14}} = \bar{f_{20}^{20}} = 2, \hat{f_2^2} = \hat{f_8^8} = \hat{f_{14}^{14}} = \hat{f_{20}^{20}} = 1$, other $s_n = B_S$ and $f_n = 0$.
10	24	2	15	2:10, 3:3, 4:2	6	$\bar{s_1^1} = \bar{s_2^2} = \bar{s_7^7} = \bar{s_8^8} = \bar{s_{13}^{13}} = \bar{s_{14}^{14}} = \bar{s_{19}^{19}} = \bar{s_{20}^{20}} = 2, \hat{s_1^1} = \hat{s_7^7} = \hat{s_{13}^{13}} = \hat{s_{19}^{19}} = 1, \bar{f_1^1} = \bar{f_7^7} = \bar{f_{13}^{13}} = \bar{f_{19}^{19}} = 1, \bar{f_2^2} = \bar{f_8^8} = \bar{f_{14}^{14}} = \bar{f_{20}^{20}} = 2, \hat{f_2^2} = \hat{f_8^8} = \hat{f_{14}^{14}} = \hat{f_{20}^{20}} = 1$, other $s_n = B_S$ and $f_n = 0$.
11	24	2	18	2:12, 3:3, 4:3	6	$\bar{s_1^1} = \bar{s_2^2} = \bar{s_7^7} = \bar{s_8^8} = \bar{s_{13}^{13}} = \bar{s_{14}^{14}} = \bar{s_{19}^{19}} = \bar{s_{20}^{20}} = 2, \hat{s_1^1} = \hat{s_7^7} = \hat{s_{13}^{13}} = \hat{s_{19}^{19}} = 1, \bar{f_1^1} = \bar{f_7^7} = \bar{f_{13}^{13}} = \bar{f_{19}^{19}} = 1, \bar{f_2^2} = \bar{f_8^8} = \bar{f_{14}^{14}} = \bar{f_{20}^{20}} = 2, \hat{f_2^2} = \hat{f_8^8} = \hat{f_{14}^{14}} = \hat{f_{20}^{20}} = 1$, other $s_n = B_S$ and $f_n = 0$.
12	36	2	18	2:12, 3:3, 4:3	6	$\bar{s_1^1} = \bar{s_2^2} = \bar{s_7^7} = \bar{s_8^8} = \bar{s_{13}^{13}} = \bar{s_{14}^{14}} = \bar{s_{19}^{19}} = \bar{s_{20}^{20}} = \bar{s_{25}^{25}} = \bar{s_{26}^{26}} = \bar{s_{31}^{31}} = \bar{s_{32}^{32}} = 2, \hat{s_1^1} = \hat{s_7^7} = \hat{s_{13}^{13}} = \hat{s_{19}^{19}} = \hat{s_{25}^{25}} = \hat{s_{31}^{31}} = 1, \bar{f_1^1} = \bar{f_7^7} = \bar{f_{13}^{13}} = \bar{f_{19}^{19}} = \bar{f_{25}^{25}} = \bar{f_{31}^{31}} = 1, \bar{f_2^2} = \bar{f_8^8} = \bar{f_{14}^{14}} = \bar{f_{20}^{20}} = \bar{f_{26}^{26}} = \bar{f_{32}^{32}} = 2 \hat{f_2^2} = \hat{f_8^8} = \hat{f_{14}^{14}} = \hat{f_{20}^{20}} = \hat{f_{26}^{26}} = \hat{f_{32}^{32}} = 1$, other $s_n = B_S$ and $f_n = 0$.

†x:y: there are y requests whose length are x.

TABLE X
EVALUATION RESULTS IN SECTION V-D

Case	Obtained from	(Γ_2^F, Γ_1^S)					
		$(0, \frac{1}{3})$	$(0, \frac{2}{3})$	$(0, 1)$	$(1, \frac{1}{3})$	$(1, \frac{2}{3})$	$(1, 1)$
9	Heuristic algorithm	7.00	7.00	7.00	7.00	7.00	7.00
	Double-slot allocation model	4.82	4.82	4.82	4.82	4.82	4.82
	Single-slot allocation model	2.33	2.33	2.33	2.33	2.33	2.33
	Persistent allocation model	2.67	2.67	2.67	1.58	1.58	1.58
10	Heuristic algorithm	2.91	2.91	3.50	2.91	2.87	2.87
	Double-slot allocation model	2.76	2.76	2.76	2.76	2.76	2.76
	Single-slot allocation model	2.33	2.33	2.33	1.31	1.31	1.31
	Persistent allocation model	2.69	2.69	2.69	1.62	1.62	1.62
11	Heuristic algorithm	2.83	2.81	2.85	2.41	1.81	2.56
	Double-slot allocation model	2.67	2.67	2.67	2.67	2.67	2.67
	Single-slot allocation model	2.33	2.33	2.33	1.27	1.27	1.27
	Persistent allocation model	2.67	2.67	2.67	1.59	1.59	1.59
12	Heuristic algorithm	7.00	7.00	7.00	7.00	7.00	7.00
	Double-slot allocation model	4.82	4.82	4.82	4.82	4.82	4.82
	Single-slot allocation model	2.33	2.33	2.33	2.33	2.33	2.33
	Persistent allocation model	2.66	2.66	2.66	1.59	1.59	1.59

According to the linearization process in [9] and Appendix A, (14) is linearized and (29a)-(29h) are replaced by the following equations:

$$t - s_n^a + \epsilon_1 \le \eta_{tn}^a \cdot B, \ \forall t \in T, n \in N, a \in A_n, \tag{15a}$$
$$t - s_n^a + \epsilon_1 \ge (\eta_{tn}^a - 1) \cdot B, \ \forall t \in T, n \in N, a \in A_n, \tag{15b}$$
$$s_n^a + f_n^a - t + \epsilon_1 \le \rho_{tn}^a \cdot B, \ \forall t \in T, n \in N, a \in A_n, \tag{15c}$$
$$s_n^a + f_n^a - t + \epsilon_1 \ge (\rho_{tn}^a - 1) \cdot B, \ \forall t \in T, n \in N, a \in A_n, \tag{15d}$$
$$e'_{atn} \le \eta_{tn}^a, \ \forall t \in T, n \in N, a \in A_n, \tag{15e}$$
$$e'_{atn} \le \rho_{tn}^a, \ \forall t \in T, n \in N, a \in A_n, \tag{15f}$$
$$e'_{atn} \ge \eta_{tn}^a + \rho_{tn}^a - 1, \ \forall t \in T, n \in N, a \in A_n, \tag{15g}$$
$$e_t^n \ge \frac{1}{|A_n|} \cdot \sum_{a \in A_n} e'_{atn}, \ \forall t \in T, n \in N, \tag{15h}$$
$$e_t^n \le \sum_{a \in A_n} e'_{atn}, \ \forall t \in T, n \in N, \tag{15i}$$
$$e'_{atn}, \eta_{tn}^a, \rho_{tn}^a \in [0, 1], \ \forall t \in T, n \in N, a \in A_n, \tag{15j}$$
$$e_t^n \in \{0, 1\}, \ \forall n \in N, t \in T. \tag{15k}$$

2) Uncertain Unavailability Period: Confronted with uncertain unavailability periods and multiple unavailability periods, Γ_n^S still controls the size of uncertainty set of beginning time slots $\mathcal{P}_n$ on node n. The larger Γ_n^S is, the more the considered possible beginning time slots of unavailability periods on node n are. For unavailability period $a \in A_n$, we choose $\lfloor \Gamma_n^S \cdot (2 \cdot \hat{s_n^a} + 1) \rfloor$ different beginning time slots from $[\bar{s_n^a} - \hat{s_n^a}, \bar{s_n^a} + \hat{s_n^a}]$ and form a set which is an element in $\mathcal{P}_n^a$. $\mathcal{P}_n^a$ contains all possible choices of uncertainty period a on node n. The size of $\mathcal{P}_n^a$ is $|\mathcal{P}_n^a| = \binom{2 \cdot \hat{s_n^a} + 1}{\lfloor \Gamma_n^S \cdot (2 \cdot \hat{s_n^a} + 1) \rfloor}$. The uncertainty set for all nodes is denoted by $\mathcal{P} = \{\{\mathcal{P}_1^1, \ldots, \mathcal{P}_1^{|A_n|}\}, \{\mathcal{P}_2^1, \ldots, \}, \ldots, \{, \ldots, \mathcal{P}_{|N|}^{|A_n|}\}\}$. One element in $\mathcal{P}$ is a combination of one element in each $\mathcal{P}_n$. One element in $\mathcal{P}_n$ is a combination of one element in each $\mathcal{P}_n^a$. Let p denote an element in $\mathcal{P}$; p_n denotes the selected set from $\mathcal{P}_n$; p_n^q denotes the qth element in p_n; p_n^{qa} denotes the selected starting time slot of a*th* unavailability period in the element in p_n^q.

Γ_n^F controls the considered length of the duration of unavailability period on node n, which is $\bar{f_n^a} + \hat{f_n^a} \cdot \lfloor \Gamma_n^F \rfloor$.

Equations (5a)-(5h) are replaced by:

$$t - p_n^{qa} + \epsilon_1 \le \eta_{tn}^{qa} \cdot B,\ \forall t \in T, n \in N, q \in [1, |p_n|], a \in A_n, \tag{16a}$$

$$t - p_n^{qa} + \epsilon_1 \ge \left(\eta_{tn}^{qa} - 1\right) \cdot B,\ \forall t \in T, n \in N, q \in [1, |p_n|], a \in A_n, \tag{16b}$$

$$p_n^{qa} + \bar{f}_n^a + \Gamma_n^{\mathrm{F}} \cdot \hat{f}_n^a - t + \epsilon_1 \le \rho_{tn}^{qa} \cdot B,\ \forall t \in T, n \in N, q \in [1, |p_n|], a \in A_n, \tag{16c}$$

$$p_n^{qa} + \bar{f}_n^a + \Gamma_n^{\mathrm{F}} \cdot \hat{f}_n^a - t + \epsilon_1 \ge \left(\rho_{tn}^{qa} - 1\right) \cdot B,\ \forall t \in T, n \in N, q \in [1, |p_n|], a \in A_n, \tag{16d}$$

$$\tau'_{tnqa} \le \eta_{tn}^{qa},\ \forall t \in T, n \in N, q \in [1, |p_n|], a \in A_n, \tag{16e}$$

$$\tau'_{tnqa} \le \rho_{tn}^{qa},\ \forall t \in T, n \in N, q \in [1, |p_n|], a \in A_n, \tag{16f}$$

$$\tau'_{tnqa} \ge \eta_{tn}^{qa} + \rho_{tn}^{qa} - 1,\ \forall t \in T, n \in N, q \in [1, |p_n|], a \in A_n, \tag{16g}$$

$$\tau_{tn}^{q} \ge \frac{1}{|A_n|} \cdot \sum_{a \in A_n} \tau'_{tnqa},\ \forall t \in T, n \in N, q \in [1, |p_n|], \tag{16h}$$

$$\tau_{tn}^{q} \le \sum_{a \in A_n} \tau'_{tnqa},\ \forall t \in T, n \in N, q \in [1, |p_n|], \tag{16i}$$

$$\tau'_{tnqa}, \eta_{tn}^{qa}, \rho_{tn}^{qa} \in [0, 1],\ \forall t \in T, n \in N, q \in [1, |p_n|], a \in A_n, \tag{16j}$$

$$\tau_{tn}^{q} \in [0, 1],\ \forall t \in T, n \in N, q \in [1, |p_n|]. \tag{16k}$$

C. Unpredictable Unavailabilities

We assume that the availability schedules used in the proposed model are known by maintenance schedules in the proposed model. In maintenance schedules, the locations of availabilities and unavailabilities are given so that the availability schedules can be expressed by binary matrixes. By comparison, there are some availabilities and unavailabilities whose locations cannot be specified, such as burst unavailabilities and probabilistic unavailabilities.

Burst unavailabilities are caused by burst hardware and software failures of servers. The influences of burst unavailabilities cannot be avoided by the scheduling of the proposed model. The common ways to suppress the influence of the burst unavailabilities are backup and replication. Probabilistic unavailabilities from prediction systems give probabilistic results instead of the exact results. If we want to use the probabilistic results as binary availability schedules, quantization of the probabilistic results are necessary, which will bring extra uncertainty. If the administrators consider that the cost is acceptable, they can set a threshold of probability. If the probability exceeds the threshold, it is considered to be one; otherwise, zero.

VII. Directions to Extend Proposed Model

A. Separate Request From SFC

We assume that one request corresponds to one SFC in Section III. We do not consider sharing VNFs among different SFCs in the proposed model. We give a direction on how to separate the requests from the SFCs in this subsection. We redefine the requests, SFCs, and VNFs to replace the definitions in paragraph 5 of Section III-A. The definition of decision variable x_{tn}^{rk} in paragraph 7 of Section III-A is 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 $c \in C$. 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.

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

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

$$o_t^r = \prod_{n \in N} \left\{ \left(\prod_{c \in C} \prod_{f \in F^c} \psi_{fc} \gamma_{cr} \left(x_{tn}^f \odot x_{t-1,n}^f \right) \right) \wedge \left(1 - \left(\prod_{c \in C} \prod_{f \in F^c} \psi_{fc} \gamma_{cr} x_{tn}^f \right) \wedge e_t^n \right) \wedge \left(1 - \left(\prod_{c \in C} \prod_{f \in F^c} \psi_{fc} \gamma_{cr} x_{t-1,n}^f \right) \wedge e_{t-1}^n \right) \right\}, \quad \forall r \in R, t \in T \backslash \{1\}, \tag{17a}$$

$$\sum_{f \in F} x_{tn}^f q_s^f \le c_{nt}^s,\ \forall n \in N, t \in T, s \in S, \tag{17b}$$

$$\sum_{f \in F^c} x_{tn}^f \le 1,\ \forall c \in C, n \in N, t \in T, \tag{17c}$$

$$\sum_{n \in N} x_{tn}^f = 1,\ \forall f \in F, t \in T, \tag{17d}$$

$$x_{tn}^f \wedge e_t^n = 0,\ \forall t \in T, n \in N, f \in F. \tag{17e}$$

Equation (17a) replaces x_{tn}^{rk} in (2) with $\prod_{c \in C} \prod_{f \in F^c} \psi_{fc} \gamma_{cr} x_{tn}^f$. Equation (17b) replaces [10, eq. (9)] and ensures that each node's computational resources must not exceed its capacity during allocation. Equation (17c) replaces [10, eq. (10)] 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 [23]. Equation (17d) replaces [10, eq. (11)] and ensures that all functions are allocated in the network.

B. Multiple Active Replicas for a VNF

We assume that only one replica is active for each VNF at the same time regardless of the required processing abilities of requests and the processing abilities which can be provided by VNFs in Section III. Actually, multiple replicas can be active at the same time and the processing abilities of the active replicas need to meet the required processing abilities of the requests. Based on Section VII-A, we introduce the following parameters about replicas and constraints.

Each VNF $f \in F$ can be replaced by a pool of A^f replica VNFs, each of which requires different capacities with an extra

overhead on capacity but behaves collectively as the original one; the capacity of each replica VNF can be different from each other associated with the same original VNF. p_{fa} represents the processing ability of the ath replica of VNF $f \in F$. q_s^{fs} represents the resource requirement of the ath replica of VNF $f \in F$ for resource $s \in S$. The required processing ability of request $r \in R$ is denoted by g_r, which is a given parameter. We use binary decision variable x_{tn}^{fa} to represent the allocation; x_{tn}^{fa} is set to 1 if the ath replica of function $f \in F$ is assigned to node $n \in N$ at time slot $t \in T$, and 0 otherwise.

$$o_t^r = \prod_{n\in N}\Bigg\{\Bigg(\prod_{c\in C}\prod_{f\in F^c}\psi_{fc}\gamma_{cr}\prod_{a\in A^f}\left(x_{tn}^{fa}\odot x_{t-1,n}^{fa}\right)\Bigg) \wedge\left(1-\left(\prod_{c\in C}\prod_{f\in F^c}\psi_{fc}\gamma_{cr}\prod_{a\in A^f}x_{tn}^{fa}\right)\wedge e_t^n\right) \wedge\left(1-\left(\prod_{c\in C}\prod_{f\in F^c}\psi_{fc}\gamma_{cr}\prod_{a'\in A^f}x_{t-1,n}^{fa}\right) \wedge e_{t-1}^n\right)\Bigg\},\ \forall r\in R, t\in T\backslash\{1\}, \tag{18a}$$

$$\sum_{f\in F}\sum_{a\in A^f}x_{tn}^{fa}q_s^{fa}\le c_{nt}^s,\ \forall n\in N, t\in T, s\in S, \tag{18b}$$

$$\sum_{a\in A^f}x_{tn}^{fa}\le 1,\ \forall c\in C, n\in N, t\in T, f\in F, \tag{18c}$$

$$x_{tn}^{fa}\wedge e_t^n=0,\ \forall t\in T, n\in N, f\in F, a\in A^f, \tag{18d}$$

$$\sum_{n\in N}\sum_{a\in A^f}p_{fa}x_{tn}^{fa}\ge\sum_{c\in C}\psi_{fc}\sum_{r\in R}\gamma_{cr}g_r,\ \forall t\in T, f\in F. \tag{18e}$$

Equations (17a)-(17e) are replaced by (18a)-(18d). Equation (18b) ensures that each node's computational resources must not exceed its capacity during allocation. Equation (18c) ensures that the replicas of the same VNF cannot be assigned to the same node. Equation (18e) ensures that the sum of the processing abilities of the replicas of each VNF meets the required processing abilities from the requirements at each time slot.

C. Network-Aware Placement

The proposed model does not consider the routing between VNFs in the same SFC or the recovery path from the unavailable VNFs to their new locations. In a real deployment, some paths cannot be chosen because of the limitation of the characteristic of links, such as bandwidth and latency. This subsection gives a direction on how to handle the routing problems by taking advantage of the network topology. Each virtual link $(i,j)\in L$ corresponds to a connection between two VMs with transmission resource b_{ij} and length l_{ij}. The transmission resources demanded by request $r\in R$ is d_r. The latency of request $r\in R$ is required to be less than ι_r.

1) Routing of SFCs: We present the following constraints to compute the VNF allocation and the routes of all SFCs.

The flow constraint of SFC paths is given by: $\forall w\in N, r\in R, k\in K_r, t\in T$,

$$\sum_{(i,j)\in L}\alpha_{w,ij}\rho_{rt}^{k,ij}=\begin{cases}-1, & \text{if } x_{tw}^{rk}=1 & \text{(19a)}\\ 1, & \text{if } x_{tw}^{r,k+1}=1 & \text{(19b)}\\ 0, & \text{if } x_{tw}^{rk}=x_{tw}^{r,k+1}=0. & \text{(19c)}\end{cases}$$

For each request, there are three types of nodes: source node (a node at which the first function of a request is allocated), destination node (a node at which the last function of a request is allocated), and others. We define indicator $\alpha_{w,ij}, w\in N, (i,j)\in L$, to represent the adjacency of nodes on directed graph G, where $\alpha_{w,ij}=1$ if node w is the tail of the directed link (i, j), i.e., $w=j$; $\alpha_{w,ij}=-1$ if node w is the head of the directed link (i, j), i.e., $w=i$; $\alpha_{w,ij}=0$ otherwise. We use binary variable $\rho_{rt}^{k,ij}$ to express the route. If link (i, j) is a segment link between the kth function and the $k+1$th function of request $r\in R$ at time slot $t\in T$, $\rho_{rt}^{k,ij}=1$, and 0 otherwise. We have the following constraints. According to [10, eq. (10)], x_{tw}^{rk} and $x_{tw}^{r,k+1}$ cannot be 1 at the same time. Thus (19a), (19b), and (19c) can be simplified to:

$$\sum_{(i,j)\in L}\alpha_{w,ij}\rho_{rt}^{k,ij}=-x_{tw}^{rk}+x_{tw}^{r,k+1},\quad \forall k\in K_r\backslash\{|K_r|\}, r\in R, t\in T, w\in N. \tag{20}$$

The link capacity constraint is given by:

$$\sum_{r\in R}\sum_{k\in K_r}\rho_{rt}^{k,ij}d_r\le b_{ij},\ \forall (i,j)\in L, t\in T, \tag{21}$$

which ensures that each link's transmission resource is not overused. The latency constraint is given by:

$$\sum_{t\in T}\sum_{k\in K_r}\sum_{(i,j)\in L}l_{ij}\rho_{rt}^{k,ij}\le\iota_r,\ \forall r\in R, \tag{22}$$

which ensures that each request meets the requirement of latency.

2) Routing During Recovery: For stateful application, the state information needs to be synchronized between the old and new nodes, which consumes the transmission resource of links. The flow constraint of recovery is given by: $\forall w\in N, r\in R, k\in K_r, t\in T\backslash\{1\}$,

$$\sum_{(i,j)\in L}\alpha_{w,ij}\tau_{rt}^{k,ij}=\begin{cases}-1, & \text{if } x_{t-1,w}^{rk}=1 & \text{(23a)}\\ 1, & \text{if } x_{tw}^{r,k}=1 & \text{(23b)}\\ 0, & \text{if } x_{t-1,w}^{rk}=x_{tw}^{r,k}=0. & \text{(23c)}\end{cases}$$

We use binary variable $\tau_{rt}^{k,ij}$ to express the route. If link (i, j) is a segment link between the nodes where the kth function of request $r\in R$ is allocated at time slot $t\in T\backslash\{1\}$ and time slot $t-1$, $\tau_{rt}^{k,ij}=1$, and 0 otherwise. The link capacity constraint except for the first time slot is given by:

$$\sum_{r\in R}\sum_{k\in K_r}\left(\rho_{rt}^{k,ij}+\tau_{rt}^{k,ij}\right)d_r\le b_{ij},\ \forall (i,j)\in L, t\in T\backslash\{1\}. \tag{24}$$

D. Backup Functions

We focus on the primary function allocation in the proposed model. The proposed model suppresses the interruptions which decrease SSCAT with determining a suitable allocation of VNFs. If we take backup functions into consideration, the interruptions can be avoided and continuous available time slots of SFCs can be extended. Moreover, we assume that the availability schedules provided by the administrators are relatively believable regardless of the detailed recovery mechanism. As a direction of progress, the recovery mechanism during unavailable periods is worthy to be studied. The work in [27] addressed a primary and backup VNF placement model for improving the continuous available time of SFCs by avoiding the interruptions caused by unavailable nodes and function reallocations.

VIII. Related Work

This paper addresses the uncertainty of availability schedule in the proposed model. In the past researches, robust techniques were applied to deal with uncertainty in the problems. To deal with the data uncertainty in linear programming, Soyster [28] introduced a linear optimization model to construct a solution that is feasible for all data that belong to a convex set, which is too conservative and gives up much of the optimality for the problem. To overcome the over-conservation, Ben-Tal *et al.* [29] introduced less conservative models by considering uncertain linear problems with ellipsoidal uncertainties. However, the above models are designed for convex uncertainty sets. In our model, the uncertainty set is not a convex set, which is discrete. To deal with the robust discrete optimization problem, Bertsimas and Sim [30] introduced an approach for robust linear optimization problem based on [28]; it offers an ability to control the degree of conservatism for each constraint.

Robust optimization techniques have been applied to different network design problems. By using the approach in [30], [31] introduced a robust integer programming problem with the data uncertainty in network flow problems and solved the minimum cost flow problems. The work in [13] introduced a problem of backup network design for general link loads, where the uncertainty is the number of primary links that fail. The work in [32] applied the robust optimization to primary and backup allocation problem, where the number of failing PMs is given but which PM fails is uncertain. The work in [14] adopted the robust optimization technique on minimizing the required backup capacity with probabilistic protection against multiple PM failures, where the uncertainty of capacity was considered. The above researches consider the discrete uncertainty sets and provide an approximate solution. Our model provides an exact solution without gaps by taking advantage of the limited size of uncertainty set in the proposed model.

Various studies have considered the impact of service interruptions on VNF allocation problems. The work in [33] concerned the low reliability of softwarized networks caused by service interruptions. The model presented in [23] addressed to minimize the cost of migration which is evaluated by the number of migrations. The work in [34] introduced a model of the adaptive and dynamic VNF allocation problem considering the interruptions caused by VNF migration. The work in [35] introduced a model for dynamic VNF placement under changing traffic load. A static placement decreases the operation cost. Reconfiguration causes service interruption and the operation cost increases. Compared with existing models which also care about the interruptions of services, the proposed model does not need extra server resources including storage and computation resources with a specific aim of continuous available time. 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 the evaluations presented in Section V, by considering the availability schedule, we use the three baseline models, which are different from the previous models mentioned above and have different objective functions without considering the availability schedule. Several works introduced their suitable benchmark models by capturing the features of each evaluation scenario. The work in [13] considered three protection types of links: cycle protection, two-hop protection, and one-hop protection, which were used to be compared with the introduced model. The work in [14] formulated a conventional deterministic-capacity model, which was compared with the three introduced models. The work in [23] presented a comparison among three different application conditions of the introduced model with considering the case that only migrations are used, the case that only replications are used, and the case that both migrations and replication can be used. The authors in [32] evaluated their introduced model under different failure probabilities with guaranteeing different availability probabilities. The work in [33] developed three algorithms for bench-marking purposes. The authors in [34] compared their developed heuristic algorithm and the introduced model with three different objective functions.

The features of the existing models mentioned in Section VIII and the proposed model are summarized in Table XI.

IX. Conclusion

This paper proposed a robust VNF allocation model for improving the continuous available time of service function chains with considering the uncertain availability schedule. We formulated the proposed model as an MILP problem. Numerical results showed that the proposed model improves the continuous available time of SFCs, compared with the persistent allocation model, the single-slot allocation model, and the double-slot allocation model in both deterministic and uncertain availability schedules. In the cases examined, the proposed model can provide longer continuous available time slots compared with the three baseline models under different levels of the robustness of uncertain availability schedule. The developed heuristic algorithm reduces the computation time by 99.85% compared with the MILP approach with a limited performance penalty by 3.37% in our evaluations. We evaluated the relationship between availability schedules and objective values. The size of the uncertainty set can be reduced

TABLE XI
COMPARISON OF EXISTING MODELS IN SECTION VIII AND PROPOSED MODEL

Ref.	Handled issue	Objective	Optimization problem	Solution	Models for comparison
[13]	Provide protection from multiple random link failures involving probabilistic survivability guarantees.	Minimize the cost of dedicated backup network by guaranteeing recovery from failures with high probability	ILP	Solved by optimization tool (CPLEX), and a simulated annealing heuristic.	Cycle protection, two-hop protection, and one-hop protection.
[14]	Decide the allocation of virtual machines under two uncertainties, failure event, and virtual machine capacity.	Minimize the total required capacity for backup PMs.	MILP	Solved by optimization tool (COIN-OR CBC).	Deterministic-capacity model.
[23]	Trade-off between the resources for VNF replica and number of migrations.	Minimize the sum of migration, server, and link costs with given weights.	LP	Solved by optimization tool (Gurobi).	Only migrations are used, only replications are used, and when both migrations and replication can be used.
[32]	Provide a probabilistic protection for primary physical machines with backup resources against random failures	Minimize the total required backup capacity.	ILP	Solved by optimization tool (CPLEX) and a simulated annealing heuristic.	Introduced model with different parameters.
[33]	Determine the number of required VNF backups in order to guarantee the required reliability.	Establish service chains while efficiently utilizing network resources.	ILP	Solved by optimization tool (CPLEX) and a heuristic algorithm.	A reliability agnostic scheme with bi-directional search-based greedy shortest path scheme, a reliability aware routing without resource sharing based on bi-directional search and a single-direction search-based greedy shortest path scheme with resource sharing.
[34]	The allocation model of a virtual network function forwarding graph in a network where the traffic is changing over time.	Minimize the cost of reallocation, the migration distance, and transmission distance are considered.	ILP	Solved by an alternating direction method of multipliers-based algorithm.	Developed heuristic algorithm and the introduced models with different objective function: with migration and end-to-end distance costs and without them.
[35]	Trade-off between the reconfiguration of SFCs and the optimality of the resulting placement and (re)routing.	Minimize the new placement cost, and the reconfiguration cost with given weights.	INLP	Solved by optimization tool (MATLAB toolbox YALMIP and Gurobi).	-
Proposed model	Improve the continuous available time for services with availability schedule.	Maximize the shortest service continuous available time among all services in the network.	ILP	Solved by optimization tool (CPLEX) and a genetic algorithm based algorithm.	Persistence allocation model, single-slot allocation model, and double-slot allocation model.

ILP: integer linear programming, LP: linear programming, INLP: integer nonlinear program, MILP: mixed ILP.

according to our observations. We gave two discussions of the proposed model: one for maintenance ability and the other for multiple unavailability periods on each node. In addition, we provided four directions to extend the proposed model.

APPENDIX A
LINEARIZATION OF PROPOSED MODEL

We introduce the following linearization process for the proposed model given in Section III. $x = \min_i\{y_i\}$ can be expressed in linear form by:

$$x = \min_i\{y_i\} \Leftrightarrow \begin{cases} x \le y_i + (1-\delta_i)\cdot B, \forall i \in Y & (25a)\\ x \ge y_i - (1-\delta_i)\cdot B, \forall i \in Y & (25b)\\ y_i \le (1-\delta_i)\cdot B + y_j, \forall i \in Y, \\ \quad j \in Y\backslash\{i\} & (25c)\\ \sum_{i\in Y}\delta_i = 1 & (25d)\\ x \le y_i, \forall i \in Y & (25e)\\ \delta_i \in \{0,1\}, \forall i \in Y, & (25f) \end{cases}$$

where B is sufficiently large to ensure that its value is larger than $y_i, i \in Y$.

For binary decision variables $y_i \in Y$, the operation $x = \vee_{i\in Y} y_i = y_1 \vee y_2 \vee \cdots \vee y_{|Y|}$ can be expressed in linear form as follows:

$$x = \vee_{i\in Y} y_i \Leftrightarrow \begin{cases} x \ge \frac{1}{|Y|}\cdot\sum_{i\in Y} y_i & (26a)\\ x \le \sum_{i\in Y} y_i & (26b)\\ x, y_i \in \{0,1\}, \forall i \in Y. & (26c) \end{cases}$$

The relationship

$$\begin{aligned} &\textbf{If} \quad a \le x \le b \quad \textbf{then}\\ &\qquad e = 1\\ &\textbf{Else}\\ &\qquad e = 0 \qquad (27a)\\ &e \in [0,1] \qquad (27b)\\ &x \in \mathbb{Z}. \qquad (27c) \end{aligned}$$

can be linearized by using the following equations:

$$x - a + \epsilon \leq \alpha \cdot B \tag{28a}$$
$$x - a + \epsilon \geq (\alpha - 1) \cdot B \tag{28b}$$
$$b - x + \epsilon \leq \beta \cdot B \tag{28c}$$
$$b - x + \epsilon \geq (\beta - 1) \cdot B \tag{28d}$$
$$e \leq \alpha \tag{28e}$$
$$e \leq \beta \tag{28f}$$
$$e \geq \alpha + \beta - 1 \tag{28g}$$
$$e, \alpha, \beta \in [0, 1] \tag{28h}$$
$$x \in \mathbb{Z}. \tag{28i}$$

a, b are given integer parameters. ϵ is a given positive parameter: $0 < \epsilon < 1$. B is a number which is larger than $x - a + \epsilon$ and $b - x + \epsilon$.

APPENDIX B
LINEARIZATION IN SECTION III-A

$$t - s_n + \epsilon_1 \leq \eta_t^n \cdot B, \ \forall t \in T, n \in N, \tag{29a}$$
$$t - s_n + \epsilon_1 \geq (\eta_t^n - 1) \cdot B, \ \forall t \in T, n \in N, \tag{29b}$$
$$s_n + f_n - t + \epsilon_1 \leq \rho_t^n \cdot B, \ \forall t \in T, n \in N, \tag{29c}$$
$$s_n + f_n - t + \epsilon_1 \geq (\rho_t^n - 1) \cdot B, \ \forall t \in T, n \in N, \tag{29d}$$
$$e_t^n \leq \eta_t^n, \ \forall t \in T, n \in N, \tag{29e}$$
$$e_t^n \leq \rho_t^n, \ \forall t \in T, n \in N, \tag{29f}$$
$$e_t^n \geq \eta_t^n + \rho_t^n - 1, \ \forall t \in T, n \in N, \tag{29g}$$
$$e_t^n, \eta_t^n, \rho_t^n \in [0, 1], \ \forall t \in T, n \in N, \tag{29h}$$
$$\beta_r - 1 \leq j y_j^r + \left(1 - \delta_j^r\right) \cdot B, \ \forall j \in T, r \in R, \tag{30a}$$
$$\beta_r - 1 \geq j y_j^r - \left(1 - \delta_j^r\right) \cdot B, \ \forall j \in T, r \in R, \tag{30b}$$
$$j y_j^r \geq \left(\delta_j^r - 1\right) \cdot B + y_{j'}^r, \ \forall j \in T, j' \in T \backslash \{j\}, r \in R, \tag{30c}$$
$$\sum_{j \in T} \delta_j^r = 1, \ \forall r \in R, \tag{30d}$$
$$\beta_r - 1 \geq j y_j^r, \ \forall j \in T, r \in R, \tag{30e}$$
$$\delta_j^r \in \{0,1\}, \ \forall j \in T, r \in R, \tag{30f}$$
$$\phi_{tn}^{rk} = 1 - x_{tn}^{rk} - x_{t-1,n}^{rk} + 2 \cdot h_{tn}^{rk}, \ \forall r \in R, t \in T \backslash \{1\}, k \in K_r, n \in N, \tag{31a}$$
$$h_{tn}^{rk} \leq x_{tn}^{rk}, \ \forall r \in R, t \in T \backslash \{1\}, k \in K_r, n \in N, \tag{31b}$$
$$h_{tn}^{rk} \leq x_{t-1,n}^{rk}, \ \forall r \in R, t \in T \backslash \{1\}, k \in K_r, n \in N, \tag{31c}$$
$$h_{tn}^{rk} \geq x_{tn}^{rk} + x_{t-1,n}^{rk} - 1, \ \forall r \in R, t \in T \backslash \{1\}, k \in K_r, n \in N, \tag{31d}$$
$$\alpha_{tn}^{rk} \leq x_{tn}^{rk}, \ \forall r \in R, t \in T, k \in K_r, n \in N, \tag{31e}$$
$$\alpha_{tn}^{rk} \leq e_t^n, \ \forall r \in R, t \in T, k \in K_r, n \in N, \tag{31f}$$
$$\alpha_{tn}^{rk} \geq x_{tn}^{rk} + e_t^n - 1, \ \forall r \in R, t \in T, k \in K_r, n \in N, \tag{31g}$$
$$\theta_{tn}^{rk} \leq \phi_{tn}^{rk}, \ \forall r \in R, t \in T \backslash \{1\}, k \in K_r, n \in N, \tag{31h}$$
$$\theta_{tn}^{rk} \leq 1 - \alpha_{tn}^{rk}, \ \forall r \in R, t \in T \backslash \{1\}, k \in K_r, n \in N, \tag{31i}$$
$$\theta_{tn}^{rk} \geq \phi_{tn}^{rk} - \alpha_{tn}^{rk}, \ \forall r \in R, t \in T \backslash \{1\}, k \in K_r, n \in N, \tag{31j}$$
$$\pi_{tn}^{rk} \leq \theta_{tn}^{rk}, \ \forall r \in R, t \in T \backslash \{1\}, k \in K_r, n \in N, \tag{31k}$$
$$\pi_{tn}^{rk} \leq 1 - \alpha_{t-1,n}^{rk}, \ \forall r \in R, t \in T \backslash \{1\}, k \in K_r, n \in N, \tag{31l}$$
$$\pi_{tn}^{rk} \geq \theta_{tn}^{rk} - \alpha_{t-1,n}^{rk}, \ \forall r \in R, t \in T \backslash \{1\}, k \in K_r, n \in N, \tag{31m}$$
$$w_t^{rk} \leq \pi_{tn}^{rk}, \ \forall r \in R, t \in T \backslash \{1\}, k \in K_r, n \in N, \tag{31n}$$
$$w_t^{rk} \geq \sum_{n \in N} \pi_{tn}^{rk} - |N| + 1, \ \forall r \in R, t \in T \backslash \{1\}, k \in K_r, \tag{31o}$$
$$o_t^r \leq w_t^{rk}, \ \forall r \in R, t \in T \backslash \{1\}, k \in K_r, \tag{31p}$$
$$o_t^r \geq \sum_{k \in K_r} w_t^{rk} - |K_r| + 1, \ \forall r \in R, t \in T \backslash \{1\}, \tag{31q}$$
$$o_1^r = 0, \ \forall r \in R, \tag{31r}$$
$$o_t^r \in \{0,1\}, \ \forall r \in R, t \in T, \tag{31s}$$
$$y_j^r \in \{0,1\}, \ \forall r \in R, j \in T, \tag{31t}$$
$$w_t^{rk} \in \{0,1\}, \ \forall r \in R, k \in K_r, t \in T \backslash \{1\}, \tag{31u}$$
$$\phi_{tn}^{rk}, h_{tn}^{rk}, \theta_{tn}^{rk}, \pi_{tn}^{rk} \in \{0,1\}, \ \forall r \in R, k \in K_r, t \in T \backslash \{1\}, n \in N, \tag{31v}$$
$$\alpha_{tn}^{rk}, x_{tn}^{rk}, \in \{0,1\}, \ \forall r \in R, k \in K_r, t \in T, n \in N. \tag{31w}$$

In the above equations, ϵ_1, ϵ_2, and B are given parameters, where $0 < \epsilon_1 < \frac{1}{s_n + f_n}$, $0 < \epsilon_2 < 1$, and B is larger than $s_n + f_n + 1 - t, t - s_n + \epsilon_2, n \in N, t \in T$, and 1. The minimum of B can be taken to be $|T| + 2$.

APPENDIX C
OBJECTIVE FUNCTION AND CONSTRAINTS UNDER UNCERTAINTY SET $p \in \mathcal{P}$ IN SECTION III-B

$$\left(\sum_{t=i}^{i+j-1} o_t^{rp}\right) - j + 1 \leq z_i^{jrp}, \ \forall i \in T, j \in T_i, r \in R, p \in \mathcal{P}, \tag{32a}$$
$$z_i^{jrp} \leq \frac{\sum_{t=i}^{i+j-1} o_t^{rp}}{j}, \ \forall i \in T, j \in T_i, r \in R, p \in \mathcal{P}, \tag{32b}$$
$$z_i^{jrp} \leq y_j^{rp}, \ \forall i \in T, j \in T_i, r \in R, p \in \mathcal{P}, \tag{32c}$$
$$y_j^{rp} \leq \sum_{t=1}^{|T|-j+1} z_t^{jrp}, \ \forall j \in T, r \in R, p \in \mathcal{P}, \tag{32d}$$
$$\lambda^p \leq \beta_r^p, \ \forall r \in R, p \in \mathcal{P}, \tag{33a}$$
$$\sum_{r \in R} \sum_{k \in K_r} x_{tn}^{rkp} \leq c_{nt}^s, \ \forall n \in N, t \in T, p \in \mathcal{P}, s \in S, \tag{34a}$$
$$\sum_{k \in K_r} x_{tn}^{rkp} \leq 1, \ \forall r \in R, n \in N, t \in T, p \in \mathcal{P}, \tag{34b}$$
$$\sum_{n \in N} x_{tn}^{rkp} = 1, \ \forall r \in R, k \in K_r, t \in T, p \in \mathcal{P}, \tag{34c}$$
$$\beta_r^p - 1 \leq j y_j^{rp} + \left(1 - \delta_j^{rp}\right) \cdot B, \ \forall j \in T, r \in R, p \in \mathcal{P}, \tag{35a}$$

$$\beta_r^p - 1 \geq jy_j^{rp} - \left(1 - \delta_j^{rp}\right) \cdot B,\ \forall j \in T, r \in R, p \in \boldsymbol{\mathcal{P}}, \tag{35b}$$

$$jy_j^{rp} \geq \left(\delta_j^{rp} - 1\right) \cdot B + y_{j'}^{rp},\ \forall j \in T, j' \in T \backslash \{j\}, r \in R, p \in \boldsymbol{\mathcal{P}}, \tag{35c}$$

$$\sum_{j \in T} \delta_j^{rp} = 1,\ \forall r \in R, p \in \boldsymbol{\mathcal{P}}, \tag{35d}$$

$$\beta_r^p - 1 \geq jy_j^{rp},\ \forall j \in T, r \in R, p \in \boldsymbol{\mathcal{P}}, \tag{35e}$$

$$\delta_j^{rp} \in \{0,1\},\ \forall j \in T, r \in R, p \in \boldsymbol{\mathcal{P}}, \tag{35f}$$

$$\phi_{tn}^{rkp} = 1 - x_{tn}^{rkp} - x_{t-1,n}^{rkp} + 2 \cdot h_{tn}^{rkp},\ \forall r \in R, t \in T \backslash \{1\}, k \in K_r, n \in N, p \in \boldsymbol{\mathcal{P}}, \tag{36a}$$

$$h_{tn}^{rkp} \leq x_{tn}^{rkp},\ \forall r \in R, t \in T \backslash \{1\}, k \in K_r, n \in N, \tag{36b}$$

$$h_{tn}^{rkp} \leq x_{t-1,n}^{rkp},\ \forall r \in R, t \in T \backslash \{1\}, k \in K_r, n \in N, p \in \boldsymbol{\mathcal{P}}, \tag{36c}$$

$$h_{tn}^{rkp} \geq x_{tn}^{rkp} + x_{t-1,n}^{rkp} - 1, \forall r \in R, t \in T \backslash \{1\}, k \in K_r, n \in N, p \in \boldsymbol{\mathcal{P}}, \tag{36d}$$

$$\alpha_{tn}^{rkp} \leq x_{tn}^{rkp},\ \forall r \in R, t \in T, k \in K_r, n \in N, p \in \boldsymbol{\mathcal{P}}, \tag{36e}$$

$$\alpha_{tn}^{rkp} \leq e_t^{np},\ \forall r \in R, t \in T, k \in K_r, n \in N, p \in \boldsymbol{\mathcal{P}}, \tag{36f}$$

$$\alpha_{tn}^{rkp} \geq x_{tn}^{rkp} + e_t^{np} - 1,\ \forall r \in R, t \in T, k \in K_r, n \in N, p \in \boldsymbol{\mathcal{P}}, \tag{36g}$$

$$\theta_{tn}^{rkp} \leq \phi_{tn}^{rkp},\ \forall r \in R, t \in T \backslash \{1\}, k \in K_r, n \in N, p \in \boldsymbol{\mathcal{P}}, \tag{36h}$$

$$\theta_{tn}^{rkp} \leq 1 - \alpha_{tn}^{rkp},\ \forall r \in R, t \in T \backslash \{1\}, k \in K_r, n \in N, p \in \boldsymbol{\mathcal{P}}, \tag{36i}$$

$$\theta_{tn}^{rkp} \geq \phi_{tn}^{rkp} - \alpha_{tn}^{rkp},\ \forall r \in R, t \in T \backslash \{1\}, k \in K_r, n \in N, p \in \boldsymbol{\mathcal{P}}, \tag{36j}$$

$$\pi_{tn}^{rkp} \leq \theta_{tn}^{rkp},\ \forall r \in R, t \in T \backslash \{1\}, k \in K_r, n \in N, p \in \boldsymbol{\mathcal{P}}, \tag{36k}$$

$$\pi_{tn}^{rkp} \leq 1 - \alpha_{t-1,n}^{rkp},\ \forall r \in R, t \in T \backslash \{1\}, k \in K_r, n \in N, p \in \boldsymbol{\mathcal{P}}, \tag{36l}$$

$$\pi_{tn}^{rkp} \geq \theta_{tn}^{rkp} - \alpha_{t-1,n}^{rkp}, \forall r \in R, t \in T \backslash \{1\}, k \in K_r, n \in N, p \in \boldsymbol{\mathcal{P}}, \tag{36m}$$

$$w_t^{rkp} \leq \pi_{tn}^{rkp},\ \forall r \in R, t \in T \backslash \{1\}, k \in K_r, n \in N, p \in \boldsymbol{\mathcal{P}}, \tag{36n}$$

$$w_t^{rkp} \geq \sum_{n \in N} \pi_{tn}^{rkp} - |N| + 1,\ \forall r \in R, t \in T \backslash \{1\}, k \in K_r, p \in \boldsymbol{\mathcal{P}}, \tag{36o}$$

$$o_t^{rp} \leq w_t^{rkp},\ \forall r \in R, t \in T \backslash \{1\}, k \in K_r, p \in \boldsymbol{\mathcal{P}}, \tag{36p}$$

$$o_t^{rp} \geq \sum_{k \in K_r} w_t^{rkp} - |K_r| + 1,\ \forall r \in R, t \in T \backslash \{1\}, p \in \boldsymbol{\mathcal{P}}, \tag{36q}$$

$$o_1^{rp} = 0,\ \forall r \in R, p \in \boldsymbol{\mathcal{P}}, \tag{36r}$$

$$o_t^{rp} \in \{0,1\},\ \forall r \in R, t \in T, p \in \boldsymbol{\mathcal{P}}, \tag{36s}$$

$$y_j^{rp} \in \{0,1\},\ \forall r \in R, j \in T, p \in \boldsymbol{\mathcal{P}}, \tag{36t}$$

$$w_t^{rkp} \in \{0,1\},\ \forall r \in R, k \in K_r, t \in T \backslash \{1\}, p \in \boldsymbol{\mathcal{P}}, \tag{36u}$$

$$\phi_{tn}^{rkp}, h_{tn}^{rkp}, \theta_{tn}^{rkp}, \pi_{tn}^{rkp} \in \{0,1\},\ \forall r \in R, k \in K_r, t \in T \backslash \{1\}, n \in N, p \in \boldsymbol{\mathcal{P}}, \tag{36v}$$

$$\alpha_{tn}^{rkp}, x_{tn}^{rkp}, \in \{0,1\},\ \forall r \in R, k \in K_r, t \in T, n \in N, p \in \boldsymbol{\mathcal{P}}, \tag{36w}$$

$$t - p_n^{qp} + \epsilon_1 \leq \eta_{tn}^{qp} \cdot B,\ \forall t \in T, n \in N, q \in [1, |p_n|], p \in \boldsymbol{\mathcal{P}} \tag{37a}$$

$$t - p_n^{qp} + \epsilon_1 \geq \left(\eta_{tn}^{qp} - 1\right) \cdot B,\ \forall t \in T, n \in N, q \in [1, |p_n|], p \in \boldsymbol{\mathcal{P}} \tag{37b}$$

$$p_n^{qp} + \bar{f}_n + \Gamma_n^{\mathrm{F}} \cdot \hat{f}_n - t + \epsilon_1 \leq \rho_{tn}^{qp} \cdot B,\ \forall t \in T, n \in N, q \in [1, |p_n|], p \in \boldsymbol{\mathcal{P}} \tag{37c}$$

$$p_n^{qp} + \bar{f}_n + \Gamma_n^{\mathrm{F}} \cdot \hat{f}_n - t + \epsilon_1 \geq \left(\rho_{tn}^{qp} - 1\right) \cdot B,\ \forall t \in T, n \in N, q \in [1, |p_n|], p \in \boldsymbol{\mathcal{P}} \tag{37d}$$

$$\tau_{tn}^{qp} \leq \eta_{tn}^{qp},\ \forall t \in T, n \in N, q \in [1, |p_n|], p \in \boldsymbol{\mathcal{P}} \tag{37e}$$

$$\tau_{tn}^{qp} \leq \rho_{tn}^{qp},\ \forall t \in T, n \in N, q \in [1, |p_n|], p \in \boldsymbol{\mathcal{P}} \tag{37f}$$

$$\tau_{tn}^{qp} \geq \eta_{tn}^{qp} + \rho_{tn}^{qp} - 1,\ \forall t \in T, n \in N, q \in [1, |p_n|], p \in \boldsymbol{\mathcal{P}} \tag{37g}$$

$$e_t^{np}, \eta_{tn}^{qp}, \rho_{tn}^{qp}, \tau_{tn}^{qp} \in [0,1],\ \forall t \in T, n \in N, q \in [1, |p_n|], p \in \boldsymbol{\mathcal{P}} \tag{37h}$$

$$e_t^{np} \geq \frac{1}{|p_n|} \cdot \sum_{q \in [1, |p_n|]} \tau_{tn}^{qp},\ \forall t, \in T, n \in N, p \in \boldsymbol{\mathcal{P}}, \tag{38a}$$

$$e_t^{np} \leq \sum_{q \in [1, |p_n|]} \tau_{tn}^{qp},\ \forall t, \in T, n \in N, p \in \boldsymbol{\mathcal{P}}. \tag{38b}$$

From (32a) to (38b), a variable with the subscript p means the value of this variable under the uncertainty set $p \in \boldsymbol{\mathcal{P}}$.

REFERENCES

[1] 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.

[2] E. J. Halpern and E. C. Pignataro, "Service function chaining (SFC) architecture," IETF, RFC 2481, Oct. 2015.

[3] D. Zhao, D. Liao, G. Sun, and S. Xu, "Towards resource-efficient service function chain deployment in cloud-fog computing," *IEEE Access*, vol. 6, pp. 66754–66766, 2018.

[4] J. Fan, Z. Ye, C. Guan, X. Gao, K. Ren, and C. Qiao, "GREP: Guaranteeing reliability with enhanced protection in NFV," in *Proc. ACM SIGCOMM Workshop Hot Topics Middleboxes Netw. Funct. Virtualization*, Aug. 2015, pp. 13–18.

[5] A. Kawabata, B. C. Chatterjee, and E. Oki, "Participating-domain segmentation based delay-sensitive distributed server selection scheme," *IEEE Access*, vol. 7, pp. 20689–20697, 2019.

[6] A. Zamani and S. Sharifian, "A novel approach for service function chain (SFC) mapping with multiple SFC instances in a fog-to-cloud computing system," in *Proc. 4th Iran. Conf. Signal Process. Intell. Syst. (ICSPIS)*, Tehran, Iran, Dec. 2018, pp. 48–52.

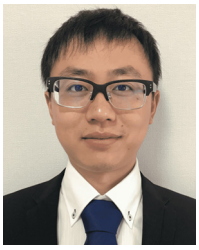
[7] F. He, T. Sato, and E. Oki, "Backup resource allocation model for virtual networks with probabilistic protection against multiple facility node failures," in *Proc. 15th Int. Conf. Design Rel. Commun. Netw. (DRCN)*, Coimbra, Portugal, Mar. 2019, pp. 37–42.

[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)*, Coimbra, Portugal, 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.*, early access, Jul. 7, 2020, doi: 10.1109/TNSM.2020.3007712.
[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)*, Ghent, Belgium, May 2013, pp. 1154–1160.
[12] M. Raza, V. Samineni, and W. Robertson, "Physical and logical topology slicing through SDN," in *Proc. IEEE Can. Conf. Elect. Comput. Eng. (CCECE)*, Vancouver, BC, Canada, May 2016, pp. 1–4.
[13] M. Johnston, H.-W. Lee, and E. Modiano, "A robust optimization approach to backup network design with random failures," *IEEE/ACM Trans. Netw.*, vol. 23, no. 4, pp. 1216–1228, Aug. 2015.
[14] M. Ito, F. He, and E. Oki, "Robust optimization model for probabilistic protection under uncertain virtual machine capacity in cloud," in *Proc. 16th Int. Conf. Design Rel. Commun. Netw. (DRCN)*, Milan, Italy, Mar. 2020, pp. 1–8.
[15] J. F. G. Fernández and A. C. Márquez, *Maintenance Management in Network Utilities: Framework and Practical Implementation*. London, U.K.: Springer, 2012.
[16] H. Maleki and Y. Yang, "An uncertain programming model for preventive maintenance scheduling," *Grey Syst. Theory Appl.*, vol. 7, no. 1, pp. 111–122, 2017.
[17] W. Luo, T. C. E. Cheng, and M. Ji, "Single-machine scheduling with a variable maintenance activity," *Comput. Ind. Eng.*, vol. 79, pp. 168–174, Jan. 2015.
[18] D. Xu, L. Wan, A. Liu, and D.-L. Yang, "Single machine total completion time scheduling problem with workload-dependent maintenance duration," *Omega*, vol. 52, pp. 101–106, Apr. 2015.
[19] 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)*, San Diego, CA, USA, 2015, pp. 1–7.
[20] 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.*, Budapest, Hungary, Apr. 2020, pp. 1–2.
[21] The Kubernetes Authors. *Kubernetes*. Accessed: Nov. 10, 2020. [Online]. Available: https://kubernetes.io/
[22] The Kubernetes Authors. *Scheduling Framework*. Accessed: Nov. 8, 2020. [Online]. Available: https://kubernetes.io/docs/concepts/scheduling-eviction/scheduling-framework/
[23] 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.*, Taipei, Taiwan, Apr. 2018, pp. 1–8.
[24] M. Mitchell, *An Introduction to Genetic Algorithms*. Cambridge, MA, USA: MIT Press, 1998.
[25] J. R. Cheng and M. Gen, "Accelerating genetic algorithms with GPU computing: A selective overview," *Comput. Ind. Eng.*, vol. 128, pp. 514–525, Feb. 2019.
[26] *Introducing IBM ILOG CPLEX Optimization Studio V12.9.0.0*, IBM Knowl. Center, Armonk, NY, USA. Accessed: May 8, 2020. [Online]. Available: https://www.ibm.com/support/knowledgecenter/SSSA5P_12.9.0/ilog.odms.studio.help/Optimization_Studio/topics/COS_home.html
[27] 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)*, Piscataway, NJ, USA, Nov. 2020, pp. 1–6.
[28] A. L. Soyster, "Convex programming with set-inclusive constraints and applications to inexact linear programming," *Oper. Res.*, vol. 21, no. 5, pp. 1154–1157, Sep./Oct. 1973.
[29] A. Ben-Tal, L. El Ghaoui, and A. Nemirovski, *Robust Optimization*, vol. 28. Princeton, NJ, USA: Princeton Univ. Press, 2009.
[30] D. Bertsimas and M. Sim, "The price of robustness," *Oper. Res.*, vol. 52, no. 1, pp. 35–53, Feb. 2004.
[31] D. Bertsimas and M. Sim, "Robust discrete optimization and network flows," *Math. Program.*, vol. 98, nos. 1–3, pp. 49–71, May 2003.
[32] F. He, T. Sato, B. C. Chatterjee, T. Kurimoto, S. Urushidani, and E. Oki, "Robust optimization model for backup resource allocation in cloud provider," in *Proc. IEEE Int. Conf. Commun. (ICC)*, Kansas City, MO, USA, May 2018, pp. 1–6.
[33] 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.
[34] 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. 16, no. 1, pp. 98–112, Mar. 2019.
[35] 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)*, Coimbra, Portugal, Nov. 2019, 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 has been a Research Fellow with the Japan Society for the Promotion of Science since 2021. 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 the Polytechnic Institute of New York University, Brooklyn. His research interests include routing, switching, protocols, optimization, and traffic engineering in communication and information networks.