Delay-Sensitive and Availability-Aware Virtual Network Function Scheduling for NFV

Song Yang, *Member, IEEE*, Fan Li, *Member, IEEE*,
Ramin Yahyapour, and Xiaoming Fu, *Senior Member, IEEE*

Abstract—Network Function Virtualization (NFV) has been emerging as an appealing solution that transforms from dedicated hardware implementations to software instances running in a virtualized environment. In NFV, the requested service is implemented by a sequence of Virtual Network Functions (VNF) that can run on generic servers by leveraging the virtualization technology. These VNFs are pitched with a predefined order, and it is also known as the Service Function Chaining (SFC). Considering that the delay and resiliency are two important Service Level Agreements (SLA) in a NFV service, in this paper, we first investigate how to quantitatively model the traversing delay of a flow in both totally ordered and partially ordered SFCs. Subsequently, we study how to calculate the VNF placement availability mathematically for both unprotected and protected SFCs. After that, we study the delay-sensitive Virtual Network Function placement and routing problem with and without resiliency concerns. We prove that this problem is NP-hard under two cases. We subsequently propose an exact Integer Nonlinear Programming (INLP) formulation and an efficient heuristic for this problem in each case. Finally, we evaluate the proposed algorithms in terms of acceptance ratio, average number of used nodes and total running time via extensive simulations.

Index Terms—Network function virtualization, service function chaining, delay, resiliency, availability, placement, routing

1 Introduction

In the traditional network services provisioning paradigm, network functions (e.g., firewall or load balancer) which are also called middleboxes are usually implemented by the dedicated hardware appliances. Deploying hardware middleboxes is costly due to their high cost for design and production and also these middleboxes need to be configured and managed manually, which further increases the costs of service providers. Network Function Virtualization (NFV) which is first proposed by ETSI [1] has been emerged as an appealing solution, since it enables to replace dedicated hardware implementations with software instances running in a virtualized environment. In NFV, the requested service is implemented by a sequence of Virtual Network Functions (VNF) that can run on generic servers by leveraging the virtualization technology. Service Function Chaining (SFC) is therefore defined as a chain-ordered set of placed VNFs that handles the traffic of the delivery and control of a specific application [2]. NFV allows to allocate network resources in a more scalable and elastic manner, offer a more efficient and agile management and operation mechanism for network functions and hence can largely reduce the overall costs.

In a SFC, the end-to-end delay [3] of the packets which traverse each VNF in this chaining is an important Service Level Agreement (SLA) to measure the performance of NFV. To guarantee a reliable network service, service providers need to promise a delay-sensitive performance and service to the customers. In this sense, the service providers may get a revenue loss if the promised total delay is violated. This deals with how to place the required ordered VNFs in the network and how to route the packets among these VNFs.

Resiliency [4] is another important SLA of NFV that defines the level the provided service can survive in the face of failures. Since, once a node or a link in a SFC fails, the whole SFC cannot operate and hence the provided service has to be stopped. To tackle this concern, we need to provide redundant SFCs to protect the primary SFC, although this comes at the expense of more network resource consumption. As a result, how to provide a customer's satisfied resilient NFV service by using minimum costs remains to be an important problem for network providers to solve.

In this paper, we first study the Delay-Sensitive VNF Scheduling problem, which is to place all the requested VNFs according to their predefined order and route the traffic among these VNFs without exceeding the end-to-end delay such that the number of used nodes is minimized. On the basis of the DSVS problem, we study the Delay-sensitive and Availability-aware NFV Scheduling problem, which additionally takes the VNF placement availability constraint into account. Our key contributions are as follows:

- *S. Yang and F. Li are with the School of Computer Science and Technology, Beijing Institute of Technology, Beijing 100081, China. E-mail: {S.Yang, fli}@bit.edu.cn.*
- *R. Yahyapour is with Gesellschaft für wissenschaftliche Datenverarbeitung mbH Göttingen and Institute of Computer Science, University of Göttingen, Göttingen 37077, Germany. E-mail: Ramin.Yahyapour@gwdg.de.*
- *X. Fu is with Institute of Computer Science, University of Göttingen, Göttingen 37077, Germany. E-mail: Fu@cs.uni-goettingen.de.*

Manuscript received 8 May 2019; accepted 2 July 2019. Date of publication 9 July 2019; date of current version 4 Feb. 2022.
(Corresponding author: Song Yang.)
Digital Object Identifier no. 10.1109/TSC.2019.2927339

- We propose a model to quantitatively calculate the traversing delay for a flow in a SFC.
- We present a model to quantitatively measure the SFC availability for NFV resiliency.
- We propose an exact Integer Nonlinear Programming (INLP) formulation and an efficient heuristic for placing and routing the delay-sensitive virtual network functions with and without resiliency concerns.
- We validate and investigate the performance of the proposed algorithms via extensive simulations.

The outline of this paper is organized as follows: Section 2 presents the related work. Section 3 analyzes and defines the delay calculation model in both totally ordered and partially ordered SFCs, and Section 4 presents the VNF placement availability calculation model. Section 5 defines the Delay-Sensitive VNF Scheduling (DSVS) problem and the Delay-sensitive and Availability-aware VNF Scheduling (DAVS) problem and analyzes their complexities. In Section 6, we propose an exact INLP formulation and an efficient heuristic for each problem. Section 7 provides the simulation results and we conclude in Section 8.

2 Related Work

A comprehensive survey about NFV can be found in [5], [6], [7], [8]. [9] provides a survey about resource allocation in NFV.

2.1 Traffic and Cost-Aware VNF Placement and Routing

Eramo et al. [10] first target the VNF routing and placement problem with the goal of maximizing the amount of data that can be processed within the network at peak traffic. They subsequently study how to consolidate VNFs and shut down unused servers such that the total operation cost (energy consumption+revenue loss) is minimized. They propose an exact Integer Linear Programming (ILP) and an efficient heuristic to solve each problem. Li et al. [11] consider a fat tree data center topology, and study (1) service chaining consolidation, (2) pod assignment, and (3) machine assignment per pod problems in both offline and online situation. The delay concern is also considered in these 3 problems. Consequently, efficient heuristics are proposed to solve these problems. Ma et al. [12] study the VNF placement problem to minimize the maximum link load in three cases: (1) when there is no dependency between VNFs, (2) totally-ordered: there is a total dependency order on the VNF set, and (3) partially-ordered: there exists dependency among a subset of VNFs. Assuming that the path between any network node pair is precalculated, they propose a polynomial-time algorithm for case (1), and prove that cases (2) and (3) are NP-hard. To solve them, a dynamic programming and an efficient heuristic are proposed to solve (2) and (3), respectively. Cohen et al. [13] propose a near-optimal approximation algorithm to place VNF without considering network function dependency. Kuo et al. [14] relax/approximate the VNF placement and routing problem based on the intuition that placing VNFs on a shorter path consumes less link bandwidth, but might also reduce VM reuse opportunities; reusing more VMs might lead to a longer path, and so it consumes more link bandwidth. Feng et al. [15] propose an approximation algorithm to find the minimum cost solution for placing VNFs and steering traffic between them for a given set of requests. Pham et al. [16] propose a sampling-based Markov approximation algorithm to jointly minimize the operational and network traffic cost for placing VNFs. Zhang et al. [17] transform the VNF placement problem into min-cost flow problem and the devise an efficient heuristic to solve this problem with the gola of minimizing total costs. Guo et al. [18] jointly consider the VNF placement and routing problem in data centers. They propose a randomized approximation algorithm when the traffic matrix is known in advance and a competitive online algorithm when the future arriving traffic is not known. However, they assume that one configuration in data centers consists of one VNF placement and one routing path solution, and a (limited) set of configurations is given. Nevertheless, the delay is not taken into account in these papers.

2.2 Delay-Sensitive VNF Scheduling

Qu et al. [19] consider the VNF transmission and processing delays, and formulate the joint problem of VNF scheduling and traffic steering as a Mixed Integer Linear Program (MILP). However, they assume that the virtual link between two physical nodes can at most process one traffic flow at one time, irrespective of the flow size. Vizarreta et al. [20] solve the QoS VNF placement problem, where the end-to-end delay and service chaining availability are considered by proposing an ILP and a heuristic. However, their adopted delay model does not relate with packet size and also the VNF placement availability in the VNF protection case is not considered in their work. Zhang et al. [21] devise an Alternating Direction Method of Multipliers (ADMM)-based algorithm to solve the delay-aware VNF placement and routing problem. However, they do not consider the nodes' delay in their problem. Li et al. [22] address the latency-aware middlebox routing and placement problem by leveraging a packet queuing model. However, fixed link transmission delay is assumed in their model. Dwaraki and Wolf [23] devise a layered-graph based heuristic to find delay-aware routes when the requested VNFs have been placed on network nodes. Allybokus et al. [24] propose an exact ILP and a greedy heuristic to solve the VNF placement and routing problem for both full and partially ordered SFCs. However, their solutions only consider a simple path within a SFC. Chen et al. [25] propose a hidden Markov Chain-based heuristic to place VNFs which jointly minimize the cost and delay for a set of given flows. Sun et al. [26] propose a framework that enables (independent) network function to work in parallel, which largely improve NFV performance in terms of delay. Gouareb et al. [27] present several heuristics to solve the VNF placement and routing problem from holistic view by leveraging queueing theory in edge clouds. Nevertheless, different from above work, our proposed delay calculation model is related with the packet size of the flow. Moreover, we consider both node and link delay and the required route traversing an entire SFC in the proposed problems is not necessarily a simple path, which is more general.

2.3 NFV Resiliency

Hmaity et al. [28] study the VNF protection in three cases, namely, (1) end-to-end protection: a primary SFC and a backup SFC are completely (both node- and link-) disjoint;

(2) link-protection, and (3) node-protection. They propose an ILP for each case in order to place VNFs to minimize the number of used nodes. Beck et al. [29] propose a recursive heuristic for survivable VNF placement to guarantee an end-to-end VNF protection. Han et al. [30] explore resilient respects of the individual VNF in terms of fault management (e.g., failure detection and automated recovery), state management, etc. They also discuss existing solutions for these aspects. Herker et al. [31] formulate how to calculate the VNF placement availability in data center topology. They also present an efficient heuristic on how to place VNFs and their backups to satisfy the requested availability. Fan et al. [32] target how to compute one backup SFC when the primary SFC is given such that the overall availability is satisfied. However, the considered availability model ignores the global information of the entire SFC, which is not precise enough. Ding et al. [33] formulate how to calculate VNF placement availability when at most one backup chaining is allowed. On basis of [32], given that the primary SFCs are already placed in the network, they [33] propose a heuristic on how to calculate the respective backup SFCs with minimum cost such that the total availability for each request is satisfied. Qu et al. [34] consider both delay and availability constraints to route and place SFCs. However, both node delay and link availability are not taken into account in their model. In addition, in their proposed VNF placement availability calculation model, the delay is not considered in each SFC when calculating the whole availability, which is not precise enough. Different from above work, our VNF placement availability model is general since it can precisely calculate the total availability of any $k \geq 1$ SFCs in both fully protected and partially protection scenario.

3 Delay Calculation for a Flow in a Service Function Chaining

We consider traversing delays under two cases for VNFs in a SFC, namely, (1) Totally Ordered: there is a total dependency order on the VNF set, and (2) Partially Ordered [26]: there exists dependency among a subset of VNFs. That is, $\exists$ at least two VNFs m and n, such that they have the same predecessor function and successor function but they have no dependencies with each other. For example, Fig. 1a is a totally ordered SFC. However, since function monitor only maintains the packets and does not change the packets, firewall and monitor can work in parallel as shown in Fig. 1b. After that, firewall and monitor functions send their individual processed packets to load balancer. As a result, load balancer only needs to select the packets from firewall and process them. In this way, as we will show later, the delay in Fig. 1b is less than the one in Fig. 1a. In the following, we will quantitatively show how to calculate the traversing delay for a flow in both totally ordered and partially ordered chainings.

3.1 Totally Ordered SFC

A network is represented by $\mathcal{G}(\mathcal{N}, \mathcal{L})$, where $\mathcal{N}$ denotes a set of N nodes and $\mathcal{L}$ stands for a set of L links. Each node $n \in \mathcal{N}$ has a maximum processing capability of $Cap(n)$ and each link $l \in \mathcal{L}$ is associated with bandwidth $b(l)$. A request is represented by $r(v, e, F, D)$, where v denotes the traffic volume (or total packet size) that needs to be delivered

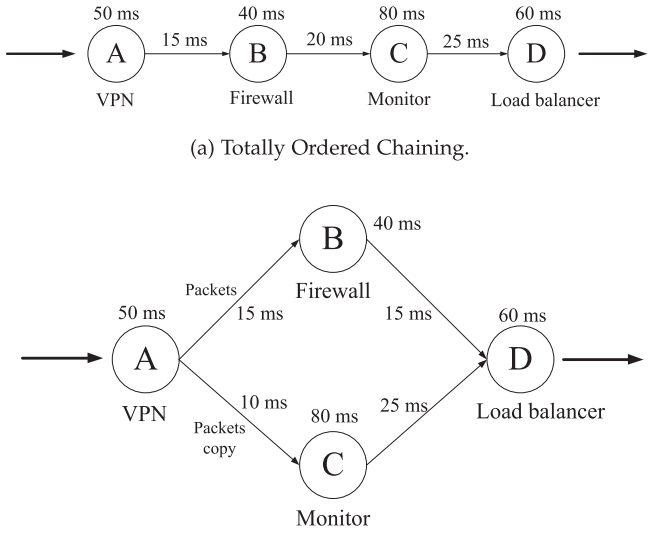


Fig. 1. Totally ordered chaining versus partially ordered chaining, derived from [26].

through the SFC, e means the requested network transmitting rate, F indicates the set of VNFs with orders, and D stands for the total requested end-to-end SFC delay. The VNF $f \in F$[1] (with required processing capacity $\eta(f)$) on node $n \in \mathcal{N}$ requires time of p_n^f to process it. The delay on a link l for a flow with total size of v can be calculated as:[2]

$$\frac{v}{e} \cdot \theta(l), \tag{1}$$

where $\theta(l)$ is a link attenuation parameter value and reflects the factors of the link that impact the transmitting delay such as link distance, signal impairmentness, etc. We assume that a SFC contains a set of $|F|$ ordered VNFs, f_1, f_2,..., $f_{|F|}$ which need to be placed in the network. Suppose there is a total dependency order on $|F|$ VNFs, then the traversing delay in this SFC is calculated as follows:

$$\sum_{f \in F, n \in \mathcal{N}} p_n^f + \sum_{l \in \mathcal{L}^s} T(l), \tag{2}$$

where p_n^f represents the processing delay for function f on node n, $\mathcal{L}^s$ denotes the link set that connects the placed VNF, and $T(l)$ indicates the flow delivering delay on link l. For example, in Fig. 1a where the flow delivering delay and node processing delay are associated, the traversing delay is $\underbrace{(50+40+80+60)}_{\text{total node delay}} + \underbrace{(15+20+25)}_{\text{total link delay}} = 290$ ms.

3.2 Partially Ordered SFC

Suppose at least two VNFs are not dependent with each other, and both of them need to receive the packets from the same

1. In this paper, F is the set of required ordered VNFs of request, and SFC is defined as a chain-ordered set of placed VNFs that handles the traffic of the delivery and control of a specific application. In this sense, F means the VNFs are to be deployed, and SFC represents the VNFs that have already been placed for traffic delivery and packet processing in network.

2. In this paper, we assume a proportional-to-flow-size link delay model, but our work can also be applied to the fixed link delay scenario.

prior VNF f_r and need to send packets to the same VNF f_s. We first partition the SFC into several segments and thus there is one or more independent VNFs in each segment. For example, there are 3 segments in Fig. 1b: Segment 1 contains the function VPN, segment 2 is composed of functions firewall and monitor, and segment 3 has the function load balancer. As a result, the traversing delay in a partially ordered SFC is equal to the longest path delay from the node which hosts one of the VNFs in the first segment to the node which contains one of the VNFs in the last segment. For example, the traversing delay of SFC in Fig. 1b is equal to $\underbrace{(50+80+60)}_{\text{total node delay}} + \underbrace{(10+25)}_{\text{total link delay}} = 225$ ms, which is less than the one in Fig. 1a. This example also illustrates that the partially ordered SFC can reduce delay compared to the totally ordered SFC. We notice that the totally ordered SFC is a special case of the partially ordered SFC, since when each segment contains only one VNF, the partially ordered SFC becomes the totally ordered SFC. Without loss of generality, we assume a (totally ordered or partially ordered) SFC consists of a set of $|M|$ segments M, and each segment $m \in M$ has $|m|$ functions. We use m_i and m_j to represent two consecutive segments, where $j = i + 1$ and $1 \leq i \leq |M| - 1$. In order to calculate the traversing delay in a partially ordered SFC with segments M, we first make the definition of *totally ordered sub-SFC* as follows:

Definition 1 Totally ordered sub-SFC. *Suppose there is a partially ordered SFC $\mathcal{C}$ with segment set M. A totally ordered sub-SFC is a chaining consisting of exactly one VNF in each segment.*

According the above definition, there are in total $\prod_{i=1}^{|M|} |m_i|$ totally ordered sub-SFCs. For example in Fig. 1b, there are $1 \cdot 2 \cdot 1 = 2$ totally ordered sub-SFCs, namely $A \rightarrow B \rightarrow D$ and $A \rightarrow C \rightarrow D$. As a result, the delay value of a partially ordered SFC is the maximum delay value among all the totally ordered sub-SFCs composing it.

4 VNF Placement Availability Calculation

The availability of a system is the fraction of time that the system is operational during the entire service time. The availability A_j of a network component j can be calculated as [35]:

$$A_j = \frac{MTTF}{MTTF + MTTR}, \tag{3}$$

where $MTTF$ represents Mean Time To Failure and $MTTR$ denotes Mean Time To Repair. In this paper, a node in the network represents a server, and a link denotes a physical link connecting two physical servers. Their availabilities are equal to the product of the availabilities of all their components (e.g., hard disk and memory for a node, amplifiers and fibers for a link). For a server (or any other equipment) n, let A_n and B_n denote its availability value and failure probability, respectively, then we have $A_n = 1 - B_n$. In reality, we can obtain the equipment's availability value by accessing the detailed logs recording every hardware component repair/failure incident during its lifetime. The details for characterizing e.g., server node failures and statistically calculating node availability can be found in [36], [37] and papers therein.

We consider a general multiple node and link failure scenario. That is, multiple nodes and links may fail at one certain time point. We distinguish and analyze the VNF placement availability under two different cases, namely (1) Unprotected SFC: only one SFC is allowed to be placed in the network, and (2) Protected SFC: at most $k \geq 2$ SFCs [28] can be placed in the network.

Suppose an unprotected SFC places VNFs on w nodes n_1, n_2, ..., n_w and traverses m links l_1, l_2, ..., l_m, its availability is calculated as:

$$\prod_{i=1}^{w} A_{n_i} \cdot \prod_{j=1}^{m} A_{l_j}, \tag{4}$$

where $\prod_{i=1}^{w} A_{n_i}$ denotes the availability of all the used nodes and $\prod_{j=1}^{m} A_{l_j}$ indicates the availability of all the traversed links.[3]

In the protected SFC, we regard that it is composed of (maximum) k unprotected SFCs. For clarification, we further term each of the k unprotected SFC in the protected placement as placement group Φ_i. We regard Φ_1 as the primary placement group. Since different placement groups may place one or more VNFs on the same node and/or traverse the same link, we distinguish the protected placement as two cases, namely (1) *fully protected SFC*, each one of k placement groups places VNF on different nodes and traverses different links and (2) *partially protected SFC*, at least two placement groups place one or more VNFs on the same node and/or traverse the same link. In the fully protected placement case, the availability can be calculated as:

$$A_{FCH}^{k} = 1 - \prod_{i=1}^{k} (1 - A_{\Phi_i}). \tag{5}$$

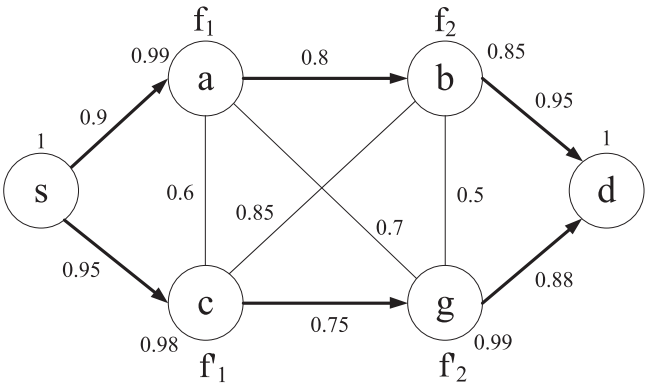
Eq. (5) indicates that the availability of k SFCs is equal to the probability that at least one SFC can work normally (does not fail). For example, in Fig. 2a where the node and link availabilities are associated, SFC s-a-b-d and s-c-g-d are fully protected, since they do not contain any same link or node. Although s and d are overlapped by these two SFCs, these two nodes are the ingress and egress nodes in the SFC and we do not consider their availabilities in this paper or we assume their availabilities are always 1. As a result, the total availability of these two SFCs are: $1 - (1 - 0.9 \cdot 0.99 \cdot 0.8 \cdot 0.85 \cdot 0.95) \cdot (1 - 0.95 \cdot 0.98 \cdot 0.75 \cdot 0.99 \cdot 0.88) \approx 0.8338$.

Next, we consider a general scenario where at least two among the k placement groups place the same VNF on the same node[4] or traverse the same links. In this case, Eq. (5) cannot be used to calculate the availability in this scenario since the availabilities of overlapped nodes or links will be counted more than once. To amend this, we use a new operator $\circ$.[5] Suppose there are m different nodes $n_1, n_2, \ldots, n_m$ with availabilities $A_{n_1}, A_{n_2}, \ldots, A_{n_m}$. For a node n_x with availability A_{n_x}, $\circ$ can be defined as follows:

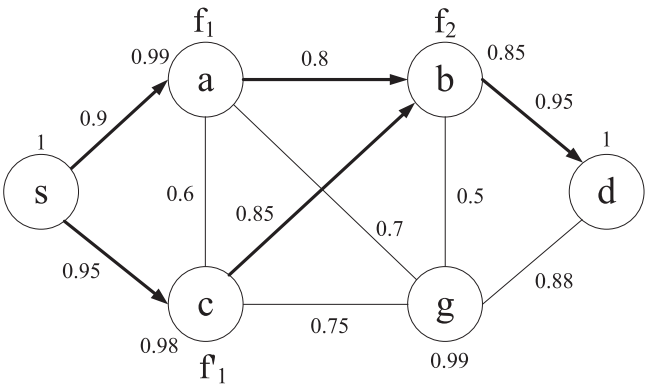
3. It is possible that one SFC traverses one link multiple times, but we consider the availability of each of the traversed links only once.

4. In this scenario, one VNF may serve for more than one placement groups. In this paper we assume each VNF placed on some node can have sufficient capabililty for servicing at most k placement groups [28].

5. As in [38] for the partially link-disjoint paths connection availability and in [39] for the virtual machine partially protection.



(a) Fully Protected SFC.



(b) Partially Protected SFC.

Fig. 2. SFC protection.

$$A_{n_1} \cdot A_{n_2} \cdot \cdots \cdot A_{n_m} \circ A_{n_x} = \begin{cases} \prod_{i=1}^{m} A_{n_i} & \text{if} \quad \exists n_i = n_x \\ \prod_{i=1}^{m} A_{n_i} \cdot A_{n_x} & \text{otherwise,} \end{cases} \tag{6}$$

where the $\circ$ computations for link availabilities can be defined analogously.

Let $\coprod$ denote consecutive $\circ$ operations of the different sets, then the availability (represented by A^k_{PCH}) of k SFCs can now be represented as:

$$\begin{aligned} A^k_{PCH} &= 1 - \coprod_{i=1}^{k} (1 - A_{\Phi_i}) \\ &= 1 - (1 - A_{\Phi_1}) \circ (1 - A_{\Phi_2}) \circ \cdots \circ (1 - A_{\Phi_k}) \\ &= \sum_{i=1}^{k} A_{\Phi_i} - \sum_{0 < i < j \leq k} A_{\Phi_i} \circ A_{\Phi_j} \\ &\quad + \sum_{0 < i < j < u \leq H} A_{\Phi_i} \circ A_{\Phi_j} \circ A_{\Phi_u} + \cdots + (-1)^{H-1} \coprod_{i=1}^{k} A_{\Phi_i}, \end{aligned} \tag{7}$$

where A_{Φ_i} denotes the availability of placement group Φ_i and can be calculated from Eq. (4). Let us take Fig. 2b as an example. SFCs s-a-b-d and s-c-b-d are partially protected, since they jointly place VNF f_2 on node b and they traverse the same link (b, d) as well. Consequently, the total availability is calculated as $1 - (1 - A_{sa} \circ A_a \circ A_{ab} \circ A_b \circ A_{bd}) \circ (1 - A_{sc} \circ A_c \circ A_{cb} \circ A_b \circ A_{bd}) = A_a \circ A_b \circ A_d \circ A_{sa} \circ A_{ab} \circ A_{bd} + A_c \circ A_b \circ A_d \circ A_{sc} \circ A_{cb} \circ A_{bd} - A_a \circ \underline{A_b} \circ A_d \circ A_{sa} \circ A_{ab} \circ \underline{A_{bd}} \circ A_c \circ \underline{A_b} \circ A_d \circ A_{sc} \circ A_{cb} \circ \underline{A_{bd}} = A_a \cdot A_b \cdot A_d \cdot A_{sa} \cdot A_{ab} \cdot A_{bd} + A_c \cdot A_b \cdot A_d \cdot A_{sc} \cdot A_{cb} \cdot A_{bd} - A_a \cdot \underline{A_b} \cdot A_d \cdot A_{sa} \cdot A_{ab} \cdot \underline{A_{bd}} \cdot A_c \cdot A_d \cdot A_{sc} \cdot A_{cb} \approx 0.759$.

5 Problem Definition and Complexity Analysis

5.1 Delay-Sensitive VNF Scheduling

Formally, the Delay-Sensitive VNF Scheduling problem is defined as follows:

Definition 2. *For a request $r(v, e, F, D)$, the Delay-Sensitive VNF Scheduling problem is to place all the requested VNFs according to their predefined order and route the traffic among these VNFs without violating the link bandwidth constraint and exceeding the end-to-end delay such that the number of used nodes is minimized.*

5.2 Delay-Sensitive and Availability-Aware VNF Scheduling

On basis of the DSVS problem, we assume that the node availabilities and link availabilities are associated in the given network $\mathcal{G}(\mathcal{N}, \mathcal{L})$. A request $r(v, e, F, D, \delta)$ indicates the same meaning with the defined request $r(v, e, F, D)$ in the DSVS problem, except that we additionally use δ to denote the requested availability value. Moreover, we assume each requested VNF can be placed at most k nodes in order to achieve a certain VNF placement availability, that is, each requested VNF has $k - 1$ copies (k is given as an input). Formally, the Delay-sensitive and Availability-aware VNF Scheduling problem is defined as follows:

Definition 3. *For a request $r(v, e, F, D, \delta)$, the Delay-sensitive and Availability-aware VNF Scheduling problem is to place at most k groups of the requested VNFs (k SFCs) according to their predefined order and route the traffic among these VNFs within each SFC, such that:*

- *The link bandwidth constraint is not violated.*
- *The end-to-end delay within each SFC is no greater than D.*
- *The VNF placement availability is no less than δ.*
- *The number of used nodes is minimized.*

By minimizing the number of used nodes, we want to save network cost consumption in terms of servers. Since the addressed problem and later proposed solutions are general, the other metrics such as number of used links can also be used as the objective of the problem. In that case, both the exact solution and proposed heuristic proposed in Section 6 only need to be slightly changed in order to solve the problem. Moreover, for simplicity, we do not consider the ingress and egress nodes in the SFC in both problems in this paper, since our focus is on finding a delay-sensitive SFC with and without resiliency concerns. We also do not consider the case when the traffic volume are changed during the packets are processed on different nodes [12], but our approach can be easily extended to this scenario.

5.3 Complexity Analysis

Remark 1. The DSVS problem and DAVS problem are NP-hard.

TABLE 1
Notations

Notation	Description
$\mathcal{G}(\mathcal{N},\mathcal{L})$	A network $\mathcal{G}$ with node set $\mathcal{N}$ and link set $\mathcal{L}$
$b(l),\theta(l),Cap(n)$	Bandwidth of link l, link attenuation parameter, maximum processing capacity of node n
$r(v,e,F,D)$	Request r, where v denotes the traffic volume that needs to be delivered through the SFC, e means the requested network transmitting rate, and F indicates the set of VNFs with orders in the DSVS problem.
$r(v,e,F,D,\delta)$	On basis of $r(v,e,F,D)$, δ indicates the requested VNF placement availability requirement in the DAVS problem.
k	Maximum times of a VNF can be placed on a network for the DAVS problem
$\mathbb{B}_r$	The set of totally ordered sub-SFCs of request r
$p_n^f,\eta(f)$	Processing delay if VNF f is placed on node n, required processing capacity of f
A_l,A_n	Availability of link l, node n.
M	A set of segments in a (partially ordered) SFC.
m_i,m_j	Two consecutive segments in a SFC.
$X_n^{f,h},Y_{f_i,f_j}^{l,h}$	INLP boolean variables, which identify whether f is placed on node n and whether the flows between f_i and f_j traverse link l by the hth placement group, respectively.

Proof. When only the totally sequential SFC is considered and the delay constraint is not imposed, Ghaznavi et al. [40] have proved that the DSVS problem is NP-hard. Since the DSVS problem is a special case of the DAVS problem (when $k=1$ and $\delta=0$), the DAVS problem is also NP-hard. The proof is therefore complete. □

Remark 2. When all the VNFs are placed on network nodes and there is sufficient link bandwidth, the DSVS problem is polynomial-time solvable.

Proof. When the link bandwidth constraint is not considered, for each two adjacent segments (m_i,m_j) and any two VNFs $f_i\in m_i$ which is hosted on node n_i and $f_j\in m_j$ which is located on node n_j, if $n_i\neq n_j$, we apply shortest path algorithm from n_i to n_j. By doing this, we find routes for every adjacent VNF pairs according to the orders in the SFC in the network. Therefore, the proof is complete. □

6 Solutions

6.1 Exact Solution

In this subsection, we propose an exact INLP formulation to solve the DSVS and DAVS problem. We first solve the DSVS problem where $k=1$ in the respective variables and start by some necessary notations and variables:

$r(v,e,F,D)$, $r(v,e,F,D,\delta)$: Request r for the DSVS problem and DAVS problem, respectively, and the details can be seen in Table 1.

$\mathbb{B}_r$: The set of totally ordered sub-SFCs of request r.

$p_n^{f,h}$: Processing delay if VNF f is placed on node n by the placement group h.

(m_i,m_j): Segment pairs that partition the whole SFC. Segment m_i contains F_i VNFs and Segment m_j contains F_j VNFs. VNFs pairs f_i and f_j which are dependent with each other in a SFC. That is, the packets which are processed by VNF f_i needs to traverse function f_j subsequently.

$X_n^{f,h}$: A boolean variable which returns 1 if the VNF f is placed on node $n\in\mathcal{N}$ by the hth placement group, and 0 otherwise.

$Y_{f_i,f_j}^{l,h}$: A boolean variable which returns 1 if the flows between VNFs f_i and f_j traverse link l by the hth placement group, and 0 otherwise.

Objective:

$$\min \sum_{n\in\mathcal{N}} \max_{f\in F,1\le h\le k} X_n^{f,h} \tag{8}$$

Constraints:
Routing constraints:

$$\begin{aligned}\sum_{(u,v)\in\mathcal{L}} Y_{f_i,f_j}^{(u,v),h}=1\ \ \forall 1\le h\le k,\\ (m_i,m_j):f_i\in m_i,f_j\in m_j,\\ u\in\mathcal{N}:X_u^{f_i,h}=1\&\&X_u^{f_j,h}\neq 1\end{aligned} \tag{9}$$

$$\begin{aligned}\sum_{(u,v)\in\mathcal{L}} Y_{f_i,f_j}^{(u,v),h}=1\ \ \forall 1\le h\le k,\\ (m_i,m_j):f_i\in m_i,f_j\in m_j,\\ v\in\mathcal{N}:X_v^{f_j,h}=1\&\&X_v^{f_i,h}\neq 1\end{aligned} \tag{10}$$

$$\begin{aligned}\sum_{(u,v)\in\mathcal{L}} Y_{f_i,f_j}^{(u,v),h}=\sum_{(v,w)\in\mathcal{L}} Y_{f_i,f_j}^{(v,w),h}\\ \forall 1\le h\le k,(m_i,m_j):f_i\in m_i,f_j\in m_j,\\ v\in\mathcal{N}:X_v^{f_i,h}\neq 1\&\&X_v^{f_j,h}\neq 1\end{aligned} \tag{11}$$

Placement constraint:

$$\sum_{n\in\mathcal{N},1\le h\le k} X_n^{f,h}\ge 1\ \ \forall f\in r.F \tag{12}$$

Link bandwidth constraint:

$$\sum_{(m_i,m_j),1\le h\le k} Y_{f_i,f_j}^{l,h}\cdot e\le b(l)\ \ \forall l\in\mathcal{L} \tag{13}$$

Node processing capacity constraint:

$$\sum_{f\in F,1\le h\le k} X_n^{f,h}\cdot\eta(f)\le Cap(n)\ \ \forall n\in\mathcal{N} \tag{14}$$

Traversing delay constraint:

$$\sum_{f\in B}\sum_{n\in\mathcal{N}} X_n^{f,h}\cdot p_n^f + \sum_{f_i,f_j\in B}\sum_{l\in\mathcal{L}}\left(\frac{v}{e}\cdot\theta(l)\cdot Y_{f_i,f_j}^{l,h}\right) \le D \quad \forall 1\le h\le k, B\in\mathbb{B}_r \tag{15}$$

Eq. (8) minimizes the number of used nodes. For instance, we first calculate the maximum value of $X_n^{f,h}$ for node n, and as long as $X_n^{f,h}=1$ for some $1\le h\le k$ and $f\in F$, it means that node n is in use to host VNF f. After that, we take the sum of $\max_{f\in F,1\le h\le k} X_n^{f,h}$ for all the nodes in $\mathcal{N}$ and try to minimize this value. Eqs. (9), (10), (11) are the multicommunity constraints that account for the routing from a source to a destination and ensures to find a path from u to v. More specially, Eq. (9) ensures that if f_i in the segment m_i is placed on node u and f_j is not placed on node u, then the sum of outgoing traffic from u is equal to 1. Eq. (10) ensures that if f_j in the segment m_j is placed on node v and f_i is not placed on node v, then the sum of incoming traffic to v is equal to 1. Eq. (11) ensures that if f_i and f_j are not placed on node v, then the sum of incoming traffic to v is equal to its outgoing traffic. Eq. (12) ensures that each required VNF f must be placed on at least one node in the network. Eq. (13) ensures that the total allocated bandwidth on each link does not exceed its maximum bandwidth. Eq. (14) ensures that the total assigned processing capacity on each node does not exceed its maximum processing capacity. Eq. (15) means that the total traversing delay of the SFC is no greater than the requested for both the totally (when $k=1$) and the partially ordered SFC (when $k>1$). More specifically, for each placement group h and each totally ordered sub-SFC B, $\sum_{f\in B,n\in\mathcal{N}} X_n^{f,h}\cdot p_n^f$ calculates the total processing delay, and $\sum_{f_i,f_j\in B}\sum_{l\in\mathcal{L}}(\frac{v}{e}\cdot\theta(l)\cdot Y_{f_i,f_j}^{l,h})$ computes the total path traversing delay. Consequently, by letting each totally ordered sub-SFC of each placement group be less than D, the total traversing delay requirement is satisfied.

When solving the DAVS problem, we keep Eqs. (8), (9), (10), (11), (12), (14), (13), (15) the same and add one more constraint in Eq. (16). Eq. (16) ensures that the availability of total k groups of requested VNFs requirement is obeyed, and it is derived based on inclusive-exclusive principle. For example, when $k=2$, Eq. (16) turns into Eq. (17). More specifically, in the first term of Eq. (17), $\prod_{n\in\mathcal{N}}\min_{f\in F}(1-X_n^{f,1}+X_n^{f,1}A_n)$ calculates the product of all the traversed nodes by group 1, and $\prod_{l\in\mathcal{L}}\min_{m_i,m_j}(1-Y_{f_i,f_j}^{l,1}+Y_{f_i,f_j}^{l,1}A_l)$ calculates the product of all the traversed links by group 1. The second term in Eq. (17) follows similarly. In the third term of Eq. (16), $\prod_{n\in\mathcal{N}}\min\big(\min_{f\in F}(1-X_n^{f,1}+X_n^{f,1}A_n),\min_{f\in F}(1-X_n^{f,2}+X_n^{f,2}A_n)\big)$ calculates the distinct product of all the traversed nodes by group 1 and 2, and $\prod_{l\in\mathcal{L}}\min\big(\min_{m_i,m_j}(1-Y_{f_i,f_j}^{l,1}+Y_{f_i,f_j}^{l,1}A_l),\min_{m_i,m_j}(1-Y_{f_i,f_j}^{l,2}+Y_{f_i,f_j}^{l,2}A_l)\big)$ calculates the distinct product of all the traversed links by group 1 and group 2. It is worthy to mention that our INLP can solve the DSVS and DAVS problem exactly. It means that it can return a solution with both VNF placement and how to route the traffic among those VNFs without violating the delay constraint (and the SFC placement availability) if any.

6.2 Heuristic

We emphasize that the route that packets traverse through a whole set of functions which are placed in the network is not necessarily a simple path. Especially with link bandwidth constraint, finding routes among VNFs is more difficult. Therefore, classical shortest path-based algorithms cannot apply well in the delay-sensitive VNF placement and routing problem. As we will show in Section 7, the shortest path based heuristic performs poorly due to nature of the DSVS and DAVS problem.

Our proposed heuristic, called Recursive VNF Scheduling Algorithm (RVSA), is presented in Algorithm 1. The general idea of VNF is that it iteratively places VNFs per-segment, when it cannot place VNFs of segment m because of node processing capacity or delay violating, it goes back the previous iteration and place the VNFs again. This procedure continues until when all the VNFs are placed or the total running time is exceeded. We first present the details of RVSA to solve the DSVS problem. More specifically, in Step 1 of Algorithm 1, T_x represents the "remaining" allowed delay value, $b(l)$ denotes the available bandwidth of link l, $\pi[f][n]$ is an indicator which records whether VNF f can be placed on node n,

$$\begin{aligned}
&\sum_{h=1}^{k}\left(\prod_{n\in\mathcal{N}}\min_{f\in F}(1-X_n^{f,h}+X_n^{f,h}A_n)\cdot\prod_{l\in\mathcal{L}}\min_{m_i,m_j}(1-Y_{f_i,f_j}^{l,h}+Y_{f_i,f_j}^{l,h}A_l)\right)\\
&-\sum_{1\le g<h\le k}\left(\prod_{n\in\mathcal{N}}\min\left(\min_{f\in F}(1-X_n^{f,g}+X_n^{f,g}A_n),\min_{f\in F}(1-X_n^{f,h}+X_n^{f,h}A_n)\right)\cdot\prod_{l\in\mathcal{L}}\min\left(\min_{m_i,m_j}(1-Y_{f_i,f_j}^{l,h}+Y_{f_i,f_j}^{l,h}A_l),\min_{m_i,m_j}(1-Y_{f_i,f_j}^{l,h}+Y_{f_i,f_j}^{l,h}A_l)\right)\right)\\
&+\cdots+(-1)^{k-1}\left(\prod_{n\in\mathcal{N}}\min_{1\le h\le k}(\min_{f\in F}(1-X_n^{f,g}+X_n^{f,g}A_n))\cdot\prod_{l\in\mathcal{L}}\min_{1\le h\le k}(\min_{m_i,m_j}(1-Y_{f_i,f_j}^{l,h}+Y_{f_i,f_j}^{l,h}A_l))\right)\ge\delta
\end{aligned} \tag{16}$$

$$\begin{aligned}
&\prod_{n\in\mathcal{N}}\min_{f\in F}(1-X_n^{f,1}+X_n^{f,1}A_n)\cdot\prod_{l\in\mathcal{L}}\min_{m_i,m_j}(1-Y_{f_i,f_j}^{l,1}+Y_{f_i,f_j}^{l,1}A_l)+\prod_{n\in\mathcal{N}}\min_{f\in F}(1-X_n^{f,2}+X_n^{f,2}A_n)\cdot\prod_{l\in\mathcal{L}}\min_{m_i,m_j}(1-Y_{f_i,f_j}^{l,2}+Y_{f_i,f_j}^{l,1}A_l)\\
&-\prod_{n\in\mathcal{N}}\min\left(\min_{f\in F}(1-X_n^{f,1}+X_n^{f,1}A_n),\min_{f\in F}(1-X_n^{f,2}+X_n^{f,2}A_n)\right)\cdot\prod_{l\in\mathcal{L}}\min\left(\min_{m_i,m_j}(1-Y_{f_i,f_j}^{l,1}+Y_{f_i,f_j}^{l,1}A_l),\min_{m_i,m_j}(1-Y_{f_i,f_j}^{l,2}+Y_{f_i,f_j}^{l,2}A_l)\right)
\end{aligned} \tag{17}$$

and $p[f][n]$ stores the solution found by the algorithm which means whether VNF f has been placed on node n. We initialize these variables as shown in Step 1. As long as $T_x \geq 0$ and the time limit is not reached, Steps 2-15 are going to find an VNF placement and routing solution. More specifically, for each segment $m \in M$ and for each VNF $f \in m$, we first try to place f on a node with sufficient available processing capacity and also the induced delay composed of processing delay and (maximum) traversing delay is less than T_x. If there exists such a node n_x, we let $p[f][n_x] = 1$ and $T_x \leftarrow T_x - p^f_{n_x} - \max(SP[n_x][N_y])$ in Step 7 and 8, where N_y denotes the set of nodes that host VNFs in the previous segment $prev[m]$ and $SP[n_x][N_y]$ represents the shortest paths between n_x and the nodes in N_y. In Step 9, we establish shortest paths between n_x and all the nodes in N_y. We subsequently let each of the traversed link's bandwidth be decreased by e in Step 10. If all the required VNFs are placed on network, it indicates that we find a feasible solution and return the result in Step 12. When Step 5 fails, it indicates that the algorithm cannot find a placement solution for placing VNFs belonging to Segment m. Therefore, we change the placement results in segment m by letting $p[f_v][n_v] \leftarrow 1$ and $\pi[f_v][n_v] \leftarrow false$ in Step 14. After that, we set $m \leftarrow prev[m]$ and the algorithm goes back to Step 2. In Step 16, when there is no feasible solution found by the algorithm, it outputs that there is no solution.

Algorithm 1. RVSA $(\mathcal{G}(\mathcal{N}, \mathcal{L}), M, r, H, TL)$

1 $T_x \leftarrow D, b(l) \leftarrow Cap(l), \pi[f][n] \leftarrow true, p[f][n] \leftarrow 0, \quad \forall l \in \mathcal{L}, \forall f \in F, n \in \mathcal{N}$;
2 **while** $T_x \geq 0$ & *Time limit TL is not reached* **do**
3 **for each** *segment* $m \in M$ **do**
4 **for each** VNF $f \in m$ **do**
5 Place f on node n_x such that $\pi[f][n_x] = true$, $T_x \geq p^f_{n_x} + \max(SP[n_x][N_y])$ and there are sufficient link bandwidth, where N_y denotes the set of nodes that host VNFs in the previous segment $prev[m]$.
6 **if** Step 5 succeeds **then**
7 $p[f][n_x] \leftarrow 1$;
8 $T_x \leftarrow T_x - p^f_{n_x} - \max(SP[n_x][N_y])$;
9 Establish shortest paths Ψ_u between n_x and all the nodes in N_y.
10 $b(l) \leftarrow b(l) - e, \forall$ traversed link $l \in \Psi_u$.
11 **if** All the VNFs are placed **then**
12 Return p and Ψ_u;
13: **else**
14 $p[f_v][n_v] \leftarrow 0, \pi[f_v][n_v] \leftarrow false$, $\forall f_v \in m$ which is originally placed by node n_v in the segment m
15 $m \leftarrow prev[m]$ and goes back Step 2.
16 Output there is no solution.

To solve the DAVS problem, we sequentially run the (slightly) modified RVSA algorithm at most k times. The only modification is that in Step 9, instead of running shortest path, RVSA applies the TAMCRA [41], which is a heuristic multi-constrained path selection algorithm we use to find restricted shortest path. In the restricted shortest path problem, each link is associated with two weights (say delay and cost), and the problem is to find a path from the source to the destination such that the the total delay is no greater than T and the total cost is minimized. The multi-constrained path selection problem is similar to the Restricted Shortest path problem, and it is assumed that there can be multiple additive link weights on each link. The goal is to find a path from a source to a destination such that the link weight sum in each dimension during the path is no greater than the specified. Since each link has link and availability weights in the DVAS problem, we employ TAMCRA to find a path with minimum delay and the product of link availabilities is less than a certain value κ. We let $\kappa = \delta$ and the number of stored paths for each node in TAMCRA is equal to 1 for simplicity. Consequently, we first run RVSA on the network and if it can return the solutions with satisfied delay constraint, we record the solution as P_1. In particular, in Step 5, we always select a node with the largest availability. After that, we decrease the link bandwidth of all the links that are traversed by P_1, and reduce the node's available processing capacity accordingly of all the nodes that are used to place VNFs. If P_1 cannot satisfy the VNF placement availability, we run RVSA again on the modified network and record the feasible solution as P_2 if any. This procedure continues until the VNF placement availability requirement is satisfied by the so-far found solutions or we run RVSA at most k times but there is no feasible solution for the DAVS problem. In all, the time complexity of RVSA to solve the DSVS problem and DAVS problem are $O(TL)$ and $O(TL \cdot k)$, respectively.

7 Simulations

7.1 Simulation Setup

We conduct simulations on two realistic carrier backbone networks [42], namely the NSFNet of 14 nodes and 21 links shown in Fig. 3 and the USANet of 24 nodes and 43 links displayed in Fig. 4. Each node has a processing capacity ranging from 10 to 15 units, and the node availability value is randomly distributed in the set of $\{0.999, 0.9995, 0.9999, 0.99999\}$. The link bandwidth is randomly generated from 30 to 50, and the link availability is randomly chosen from the set of $\{0.99, 0.999, 0.9999\}$. Moreover, the value of $\theta(l)$ falls in the range of $[20, 50]$. There are in total 20 different VNFs that can be requested in the service and the size of the VNF is randomly distributed between 5 and 10. The time for different node to process different VNF is randomly generated from 10 to 25 ms. For each request $r(v, e, F, D)$ in the DSVS problem and each request $r(v, e, F, D, \delta)$ in the DAVS problem which demand totally ordered SFCs, $v \in [1, 10]$, $e \in [50, 100]$, $|F| \in [5, 10]$ and $D \in [100, 200]$. The traffic request setting is similar to [43], and reflects the conventional application of e.g., Web Service or VoIP. Moreover, δ is randomly distributed in the set of $[0.95, 0.99, 0.995, 0.999, 0.9995, 0.9999]$, similar to [39]. The request which asks for partially ordered SFCs follow the same with the request generated for the totally ordered SFCs, but we let that each segment contains 2 VNFs and the number of segments ranges from 2 to 5. In the DSVS problem, we compare our proposed algorithms with two heuristics that are derived from literature:[6]

1) Traffic And Space Aware Routing (TASAR) [12]: It first selects one node to place as many VNFs as

6. Most of current relevant literature deal with different problems in NFV or adopt diffrent NFV models with us, so we have to modify two relevant proposed algorithms from [12], [34] in order to solve our proposed problems accordingly.

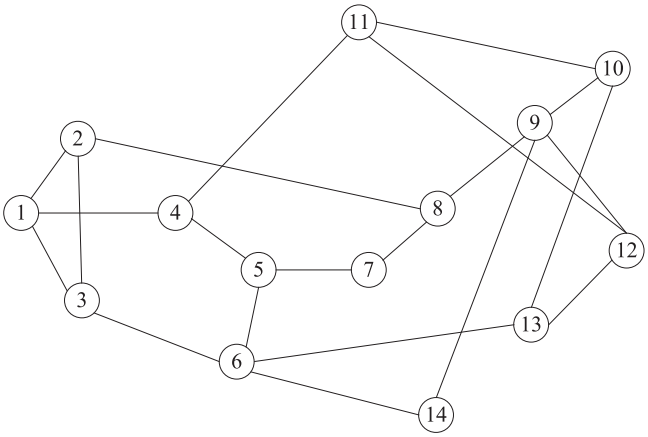


Fig. 3. NSFNet carrier backbone network.

possible, and then iteratively route to a nearby node with spaces until sufficient spaces have been accumulated for all the required VNFs.

2) Greedy Shortest Path (GSP) [34]: For each node pair (s,t) in the network, it first finds z-shortest paths from s to t. After that, it iteratively places VNFs according to the requested SFC one by one on the nodes along each of these z paths. It selects one feasible placement solution which uses the minimum number of nodes from those z paths for each (s,t) pair. Finally, it returns the "best" feasible placement solution with the minimum node usage among $\frac{N\cdot(N-1)}{2}$ different (s,t) pairs of solutions[7] if any. We set $z=10$ for GSP in the simulation.

In the DAVS problem where we set $k=2$, we first run TASAR or GSP in the network, if it returns a feasible solution with satisfied delay constraint (say P_1), we prune the nodes and links that are used by them by decreasing their node available processing capacity and link bandwidth, respectively. Subsequently, we apply TASAR or GSP again on the remainder of the network. The returned feasible solution in this time is denoted by P_2. Finally, if P_1 and P_2 satisfy the availability requirement, then we return the solution, otherwise we regard that TASAR or GSP cannot find solutions for the DAVS problem. We randomly and respectively generate $R=100$, 200, and 300 requests for both problems and for both totally ordered and partially ordered SFCs. In this sense, all the compared algorithms run R times[8] for corresponding R requests in each scenario in order to collect the simulation results. For example, for each generated request the INLP runs one time on $\mathcal{G}$ (the parameters in $\mathcal{G}$ like link bandwidth keep the same for each time of algorithm's running) and we evaluate the algorithm's performance over all the requests.

The simulations are run on a high-performance desktop PC with 3.40 GHz and 16 GB memory. We use IBM ILOG CPLEX 12.6 to implement the proposed INLP. All the heuristics are implemented by C# and compiled on Visual Studio 2015 (using NET Framework 4.5). In order that the INLP can return a feasible solution in a reasonable time, we set the time limit for the INLP to be 20 minutes for each request and the gap of the INLP is set to 0.02. Moreover, we let $TL=1$ second for RVSA.

7. Since the returned solutions by GSP for node pair (s,t) and (t,s) are the same, we only need to consider possible solutions from $\frac{N\cdot(N-1)}{2}$ node pairs.

8. In this paper, the proposed and compared algorithm take the input of single request to find corresponding VNF placement and traffic routing solution. However, we can also extend the proposed algorithms to simultaneously find solutions for multiple requests.

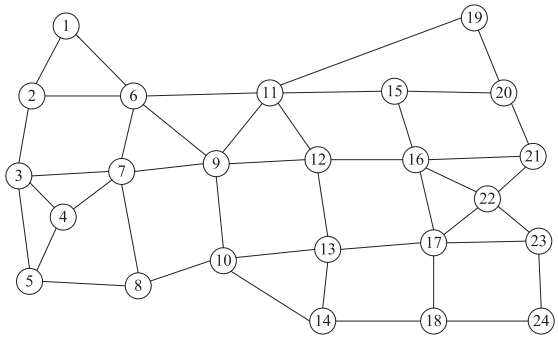


Fig. 4. USA carrier backbone network.

7.2 Simulation Results

7.2.1 Delay-Sensitive VNF Scheduling

We first evaluate the performance of the algorithms in terms of Acceptance Ratio (AR), which is defined as the number of accepted requests over all the requests (between 0 and 1). Here, the "accepted request" means that its SFC delay requirement is satisfied and each link's consumed bandwidth is not exceeded. Figs. 5a and 6a give the AR value returned by all the algorithms in NSFNet. More specifically, when totally ordered SFC is requested, in Fig. 5a, we find out that the INLP achieves the highest AR value, followed by our proposed heuristic RVSA with a slight difference. TASAR ranks the third, but its returned AR value is largely less than RVSA. This is because TASAR does not consider link delay factor when placing VNFs, which may violate the requested delay constraint and results in request blocking. GSP performs the worst. The reason is that although it first determines z-shortest paths which may have the least path delay, it ignores the VNF size and node processing capacity. This means that the VNF (s) cannot be placed on the node along the path whose maximum processing capacity is fewer than the VNF's. When partially ordered SFC is requested, the AR value achieved for all the algorithms except for GSP in this case (see Fig. 6a) is larger than the case for the totally ordered SFC (see Fig. 5a). This can be explained that there are at most 5 segments in the partially ordered SFC, so the total link delay value is the sum of at most 4 subpaths' delay (between 5 segment pairs) which is much less than the case in the totally ordered SFC with the sum of at most 9 subpaths' delay (between 10 segment pairs). Consequently, the total delay in the partially ordered SFC become less and hence becomes easier for the algorithms to find feasible solutions to satisfy the requested delay constraint. Nevertheless, INLP can accept almost all the requests, while RVSA and TASAR can achieve a close to optimal AR performance. GSP, on the other hand, still performs very poorly, due to the the reasons as we analyzed above. We found that the results in USANet (Figs. 7, 8) follow similarly with the ones in the NSFNet (Figs. 5, 6). The reason is that both NSFNet and USANet are well connected and their parameters such as node's maximum processing capacity are generated similarly. Hence, the routing aspect of variety in terms of topology does not affect the DSVS problem too much, since the solution is

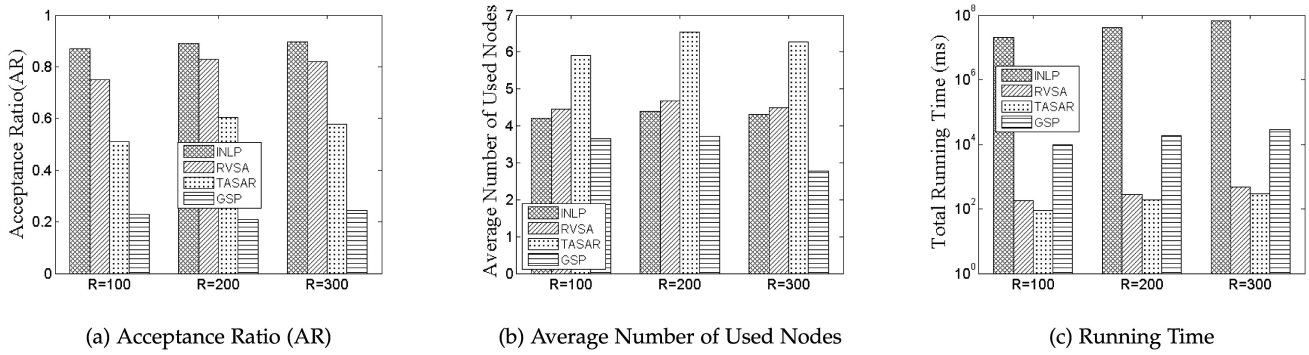


Fig. 5. Simulation results on NSFNet over different amount of requests with totally ordered SFC for the DSVS problem: (a) Acceptance Ratio (AR), (b) Average Number of Used Nodes (ANUN) and (c) Running Time.

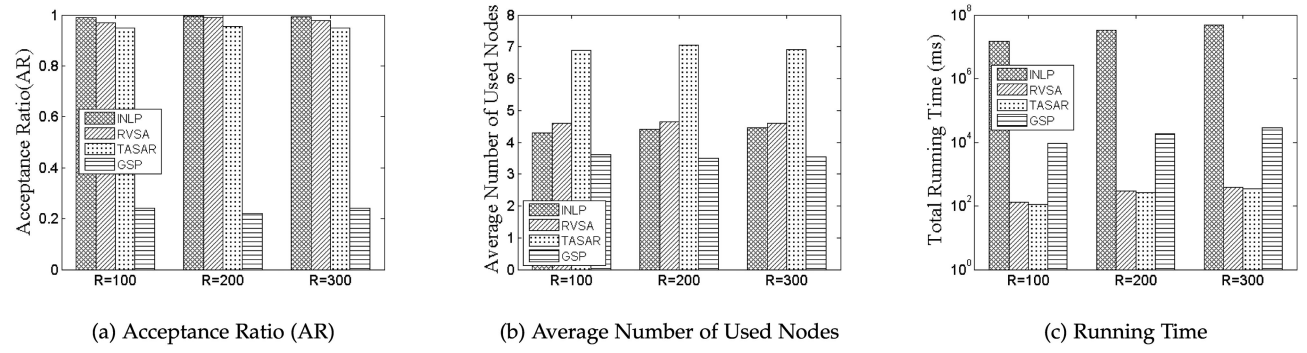


Fig. 6. Simulation results on NSFNet over different amount of requests with partially ordered SFC for the DSVS problem: (a) Acceptance Ratio (AR), (b) Average Number of Used Nodes and (c) Running Time.

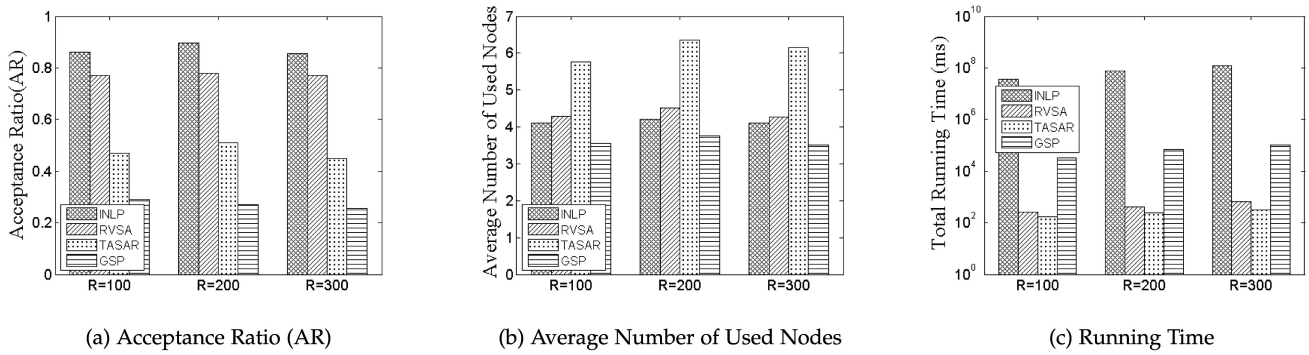
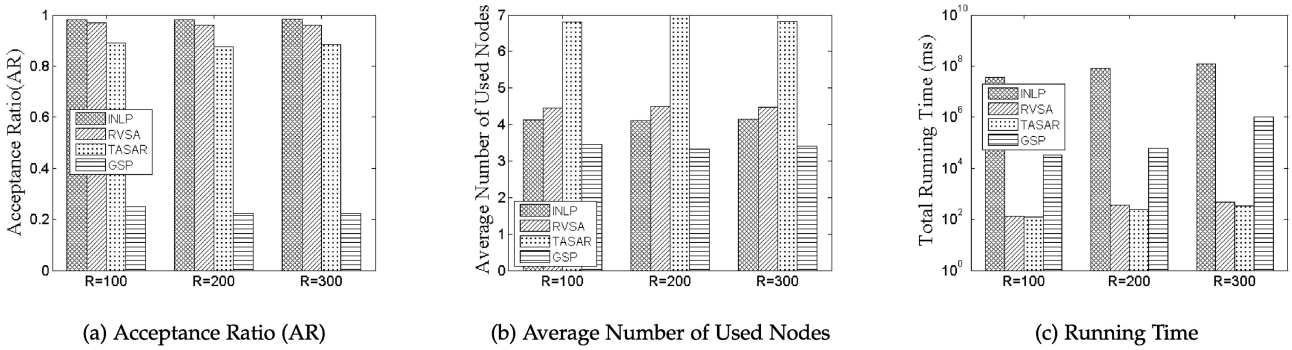


Fig. 7. Simulation results on USANet over different amount of requests with totally ordered SFC for the DSVS problem: (a) Acceptance Ratio (AR), (b) Average Number of Used Nodes and (c) Running Time.

dominated by how to place the VNFs in the network in this context. For simplicity, in the following we only demonstrate the results for the algorithms in NSFNet.

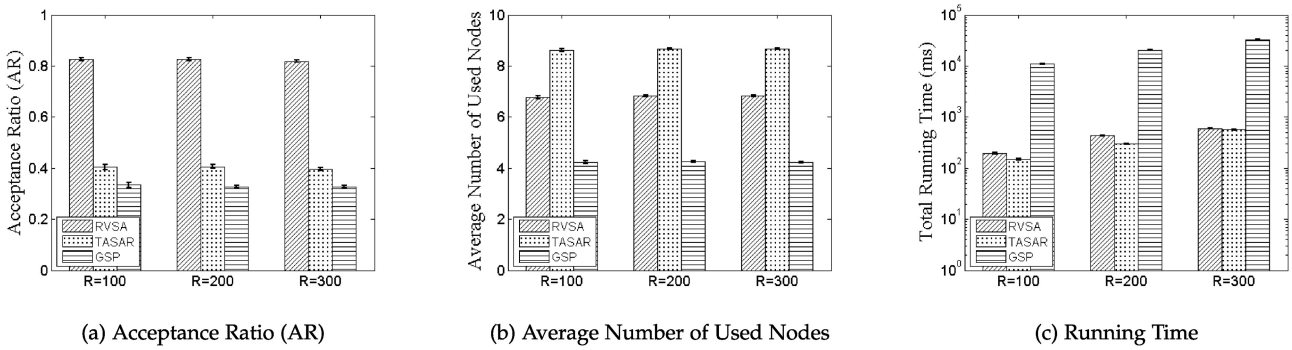
Next, we compare the algorithms in terms of Average Number of Used Nodes. The ANUN is defined as the total number of nodes consumed by the accepted requests divided by the number of accepted requests of the algorithm. In Figs. 5b and 6b, we see that the achieved ANUN value by GSP is the lowest. This is because its acceptance ratio in those scenarios is too low, and it only finds solutions for some "easier" requests. Except for those cases, the INLP achieves the minimum value of ANUN, and our proposed RVSA obtains the second lowest ANUN value. From above, we observe that even under the constrained simulation setup, the exact INLP can always accept most requests and consume the least amount of nodes as well, which validates its correctness.

Finally, Figs. 5c and 6c present the total running time for all the algorithms (in log scale). We found that the running time for the same algorithm increases as the number of traffic requests raises. The INLP is significantly more time-consuming than all the 3 heuristics. The GSP consumes more time than the other 2 heuristics, since it iterates $\frac{N\cdot(N-1)}{2}$ possible solutions (rounds) by finding $z = 10$ shortest paths in each solution. RVSA is only slightly higher than TASAR, but both of them are very quick. In addition, although we set $TL = 1$ second in RVSA, we found it returns a feasible solution within less than 1 second in most of the times. From above we see that, due to the nature that RVSA applies the recursive technique whose returned route is not necessarily a simple path, it can efficiently find feasible (close-to-optimal) solutions for the DSVS (and DAVS problem in below) in most of times at a reasonable short time.



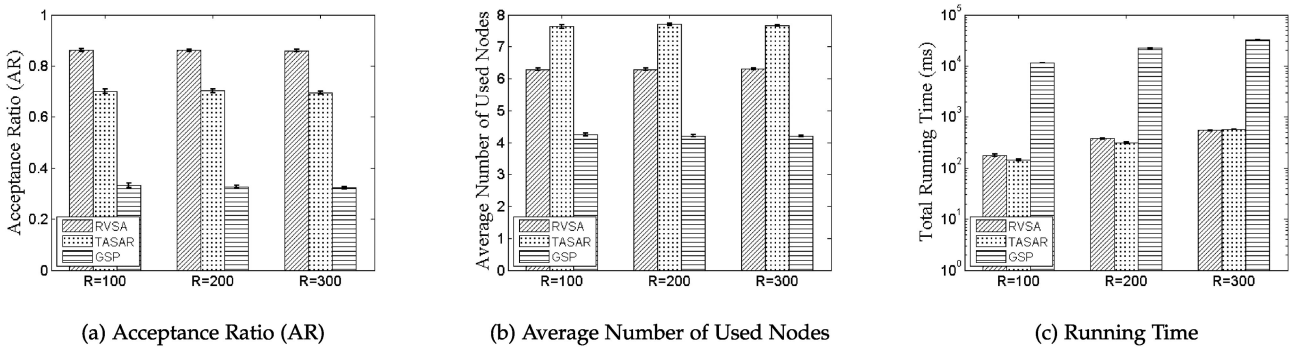
(a) Acceptance Ratio (AR) (b) Average Number of Used Nodes (c) Running Time

Fig. 8. Simulation results on USANet over different amount of requests with partially ordered SFC for the DSVS problem: (a) Acceptance Ratio (AR), (b) Average Number of Used Nodes and (c) Running Time.



(a) Acceptance Ratio (AR) (b) Average Number of Used Nodes (c) Running Time

Fig. 9. Simulation results on NSFNet over 100 sets of different amount of requests with totally ordered SFC for the DAVS problem: (a) Acceptance Ratio (AR), (b) Average Number of Used Nodes and (c) Running Time.



(a) Acceptance Ratio (AR) (b) Average Number of Used Nodes (c) Running Time

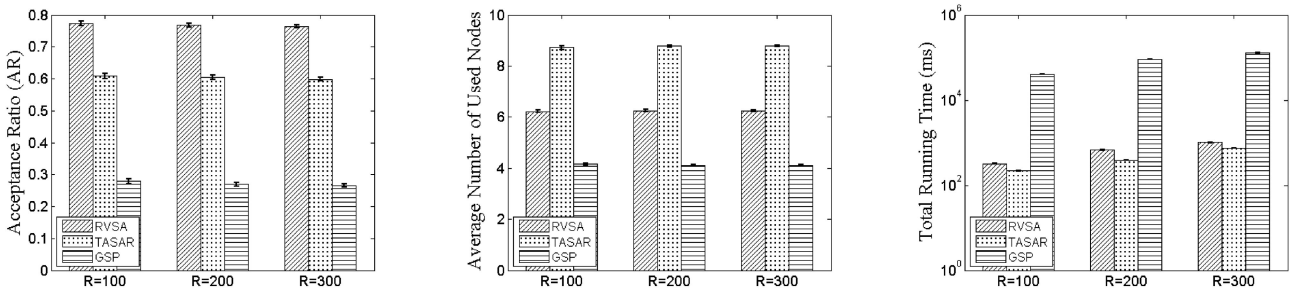
Fig. 10. Simulation results on NSFNet over 100 sets of different amount of requests with partially ordered SFC for the DAVS problem: (a) Acceptance Ratio (AR), (b) Average Number of Used Nodes and (c) Running Time.

7.2.2 Delay-Sensitive and Availability-Aware VNF Scheduling

In this subsection, we evaluate the performance of the algorithms to solve the DAVS problem. Due to the nonlinear constraint in Eq. (16) for the INLP, it becomes very time-consuming and keeps running for at least one day without returning a feasible solution. We therefore only evaluate the performance of heuristic algorithms. Due to the lack of the exact solution, we generate 100 sets of $R = 100, 200, 300$ traffic requests, respectively, and evaluate all the heuristic algorithms for those sets of traffic requests (100 runs). By doing this, we want to establish confidence on the performance of heuristics. Similar to Section 7.2.1, for all the algorithms, the results in the USANet (Figs. 11 and 12) follow similarly to the results in the NSFNet (Figs. 9 and 10), we therefore only describe and analyze the results in the NSFNet for all the heuristics for simplicity.

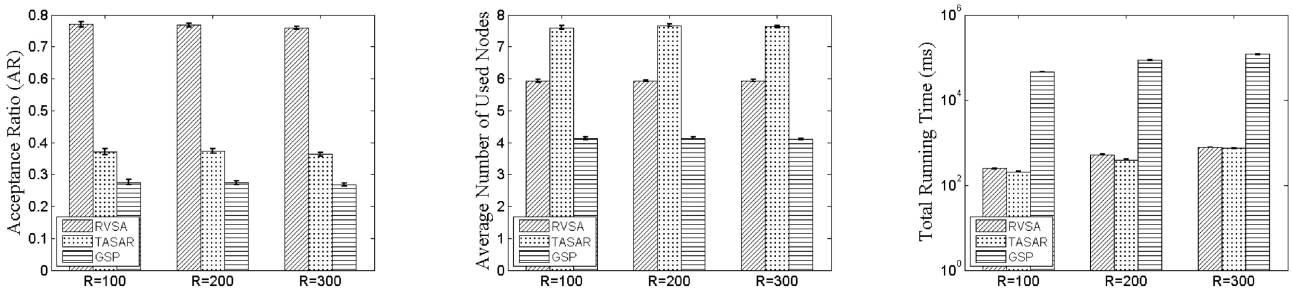
Figs. 9 and 10 depict the AR, ANUN and running time (in log scale) of all these algorithms, where the confidence interval is set to 95%. The 95% confidence interval is calculated for all the figures, but in those where it is not visible, the interval is negligibly small.[9] RVSA always achieves better performance than the other 2 heuristics in terms of AR (see Figs. 9a and 10a) and ANUN (see Figs. 9b and 10b). The reason can be explained that RVSA leverages TAMCRA [41] which takes both availability and delay factors into account when finding a path. It indicates that

9. We note here that some plots are log-scale that additionally contributes to the confidence interval visibility.



(a) Acceptance Ratio (AR) (b) Average Number of Used Nodes (c) Running Time

Fig. 11. Simulation results on USANet over 100 sets of different amount of requests with totally ordered SFC for the DAVS problem: (a) Acceptance Ratio (AR), (b) Average Number of Used Nodes and (c) Running Time.



(a) Acceptance Ratio (AR) (b) Average Number of Used Nodes (c) Running Time

Fig. 12. Simulation results on USANet over 100 sets of different amount of requests with partially ordered SFC for the DAVS problem: (a) Acceptance Ratio (AR), (b) Average Number of Used Nodes and (c) Running Time.

adopting TAMCAR together with its recursive nature can make RVSA to efficiently solve DAVS problem as well. Different from Section 7.2.1, for the same algorithm, the AR value in Fig. 10a is not always (much) higher the AR value in Fig. 9a. This is due to the fact that apart from the delay requirement (constraint), the VNF placement availability requirement (constraint) is also imposed in the DAVS problem. In this sense, the acceptance of a request depends on satisfying both two requirements. Moreover, Figs. 9c and 10c show that GSP is more time consuming than the other heuristics, and our proposed heuristic RVSA is very comparable with TASAR with much quicker running time.

In all, we conclude that the exact solution can always find optimal solution (if there exists), but this comes at the expense of much higher running time. Especially when to solve the DAVS problem, the exact solution cannot return a solution even within quite a long time. Our proposed RVSA, can return a feasible solution in most of times at a much quicker time, which can be used when the traffic requests arrive in an online fashion.

8 Conclusion

In this paper, we first investigate how to quantitatively model the traversing delay of a flow in a SFC and study how to calculate VNF placement availability mathematically for both the totally ordered SFC and partially ordered SFC. After that, we study the Delay-Sensitive VNF Scheduling problem and the Delay-Sensitive and Availability-aware VNF Scheduling problem. We have proved that both problems are NP-hard. To solve them, we propose an exact INLP and an efficient heuristic. Extensive simulations reveal that the exact INLP can always return feasible solutions if there exists, but its running time is significantly higher. On the other hand, our proposed heuristic RVSA, can achieve a close to optimal performance with a much quicker running time. In our future work, we plan to study the DAVS problem and the DSVS problem when traffic demands vary in different time intervals, and study how to consolidate SFCs to satisfy both delay and availability requirements.

Acknowledgments

The work of Song Yang is partially supported by the National Natural Science Foundation of China (NSFC, No. 61802018) and Beijing Institute of Technology Research Fund Program for Young Scholars. The work of Fan Li is partially supported by the NSFC (No. 61772077, 61370192), and the Beijing Natural Science Foundation (No. 4192051). The work of Xiaoming Fu is partially supported by EU FP7 CleanSky ITN (No. 607584) and H2020 RISE COSAFE (No. 824019) projects.

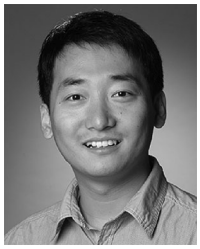
References

[1] "ETSI Publishes First Specifications for Network Functions Virtualisation." [Online]. Available: http://www.etsi.org/news-events/news/700–2013-10-etsi-publishes-first-n fv-specifications

[2] A. M. Medhat, T. Taleb, A. Elmangoush, G. A. Carella, S. Covaci, and T. Magedanz, "Service function chaining in next generation networks: State of the art and research challenges," *IEEE Commun. Mag.*, vol. 55, no. 2, pp. 216–223, Feb. 2017.

[3] D. C. Verma, "Service level agreements on IP networks," *Proc. IEEE*, vol. 92, no. 9, pp. 1382–1388, Sep. 2004.

[4] U. Franke, "Optimal IT service availability: Shorter outages, or fewer?" *IEEE Trans. Netw. Serv. Manage.*, vol. 9, no. 1, pp. 22–33, Mar. 2012.
[5] D. Bhamare, R. Jain, M. Samaka, and A. Erbad, "A survey on service function chaining," *J. Netw. Comput. Appl.*, vol. 75, no. 3, pp. 138–155, 2016.
[6] R. Mijumbi, J. Serrat, J.-L. Gorricho, N. Bouten, F. De Turck, and R. Boutaba, "Network function virtualization: State-of-the-art and research challenges," *IEEE Commun. Surv. Tuts.*, vol. 18, no. 1, pp. 236–262, Jan.-Mar. 2016.
[7] B. Yi, X. Wang, K. Li, M. Huang, et al., "A comprehensive survey of network function virtualization," *Comput. Netw.*, vol. 14, pp. 212–262, 2018.
[8] H. Hantouti, N. Benamar, T. Taleb, and A. Laghrissi, "Traffic steering for service function chaining," *IEEE Commun. Surv. Tuts.*, vol. 21, no. 1, pp. 487–507, Jan.-Mar. 2019.
[9] J. G. Herrera and J. F. Botero, "Resource allocation in NFV: A comprehensive survey," *IEEE Trans. Netw. Serv. Manage.*, vol. 13, no. 3, pp. 518–532, Sep. 2016.
[10] V. Eramo, E. Miucci, M. Ammar, and F. G. Lavacca, "An approach for service function chain routing and virtual function network instance migration in network function virtualization architectures," *IEEE/ACM Trans. Netw.*, vol. 25, no. 4, pp. 2008–205, Aug. 2017.
[11] Y. Li, L. T. X. Phan, and B. T. Loo, "Network functions virtualization with soft real-time guarantees," in *Proc. 35th Annu. IEEE Int. Conf. Comput. Commun.*, 2016, pp. 1–9.
[12] W. Ma, O. Sandoval, J. Beltran, D. Pan, and N. Pissinou, "Traffic aware placement of interdependent NFV middleboxes," in *Proc. IEEE Conf. Comput. Commun.*, 2017, pp. 1–9.
[13] R. Cohen, L. Lewin-Eytan, J. S. Naor, and D. Raz, "Near optimal placement of virtual network functions," in *Proc. IEEE Conf. Comput. Commun.*, 2015, pp. 1346–1354.
[14] T.-W. Kuo, B.-H. Liou, K. C.-J. Lin, and M.-J. Tsai, "Deploying chains of virtual network functions: On the relation between link and server usage," in *Proc. 35th Annu. IEEE Int. Conf. Comput. Commun.*, 2016, pp. 1–9.
[15] H. Feng, J. Liorca, A. M. Tulino, D. Raz, and A. F. Molisch, "Approximation algorithms for the NFV service distribution problem," in *Proc. IEEE Conf. Comput. Commun.*, 2017, pp. 1–9.
[16] C. Pham, N. H. Tran, S. Ren, W. Saad, and C. S. Hong, "Traffic-aware and energy-efficient vNF placement for service chaining: Joint sampling and matching approach," in *Proc. IEEE Trans. Serv. Comput.*, 2017, doi: 10.1109/TSC.2017.2671867.
[17] J. Zhang, W. Wu, and J. Lui, "On the theory of function placement and chaining for network function virtualization," in *Proc. 18th ACM Int. Symp. Mobile Ad Hoc Netw. Comput.*, 2018, pp. 91–100.
[18] L. Guo, J. Pang, and A. Walid, "Joint placement and routing of network function chains in data centers," in *Proc. IEEE Conf. Comput. Commun.*, 2018, pp. 612–620.
[19] L. Qu, C. Assi, and K. Shaban, "Delay-aware scheduling and resource optimization with network function virtualization," *IEEE Trans. Commun.*, vol. 64, no. 9, pp. 3746–3758, Sep. 2016.
[20] P. Vizarreta, M. Condoluci, C. Mahuca, T. Mahmoodi, and W. Kellerer, "QoS-driven function placement reducing expenditures in NFV deployments," in *Proc. IEEE Int. Conf. Commun.*, 2017, pp. 1–7.
[21] Z. Zhang, Z. Li, C. Wu, and C. Huang, "A scalable and distributed approach for NFV service chain cost minimization," in *Proc. IEEE Int. Conf. Distrib. Comput. Syst.*, 2017, pp. 2151–2156.
[22] Q. Li, Y. Jiang, P. Duan, M. Xu, and X. Xiao, "Quokka: Latency-aware middlebox scheduling with dynamic resource allocation," *J. Netw. Comput. Appl.*, vol. 78, pp. 253–266, 2017.
[23] A. Dwaraki and T. Wolf, "Adaptive service-chain routing for virtual network functions in software-defined networks," in *Proc. Workshop Hot Top. Middleboxes Netw. Function Virtualization*, 2016, pp. 32–37.
[24] Z. Allybokus, N. Perrot, J. Leguay, L. Maggi, and E. Gourdin, "Virtual function placement for service chaining with partial orders and anti-affinity rules," *Netw.*, vol. 71, no. 2, pp. 97–106, 2017.
[25] H. Chen, X. Wang, Y. Zhao, T. Song, Y. Wang, S. Xu, and L. Li, "Mosc: A method to assign the outsourcing of service function chain across multiple clouds," *Comput. Netw.*, vol. 133, pp. 166–182, 2018.
[26] C. Sun, J. Bi, Z. Zheng, H. Yu, and H. Hu, "NFP: Enabling network function parallelism in NFV," in *Proc. Conf. ACM Special Interest Group Data Commun.*, 2017, pp. 43–56.
[27] R. Gouareb, V. Friderikos, and A.-H. Aghvami, "Virtual network functions routing and placement for edge cloud latency minimization," *IEEE J. Select. Areas Commun.*, vol. 36, no. 10, pp. 2346–2357, Oct. 2018.
[28] A. Hmaity, M. Savi, F. Musumeci, M. Tornatore, and A. Pattavina, "Virtual network function placement for resilient service chain provisioning," in *Proc. 8th IEEE/IFIP Int. Workshop Resilient Netw. Des. Model.*, 2016, pp. 245–252.
[29] M. T. Beck, J. F. Botero, and K. Samelin, "Resilient allocation of service function chains," in *Proc. IEEE Conf. Netw. Function Virtualization Softw. Defined Netw.*, 2016, pp. 128–133.
[30] B. Han, V. Gopalakrishnan, G. Kathirvel, and A. Shaikh, "On the resiliency of virtual network functions," *IEEE Commun. Mag.*, vol. 55, no. 7, pp. 152–157, Jul. 2017.
[31] S. Herker, X. An, W. Kiess, S. Beker, and A. Kirstaedter, "Datacenter architecture impacts on virtualized network functions service chain embedding with high availability requirements," in *Proc. IEEE Globecom Workshops*, 2015, pp. 1–7.
[32] 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 Top. Middleboxes Netw. Function Virtualization*, 2015, pp. 13–18.
[33] W. Ding, H. Yu, and S. Luo, "Enhancing the reliability of services in nfv with the cost-efficient redundancy scheme," in *Proc. IEEE Int. Conf. Commun.*, 2017, pp. 1–6.
[34] L. Qu, C. Assi, K. Shaban, and M. J. Khabbaz, "A reliability-aware network service chain provisioning with delay guarantees in NFV-enabled enterprise datacenter networks," *IEEE Trans. Netw. Serv. Manage.*, vol. 14, no. 3, pp. 554–568, Sep. 2017.
[35] J. I. McCool, *Probability and Statistics With Reliability, Queuing and Computer Science Applications*. New York, NY, USA: Taylor & Francis, 2003.
[36] K. V. Vishwanath and N. Nagappan, "Characterizing cloud computing hardware reliability," in *Proc. 1st ACM Symp. Cloud Comput.*, 2010, pp. 193–204.
[37] D. Ford, F. Labelle, F. I. Popovici, M. Stokely, V.-A. Truong, L. Barroso, C. Grimes, and S. Quinlan, "Availability in globally distributed storage systems," in *Proc. 9th USENIX Conf. Operating Syst. Des. Implementation*, 2010, vol. 10, pp. 1–7.
[38] S. Yang, S. Trajanovski, and F. A. Kuipers, "Availability-based path selection and network vulnerability assessment," *Netw.*, vol. 66, no. 4, pp. 306–319, 2015.
[39] S. Yang, P. Wieder, R. Yahyapour, S. Trajanovski, and X. Fu, "Reliable virtual machine placement and routing in clouds," *IEEE Trans. Parallel Distrib. Syst.*, vol. 28, no. 10, pp. 2965–2978, Oct. 2017.
[40] M. Ghaznavi, N. Shahriar, S. Kamali, R. Ahmed, and R. Boutaba, "Distributed service function chaining," *IEEE J. Select. Areas Commun.*, vol. 35, no. 11, pp. 2479–2489, Nov. 2017.
[41] P. Van Mieghem and F. A. Kuipers, "Concepts of exact QoS routing algorithms," *IEEE/ACM Trans. Netw.*, vol. 12, no. 5, pp. 851–864, Oct. 2004.
[42] G. Shen and R. S. Tucker, "Energy-minimized design for IP over WDM networks," *IEEE/OSA J. Opt. Commun. Netw.*, vol. 1, no. 1, pp. 176–186, Jun. 2009.
[43] M. Savi, M. Tornatore, and G. Verticale, "Impact of processing costs on service chain placement in network functions virtualization," in *Proc. IEEE Conf. Netw. Function Virtualization Softw. Defined Netw.*, 2015, pp. 191–197.



Song Yang received the PhD degree from Delft University of Technology, The Netherlands, in 2015. From August 2015 to July 2017, he worked as postdoc researcher for the EU FP7 Marie Curie Actions CleanSky Project in Gesellschaft für wissenschaftliche Datenverarbeitung mbH Göttingen (GWDG), Göttingen, Germany. He is currently an associate professor with the School of Computer Science and Technology, Beijing Institute of Technology, China. His research interests include data communication networks, cloud/edge computing, and network function virtualization. He is a member of the IEEE.

Fan Li received the BEng and MEng degrees in communications and information system from Huazhong University of Science and Technology, China, respectively, the MEng degree in electrical engineering from the University of Delaware, and the PhD degree in computer science from the University of North Carolina at Charlotte. She is a professor with the School of Computer Science, Beijing Institute of Technology, China. Her current research focuses on wireless networks, smart sensing, crowd sensing and mobile computing. Her papers received Best Paper Awards from IEEE MASS (2013), IEEE IPCCC (2013), ACM MobiHoc (2014), and Tsinghua Science and Technology (2015). She is a member of the ACM and IEEE.

Ramin Yahyapour received the doctoral degree in electrical engineering. He is a full professor with the Georg-August University of Göttingen. He is also managing director of the GWDG, a joint compute and IT competence center of the university and the Max Planck Society. His research interest include efficient resource management in its application to service-oriented infrastructures, clouds, and data management. He is especially interested in data and computing services for eScience. He gives lectures on parallel processing systems, service computing, distributed systems, cloud computing, and grid technologies. He was and is active in several national and international research projects. He serves regularly as reviewer for funding agencies and consultant for IT organizations. He is organizer and program committee member of conferences and workshops as well as reviewer for journals.

Xiaoming Fu received the PhD degree in computer science from Tsinghua University, Beijing, China, in 2000. He was then a research staff with the Technical University Berlin until joining the University of Göttingen, Germany, in 2002, where he has been a professor in computer science and heading the Computer Networks Group since 2007. He has spent research visits with universities of Cambridge, Uppsala, UPMC, Columbia, UCLA, Tsinghua, Nanjing, Fudan, and PolyU of Hong Kong. His research interests include network architectures, protocols, and applications. He is currently an editorial board member of the *IEEE Communications Magazine*, the *IEEE Transactions on Network and Service Management*, and the *Elsevier Computer Communications*, and has served on the organization or program committees of leading conferences such as INFOCOM, ICNP, ICDCS, MOBICOM, MOBIHOC, CoNEXT, ICN and COSN. He is a senior member of the IEEE, an IEEE Communications Society Distinguished Lecturer, a fellow of IET and member of the Academia Europaea.