On the Upgrade of Service Function Chains with Heterogeneous NFV Platforms

Yuhan Xue and Zuqing Zhu, *Senior Member, IEEE*

***Abstract*—The fast development of high-performance and flexible SmartNICs and programmable data plane switches (PDP-SWs) has motivated people to consider the deployment of virtual network functions (vNFs) on them. Hence, together with traditional virtual machines (VMs), SmartNICs and PDP-SWs form heterogeneous network function virtualization (NFV) platforms for realizing vNF service chains (vNF-SCs). In this work, we consider the transition from software-based homogeneous NFV platforms to the heterogeneous ones, and study how to optimize the service upgrade of vNF-SCs. Specifically, the service upgrade is divided into two steps, which are 1) selecting servers/switches in the substrate network (SNT) to upgrade, which is done by adding SmartNICs to servers and replacing traditional switches with PDP-SWs, under a fixed budget, and 2) redeploying the existing vNF-SCs in the updated SNT to maximize the quality-of-service (QoS) improvement on latency reductions. We first formulate an integer linear programming (ILP) model to optimize the overall service upgrade, then design two correlated optimizations for its two steps, and finally propose polynomial-time approximation algorithms to solve the optimizations. The results of extensive simulations confirm that our proposed algorithm outperforms the existing benchmarks in various network scenarios, and achieves better tradeoff between performance and time-efficiency.**

***Index Terms*—Network function virtualization (NFV), Heterogeneous NFV platforms, Service function chain (SFC), Approximation algorithm, Facility location, Multiple knapsack, Linear programming (LP) relaxation and randomized rounding.**

I. Introduction

NOWADAYS, the Internet is undergoing dramatic changes to adapt to the unprecedented and ever-growing amounts of data traffic, end users, and network services [1–3]. Hence, network infrastructures have been reshaped by advanced physical-layer technologies [4–9] to be better prepared for tremendous volumes of highly-dynamic traffic. Although this has made network pipes wider and more flexible, the packet processing on nodes should be more powerful and agile to better serve heterogeneous network services with various quality-of-service (QoS) demands. To this end, novel programmable packet processing hardware (*e.g.*, smart network interface cards (SmartNICs) [10] and programmable data plane switches (PDP-SWs) [11]) have been developed and attracted intensive interests recently. The aforementioned advances on network pipes and nodes have promoted the idea of in-network computing [12], *i.e.*, computing tasks are handled simultaneously with packet switching on forwarding devices such that both the latencies of packet processing and transmission and the volume of traffic can be effectively reduced.

Y. Xue and Z. Zhu are with the School of Information Science and Technology, University of Science and Technology of China, Hefei, Anhui 230027, P. R. China (email: zqzhu@ieee.org).

Manuscript received on May 21, 2021.

As in-network computing deploys virtual network functions (vNFs) on forwarding devices to handle computing tasks [13, 14], it belongs to the regime of network function virtualization (NFV) [15, 16]. Specifically, the hardware-based forwarding devices (*e.g.*, SmartNICs and PDP-SWs) can be considered together with their software-based counterparts (*e.g.*, virtual machines (VMs) on servers) to come up with a network environment with heterogeneous NFV platforms [17–20]. Due to their programmability, SmartNICs and PDP-SWs are all general-purpose hardware, and thus realizing vNFs over them will not violate the principle of NFV. On the contrary, they actually further extend the success of NFV, because heterogeneous NFV platforms can unify the benefits of hardware- and software-based systems, and provide service providers (SPs) more flexibility and programmability to support various QoS requirements cost-efficiently [20]. For instance, the hardware platforms have superior packet processing capacity and only induce very short latency when supporting bandwidth-intensive vNFs [19], while the software ones are runtime programmable and can provide enough computing and memory resources for computing-intensive vNFs [16].

Previously, people studied the provisioning of vNF service chains (vNF-SCs) over heterogeneous NFV platforms [17–20], to explore the flexibility and programmability mentioned above. However, these studies were all based on a fixed substrate network (SNT), which means that the deployment of heterogeneous NFV platforms (*e.g.*, servers, SmartNICs and PDP-SWs) in the SNT was assumed to be unchanged. Note that, in addition to the network services whose lifetime is relatively short, there are also a fairly amount of long-term ones in today's Internet, which can even run semi-permanently to handle the business of SPs [21–23]. For instance, the network services for online booking/shopping in e-commerce need to steer traffic through a series of network functions (*e.g.*, firewall, load-balancer, and request handler), whose configuration is normally for long-term [23].

Similar to the short-term ones, it is cost-effective for SPs to build these long-term network services with vNF-SCs. This is because they need to address dynamic service requests and the population change and mobility of end-users, even though the configuration of the network functions is relatively static. Moreover, to adapt to the spatial and temporal increases of service demands, it will be inevitable for the SPs to upgrade the substrate network elements on which to deploy the vNF-SCs regularly. In other words, to ensure the QoS of long-term vNF-SCs, we need to jointly optimize the service upgrade of an SNT with heterogeneous NFV platforms and the reconfiguration of active vNF-SCs in the upgraded SNT, under a fixed

equipment budget. Note that, this optimization is intrinsically more complex than the provisioning or reconfiguration of vNF-SCs in a fixed SNT [24], because the upgrade of the SNT and the reconfiguration of active vNF-SCs are correlated with each other and thus cannot be tackled independently. To the best of our knowledge, this problem has not been fully explored yet.

Previously, in [24], we conducted an initial study on the problem. Specifically, we considered one of the most relevant network environments for vNF-SC provisioning, *i.e.*, a data-center network (DCN), as the SNT, and optimized the service upgrade of vNF-SCs in it by leveraging heterogeneous NFV platforms (*i.e.*, SmartNICs, PDP-SWs and VMs). The service upgrade includes two steps, 1) selecting servers/switches in the DCN to upgrade, which is done by adding SmartNICs to servers and replacing traditional switches with PDP-SWs, under a fixed budget, and 2) redeploying active vNF-SCs in the updated DCN to maximize the QoS improvement on latency reductions. To solve the correlated optimizations of the two steps, we designed a time-efficient heuristic in [24].

However, the algorithm design in [24] was still preliminary, because the heuristic can hardly obtain near-optimal solutions whose performance gaps to the optimal ones are bounded. More importantly, the algorithms were designed based on the assumption that the SNT is a DCN, and thus it is not generic. This motivates us to extend the study in this work. Specifically, by revisiting the two steps of the budget-constrained service upgrade, we first formulate an overall integer linear programming (ILP) model to optimize them jointly for the exact solution, then obtain two correlated optimizations for the steps, respectively, and finally propose polynomial-time approximation algorithms to solve both optimizations.

For the first step, we design the optimization as to select servers/switches (*i.e.*, substrate nodes (SNs)) in a generic SNT to upgrade such that the number of existing vNFs, which can be redeployed on the upgraded SNs, is maximized. To design an approximation algorithm for it, we divide the problem-solving into two phases. In *Phase* I, we relax the budget constraint, transform the SN selection problem into a capacitated facility location problem (CFLP) [25], and solve it with linear programming (LP) rounding. Next, in *Phase* II, we remove the SNs to be upgraded in iterations until the budget constraint is satisfied. In each iteration, we first recalculate the mapping between the existing vNFs and the SNs for upgrading, and then remove the SN on which the smallest number of existing vNFs will be redeployed. More specifically, the recalculation of the mapping is first transformed into the multiple knapsack problem with assignment restrictions and capacity constraints (MK-AR-CC), and is then solved by leveraging the approximation algorithm developed in [26].

After tackling the first step with the two-phase approach, we move to the second step, and propose an approximation algorithm based on LP relaxation and randomized rounding [27] to determine how to redeploy the existing vNF-SCs in the updated SNT. Finally, we perform extensive simulations to demonstrate that the algorithm proposed in this work outperforms the heuristic in [24], and achieves better tradeoff between performance and time-efficiency.

The rest of the paper is organized as follows. We survey the related work in Section II. Section III explains our network model and shows the ILP for the overall optimization. We formulate two ILPs for the steps of the budget-constrained service upgrade and propose polynomial-time approximation algorithms to solve them, in Sections IV and V, respectively. Numerical simulations are discussed in Section VI for performance evaluation. Finally, Section VII summarizes the paper.

II. Related Work

The basic idea of NFV is to deploy network services with vNFs running on general-purpose software/hardware platforms instead of relying on proprietary hardware systems [15]. Depending on how the vNFs are organized and how the application traffic is steered through them, an SP can leverage NFV to realize network services with vNF-SCs [28, 29], vNF multicast trees [30], and generic vNF forwarding graphs [31]. To provision these network services in an SNT, the SP needs to embed vNFs on SNs (*i.e.*, instantiating vNFs with the memory resources in SNs) and route the traffic among vNFs over substrate paths. Although this looks similar to the procedure of the well-known virtual network embedding (VNE) [32–34], they are different because an SP can embed multiple vNFs, which are in the same network service, on one SN (*i.e.*, violating the one-to-one mapping in VNE [30]). For a comprehensive survey on NFV, one can refer to [35].

For vNF-SCs specifically, many previous studies have been devoted to optimizing their service provisioning schemes [36], and the technical specifications of vNF-SC have been published in [37]. However, these studies assumed that the SNT for vNF-SC provisioning has fixed configuration, and did not address the service upgrade that involves incremental deployment and adjustment of substrate network elements. Meanwhile, still based on the assumption that the hardware configuration of the SNT is unchanged, people have investigated the reconfiguration of vNF-SCs in [38–42], where the studies in [38–41] were based on an SNT with homogeneous NFV platforms, and the SNT that consists of heterogeneous NFV platforms was considered in [42]. The study in [38] addressed how to reconfigure vNF-SCs to adapt to the movement of end users. Eramo *et al.* [39] proposed algorithms to migrate vNFs such that the dynamics of vNF-SC requests in terms of volume and spatial distribution can be handled well. In [40], the authors tried to optimize the reconfiguration of vNF-SCs under time-varying service demands, for minimizing the total cost of vNF deployment and vNF-SC reconfiguration. The study in [41] designed an online scaling algorithm to readjust vNFs according to dynamic demands and minimize the operational cost of vNF-SCs. Hu *et al.* [42] reduced the overheads of VM-based vNF-SC provisioning by reconfiguring vNF-SCs to use the existing heterogeneous NFV platforms in an SNT. Nevertheless, all of these studies did not address how to upgrade the hardware of an SNT with heterogeneous NFV platforms to better serve the long-term vNF-SCs in it.

Previously, there also have been some studies on the network upgrade related to NFV [43–45]. Poularakis *et al.* [43] tackled the problem of how to gradually upgrade a traditional network to the one that enables software-defined networking

(SDN) (*i.e.*, hardware upgrade). The studies in [44, 45] jointly considered SDN and NFV and devoted themselves to supporting fast and consistent network policy updates under different constraints (*i.e.*, software upgrade). As these investigations did not jointly optimize the hardware upgrade of an SNT and the reconfiguration of active vNF-SCs in the upgraded SNT, our work is different from them.

Inspired by the principle of in-network computing [12], *i.e.*, extending the functionalities of forwarding devices beyond packet switching, people recently started to consider the deployment of vNFs on hardware-based programmable forwarding devices, such as SmartNICs and PDP-SWs. For instance, in [46, 47], the authors offloaded key-value stores to PDP-SWs and SmartNICs, respectively, while NetHCF [48] realized the vNF for filtering spoofed traffic on PDP-SWs. Previously, the studies in [17–20, 42] have addressed the provisioning of vNF-SCs in an SNT that includes heterogeneous NFV platforms. Nevertheless, they did not consider the service upgrade that can change the hardware configuration of the SNT. Note that, in addition to the upgrade schemes involving SmartNICs and PDP-SWs, the service upgrade of vNF-SCs can also be realized by leveraging software-enabled acceleration (*e.g.*, the data plane development kit (DPDK)) [49]. However, such an upgrade is software-based, which does not change the hardware configuration of an SNT. Hence, its optimization is equivalent to that for reconfiguring active vNF-SCs in a fixed SNT, which has already been studied in [38–42] and thus does not need to be revisited. Moreover, SmartNICs and PDP-SWs provide better traffic processing performance than the software-enabled acceleration schemes [50, 51].

To the best of our knowledge, our previous work in [24] was the only one that investigated how to upgrade an SNT with heterogeneous NFV platforms (*i.e.*, SmartNICs, PDP-SWs and VMs) to serve the vNF-SCs in it better. However, the algorithm design in [24] was still preliminary, especially for the time-efficient heuristic, as it is not generic and should be improved.

III. Network Model

We model the SNT as an undirected graph $G(V, E)$, where V and E are the sets of SNs and substrate links (SLs), respectively. The SNT consists of two types of SNs, *i.e.*, switches and servers, for vNF-SC deployment. Before the service upgrade, the SNT only includes traditional switches/servers, which means that there is no SmartNIC on the servers and vNFs can only be instantiated on servers with VMs. In other words, the SNT only consists of homogeneous software-based NFV platforms before the upgrade. The upgrade equips SmartNICs on servers and replaces switches with PDP-SWs under a preset budget on equipment cost. Therefore, after the upgrade, the SNT includes heterogeneous NFV platforms (*i.e.*, SmartNICs, PDP-SWs and VMs), and the SP can redeploy the existing vNFs on SmartNICs and PDP-SWs to improve their traffic processing capacities and reduce their latencies.

Note that, the differences between PDP-SWs and SmartNICs lay in 1) PDP-SWs are stand-alone forwarding devices, which can completely replace traditional switches in network upgrades, while SmartNICs are usually equipped on commodity servers and thus cannot work by themselves, 2) the packet processing in a PDP-SW is normally much more powerful than that in a SmartNIC, and thus when carrying the same vNF, the processing latency of the PDP-SW will be shorter, 3) SmartNICs have more memory for instantiating vNFs, because they are equipped on servers and thus do not need to use as much memory as PDP-SWs on normal packet switching, and 4) the cost of PDP-SW is higher than that of a SmartNIC.

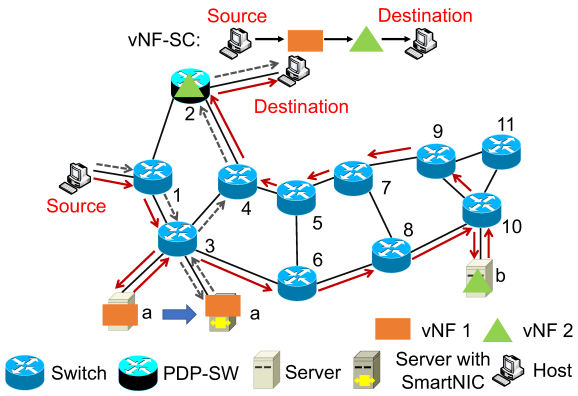


Fig. 1. Example on service upgrade with heterogeneous NFV platforms.

We define a set M to include all the vNF types that can be supported in the SNT. Then, considering the heterogeneity among the NFV platforms, we assume that if a type-m vNF ($m \in M$) gets instantiated on a PDP-SW/SmartNIC/VM, it consumes $\hat{c}_m^S/\hat{c}_m^N/\hat{c}_m^V$ units of memory, respectively, and has a traffic processing capacity of $b_m^S/b_m^N/b_m^V$ in bandwidth units, respectively. Meanwhile, if the service upgrade migrates a type-m vNF from a VM to a PDP-SW/SmartNIC, a latency reduction of δ_m^S/δ_m^N in time units can be achieved, respectively.

For the service upgrade, latency reductions can be achieved not only by migrating vNFs from VMs to PDP-SWs/SmartNICs but also by rerouting their application traffic. Fig. 1 gives an illustrative example on this. The vNF-SC before the upgrade is marked in red, and it has to take a detour to reach the vNFs on *Servers* a and b. The upgrade replaces *Switch* 2 with a PDP-SW and migrates *vNF* 2 to it from *Server* b, and equips a SmartNIC on *Server* a to carry *vNF* 1. Hence, the vNF-SC's routing path gets shortened by 8 hops (the one marked in gray), and moreover, as the traffic processing in a PDP-SW/SmartNIC is much faster than that in a VM, the end-to-end (E2E) latency of the vNF-SC is further reduced. Note that, when calculates the E2E latency of a vNF-SC, this work considers two types of latencies, *i.e.*, the processing latencies on all the vNFs in the vNF-SC, and the propagation latencies on all the SLs that the traffic of the vNF-SC goes through.

A. Overall ILP Model

With the aforementioned network model, we formulate the problem of the service upgrade as to 1) select servers/switches in the SNT to upgrade under a fixed budget, and 2) redeploy the existing vNF-SCs in the upgraded SNT to maximize the QoS improvement on latency reduction. As the output of the first step affects the performance of the second one, we first formulate an ILP model to cover the overall optimization. The ILP takes the provisioning schemes of active vNF-SCs before

the upgrade as the input, and aims to maximize the QoS improvement on the E2E latencies of these vNF-SCs with the upgrade and subsequent vNF-SC reconfiguration. Hence, in the following, we define the parameters $\{\varphi_{v,l}^{in,i}, \psi_{p,n}^{in,i}, \tau_i^{in}, \varpi_i^{in}\}$ to represent the state of active vNF-SCs before the upgrade, while the variables $\{\vartheta_{i,l}^{v}, \varphi_{v,l}^{out,i}, \psi_{p,n}^{out,i}, \tau_i^{out}, \varpi_i^{out}\}$ are introduced to denote the state of active vNF-SCs in the upgraded SNT.

Parameters:

- $G(V, E)$: the topology of the SNT.
- $B_{(u,v)}$: the bandwidth capacity of SL $(u, v) \in E$.
- M: the set of all the vNF types supported in the SNT.
- P: the set of precalculated substrate paths in the SNT.
- $SC_i = \{s_i, d_i, \{f_{i,1}, \cdots, f_{i,l}, \cdots, f_{i,N_i}\}, b_i, t_i\}$: the i-th vNF-SC, where s_i and d_i are the source and destination SNs, respectively, b_i is its bandwidth demand, t_i is its QoS demand on E2E latency, and N_i is the number of vNFs in its vNF-SC. We have $R = \{SC_i, \forall i\}$.
- $f_{i,l}^{m}$: the boolean that equals 1 if the l-th vNF ($f_{i,l}$) in SC_i is a type-m vNF ($m \in M$), and 0 otherwise.
- $\varphi_{v,l}^{in,i}$: the boolean that equals 1 if the l-th vNF in SC_i is embedded on SN v before the upgrade.
- $\psi_{p,n}^{in,i}$: the boolean that equals 1 if the n-th virtual link in SC_i is embedded on substrate path p before the upgrade.
- τ_i^{in}: the E2E latency of SC_i before the upgrade.
- ϖ_i^{in}: the boolean that equals 1 if SC_i satisfies the demand on E2E latency before the upgrade, and 0 otherwise.
- $\zeta_v^{in,i}$: the boolean that equals 1 if SC_i passes through SN v before the upgrade.
- μ_v: the boolean that equals 1 if SN v is a switch, and 0 if it is a server.
- δ_m^S: the latency reduction achieved after migrating a type-m vNF from a VM to a PDP-SW.
- δ_m^N: the latency reduction achieved after migrating a type-m vNF from a VM to a SmartNIC.
- $sp_{(u,v)}^{p}$: the boolean that equals 1 if SL (u, v) is on p.
- D_p: the E2E delay of substrate path p, which is proportional to the hop-count of p.
- $C_v^S/C_v^N/C_v^V$: the memory space of an NFV platform on SN v, if it is a PDP-SW/SmartNIC/server.
- $b_m^S/b_m^N/b_m^V$: the bandwidth capacity of a type-m vNF, if it is deployed on a PDP-SW/SmartNIC/VM.
- $\hat{c}_m^S/\hat{c}_m^N/\hat{c}_m^V$: the memory space consumed by a type-m vNF, if it is deployed on a PDP-SW/SmartNIC/VM.
- Ω: the total budget on the equipment cost of new PDP-SWs and SmartNICs used in the upgrade.
- ϕ^S/ϕ^N: the cost of a PDP-SW/SmartNIC.

Variables:

- $\vartheta_{i,l}^{v}$: the boolean variable that equals 1 if SN v is selected for being updated and the l-th vNF in SC_i is deployed on the upgraded part of SN v, and 0 otherwise.
- $\varphi_{v,l}^{out,i}$: the boolean variable that equals 1 if the l-th vNF in SC_i is deployed on SN v, and 0 otherwise.
- $\psi_{p,n}^{out,i}$: the boolean variable that equals 1 if n-th virtual link (VL) in SC_i uses path p, and 0 otherwise.
- τ_i^{out}: the E2E latency of SC_i after the upgrade.
- ϖ_i^{out}: the boolean variable that equals 1 if SC_i satisfies its E2E latency demand after the update, and 0 otherwise.
- ι_v: the boolean variable that equals 1 if SN v is selected for being updated, and 0 otherwise.
- $x_{i,l}^{v}, y_{i,l}^{v}, z_{i,n}^{p}$: the boolean auxiliary variables that are introduced for linearization.

Objective:

The overall optimization tries to maximize the additional QoS satisfaction due to E2E latency reductions, which are achieved by the upgrade. Therefore, the objective should be

$$\text{Maximize} \quad \sum_i (\varpi_i^{out} - \varpi_i^{in}). \tag{1}$$

Constraints:

$$\begin{cases} x_{i,l}^{v} = \vartheta_{i,l}^{v} \cdot \varphi_{v,l}^{out,i}, & \forall i, l, v, \\ y_{i,l}^{v} = \varpi_i^{out} \cdot x_{i,l}^{v}, & \forall i, l, v, \\ z_{i,n}^{p} = \varpi_i^{out} \cdot \psi_{p,n}^{out,i}, & \forall i, n, p. \end{cases} \tag{2}$$

Eq. (2) explains the definitions of the boolean auxiliary variables for linearization, each of which represents the multiplication of two boolean variables.

$$\begin{cases} x_{i,l}^{v} \le \vartheta_{i,l}^{v} \\ x_{i,l}^{v} \le \varphi_{v,l}^{out,i} \\ x_{i,l}^{v} \ge \varphi_{v,l}^{out,i} + \vartheta_{i,l}^{v} - 1 \end{cases}, \quad \forall i, l, v, \tag{3}$$

$$\begin{cases} y_{i,l}^{v} \le \varpi_i^{out} \\ y_{i,l}^{v} \le x_{i,l}^{v} \\ y_{i,l}^{v} \ge x_{i,l}^{v} + \varpi_i^{out} - 1 \end{cases}, \quad \forall i, l, v, \tag{4}$$

$$\begin{cases} z_{i,n}^{p} \le \varpi_i^{out} \\ z_{i,n}^{p} \le \psi_{p,n}^{out,i} \\ z_{i,n}^{p} \ge \psi_{p,n}^{out,i} + \varpi_i^{out} - 1 \end{cases}, \quad \forall i, n, p. \tag{5}$$

Eqs. (3)-(5) are the constraints for linearization.

$$\begin{aligned} (t_i - \tau_i^{in}) \cdot \varpi_i^{out} + \sum_{n,p} z_{i,n}^{p} \cdot D_p - \sum_{n,p} \psi_{p,n}^{in,i} \cdot D_p \cdot \varpi_i^{out} \\ + \sum_{l,m} f_{i,l}^{m} \cdot \sum_{v} y_{i,l}^{v} \cdot \delta_m^N \cdot (1 - \mu_v) \\ + \sum_{l,m} f_{i,l}^{m} \cdot \sum_{v} y_{i,l}^{v} \cdot \delta_m^S \cdot \mu_v \ge 0, \quad \forall i. \end{aligned} \tag{6}$$

Eq. (6) ensures that the value of ϖ_i^{out} can be derived correctly.

$$\sum_v \iota_v \cdot [\phi^S \cdot \mu_v + \phi^N \cdot (1 - \mu_v)] \le \Omega, \quad \forall i, l. \tag{7}$$

Eq. (7) ensures that the upgrade's cost is within the budget.

$$\begin{cases} \sum_{i,l,m} \hat{c}_m^S \cdot f_{i,l}^{m} \cdot x_{i,l}^{v} \cdot \mu_v \le C_v^S \\ \sum_{i,l,m} \hat{c}_m^N \cdot f_{i,l}^{m} \cdot x_{i,l}^{v} \cdot (1 - \mu_v) \le C_v^N \\ \sum_{i,l,m} \hat{c}_m^V \cdot f_{i,l}^{m} \cdot (\varphi_{v,l}^{out,i} - x_{i,l}^{v}) \cdot (1 - \mu_v) \le C_v^V \end{cases}, \quad \forall v. \tag{8}$$

Eq. (8) ensures that the vNF deployment on each SN will not use more memory than the corresponding memory space.

$$\begin{cases} \sum_i b_i \cdot \sum_l f_{i,l}^{m} \cdot x_{i,l}^{v} \cdot \mu_v \le b_m^S \\ \sum_i b_i \cdot \sum_l f_{i,l}^{m} \cdot x_{i,l}^{v} \cdot (1 - \mu_v) \le b_m^N \\ \sum_i b_i \cdot \sum_l f_{i,l}^{m} \cdot (\varphi_{v,l}^{out,i} - x_{i,l}^{v}) \cdot (1 - \mu_v) \le b_m^V \end{cases}, \quad \forall m, v. \tag{9}$$

Eq. (9) ensures that the vNF deployment on each SN will not use more bandwidth than the corresponding capacities.

$$\sum_{i,n} \psi_{p,n}^{out,i} \cdot sp_{(u,v)}^{p} \cdot b_i \leq B_{(u,v)}, \quad \forall (u,v) \in E. \tag{10}$$

Eq. (10) ensures that the bandwidth capacity of each SL will not be exceeded.

$$\begin{aligned} &\sum_{v} \varphi_{v,l}^{out,i} \cdot \vartheta_{i,l}^{v} \cdot \mu_v + \sum_{v} \varphi_{v,l}^{out,i} \cdot \vartheta_{i,l}^{v} \cdot (1-\mu_v) \\ &+ \sum_{v} \varphi_{v,l}^{out,i} \cdot (1-\vartheta_{i,l}^{v}) \cdot (1-\mu_v) = 1, \quad \forall i,l. \end{aligned} \tag{11}$$

Eq. (11) ensures that each vNF in a vNF-SC is deployed on one and only one NFV platform.

$$\varphi_{v,l}^{out,i} \leq \zeta_{v}^{in,i}, \quad \forall i,l,v. \tag{12}$$

Eq. (12) ensures that the changes of vNF deployments caused by an SN upgrade happen at the right locations.

$$\begin{aligned} &\sum_{n,p,(u,v)\in E} \psi_{p,n}^{out,i} \cdot sp_{(u,v)}^{p} - \sum_{n,p,(v,u)\in E} \psi_{p,n}^{out,i} \cdot sp_{(v,u)}^{p} \\ &= \begin{cases} 1, & u = s_i \\ -1, & u = d_i, \quad \forall i. \\ 0, & \text{otherwise} \end{cases} \end{aligned} \tag{13}$$

Eq. (13) ensures that the flow conservation conditions.

$$\sum_{p} \psi_{p,n}^{out,i} = 1, \quad \forall i,n. \tag{14}$$

Eq. (14) ensures that each VL in a vNF-SC is mapped onto one and only one substrate path.

$$\begin{cases} \sum\limits_{\{p:s_i \to v, p\in P\}} \psi_{p,1}^{out,i} \geq \varphi_{v,1}^{out,i}, \quad \forall i,v, \\ \sum\limits_{\{p:u \to v, p\in P\}} \psi_{p,l}^{out,i} \geq \varphi_{u,l-1}^{out,i} + \varphi_{v,l}^{out,i} - 1, \forall i,u,v,l, \\ \sum\limits_{\{p:v \to d_i, p\in P\}} \psi_{p,N_i+1}^{out,i} \geq \varphi_{v,N_i}^{out,i}, \quad \forall i,v. \end{cases} \tag{15}$$

Eq. (15) ensures that for each vNF-SC, its vNFs are connected in the right sequence.

$$\sum_{i,l} \vartheta_{i,l}^{v} = \iota_v, \quad \forall v. \tag{16}$$

Eq. (16) ensures that relation between $\vartheta_{i,l}^{v}$ and ι_v is correct, *i.e.*, for any v, if we have $\vartheta_{i,l}^{v} = 1$, the value of ι_v will be 1.

Note that, Eq. (6) derives the value of ϖ_i^{out} based on the E2E latencies of SC_i before and after the upgrade (*i.e.*, τ_i^{in} and τ_i^{out}, respectively), where τ_i^{out} can be expressed with τ_i^{in} and the latency reduction brought by the upgrade. The formulation of the ILP above suggests that the overall optimization is relatively complex. Meanwhile, since the optimization involves many variables and constraints, it would be difficult to design a polynomial-time approximation algorithm for it too. Hence, we decide to tackle its two steps separately. Specifically, we formulate an optimization for each step, and solve it with a polynomial-time approximation algorithm. Note that, dividing a complex optimization into sequential steps can cause an uncertain loss to the approximation ratio of the final solution. Therefore, we design the objective of the first step such that the optimization toward it can assist the second step to achieve good performance. The simulation results in Section VI will verify the performance of our two-step algorithm.

IV. First Step of Two-Step Algorithm: Determining the Upgrade Scheme

According to the discussions in Section III-A, the first step of the service update is to select servers/switches in the SNT to upgrade under a fixed budget. Note that, as this step determines the configuration of the upgraded SNT, its output will affect the performance of the second step. Hence, we should design the objective of the first step to be beneficial for the optimization in the second one. Intuitively, the upgrading of a server/switch will be more beneficial to the vNF-SC redeployment in the second step, if more vNFs can be migrated onto it to achieve latency reductions on their vNF-SCs. To this end, we define the objective of the first step as to maximize the number of vNFs that can be migrated to the upgraded SNs (*i.e.*, servers and switches), and formulate its optimization as

Variables:

- $\varphi_{v,l}^{out,i}$, ι_v and $\vartheta_{i,l}^{v}$: the boolean variables whose definitions are the same as those the overall ILP in Section III-A.
- $\gamma_{i,l}^{v}$: the boolean auxiliary variable for linearization.

Objective:

The optimization objective is designed as follows.

$$\text{Maximum} \quad \sum_{i,l,v} \varphi_{v,l}^{out,i} \cdot \vartheta_{i,l}^{v}, \tag{17}$$

which means that we would like to maximize the number of vNFs that can be embedded on upgraded SNs. As the objective in Eq. (17) is nonlinear, we introduce $\gamma_{i,l}^{v}$ to linearize it

$$\gamma_{i,l}^{v} = \varphi_{v,l}^{out,i} \cdot \vartheta_{i,l}^{v}, \quad \forall i,l,v, \tag{18}$$

where the detailed linearization will be explained in Eq. (20) below. Then, the objective in Eq. (17) is linearized as

$$\text{Maximum} \quad \sum_{i,l,v} \gamma_{i,l}^{v}. \tag{19}$$

Constraints:

The constraints in Eqs. (7), (12) and (14) are adapted from the overall ILP in Section III-A.

$$\begin{cases} \gamma_{i,l}^{v} \leq \vartheta_{i,l}^{v} \\ \gamma_{i,l}^{v} \leq \varphi_{v,l}^{out,i} \\ \gamma_{i,l}^{v} \geq \varphi_{v,l}^{out,i} + \vartheta_{i,l}^{v} - 1 \end{cases}, \quad \forall i,l,v. \tag{20}$$

Eq. (20) is the constraint for linearization.

$$\begin{cases} \sum\limits_{i,l,m} \hat{c}_m^S \cdot f_{i,l}^{m} \cdot \gamma_{i,l}^{v} \cdot \mu_v \leq C_v^S \\ \sum\limits_{i,l,m} \hat{c}_m^N \cdot f_{i,l}^{m} \cdot \gamma_{i,l}^{v} \cdot (1-\mu_v) \leq C_v^N \\ \sum\limits_{i,l,m} \hat{c}_m^V \cdot f_{i,l}^{m} \cdot (\varphi_{v,l}^{out,i} - \gamma_{i,l}^{v}) \cdot (1-\mu_v) \leq C_v^V \end{cases}, \quad \forall v. \tag{21}$$

Eq. (21) ensures that the vNF deployment on each SN will not use more memory than the corresponding memory space.

$$\begin{cases} \sum\limits_{i} b_i \cdot \sum\limits_{l} f_{i,l}^{m} \cdot \gamma_{i,l}^{v} \cdot \mu_v \leq b_m^S \\ \sum\limits_{i} b_i \cdot \sum\limits_{l} f_{i,l}^{m} \cdot \gamma_{i,l}^{v} \cdot (1-\mu_v) \leq b_m^N \\ \sum\limits_{i} b_i \cdot \sum\limits_{l} f_{i,l}^{m} \cdot (\varphi_{v,l}^{out,i} - \gamma_{i,l}^{v}) \cdot (1-\mu_v) \leq b_m^V \end{cases}, \quad \forall m,v. \tag{22}$$

Eq. (22) ensures that the vNF deployment on each SN will not use more bandwidth than the corresponding capacities.

A. Phase I

To design an algorithm for the optimization mentioned above, we divide the problem-solving into two phases. In *Phase* I, we relax the budget constraint, and transform the optimization into the one that tries to minimize the budget used for the upgrade in which all the existing vNFs can be migrated onto an upgraded SN. Hence, we remove the constraint of Eq. (7), put it in the objective, and add a new constraint

$$\sum_{v} \gamma_{i,l}^{v} = 1, \quad \forall i, l, \tag{23}$$

to ensure that all the existing vNFs can be migrated onto one upgraded SN. Then, the new optimization is

$$\begin{aligned} &\text{Minimize} \quad \sum_{v} \iota_v \cdot [\phi^S \cdot \mu_v + \phi^N \cdot (1 - \mu_v)], \\ &\text{s.t.} \quad \text{Eqs. (11), (12), (16), (20)-(22), and (23).} \end{aligned} \tag{24}$$

By observing the optimization in Eq. (24), we find that it can be further transformed into a capacitated facility location problem (CFLP) with additional constraints [25], if we treat the existing vNFs as customers and all the servers/switches in the SNT as facilities. As the problem is $\mathcal{NP}$-hard [52], we leverage the idea introduced in [53] to design a polynomial-time approximation algorithm for it based on LP rounding.

Algorithm 1: Algorithm for optimization in Eq. (24)

Input: existing vNF-SCs $\{SC_i\}$, SNT $G(V, E)$.
Output: set of SNs to upgrade V', upgrade cost Ω'.

```
1 solve LP relaxation of Eq. (24) to get a solution X*;
2 for each SC_i ∈ R do
3     for the l-th vNF in SC_i do
4         include all the SNs on which the vNF can be
          embedded in set V_{i,l};
5         select an SN in V_{i,l} to upgrade for the vNF
          according to probabilities {ϑ^v_{i,l}, v ∈ V_{i,l}};
6         insert the SN in V' if it is not already in;
7         update upgrade cost Ω' and resource usages
          in the SNT;
8     end
9 end
```

Algorithm 1 shows the detailed procedure. In *Line* 1, we relax the ILP in Eq. (24) to obtain an LP, and solve it to get a solution X^*. Here, all the boolean variables are relaxed to real ones within $[0, 1]$. Then, the two for-loops that cover *Lines* 2-9 select an SN to upgrade and redeploy an existing vNF on it[1]. Specifically, for the l-th vNF in an existing vNF-SC SC_i, the SN selection works as follows. We first check the current routing path of SC_i, find all the SNs that are on it[2] and can carry the vNF, and include the SNs in set $V_{i,l}$ (*Line* 4). Here, if an SN has already been selected to upgrade, we check whether the upgraded SN has sufficient resources (*i.e.*, memory space and bandwidth capacity) to carry the vNF. If yes, the SN is included in $V_{i,l}$, and it is excluded, otherwise. On the other hand, if an SN on the routing path has not been selected to upgrade yet, we just include it in $V_{i,l}$.

Line 5 randomly selects an SN in $V_{i,l}$ to upgrade for the l-th vNF in SC_i, according to the probabilities $\{\vartheta_{i,l}^v, v \in V_{i,l}\}$ in the solution X^*. After the SN has been selected, we update the values of $\{\vartheta_{i,l}^v, v \in V_{i,l}\}$ accordingly, *i.e.*, rounding the real values within $[0, 1]$ to boolean ones. Then, we insert the selected SN in the set of SNs to upgrade V', if it is not already in (*Line* 6). *Line* 7 updates the upgrade cost Ω' to include the equipment cost of upgrading the selected SN, and it also updates the resource usages in the SNT by assuming that the vNF is redeployed on the selected SN. An LP can be solved in polynomial-time, and the time complexity of *Lines* 2-9 in *Algorithm* 1 is $O(|R|^2 \cdot \max_i(N_i)^2)$, where $|R|$ denotes the number of existing vNF-SCs and N_i is the number of vNFs in SC_i. Therefore, *Algorithm* 1 is a polynomial-time algorithm. Meanwhile, according to [53], the approximation ratio of the LP-rounding approach in *Algorithm* 1 is upper-bounded by 3.25. To this end, we can see that *Algorithm* 1 is a polynomial-time approximation algorithm for the optimization in Eq. (24).

B. Phase II

Next, in *Phase* II, we remove the SNs in the set V' obtained by *Algorithm* 1 in iterations, until the budget constraint is satisfied. In each iteration, we first recalculate the mapping between the existing vNFs and the SNs in V', and then remove the SN on which the smallest number of existing vNFs will be redeployed. Specifically, the optimization can be formulated as

$$\begin{aligned} &\text{Maximum} \quad \sum_{v \in V'} \sum_{i,l} \gamma_{i,l}^{v}, \\ &\text{s.t.} \quad \text{Eqs. (11), (12), and (20)-(22),} \end{aligned} \tag{25}$$

where all the constraints only consider SNs in V', and the value of $\gamma_{i,l}^v$ satisfies the assumption used in *Algorithm* 1 (*i.e.*, for an existing vNF, we will only consider the SNs on the current routing path of its vNF-SC). Then, if we treat each existing vNF as an item and each SN in V' as a knapsack, the optimization in Eq. (25) can be transformed into the multiple knapsack problem with assignment restrictions and capacity constraints (MK-AR-CC), where the assignment restrictions are due to the fact that each vNF can only select its SN from a subset of SNs in V'. According to [26], MK-AR-CC can also be solved with a polynomial-time approximation algorithm.

Algorithm 2 shows how to solve the MK-AR-CC in *Phase* II by leveraging the procedure developed in [26]. *Lines* 1-2 are for the initialization, where we first relax the optimization in Eq. (25) to an LP, then solve it to get a solution Y^*, and finally build a feasible solution Y to the LP relaxation with Y^*. Here, Y has the property that the number of variables $\{\gamma_{v,l}^i\}$, which equal 1, is maximized. Then, for each upgraded SN $v \in V'$, we finalize the existing vNFs that should be embedded onto it (*Lines* 3-8). Specifically, this is done by building and solving

[1]Note that, all the vNF redeployment schemes obtained in the first step are only used for selecting the SNs to upgrade, and thus they are hypothetical. The actual vNF redeployment schemes will be determined in the second step.

[2]Note that, if a server connects directly to a switch on the routing path, we also consider it as an SN on the path.

Algorithm 2: Algorithm for optimization in Eq. (25)

Input: existing vNF-SCs $\{SC_i\}$, set of SNs to upgrade V'.

Output: mapping schemes between vNFs and SNs.

1 solve LP relaxation of Eq. (25) to get a solution Y^*;
2 build a feasible solution Y to the LP relaxation with Y^* so that the number of $\gamma_{i,l}^v = 1$ is maximized;
3 **for** *each SN* $v \in V'$ **do**
4 put all the existing vNFs, which have $\gamma_{i,l}^v = 1$ according to Y, in the item set $\mathcal{F}_v$;
5 select an existing vNF whose $\gamma_{i,l}^v$ is fractional according to Y with the Hungarian method, and include it in the item set $\mathcal{F}_v$;
6 treat the vNFs in $\mathcal{F}_v$ as items and SN v as the knapsack to construct a single knapsack problem;
7 solve the single knapsack problem with an FPTAS to finalize the vNFs whose $\gamma_{i,l}^v = 1$;
8 **end**
9 output the finalized $\{\gamma_{v,l}^i,\ \forall v \in V', i, l\}$ as an approximation solution of the original problem;

a single knapsack problem as follows. *Line* 4 first puts all the existing vNFs that have $\gamma_{i,l}^v = 1$ according to Y in the item set $\mathcal{F}_v$. Then, we leverage the Hungarian method [54] to select an existing vNF whose $\gamma_{i,l}^v$ is fractional according to Y, and include the vNF in $\mathcal{F}_v$ (*Line* 5). Next, the single knapsack problem is obtained by treating the vNFs in $\mathcal{F}_v$ as items and the upgraded SN v as the knapsack (*Line* 6). In *Line* 7, the knapsack problem is solved with a fully polynomial-time approximation scheme (FPTAS) [55]. Finally, *Line* 9 outputs the finalized $\{\gamma_{v,l}^i,\ \forall v \in V', i, l\}$ as an approximation solution of the optimization in Eq. (25).

According to [26], the approximation ratio of *Algorithm* 2 is $1 + \frac{2}{K+1} + \epsilon$, where K means that the size of each knapsack (*i.e.*, each upgraded SN) is at least K times larger than the largest item (*i.e.*, an existing vNF), which is assignable to the SN, and ϵ is an arbitrarily-small positive constant.

For *Phase* II, we run *Algorithm* 2 in each iteration to optimize the mapping between the existing vNFs and the SNs in V', remove the SN on which the least number of existing vNFs will be redeployed from V', and then move to the next iteration until the budget constraint is satisfied. The approximation ratio of *Algorithm* 2 in each iteration is $1 + \frac{2}{K+1} + \epsilon$, and there will be at most

$$\mathcal{N} = \frac{\Omega' - \Omega}{\min(\phi^S, \phi^N)}, \tag{26}$$

iterations, where Ω' is the upgrade cost obtained by *Algorithm* 1, and ϕ^S and ϕ^N are the costs of a PDP-SW and a SmartNIC, respectively. Therefore, the approximation ratio of the problem-solving in *Phase* II will be

$$\eta \leq \left(1 + \frac{2}{K+1} + \epsilon\right)^{\mathcal{N}}. \tag{27}$$

C. Overall Procedure

The overall procedure to solve the optimization in the first step is shown in *Algorithm* 3. In *Line* 1, we relax the budget constraint and solve the SN selection problem with *Algorithm* 1 to get an intermediate set of SNs to upgrade (V'). Then, we use *Algorithm* 2 to determine the mapping between the existing vNFs and the SNs to upgrade and remove SNs from V' in iterations, until the budget constraint is satisfied (*Lines* 2-6). As both of *Algorithms* 1 and 2 are polynomial-time algorithms, *Algorithm* 3 runs in polynomial-time too.

Algorithm 3: Overall procedure for the first step

Input: existing vNF-SCs $\{SC_i\}$, SNT $G(V, E)$, budget of service upgrade Ω.

Output: set of SNs to update V'.

1 apply *Algorithm* 1 to get V' and Ω';
2 **while** $\Omega' > \Omega$ **do**
3 delete the SN v on which the least number of existing vNFs will be redeployed from V';
4 update Ω' to remove the upgrade cost of SN v;
5 apply *Algorithm* 2 to update the mapping between existing vNFs and SNs in V';
6 **end**

V. Second-Step of Two-Step Algorithm: Redeploying vNF-SCs in Upgraded SNT

After determining the SNs to upgrade in the first step, the second step optimizes the redeployment schemes of the existing vNF-SCs in the upgraded SNT. Specifically, for the second step, the variables $\{\iota_v\}$ in the overall ILP in Section III-A become pre-known parameters (*i.e.*, according to V'), and then the optimization can be formulated as

$$\begin{aligned} &\text{Maximize} \quad \sum_i (\varpi_i^{out} - \varpi_i^{in}), \\ &\text{s.t.} \quad \text{Eqs. (2), (4)-(6), and (8)-(16).} \end{aligned} \tag{28}$$

In the following, we leverage LP relaxation and randomized rounding [27] to design a polynomial-time approximation algorithm for the optimization in Eq. (28).

The procedure of the approximation algorithm is shown in *Algorithm* 4. We first relax all the boolean variables in the optimization in Eq. (28) to real ones within $[0, 1]$ to get an LP, and tighten the constraints in Eqs. (8) and (9) (*Line* 1). Here, the constraints are tightened to expedite the running of *Algorithm* 4. Specifically, it makes the resource constraints in Eqs. (8) and (9) tighter with preset ratios. For instance, the memory resource constraint in Eq. (8) is tighten with κ as

$$\begin{cases} \hat{C}_v^S = C_v^S \cdot (1 - \kappa) \\ \hat{C}_v^N = C_v^N \cdot (1 - \kappa), \quad \forall v, \\ \hat{C}_v^V = C_v^V \cdot (1 - \kappa) \end{cases} \tag{29}$$

where $\hat{C}_v^S$, $\hat{C}_v^N$, and $\hat{C}_v^V$ are the corresponding memory spaces considered in the LP. Similarly, by replacing C_v^S, C_v^N, C_v^V and κ in Eq. (29) with b_m^S, b_m^N, b_m^V and λ, respectively, we can tighten the constraint in Eq. (9) with λ. The LP is then

solved in *Line* 2 to obtain an objective T_{LP}, which provides an upper-bound on the objective of the original ILP in Eq. (28).

Algorithm 4: Algorithm for the second step

Input: $G(V, E)$, $\{SC_i\}$, set of upgraded SNs V'.

1 relax ILP in Eq. (28) to an LP, and tighten constraints in Eqs. (8) and (9) with ratios κ and λ, respectively;
2 solve the LP and get an objective T_{LP};
3 **for** *each* $j \in [1, \mathcal{Q}]$ **do**
4 $\mathcal{S} = \emptyset$, and initialize $p, q \in (0, 1)$ randomly;
5 **for** *each* $SC_i \in R$ **do**
6 **for** *the* l*-th vNF in* SC_i **do**
7 $F = 0$;
8 **while** $F = 0$ **do**
9 $p_1^{v,l} = 0$, $p_2^{v,l} = 0$;
10 **for** *each SN* $v \in V$ **do**
11 $p_2^{v,l} = p_2^{v,l} + \varphi_{v,l}^{out,i}$;
12 **if** $p_1^{v,l} < p \leq p_2^{v,l}$ **then**
13 $q_1^{p,n} = 0$, $q_2^{p,n} = 0$;
14 **for** *each* $\psi_{p,n}^{out,i}$ *related to* v **do**
15 $q_2^{p,n} = q_2^{p,n} + \psi_{p,n}^{out,i}$;
16 **if** $q_1^{p,n} < q \leq q_2^{p,n}$ **then**
17 $\varphi_{v,l}^{out,i} = 1$, $\psi_{p,n}^{out,i} = 1$;
18 $\{\varphi_{v,l}^{out,i}, \psi_{p,n}^{out,i}\} \rightarrow \mathcal{S}$;
19 $F = 1$, **break**;
20 **end**
21 $q_1^{p,n} = q_2^{p,n}$;
22 **end**
23 **if** $F = 1$ **then**
24 **break**;
25 **end**
26 $p_1^{v,l} = p_2^{v,l}$;
27 **end**
28 **end**
29 **end**
30 **end**
31 **end**
32 get $\{\vartheta_{i,l}^v, \tau_i^{out}, \varpi_i^{out}, y_{i,l}^v, z_{i,n}^p\}$ with $\{\varphi_{v,l}^{out,i}, \psi_{p,n}^{out,i}\}$ in $\mathcal{S}$;
33 put all variables in $\mathcal{S}$ and objective in T;
34 **if** $\mathcal{S}$ *is a feasible solution of Eq.* (28) **then**
35 **if** $T \geq \xi \cdot T_{LP}$ **then**
36 **break**;
37 **end**
38 **end**
39 **end**

Next, the for-loop that covers *Lines* 3-39 accomplishes the randomized rounding, where the largest number of iterations ($\mathcal{Q}$) is preset empirically [27]. In each iteration, *Line* 4 initializes the solution $\mathcal{S}$ and the random numbers p and q that will be used in the randomized rounding. Then, the two for-loops determine the integer values of all the variables $\{\varphi_{v,l}^{out,i}, \psi_{p,n}^{out,i}\}$ that are related to each existing vNF (*Lines* 5-31), with the standard procedure of randomized rounding.

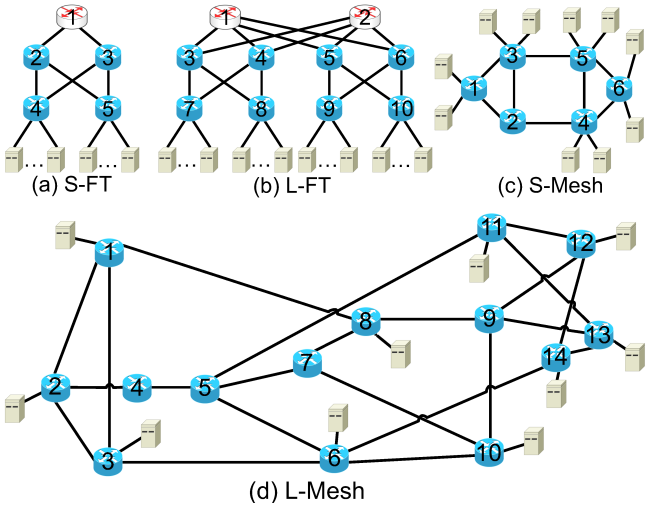


Fig. 2. Topologies used in simulations.

Finally, when all the values of $\varphi_{v,l}^{out,i}, \psi_{p,n}^{out,i}$ have been obtained, we calculate the values of the remaining variables with them, obtain an objective T with the variables, and insert all the variables in $\mathcal{S}$ (*Lines* 32-33). Here, for an SN v that is a server and has been equipped a SmartNIC in the upgrade, we deploy vNFs on the SmartNIC greedily. Specifically, if we can determine that the l-th vNF in SC_i is deployed on such an SN v after the upgrade, the vNF will be embedded on the SmartNIC there (*i.e.*, $\vartheta_{i,l}^v = 1$), as long as it has enough resources. *Line* 34 validates all the constraints in Eq. (28) with $\mathcal{S}$ to determine whether $\mathcal{S}$ represents a feasible solution. If yes, we check whether it satisfies the preset ratio ξ in *Line* 35. If yes, we get a qualified solution to the ILP in Eq. (28), and the iterations can be ended. Otherwise, the algorithm proceeds to the next iteration. Similar as $\mathcal{Q}$, the ratio ξ is preset empirically [27]. We use $\mathcal{Q}$ and ξ to adjust the tradeoff between the algorithm's time complexity and approximation ratio.

We can easily verify that for the maximization in Eq. (28), the approximation ratio of *Algorithm* 4 is at least ξ. Specifically, the objective obtained by solving the LP in *Line* 2 provides an upper-bound on the optimal objective of the original problem (T_{ILP}), and as T is the objective of a feasible solution to the original problem, it provides a lower-bound. Then, the approximation ratio of *Algorithm* 4 is obtained as

$$\eta = \frac{T}{T_{\text{ILP}}} \geq \frac{T}{T_{\text{LP}}} \geq \xi, \tag{30}$$

Meanwhile, we would like to point out that according to the principle of LP relaxation and randomized rounding and the well-known Chernoff-Bound [56], the probability of *Algorithm* 4 finding a qualified feasible solution approaches to 1, as long as $\mathcal{Q}$ and ξ are properly set (*e.g.*, the simulations in Section VI have $\mathcal{Q} = 10$ and $\xi = 0.75$). Because the time complexity of *Lines* 3-39 is $O(\mathcal{Q} \cdot |R|^2 \cdot \max_i(N_i) \cdot \max_i(N_i + 1) \cdot |V| \cdot |P|)$, *Algorithm* 4 is a polynomial-time approximation algorithm.

VI. Performance Evaluations

In this section, we conduct numerical simulations to evaluate the performance of our proposed algorithm.

A. Simulation Setup

In order to verify that our proposed algorithm is generic enough to work well with various SNTs, the simulations consider six different SNT topologies. Each topology contains two types of SNs, which are switches and servers, respectively. Four of the topologies are shown in Fig. 2, where the topologies in Figs. 2(a) and 2(b) are tree-type ones in different sizes for DCNs, and those in Figs. 2(c) and 2(d) are mesh topologies. The remaining two are large random topologies (RTs) with 30 and 45 switches, respectively. To generate each RT, we first connect the switches randomly with the GT-ITM tool in [57] using a connectivity of 0.2, and then attach servers to each switch randomly. Table I explains the setting of each SNT topology. Note that, we refer to the tree-type topologies in Figs. 2(a) and 2(b) as fat-trees (FTs), and S-FT and L-FT are the abbreviations for "small FT" and "large FT", respectively.

TABLE I
SETTINGS OF SNT TOPOLOGIES IN SIMULATIONS

Topology	S-FT	L-FT	S-Mesh	L-Mesh	RT-1	RT-2
# of Switches	5	10	6	14	45	30
# of Servers	10	60	10	60	45	60
Total SNs	15	70	16	74	90	90

The simulations consider $|M| = 4$ types of vNFs. To ensure that our simulations can represent the practical cases, we select the following parameters either according to the analysis of real-world network systems [58] or based on the observation in our own experiments [19]. The memory usage of a vNF is within $[20, 40]$ units [59–61], and its processing latency ranges within $[150, 300]$ μs before the service upgrade (*i.e.*, when the vNF runs on a VM). Each vNF-SC has its number of vNFs randomly selected from $[2, 4]$, its bandwidth demand is uniformly distributed within $[25, 50]$ Mbps, and its requirement on E2E latency is randomly selected from $\{200, 600, 1000\}$ μs with the probabilities of $\{0.3, 0.4, 0.3\}$, respectively.

We set the processing capacity of a vNF as $b_m^S = 100$ Gbps, $b_m^N = 10$ Gbps, and $b_m^V = 1$ Gbps, when it is deployed on a PDP-SW/SmartNIC/VM, respectively. We set the reduction on processing latency brought by a service upgrade as $\delta_m^S \in [75, 150]$ μs (migrating a vNF from a VM to a PDP-SW) and $\delta_m^N \in [40, 75]$ μs (migrating a vNF from a VM to a SmartNIC). The propagation delay of each SL is assumed to be 1 μs, and the memory space on an SN is set as $C_v^S = 200$, $C_v^N = 500$, and $C_v^V = 800$ units, for a PDP-SW/SmartNIC/server, respectively. The unit-costs of PDP-SWs and SmartNICs are initially set as $\phi^S = 30$ and $\phi^N = 10$ units, respectively, and we will change the ratio between them in Section VI-C to check their effects on the performance of our algorithm.

In addition to the overall ILP and the proposed two-step algorithm (TSA) that integrates *Algorithms* 3 and 4, the simulations also consider the two heuristic algorithms discussed in [24]. The first one is forwarding tree based algorithm (FTA), which tackles the service upgrade by first merging existing vNF-SCs to build vNF forwarding trees (vNF-FTs), then expanding each vNF-FT to a mapping tree (MT) according to the SNT's topology, and finally determining how to upgrade SNs and redeploy vNF-SCs based on the MTs. The second one is a simple greedy-based algorithm (NFTA), which first upgrades SNs in descending order of their resource usages, and then tries to redeploy the vNFs in each vNF-SC on the first upgraded SN that is available on the routing path of the vNF-SC. The simulations are carried out on a Ubuntu server with 4.0 GHz Intel Core i7-7400K CPU and 16 GB memory, and the software environment is MATLAB 2017b with GLPK toolbox and Gurobi v9.1.0. To ensure sufficient statistical accuracy, we average the results from 20 independent runs to get each data point in the simulations.

B. Small-Scale Simulations

We first use the two small-scale SNT topologies in Figs. 2(a) and 2(c) to compare the performance of all the algorithms (including the overall ILP). The service upgrade is allocated with different budgets ($\Omega = \{150, 300\}$ units). We use the QoS improvement in Eq. (1) and the total latency reduction ($\sum_i (\tau_i^{in} - \tau_i^{out})$) to quantify the performance improvement brought by the service upgrade.

Tables II and III show the simulation results. As expected, the overall ILP always provides the best service upgrade schemes to achieve the largest QoS improvement and longest latency reduction among the algorithms. Meanwhile, our proposed TSA can approximate the optimal solutions from the overall ILP well, and its performance is always much better than the two heuristics in [24] (FTA and NFTA). The good approximation achieved by TSA can be further verified by the results in Table III, which lists the SNs chosen by the algorithms to upgrade in randomly-selected simulation runs. It can be seen that the overall ILP and TSA select similar SNs to upgrade in different simulation scenarios. Meanwhile, we can see that even though the overall ILP and TSA can select exactly the same SNs to upgrade in Table III, their corresponding performance on QoS improvement and latency reduction in Table II is still different. This is because the two algorithms reconfigure active vNF-SCs with the SmartNICs and PDP-SWs on the upgraded SNs differently.

In the meantime, the results in Tables II and III also confirm the universality of TSA. Specifically, the performance gaps between TSA and the overall ILP are generally constant when different SNT topologies are used, while when the S-Mesh in Fig. 2(c) is used, the performance degradations of FTA and NFTA related to the overall ILP and TSA are generally larger. This is because FTA and NFTA were designed based on the assumption that the SNT is a DCN and uses a tree-type topology, which is not the case for our TSA.

C. Large-Scale Simulations

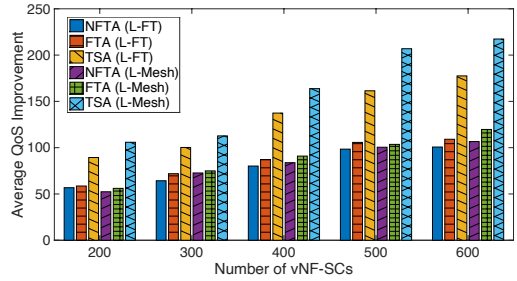
We then further evaluate the performance of TSA, FTA and NFTA with large-scale SNT topologies (*i.e.*, L-FT, L-Mesh, RT-1, and RT-2). Note that, due to its complexity, solving the overall ILP directly has become intractable for these cases, which means that it cannot finish running within tens of hours and can easily use up the memory on our server for the problem-solving. The results in Fig. 3 compare the algorithms'

TABLE II
PERFORMANCE COMPARISON OF ALGORITHMS IN SMALL-SCALE SIMULATIONS

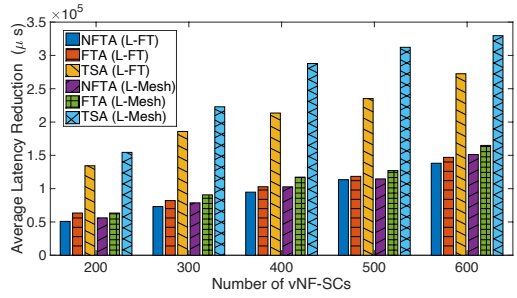
Number of vNF-SCs	Budget for Network Upgrade Ω	Topology	QoS Improvement $\sum_i (\varpi_i^{out} - \varpi_i^{in})$				Latency Reduction $\sum_i (\tau_i^{in} - \tau_i^{out})$				Running Time (Seconds)			
			ILP	TSA	FTA	NFTA	ILP	TSA	FTA	NFTA	ILP	TSA	FTA	NFTA
50	150	S-FT	16	13	9	8	6,818	6,510	4,116	3,840	223.89	19.16	0.15	0.01
50	300	S-FT	17	16	13	11	6,932	6,864	6,494	6,046	231.59	6.01	0.13	0.01
100	150	S-FT	32	27	18	17	13,064	11,066	7,930	7,240	555.23	47.18	0.22	0.02
100	300	S-FT	36	33	23	21	13,142	12,044	9,554	8,478	460.03	12.75	0.21	0.02
50	150	S-Mesh	15	13	8	7	6,620	6,048	3,830	3,396	232.93	17.01	2.16	0.01
50	300	S-Mesh	17	17	12	9	7,358	7,208	5,802	4,244	229.76	6.91	2.29	0.01
100	150	S-Mesh	30	26	14	12	12,824	11,018	6,826	6,256	599.49	53.98	4.47	0.02
100	300	S-Mesh	37	34	18	15	13,712	12,828	8,722	7,484	494.99	13.79	4.43	0.02

TABLE III
CHOICES OF SNS TO UPGRADE IN SMALL-SCALE SIMULATIONS

Number of vNF-SCs	Budget for Network Upgrade Ω	Topology	Set of Upgraded SNs			
			ILP	TSA	FTA	NFTA
50	150	S-FT	$[1, 2, 6, 8-12, 14]$	$[1, 5, 7-10, 12-14]$	$[1, 5-10, 13, 14]$	$[5-12]$
50	300	S-FT	$[1-11, 13, 15]$	$[1-15]$	$[1, 5-14]$	$[5-14]$
100	150	S-FT	$[3, 6-15]$	$[1, 2, 5, 8-10, 12-14]$	$[5-7, 9, 11-15]$	$[5-8, 10-15]$
100	300	S-FT	$[1-15]$	$[1-15]$	$[3-10, 12-14]$	$[5-15]$
50	150	S-Mesh	$[2, 6-8, 10, 11, 14, 16]$	$[2, 5, 6, 8, 10, 11, 14]$	$[4, 6, 8-14]$	$[2-4, 8-12]$
50	300	S-Mesh	$[1-14, 16]$	$[1-12, 14-16]$	$[1-4, 8-14, 16]$	$[2-4, 8-14]$
100	150	S-Mesh	$[2, 4, 8-16]$	$[2, 5, 6, 8, 10, 12, 16]$	$[1, 2, 7, 9-14]$	$[2-5, 8, 10, 14]$
100	300	S-Mesh	$[1-16]$	$[1-16]$	$[1-7, 9-15]$	$[2-5, 7-16]$



(a) Average QoS improvement



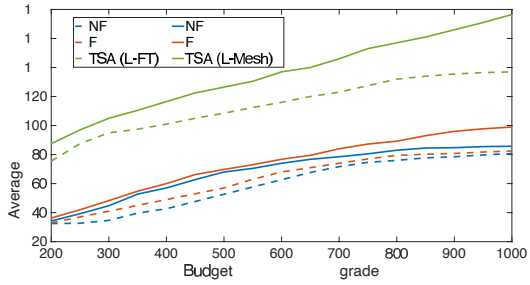
(b) Average latency reduction

Fig. 3. Results of large-scale simulations with a fixed budget ($\Omega = 800$).

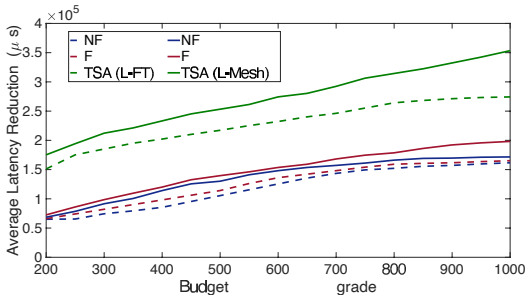
performance when the budget of the service upgrade is fixed as $\Omega = 800$ units. We observe that TSA always provides the highest QoS improvement and the largest latency reduction among the algorithms, and its performance improvements over FTA and NFTA are more significant when the L-Mesh in Fig. 2(d) is used. This verifies the superiority and university of TSA for large-scale SNT topologies. Then, we check how the algorithms' performance changes, when the budget Ω increases but the number of existing vNF-SCs is fixed as $|R| = 400$. Fig. 4 indicates that TSA still outperforms FTA and NFTA significantly, and the performance gaps between TSA and FTA/NFTA actually increase when the budget increases.

Next, we evaluate the algorithms with different unit-costs of PDP-SWs and SmartNICs. Specifically, we fix the unit-cost of SmartNICs as $\phi^N = 10$ units, but change the ratio of $\frac{\phi^S}{\phi^N}$ to be $5:1$ and $7:1$. Here, we use the L-Mesh in Fig. 2(d) and fix the number of existing vNF-SCs as $|R| = 400$. The results in Fig. 5 suggest that TSA still performs significantly better than the benchmarks, but its performance gaps over them become smaller when the unit-cost of PDP-SWs increases. This is because when the PDP-SWs become more expensive, TSA will have to use less of them in the service upgrade, while the latency reduction achieved by upgrading with a SmartNIC is shorter than that achieved by upgrading with a PDP-SW.

Finally, we test the algorithms with large RTs to confirm the effectiveness of TSA. The simulations fix the budget as $\Omega = 800$ units and increase the number of existing vNF-SCs. The results in Fig. 6 indicate that TSA still always performs the best. Meanwhile, when the budget is relatively large, the more switches that we have in the SNT, the larger performance improvement that the service upgrade can achieve. This is still due to the fact that upgrading a vNF from using a VM to using

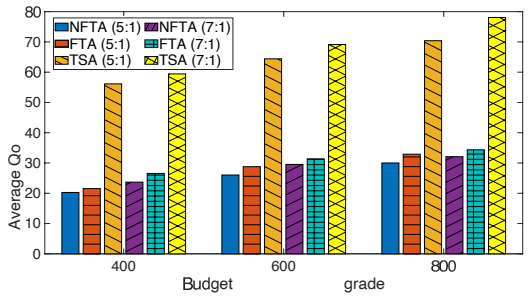


(a) Average QoS improvement

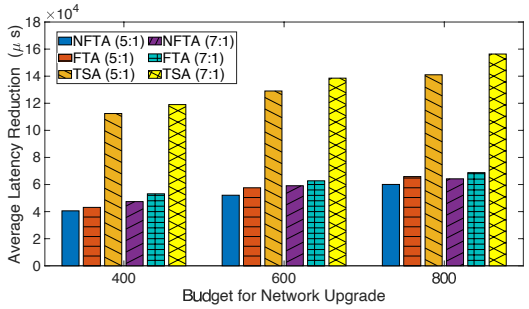


(b) Average latency reduction

Fig. 4. Results of large-scale simulations with $|R| = 400$ existing vNF-SCs.



(a) Average QoS improvement



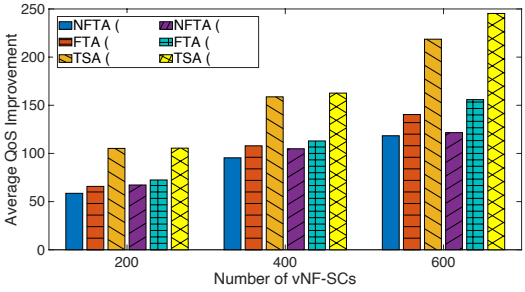
(b) Average latency reduction

Fig. 5. Results of large-scale simulations for different unit-costs of PDP-SWs and SmartNICs ($|R| = 400$ and using L-Mesh).

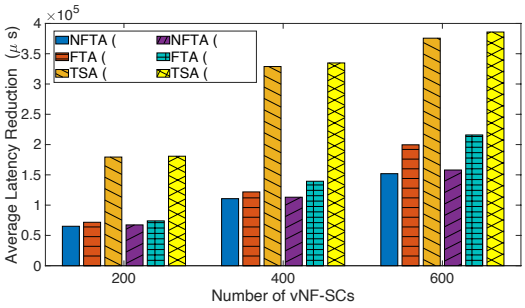
a PDP-SW brings in a larger latency reduction.

VII. Conclusion

In this paper, we studied how to optimize the service upgrade of vNF-SCs by leveraging the heterogeneous NFV platforms that include PDP-SWs and SmartNICs. The service upgrade was divided into two steps, *i.e.*, selecting servers/switches in the SNT to upgrade under a fixed budget, and redeploying the existing vNF-SCs in the updated SNT to



(a) Average QoS improvement



(b) Average latency reduction

Fig. 6. Results of large-scale simulations with large RTs ($\Omega = 800$).

maximize the QoS improvement on latency reductions. We first formulated two correlated optimizations for the steps, and then proposed polynomial-time approximation algorithms to solve the optimizations. Extensive simulations verified that our proposed algorithm outperforms the existing benchmarks in various network scenarios, and achieves better tradeoff between performance and time-efficiency.

Acknowledgments

This work was supported in part by the NSFC project 61871357, SPR Program of CAS (XDC02070300), and Fundamental Funds for Central Universities (WK3500000006).

References

[1] Cisco Visual Networking Index, 2017-2022. [Online]. Available: https://www.cisco.com/c/en/us/solutions/collateral/service-provider/visual-networking-index-vni/white-paper-c11-741490.html.

[2] P. Lu *et al.*, "Highly efficient data migration and backup for Big Data applications in elastic optical inter-data-center networks," *IEEE Netw.*, vol. 29, pp. 36–42, Sept./Oct. 2015.

[3] H. Lu, M. Zhang, Y. Gui, and J. Liu, "QoE-driven multi-user video transmission over SM-NOMA integrated systems," *IEEE J. Sel. Areas Commun.*, vol. 37, pp. 2102–2116, Sept. 2019.

[4] Z. Zhu, W. Lu, L. Zhang, and N. Ansari, "Dynamic service provisioning in elastic optical networks with hybrid single-/multi-path routing," *J. Lightw. Technol.*, vol. 31, pp. 15–22, Jan. 2013.

[5] H. Wu and H. Lu, "Delay and power tradeoff with consideration of caching capabilities in dense wireless networks," *IEEE Trans. Wireless Commun.*, vol. 18, pp. 5011–5025, Oct. 2019.

[6] L. Gong *et al.*, "Efficient resource allocation for all-optical multicasting over spectrum-sliced elastic optical networks," *J. Opt. Commun. Netw.*, vol. 5, pp. 836–847, Aug. 2013.

[7] Y. Gui, H. Lu, F. Wu, and C. Chen, "Robust video broadcast for users with heterogeneous resolution in mobile networks," *IEEE Trans. Mobile Comput., in Press*, 2020.

[8] Z. Zhu *et al.*, "Demonstration of cooperative resource allocation in an OpenFlow-controlled multidomain and multinational SD-EON testbed," *J. Lightw. Technol.*, vol. 33, pp. 1508–1514, Apr. 2015.

[9] Y. Yin *et al.*, "Spectral and spatial 2D fragmentation-aware routing and spectrum assignment algorithms in elastic optical networks," *IEEE J. Opt. Commun. Netw.*, vol. 5, pp. A100–A106, Oct. 2013.

[10] SmartNICs from Intel. [Online]. Available: https://www.intel.com/content/www/us/en/products/network-io/smartnic.html.

[11] Tofino switch. [Online]. Available: https://www.barefootnetworks.com/products/brief-tofino/.

[12] N. Zilberman, "In-network computing," Apr. 2019. [Online]. Available: https://www.sigarch.org/in-network-computing-draft/.

[13] S. Grant, A. Yelam, M. Bland, and A. Snoeren, "SmartNIC performance isolation with FairNIC: Programmable networking for the cloud," in *Proc. of ACM SIGCOMM 2020*, pp. 681–693, Jul. 2020.

[14] K. Zhang, D. Zhuo, and A. Krishnamurthy, "Gallium: Automated software middlebox offloading to programmable switches," in *Proc. of ACM SIGCOMM 2020*, pp. 283–295, Jul. 2020.

[15] "Network functions virtualization (NFV)," Tech. Rep., Oct. 2014. [Online]. Available: https://portal.etsi.org/Portals/0/TBpages/NFV/Docs/NFV_White_Paper3.pdf.

[16] K. Han *et al.*, "Application-driven end-to-end slicing: When wireless network virtualization orchestrates with NFV-based mobile edge computing," *IEEE Access*, vol. 6, pp. 26 567–26 577, 2018.

[17] C. Sun, J. Bi, Z. Zheng, and H. Hu, "HYPER: A hybrid high-performance framework for network function virtualization," *IEEE J. Sel. Areas Commun.*, vol. 35, pp. 2490–2500, Nov. 2017.

[18] L. Cui *et al.*, "Enabling heterogeneous network function chaining," *IEEE Trans. Parallel Distrib. Syst.*, vol. 30, pp. 842–854, Sept. 2019.

[19] L. Dong *et al.*, "On application-aware and on-demand service composition in heterogenous NFV environments," in *Proc. of GLOBECOM 2019*, pp. 1–6, Dec. 2019.

[20] L. Dong, N. L. S. da Fonseca, and Z. Zhu, "Application-driven provisioning of service function chains over heterogeneous NFV platforms," *IEEE Trans. Netw. Serv. Manag., in Press*, 2020.

[21] M. Carvalho, W. Cirne, F. Brasileiro, and J. Wilkes, "Long-term SLOs for reclaimed cloud computing resources," in *Proc. of SOCC 2014*, pp. 1–13, Nov. 2014.

[22] C. Reiss *et al.*, "Heterogeneity and dynamicity of clouds at scale: Google trace analysis," in *Proc. of SOCC 2012*, pp. 1–13, Oct. 2012.

[23] N. Poggi *et al.*, "Characterization of workload and resource consumption for an online travel and booking site," in *Proc. of IISWC 2010*, pp. 1–10, Dec. 2010.

[24] Y. Xue and Z. Zhu, "Leveraging heterogeneous NFV platforms to upgrade service function chains in DCNs," in *Proc. of NetSoft 2021*, pp. 1–5, Jun. 2021.

[25] Facility location problem. [Online]. Available: https://en.wikipedia.org/wiki/Facility_location_problem.

[26] S. Miyazaki, N. Morimot, and Y. Okabe, "Approximability of two variants of multiple knapsack problems," in *Proc. of CIAC 2015*, pp. 365–376, May 2015.

[27] P. Raghavan and C. Tompson, "Randomized rounding: a technique for provably good algorithms and algorithmic proofs," *Combinatorica*, vol. 7, pp. 365–374, Dec. 1987.

[28] W. Fang *et al.*, "Joint spectrum and IT resource allocation for efficient vNF service chaining in inter-datacenter elastic optical networks," *IEEE Commun. Lett.*, vol. 20, pp. 1539–1542, Aug. 2016.

[29] Q. Sun, P. Lu, W. Lu, and Z. Zhu, "Forecast-assisted NFV service chain deployment based on affiliation-aware vNF placement," in *Proc. of GLOBECOM 2016*, pp. 1–6, Dec. 2016.

[30] M. Zeng, W. Fang, and Z. Zhu, "Orchestrating tree-type VNF forwarding graphs in inter-DC elastic optical networks," *J. Lightw. Technol.*, vol. 34, pp. 3330–3341, Jul. 2016.

[31] Y. Wang, P. Lu, W. Lu, and Z. Zhu, "Cost-efficient virtual network function graph (vNFG) provisioning in multidomain elastic optical networks," *J. Lightw. Technol.*, vol. 35, pp. 2712–2723, Jul. 2017.

[32] L. Gong and Z. Zhu, "Virtual optical network embedding (VONE) over elastic optical networks," *J. Lightw. Technol.*, vol. 32, pp. 450–460, Feb. 2014.

[33] H. Jiang, Y. Wang, L. Gong, and Z. Zhu, "Availability-aware survivable virtual network embedding (A-SVNE) in optical datacenter networks," *J. Opt. Commun. Netw.*, vol. 7, pp. 1160–1171, Dec. 2015.

[34] L. Gong, H. Jiang, Y. Wang, and Z. Zhu, "Novel location-constrained virtual network embedding (LC-VNE) algorithms towards integrated node and link mapping," *IEEE/ACM Trans. Netw.*, vol. 24, pp. 3648–3661, Dec. 2016.

[35] R. Mijumbi *et al.*, "Network function virtualization: State-of-the-art and research challenges," *IEEE Commun. Surveys Tuts.*, vol. 18, pp. 236–262, First Quarter 2016.

[36] D. Bhamare, R. Jain, M. Samaka, and A. Erbad, "A survey on service function chaining," *J. Netw. Comput. Appl.*, vol. 75, pp. 138–155, Nov. 2016.

[37] "IETF service function chaining (SFC)," Tech. Rep., Apr. 2014. [Online]. Available: https://datatracker.ietf.org/wg/sfc/charter.

[38] J. Liu *et al.*, "On dynamic service function chain deployment and readjustment," *IEEE Trans. Netw. Serv. Manag.*, vol. 14, pp. 543–553, Sept. 2017.

[39] V. Eramo, E. Miucci, M. Ammar, and F. 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, pp. 2008–2025, Mar. 2017.

[40] K. Noghani, A. Kassler, and J. Taheri, "On the cost-optimality trade-off for service function chain reconfiguration," in *Proc. of CloudNet 2019*, pp. 1–6, Nov. 2019.

[41] Z. Luo and C. Wu, "An online algorithm for VNF service chain scaling in datacenters," *IEEE/ACM Trans. Netw.*, vol. 28, pp. 1061–1073, Mar. 2020.

[42] Y. Hu and T. Li, "Enabling efficient network service function chain deployment on heterogeneous server platform," in *Proc. of HPCA 2018*, pp. 27–39, Feb. 2018.

[43] K. Poularakis, G. Iosifidis, G. Smaragdakis, and L. Tassiulas, "Optimizing gradual SDN upgrades in ISP networks," *IEEE/ACM Trans. Netw.*, vol. 27, pp. 288–301, Jan. 2019.

[44] L. Wang *et al.*, "Simplifying network updates in SDN and NFV networks using GUM," in *Proc. of ICCCN 2018*, pp. 1–9, Jul. 2018.

[45] T. Hsieh, C. Chuang, S. Chou, and A. Pang, "Traffic-aware network update in software-defined NFV networks," in *Proc. of WPMC 2020*, pp. 1–6, Oct. 2020.

[46] X. Jin *et al.*, "NetCache: Balancing key-value stores with fast in-network caching," in *Proc. of SOSP 2017*, pp. 121–136, Oct. 2017.

[47] B. Li *et al.*, "KV-Direct: High-performance in-memory key-value store with programmable NIC," in *Proc. of SOSP 2017*, pp. 137–152, Oct. 2017.

[48] G. Li *et al.*, "NETHCF: Enabling line-rate and adaptive spoofed IP traffic filtering," in *Proc. of ICNP 2019*, pp. 1–12, Oct. 2019.

[49] N. Pitaev, M. Falkner, A. Leivadeas, and I. Lambadaris, "Characterizing the performance of concurrent virtualized network functions with OVS-DPDK, FD.IO VPP and SR-IOV," in *Proc. of ICPE 2018*, pp. 285–292, Mar. 2018.

[50] M. Liu *et al.*, "Offloading distributed applications onto SmartNICs using iPipe," in *Proc. of ACM SIGCOMM 2019*, pp. 318–333, Aug. 2019.

[51] D. Kim *et al.*, "TEA: Enabling state-intensive network functions on programmable switches," in *Proc. of ACM SIGCOMM 2020*, pp. 90–106, Jul. 2020.

[52] Y. Bejerano, "Efficient integration of multihop wireless and wired networks with QoS constraints," *IEEE/ACM Trans. Netw.*, vol. 12, pp. 1064–1078, Dec. 2004.

[53] M. Charikar and S. Li, "A dependent LP-rounding approach for the k-median problem," in *Proc. of ICALP 2012*, pp. 194–205, Jul. 2012.

[54] H. Kuhn, "The Hungarian method for the assignment problem," *Nav. Res. Logist. Q.*, vol. 2, pp. 83–97, Mar. 1955.

[55] O. Ibarra and C. Kim, "Fast approximation algorithms for the knapsack and sum of subset problems," *J. ACM*, vol. 22, pp. 463–468, Oct. 1975.

[56] D. Dubhashi and A. Panconesi, "Concentration of measure for the analysis of randomized algorithms," in *Cambridge University Press*, 2009.

[57] E. Zegura, K. Calvert, and S. Bhattacharjee, "How to model an internetwork," in *Proc. INFOCOM 1996*, pp. 594–602, Mar. 1996.

[58] J. Sonchack, J. Aviv, E. Keller, and M. Smith, "Turboflow: Information rich flow record generation on commodity switches," in *Proc. of EuroSys 2018*, pp. 1–16, Apr. 2018.

[59] L. Jersak and T. Ferreto, "Performance-aware server consolidation with adjustable interference levels," in *Proc. of SAC 2016*, pp. 420–425, Apr. 2016.

[60] I. Pietri and R. Sakellariou, "Mapping virtual machines onto physical machines in cloud computing: A survey," *ACM Comput. Surv.*, vol. 49, pp. 1–30, Oct. 2016.

[61] M. Liu and T. Li, "Optimizing virtual machine consolidation performance on NUMA server architecture for cloud workloads," in *Proc. of ISCA 2014*, pp. 325–336, Jun. 2014.