Application-driven Provisioning of Service Function Chains over Heterogeneous NFV Platforms

Lu Dong, Nelson L. S. da Fonseca, and Zuqing Zhu, *Senior Member, IEEE*

***Abstract*—Although network function virtualization (NFV) has been proven to be beneficial in terms of equipment cost, service delivery flexibility, and time-to-market, most of the studies in this area only addressed homogeneous NFV platforms (*e.g.*, with virtual machines (VMs) only). In this work, we argue that by leveraging heterogeneous NFV platforms such as VMs, docker containers, and programmable hardware accelerators (*e.g.*, SmartNICs), one could achieve better flexibility and cost-effectiveness to support virtual network function service chains (vNF-SCs) with various quality-of-service (QoS) requirements. Therefore, we study application-driven provisioning of vNF-SCs over heterogeneous NFV platforms, and design a polynomial-time approximation algorithm to tackle the problem for near-optimal solutions. We first introduce a layered auxiliary graph (LAG) based approach to model the problem of vNF-SC provisioning, and then formulate a novel integer linear programming (ILP) model based on it. Specifically, the ILP model minimizes the total cost of vNF-SC deployment while ensuring that the QoS requirements of all the vNF-SCs are satisfied. To solve the ILP time-efficiently, we propose an approximation algorithm based on linear programming (LP) relaxation and randomized rounding. Extensive simulations confirm that with significantly improved time-efficiency, our proposed algorithm can provide near-optimal solutions whose gaps to the exact ones are bounded.**

***Index Terms*—Network function virtualization (NFV), Heterogeneous NFV platforms, Service function chaining, Approximation algorithm, Linear relaxation.**

I. Introduction

OVER the past decade, the Internet has gone through revolutionary changes to accommodate tremendous emerging applications [1, 2], for making our daily lives much more comfortable and convenient. Traditionally, service providers (SPs) rely on special-purpose middleboxes to support new applications, which recently becomes increasingly challenging because of the unbearable costs and maintenance complexity, and the long time-to-market. To address these challenges, SPs have switched to network function virtualization (NFV) [3, 4], which can realize network applications with virtual network functions (vNFs) running on general-purpose hardware/software platforms instead of using proprietary hardware [5, 6]. For instance, they can decompose network services into atomic network functions (*e.g.*, firewall and load-balancer), instantiate the network functions with vNFs, and steer application traffic through required vNFs in sequence to realize each network service (*i.e.*, vNF service chaining (vNF-SC) [7, 8]).

L. Dong 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).

N. Fonseca is with the Institute of Computing, State University of Campinas, Campinas, SP 13083-852, Brazil.

Manuscript received on August 7, 2020.

The success of cloud computing and network slicing in 5G and datacenter networks has promoted the idea of composing network services as vNF-SCs [6, 9]. To fully explore the benefits of vNF-SC, previous studies have considered both the algorithm design for its service provisioning [10–12] and the system implementation for orchestrating IT and bandwidth resources for its deployment in real networks [13, 14]. Nevertheless, most of the existing studies on vNF-SC overlooked the possibility of using heterogeneous NFV platforms to realize vNFs, and assumed that all the vNFs would be instantiated over the same type of platforms (*e.g.*, virtual machines (VMs)).

Note that, we can utilize various software/hardward platforms, *e.g.*, VMs, docker containers [15], and programmable hardware accelerators (*i.e.*, field programmable gate arrays (FPGAs) [16] and SmartNICs [17]), to realize vNFs. Here, VMs and docker containers can be deployed on commodity servers, while FPGA and SmartNICs are also general-purpose commodity hardware due to their programmability and commercial availability. Hence, realizing vNFs over these heterogeneous platforms will not violate the basic principle of NFV. Meanwhile, the unique features of the heterogeneous NFV platforms provide SPs better programmability and flexibility to support various quality-of-service (QoS) requirements simultaneously for vNF-SCs [18, 19]. For example, vNFs can be instantiated on docker containers with setup latencies less than one second [13], while those on SmartNICs can easily achieve over 10 Gbps traffic processing capacity [19].

Specifically, vNFs based on software platforms (*i.e.*, VMs and docker containers) have advantages in terms of cost, elasticity, and setup latency, while their traffic processing capacities and data processing latencies could be the bottleneck for high-throughput and ultra-low-latency services (*e.g.*, remote surgery). This is because software platforms have inherent performance overheads compared with hardware ones. On the other hand, even though FPGAs and SmartNICs ensure superior traffic processing performance, they have relatively high costs and need to be reprogrammed for deploying new vNFs (*i.e.*, long setup latency). As each application has its own QoS demands from clients as well as other unique requirements like setup latency and service flexibility from its SP, its provisioning based on a vNF-SC should be "application-driven", *i.e.*, we need to consider the aforementioned advantages and disadvantages of each NFV platform and serve the application's vNF-SC in the way that both its SP and clients can be satisfied. For instance, in the case where certain clients of vNF-SCs can move around and have stringent QoS demands for low latency (*e.g.*, the ultra-low latency scenario in 5G), the SP will have difficulty satisfying the QoS demands if it

deploys all the vNFs in software platforms. On the other hand, deploying all the vNFs in SmartNICs might not be a good solution either, because they are relatively expensive and can hardly be reprogrammed quickly to adapt to the movement of clients. With heterogeneous NFV platforms, the SP has more flexibility to handle such situations. Hence, it will be relevant to consider application-driven provisioning of vNF-SCs over heterogeneous NFV platforms that separately consider VMs, docker containers, and SmartNICs for vNF deployments.

In our previous work [19], we studied this problem preliminarily, mainly from the perspective of system implementation. Specifically, we laid out the network model, built a simple testbed to demonstrate the benefits of serving vNF-SCs over heterogeneous platforms, and formulated an integer linear programming (ILP) model based on the measurements to optimize the application-driven provisioning of vNF-SCs in terms of cost-effectiveness. However, we did not try to optimize the ILP's formulation or solve it time-efficiently. The ILP in [19] was only solved for small-scale problems (*i.e.*, serving at most 6 vNF-SCs in a six-node topology). This is because the basic problem of vNF-SC provisioning is $\mathcal{NP}$-hard [20], and the ILP becomes intractable for large-scale problems. Although time-efficient heuristics can be leveraged to find feasible solutions, they can hardly obtain near-optimal solutions whose performance gap to the optimal ones is guaranteed.

Motivated by the aforementioned dilemma, we, in this work, revisit the problem of application-driven provisioning of vNF-SCs over heterogeneous NFV platforms, from the perspectives of theoretical analysis and algorithm design. We first introduce a layered auxiliary graph (LAG) based scheme to model the problem with a compact ILP. Then, we optimize the ILP's formulation to improve its practicalness and optimization performance, such that the total cost of vNF-SC deployment can be minimized while the QoS requirements of all the vNF-SC requests are satisfied with carefully-chosen NFV platforms. Next, we design a polynomial-time approximation algorithm based on linear programming (LP) relaxation with randomized rounding, to solve the ILP's optimization time-efficiently for near-optimal solutions. Finally, we conduct extensive simulations to evaluate our proposal and verify its performance. The major contributions of this work are summarized as follows.

- We consider application-driven provisioning of vNF-SCs over heterogeneous NFV platforms and design an LAG-based approach to model the provisioning problem.
- We formulate a compact ILP model to solve the problem exactly, based on the LAG-based problem modeling.
- Based on the ILP model, we design a polynomial-time algorithm with LP relaxation and randomized rounding to approximate the exact solutions.
- We run extensive simulations to verify the performance of our proposed approximation algorithm.

The rest of this paper is organized as follows. Section II briefly reviews the related work. In Section III, we describe the application-driven vNF-SCs provisioning problem and explain the LAGs to model it. The novel ILP model is formulated in Section IV, and we design the approximation algorithm based on LP relaxation with randomized rounding in Section V. The performance evaluation is then presented in Section VI. Finally, Section VII summarizes the paper.

II. Related Work

Nowadays, NFV has gained intensive interests from both academia and industry and thus promoted active research and development activities. The technical documents for NFV standardization have been published in [21, 22] to explain the requirements and service frameworks of NFV and its typical use-cases, respectively. Depending on how the traffic flows are organized and routed over required vNFs, NFV can assist an SP to compose its network services in the forms of vNF-SCs [7, 23], vNF multicast trees [24], and generic vNF forwarding graphs [25]. It should be noted that the problem of NFV-based service composition is fundamentally different from the famous virtual network embedding (VNE) problem [26–29], according to the analysis in [24]. Specifically, NFV-based service composition can instantiate multiple vNFs, which belong to the same network service, on one substrate network element, while this changes the virtual topology and is normally not allowed in VNE. We focus on the provisioning of vNF-SCs over heterogeneous NFV platforms in this work, while more sophisticated service compositions with the same background will be considered in our future work. For the technical standard regarding vNF-SC, one can refer to [30].

Previously, many studies have been dedicated to optimizing the vNF-SC provisioning in different types of networks [12]. Using packet networks as substrate networks (SNTs), the studies in [31–34] have formulated optimization models and designed heuristic algorithms to tackle the problem. For example, the authors of [34] proposed an ILP model and a heuristic to optimize the vNF-SC provisioning across geographically-distributed clouds. Considering an optical network (*e.g.*, the fixed-grid wavelength-division multiplexing (WDM) network or the flexible-grid elastic optical network (EON) [35–37]) as the SNT, researchers have also studied how to orchestrate IT resources together with the spectra in fiber links for assembling vNF-SCs in datacenter interconnections [10, 38]. Note that, the unique consideration of provisioning vNF-SCs in an optical network is that the bandwidth resources are represented by discrete wavelength channels. Hence, the joint optimization of IT and bandwidth resource allocations becomes more complex than its counterpart in a packet network [39].

In addition to conventional optimization techniques, recent studies have also leveraged game theory [40, 41] and machine learning [8, 11] to optimize vNF-SC provisioning. However, all the studies mentioned above resorted to either non-polynomial-time algorithms that are not scalable or heuristics whose performance gaps to the exact ones are not bounded. More importantly, none of them has considered the provisioning of vNF-SCs over heterogeneous NFV platforms.

Researchers have also tried to design approximation algorithms that can solve the provisioning of vNF-SCs with guaranteed gaps to the exact solutions [9, 42–46]. Dietrich *et al.* [9] presented a holistic solution to the problem of network service embedding in multi-provider networks. Specifically, they decomposed the problem into two subproblems and designed

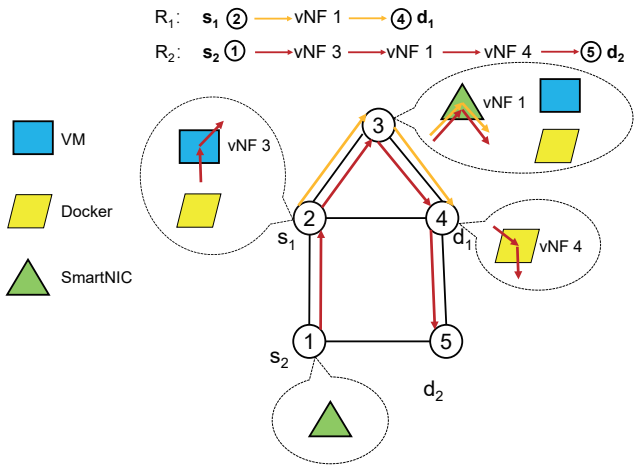


Fig. 1. Provisioning vNF-SCs over heterogeneous NFV platforms.

LP formulations to derive near-optimal solutions for both of them. The authors of [42] first modeled the problem of vNF placement as a combination of the facility location problem and the generalized assignment problem, and then designed an approximation algorithm that can provide near-optimal solutions with bi-criteria constant approximation guarantees. Nevertheless, their model did not address the flow routing for vNF-SCs. In [43], the researchers designed a system that can dynamically provision IT and bandwidth resources to vNF-SCs to provide timing guarantees. They considered a homogeneous NFV environment in datacenters, and designed an approximation algorithm to maximize the number of vNF-SC requests that can be provisioned successfully.

Sang *et al.* [44] designed two polynomial-time algorithms for vNF-SC provisioning and proved their approximation ratios, but they assumed that there was only one type of vNFs in the network, which is not practical for general cases. The study in [45] showed that vNF-SC provisioning can be mapped to an exponential number of min-cost flow problems, based on which it developed an approximation algorithm. In [46], the authors formulated a mixed integer linear programming (MILP) model and proposed a polynomial-time algorithm to maximize the acceptable flow rate under a budget on energy costs, but latency constraints were not considered. These studies also did not consider heterogeneous NFV platforms.

The heterogeneous NFV platforms considered in this work refer to the general-purpose hardware/software platforms, *e.g.*, VMs, docker-containers, and SmartNICs, which can be programmed to carry various vNFs. The network environment considered here is different from the hybrid environment in [47, 48], which consists of both special-purpose middleboxes and homogeneous NFV platforms (*e.g.*, VMs). More specifically, all the heterogeneous platforms in our NFV environment can support various types of vNFs, while each middlebox in the hybrid environment is only dedicated to a single network function. Therefore, in our case, the vNF-SC provisioning is much more flexible and thus more complex.

To the best of our knowledge, the only existing studies that have considered heterogeneous NFV platforms are the HYPER in [18] and our own study in [19]. In [18], HYPER was developed as a high-performance service framework based on OpenStack and ONetCard, for realizing vNF-SCs over heterogeneous NFV platforms. Although the authors did an excellent job on system implementation, they did not focus much on the algorithm design of application-driven provisioning of vNF-SCs. For instance, they simply modeled the vNF placement as a bin packing problem but ignored the affiliation among vNFs for a specific vNF-SC. Moreover, their algorithm did not try to optimize the flow routing for vNF-SCs.

This paper greatly extends our preliminary study in [19], by proposing a novel and compact ILP based on the LAG-based problem modeling and designing an approximation algorithm that can ensure near-optimal solutions. More specifically, in terms of the ILP formulation, we make three major improvements. Firstly, the ILP in [19] is a path-based one, which means that we need to pre-calculate K shortest paths between each node pair in the SNT. This, however, restricts the optimality of the ILP's solutions, especially when the SNT's topology is relatively large, and increasing the value of K cannot resolve the issue completely. Hence, this work designs the ILP as a link-based one, which means that all the feasible paths in the SNT can be checked to guarantee its optimality. Secondly, with the LAG-based problem modeling, the ILP in this work describes the mapping of each vNF in a vNF-SC to a VM/Docker/SmartNIC on a substrate node more clearly, and its compactness is also ensured. Finally, in addition to data processing latency, this work also considers the propagation delay on network links, which makes the ILP more practical.

III. Problem Description

In this section, we first describe the heterogeneous environment that includes software and hardware NFV platforms such as VM, docker container and SmartNIC, for vNF-SC deployment, then explain the LAGs for problem modeling, and finally define the problem of application-driven provisioning of vNF-SCs over heterogeneous NFV platforms.

A. Heterogeneous NFV Platforms

Fig. 1 shows an example on the network environment that includes heterogeneous NFV platforms. Here, each node in the substrate network (SNT) can support an arbitrary combination of three types of NFV platforms (*i.e.*, VM, docker container (Docker) and SmartNIC), which are not special-purpose but can be used to instantiate different types of vNFs and satisfy various QoS requirements of applications. Hence, they are actually the virtualized IT resources in the SNT to facilitate NFV. For instance, *Nodes* 1 and 4 only support SmartNICs and Dockers, respectively, *Node* 2 can carry both VMs and Dockers, and *Node* 3 includes all the three types of platforms. To save the total deployment cost, we allow different vNF-SCs to share a vNF. For example, vNF-SCs R_1 and R_2 in Fig. 1 share the *vNF* 1 instantiated on a SmartNIC on *Node* 3.

B. Layered Auxiliary Graphs (LAGs)

Since a vNF-SC request demands for an ordered sequence of vNFs to process application traffic along the chosen path from

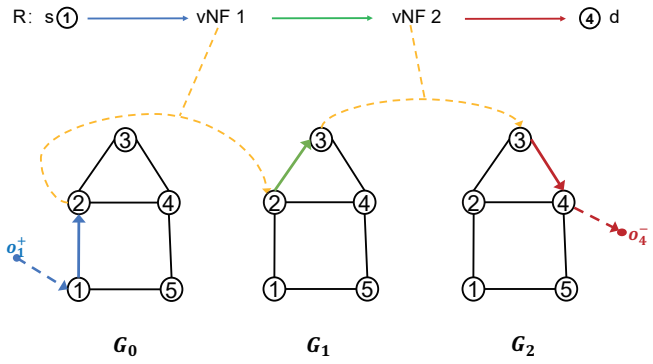


Fig. 2. Example on LAGs and provisioning a vNF-SC with them.

source to destination, we model it as $R_i(s_i, d_i, SC_i, b_i, t_i)$, where i is its unique index, s_i and d_i are the source and destination nodes, respectively, b_i represents its bandwidth requirement, t_i is its tolerable end-to-end delay, and $SC_i =< f_{i,1}, \cdots, f_{i,l}, \cdots, f_{i,N_i} >$ denotes its vNF sequence. Here, $f_{i,l}$ is the type of the l-th vNF, and N_i is the number of vNFs in the vNF-SC. Each vNF in SC_i needs to be mapped onto a VM/Docker/SmartNIC on a substrate node (SN), and we can either deploy a new vNF for it or make it use an existing vNF that is in the same type and has enough data processing throughput left. Then, the traffic of the vNF-SC needs to be steered through the deployed vNFs in the sequence required by SC_i. To model these relations with a compact ILP model and consequently facilitate the design of our approximation algorithm, we leverage the idea of LAGs in [49], *i.e.*, using a few layered graphs derived from the physical topology to help build an integrated view during modeling. Specifically, we first decompose a vNF-SC $SC_i =< f_{i,1}, \cdots, f_{i,N_i} >$ into a few segments, each of which corresponds to the connection between two adjacent vNFs, *i.e.*, the segments are $< s_i, f_{i,1} >$, $< f_{i,1}, f_{i,2} >$, $\cdots$, $< f_{i,N_i}, d_i >$. Then, we slice the SNT's topology into L^* LAGs, each of which has the same topology as that of the SNT, and we have

$$L^* = \max_i (N_i + 1). \tag{1}$$

The LAGs are denoted as $G_0, G_1, \cdots, G_{L^*-1}$. The provisioning of each vNF-SC request R_i will involve (N_i+1) LAGs. We number the segments of R_i from 0 to N_i, where $< s_i, f_{i,1} >$ is the 0-th segment and so on. Then, the provisioning of the j-th segment is handled in the j-th LAG G_j, and the edge between two adjacent LAGs represent a vNF. For instance, for R_i, the edge between G_j and G_{j+1} is for the vNF $f_{i,j+1}$.

Fig. 2 depicts an example on LAGs. The traffic of the vNF-SC originates from *Node* 1, is processed by *vNFs* 1 and 2 sequentially, and terminates at *Node* 4. As $N_i = 2$, we use three LAGs, *i.e.*, G_0, G_1, and G_2, to provision it. In G_0, since the source of the vNF-SC is *Node* 1, we have a dummy node o_1^+ to point to *Node* 1 and place *vNF* 1 on *Node* 2. Then, the segment in G_1 starts from *Node* 2 and ends at *Node* 3, where *vNF* 2 is placed. Finally, in G_2, the segments ends at another dummy node o_4^-. Hence, the path from o_1^+ to o_4^- represents of the provisioning scheme of the vNF-SC.

C. Network Model

We model the SNT's topology as an undirected graph $G(V, E)$, where V and E are the sets of SN and substrate links (SLs), respectively. Each SN $v \in V$ can carry h_v NFV platforms at most, and more specifically, the maximum numbers of NFV platforms that can be VMs/Dockers/SmartNICs in SN $v \in V$ are denoted as $h_v^U/h_v^D/h_v^S$, respectively. We use three parameters to denote the type of the k-th NFV platform on SN v, and their values are determined in advance. Specifically, $\mu_{v,k}$, $\xi_{v,k}$ and $\nu_{v,k}$ are boolean parameters that equal 1 if the k-th NFV platform on SN v is a VM/Docker/SmartNIC, respectively, and 0 otherwise. We have

$$h_v^U + h_v^D + h_v^S = h_v, \quad \forall v \in V, \tag{2}$$

$$\mu_{v,k} + \xi_{v,k} + \nu_{v,k} = 1, \quad \forall v \in V,\ k \in [1, h_v]. \tag{3}$$

By adjusting the values of $\{h_v^U, h_v^D, h_v^S, \mu_{v,k}, \xi_{v,k}, \nu_{v,k}\}$, we can make each SN support an arbitrary combination of VMs/Dockers/SmartNICs. The total IT resource capacities of the NFV platforms on SN v are $C_v^U/C_v^D/C_v^S$ for VMs/Dockers/SmartNICs, respectively.

Each vNF-SC can be built with different types of vNFs (*e.g.*, firewall and network address translator (NAT)). Therefore, we denote the vNF types that can be supported in the SNT with set M. For a vNF belonging to type $m \in M$, if it gets deployed on a VM/Docker/SmartNIC, it consumes $\hat{c}_m^U/\hat{c}_m^D/\hat{c}_m^S$ units of IT resources (*e.g.*, the percentage of memory usage), can at most process $b_m^U/b_m^D/b_m^S$ data traffic in terms of bandwidth units, and takes $r_m^U/r_m^D/r_m^S$ units of time to process application data, respectively. For the communications among vNFs, we use (u_{k_1}, v_{k_2}) to denote the link $(u, v) \in E$ between the k_1-th NFV platform on SN u and the k_2-th NFV platform on SN v. Obviously, for any two platforms on a same SN $v \in V$, the link (v_{k_1}, v_{k_2}) does not consume any bandwidth resources. Hence, we use links between NFV platforms to represent the routing path of each vNF-SC, and to denote the special cases of source and destination (*i.e.*, s_i and d_i of R_i), we add a dummy node o_v^+ or o_v^- to the LAGs if s_i or d_i is on SN $v \in V$, respectively. Fig. 3 gives an example to explain the path computation mentioned above, where the vNF-SC's traffic gets routed as $o_2^+ \rightarrow 2 \rightarrow 3 \rightarrow 4 \rightarrow o_4^-$, and *vNF* 1 is deployed on the k_1-th platform in *Node* 3, which is a SmartNIC.

IV. ILP Formulation

In this section, we formulate an ILP model to solve the problem of application-driven provisioning of vNF-SCs over an SNT that contains heterogeneous NFV platforms, based on the LAG-based modeling discussed in Section III-B. The optimization objective is to minimize the total deployment cost of all the vNF-SC requests. Specifically, to provision a vNF-SC request, we need to deploy the required vNFs on proper platforms in the SNs (*i.e.*, the platforms can be based on VMs/Dockers/SmartNICs), and connect the vNFs in sequence by steering the traffic of the request from source to destination.

Notations:

- $G(V, E)$: the topology of the SNT.
- G_j: the j-th LAG sliced from the SNT.
- $R_i(s_i, d_i, SC_i, b_i, t_i)$: the i-th vNF-SC request ($R_i \in \mathbf{R}$).

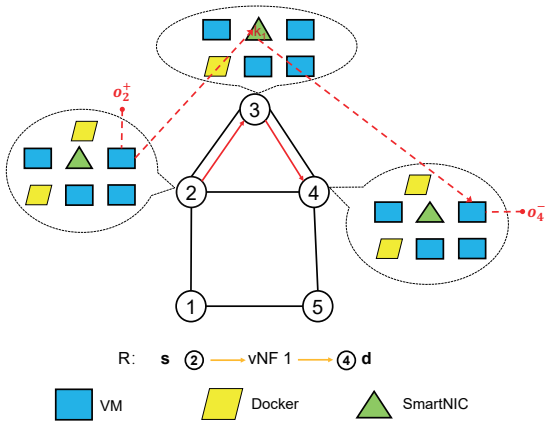


Fig. 3. Example of network model.

- N_i: the number of vNFs in request R_i.
- β: the unit cost of bandwidth usage on each SL.
- M: the set of available vNF types.
- $f_{i,l}^m$: the boolean parameter that equals 1 if the l-th vNF in SC_i is a type-m vNF ($m \in M$), and 0 otherwise.
- h_v: the number of NFV platforms that SN v has.
- $\mu_{v,k}$: the boolean parameter that equals 1 if the k-th platform on SN v is a VM, and 0 otherwise.
- $\xi_{v,k}$: the boolean parameter that equals 1 if the k-th platform on SN v is a docker container, and 0 otherwise.
- $\nu_{v,k}$: the boolean parameter that equals 1 if the k-th platform on SN v is a SmartNIC, and 0 otherwise.
- $d_{(u_{k_1},v_{k_2})}$: the boolean parameter that equals 1 if $u \neq v$ (*i.e.*, $(u,v) \in E$ and $u,v \in V$), and 0 otherwise. This parameter is introduced to determine whether a link between the vNFs deployed on any two NFV platforms consumes bandwidth resources in the SNT.
- $D_{(u_{k_1},v_{k_2})}$: the propagation latency of link (u_{k_1},v_{k_2}).
- o_v^+/o_v^-: the dummy node to/from SN v if it is the source/destination of a vNF-SC request.
- $C_v^U/C_v^D/C_v^S$: the IT resource capacity of a VM/Docker/SmartNIC on SN v, respectively.
- $b_m^U/b_m^D/b_m^S$: the processing throughput of a VM/Docker/SmartNIC, respectively, if it carries a type-m vNF.
- $r_m^U/r_m^D/r_m^S$: the unit data processing latency of a VM/Docker/SmartNIC, respectively, if carrying a type-m vNF.
- $\alpha_m^U/\alpha_m^D/\alpha_m^S$: the unit cost of IT resources for deploying a type-m vNF on a VM/Docker/SmartNIC.

Variables:

- $x_{i,(u_{k_1},v_{k_2}),j}$: the boolean variable that equals 1 if link (u_{k_1},v_{k_2}) in LAG G_j is used to provision request R_i, and 0 otherwise.
- $y_{i,l}^{v,k}$: the boolean variable that equals 1 if the l-th vNF in SC_i is deployed on the k-th platform in SN v, and 0 otherwise.
- $\phi_m^{v,k}$: the boolean variable that equals 1 if the k-th platform in SN v carries a type-m vNF, and 0 otherwise.

Objective:

The objective is to minimize the total deployment cost of all the vNF-SC requests. Here, the deployment cost of each vNF-SC request includes both the cost of IT resource usages for vNF placement and the cost of bandwidth usage. Therefore, the optimization objective is

$$\text{Minimize} \quad T = T_v + T_b, \tag{4}$$

where T_v and T_b are the costs of IT and bandwidth usages,

$$\begin{aligned} T_v &= \sum_{m,v,k} \left(\alpha_m^U \cdot \phi_m^{v,k} \cdot \mu_{v,k} + \alpha_m^D \cdot \phi_m^{v,k} \cdot \xi_{v,k} + \alpha_m^S \cdot \phi_m^{v,k} \cdot \nu_{v,k} \right), \\ T_b &= \sum_{i,u,k_1,v,k_2,j} \beta \cdot b_i \cdot d_{(u_{k_1},v_{k_2})} \cdot x_{i,(u_{k_1},v_{k_2}),j}. \end{aligned} \tag{5}$$

Constraints:

$$\begin{cases} \sum_{m,k} \hat{c}_m^U \cdot \phi_m^{v,k} \cdot \mu_{v,k} \leq C_v^U \\ \sum_{m,k} \hat{c}_m^D \cdot \phi_m^{v,k} \cdot \xi_{v,k} \leq C_v^D \\ \sum_{m,k} \hat{c}_m^S \cdot \phi_m^{v,k} \cdot \nu_{v,k} \leq C_v^S \end{cases}, \quad \forall v \in V. \tag{6}$$

Eq. (6) ensures that the vNF deployment on the NFV platforms in each SN will not exceed their IT resource capacities.

$$\begin{cases} \sum_i b_i \cdot \sum_l f_{i,l}^m \cdot y_{i,l}^{v,k} \cdot \mu_{v,k} \leq b_m^U \\ \sum_i b_i \cdot \sum_l f_{i,l}^m \cdot y_{i,l}^{v,k} \cdot \xi_{v,k} \leq b_m^D \\ \sum_i b_i \cdot \sum_l f_{i,l}^m \cdot y_{i,l}^{v,k} \cdot \nu_{v,k} \leq b_m^S \end{cases}, \quad \forall m,v,k. \tag{7}$$

Eq. (7) ensures that the vNF deployment on the platforms in each SN will not exceed their data processing capacities.

$$\begin{aligned} &\sum_{l,m} f_{i,l}^m \cdot \sum_{v,k} y_{i,l}^{v,k} \cdot (\mu_{v,k} \cdot r_m^U + \xi_{v,k} \cdot r_m^D + \nu_{v,k} \cdot r_m^S) \\ &+ \sum_{(u_{k_1},v_{k_2}),j} x_{i,(u_{k_1},v_{k_2}),j} \cdot D_{(u_{k_1},v_{k_2})} \leq t_i, \quad \forall i. \end{aligned} \tag{8}$$

Eq. (8) ensures that, for a request R_i, its end-to-end latency, which includes both the data processing latency and the link propagation delay, will not exceed the QoS requirement t_i.

$$\sum_{v,k} y_{i,l}^{v,k} = 1, \quad \forall i,l. \tag{9}$$

Eq. (9) ensures that each vNF in SC_i is deployed on one and only one NFV platform.

$$\sum_m \phi_m^{v,k} \leq 1, \quad \forall v,k. \tag{10}$$

Eq. (10) ensures that each NFV platform on an SN can only be used to carry one type of vNF at most.

$$\begin{cases} \sum_{i,l} f_{i,l}^m \cdot y_{i,l}^{v,k} > (\phi_m^{v,k} - 1) \cdot \left(1 + \sum_{i,l} f_{i,l}^m \right) \\ \sum_{i,l} f_{i,l}^m \cdot y_{i,l}^{v,k} \leq \phi_m^{v,k} \cdot \sum_{i,l} f_{i,l}^m \end{cases}, \quad \forall m,v,k. \tag{11}$$

$$\phi_m^{v,k} \geq y_{i,l}^{v,k} \cdot f_{i,l}^m, \quad \forall m,v,k,i,l. \tag{12}$$

Eqs. (11)-(12) ensure that the values of the variables are set correctly according to their inherent relations. In other words,

Eq. (11) makes sure that if $\left(\sum_{i,l} f_{i,l}^{m} \cdot y_{i,l}^{v,k}\right) > 0$, we have $\phi_m^{v,k} = 1$, and $\phi_m^{v,k}$ should be set as 0 otherwise.

$$\sum_{v,k_2} x_{i,(u_{k_1},v_{k_2}),j} - \sum_{v,k_2} x_{i,(v_{k_2},u_{k_1}),j} = y_{i,j}^{u,k_1} - y_{i,j+1}^{u,k_1}, \quad \forall k_1, u \in V, i, j \in [1, N_i - 1]. \tag{13}$$

Eq. (13) ensures the flow conservation constraint, which means that the in and out flows of each NFV platform is equal except for the ingress/egress points of a segment of SC_i in G_j.

$$\sum_{v,k_2} x_{i,(u_{k_1},v_{k_2}),0} - \sum_{v,k_2} x_{i,(v_{k_2},u_{k_1}),0} = \begin{cases} 1, \ k_1 = o_u^+ \text{ and } u = s_i, \\ -y_{i,1}^{u,k_1}, \quad \text{otherwise}, \end{cases} \quad \forall i. \tag{14}$$

Eq. (14) ensures the flow conservation constraint in the first LAG G_0, when serving R_i.

$$\sum_{v,k_2} x_{i,(u_{k_1},v_{k_2}),N_i} - \sum_{v,k_2} x_{i,(v_{k_2},u_{k_1}),N_i} = \begin{cases} -1, \ k_1 = o_u^- \text{ and } u = d_i, \\ y_{i,N_i}^{u,k_1}, \quad \text{otherwise}, \end{cases} \quad \forall i. \tag{15}$$

Eq. (15) ensures the flow conservation constraint in the N_i-th LAG G_{N_i}, when serving R_i.

$$y_{i,l}^{v,o_v^+} = y_{i,l}^{v,o_v^-} = 0, \quad \forall i, l, v, o_v^+, o_v^-. \tag{16}$$

Eq. (16) ensures that the dummy nodes to/from each SN v cannot be used for real vNF deployment.

$$\sum_{u,k_1,v,k_2} x_{i,(u_{k_1},v_{k_2}),j} \geq 1, \quad \forall i, j. \tag{17}$$

Eq. (17) ensures that at least one link in G_j is chosen for composing a path segment of R_i.

$$\begin{cases} \sum_{v,k_2} x_{i,(u_{k_1},v_{k_2}),j} \geq y_{i,j}^{u,k_1} \\ \sum_{v,k_2} x_{i,(v_{k_2},u_{k_1}),j-1} \geq y_{i,j}^{u,k_1}, \end{cases} \quad \forall u, k_1, j \in [1, N_i]. \tag{18}$$

Eq. (18) ensures the correct relation between vNF placement and routing path construction. Specifically, it makes sure that if the k_1-th platform on SN u is used to deploy the j-th vNF of SC_i, the path calculation in LAGs G_{j-1} and G_j should consider the k_1-th platform on SN u as an end-node for segments $< f_{i,j-1}, f_{i,j} >$ and $< f_{i,j}, f_{i,j+1} >$, respectively.

V. Design of Approximation Algorithm

Solving the aforementioned ILP model can obtain the exact solutions of our problem, but it would be intractable for large-scale ones. Meanwhile, it is known that even the basic vNF placement problem is $\mathcal{NP}$-hard [12]. Therefore, we resort to a polynomial-time approximation algorithm which can guarantee the performance gap to the exact solutions. Specifically, our approximation algorithm leverages LP relaxation with randomized rounding to solve the problem of application-driven provisioning of vNF-SCs over heterogeneous NFV platforms time-efficiently and obtain near-optimal solutions.

A. Overall Procedure

Algorithm 1 shows the overall procedure of our approximation algorithm. In addition to the information about the SNT and vNF-SC requests, it also takes several positive parameters (*i.e.*, Q, γ, δ, ϵ, and ζ) as the inputs. We use Q and γ to adjust the tradeoff between the time complexity of the algorithm and its approximation ratio, and their values are determined empirically [50]. Specifically, the value of γ can be understood as a preset expectation on the output of *Algorithm* 1. We will prove the relation between γ and the approximation ratio of *Algorithm* 1 later in Section V-D, and will show the effects of γ with the simulations in Section VI. δ, ϵ, and ζ are the ratios to tighten the constraints in Eqs. (6)-(8) for the LP relaxation, respectively, and they are introduced to ensure that we can get feasible solutions to the original ILP model through LP relaxation with randomized rounding. The values of δ, ϵ, and ζ are also determined empirically.

In *Line* 1, we relax all the boolean variables in the ILP model to real ones within $[0, 1]$ and get an LP model. Then, *Line* 2 tightens several constraints in the LP with the corresponding ratios. For instance, we tighten the IT resource capacities with δ, which means that in the LP model, we have

$$\begin{cases} \tilde{C}_v^U = C_v^U \cdot (1 - \delta) \\ \tilde{C}_v^D = C_v^D \cdot (1 - \delta), \\ \tilde{C}_v^S = C_v^S \cdot (1 - \delta) \end{cases} \quad \forall v \in V, \tag{19}$$

where $\tilde{C}_v^U$, $\tilde{C}_v^D$, and $\tilde{C}_v^S$ are the corresponding IT resource capacities in the LP. The similar tightening scheme applies to the constraints in Eqs. (7) and (8). The LP is solved in *Line* 3 to obtain the objective T_{LP}, which is a lower-bound of the solution to the original ILP. This can be done in polynomial-time [51] (*e.g.*, with the ellipsoid algorithm [52]).

Line 4 is for the initialization of the randomized rounding. The while-loop covering *Lines* 5-14 performs the randomized rounding for Q iterations at most. In each iteration, *Line* 6 performs randomized rounding on the real variables in $\{X_{i,j}\}$, calculates $\{y_{i,l}^{v,k}, \ \phi_m^{v,k}\}$ based on the rounding results, and obtains an integer solution $\mathbf{S}$, all with *Algorithm* 2. Then, we calculate the objective T and validate all the constraints in the original ILP with $\mathbf{S}$ (*Line* 7). If $\mathbf{S}$ is a feasible solution to the ILP, *Line* 9 checks whether its performance satisfies the approximation ratio γ. If yes, we get a qualified solution to the ILP. Otherwise, the while-loop proceeds to the next iteration.

B. Randomized Rounding

Algorithm 2 explains the randomized rounding to construct an integer solution based on the solution to the LP, *i.e.*, $\{X_{i,j}\}$. *Lines* 1-2 are for the initialization. Then, in the for-loop covering *Lines* 3-33, each iteration determines the integer solution $\{x_{i,(u_{k_1},v_{k_2}),j}\}$ that is related to provision request R_i in LAG G_j. Specifically, the operations are as follows. For each request R_i, we calculate a path segment in G_j, which starts from a link (u_{k_1}, v_{k_2}). Here, we use $(v_k, *)$ to denote all the outgoing links from the k-th platform on SN v. Then, following the chosen link, we select an outgoing link from its ending platform with a probability of $x_{i,(u_{k_1},v_{k_2}),j}$, and repeat the procedure until reaching the ending platform of R_i in G_j.

In *Line* 4, we initialize t_1 and t_2, which represent the starting SN and the platform on the SN that originates the chosen link, respectively. Then, if the $x_{i,(u_{k_1},v_{k_2}),j}$ from the LP is within $(p_1^{u,k_1}, p_2^{u,k_1}]$, link (u_{k_1}, v_{k_2}) is chosen in LAG G_j to serve R_i (*Lines* 9-23). Here, F_1 is the flag to indicate whether the provisioning in G_j has ended, while flag F_2 tells us whether a required link has been found for the segment in G_j, and *Lines* 13-15 help to avoid endless loops. Finally, when all the values of $\{x_{i,(u_{k_1},v_{k_2}),j}\}$ have been obtained, *Lines* 34-35 compute $\{y_{i,l}^{v,k},\ \phi_m^{v,k}\}$ based on them, insert the variables in $\mathbf{S}$, and return $\mathbf{S}$ as an integer solution from the randomized rounding.

Algorithm 1: Procedure of Approximation Algorithm

Input: SNT topology $G(V, E)$, set of vNF-SC requests $\mathbf{R}$, maximum number of rounding trials Q, approximation ratios γ, δ, ϵ, and ζ.

1 relax the ILP of Eqs. (4)-(18) to get an LP;
2 tighten the LP's constraints in Eqs. (6)-(8) with ratios δ, ϵ and ζ, respectively;
3 solve the LP to get $\{x_{i,(u_{k_1},v_{k_2}),j},\ y_{i,l}^{v,k},\ \phi_m^{v,k}\}$ in real numbers and the objective T_{LP} with Eqs. (4)-(5);
4 $q = 1$, $X_{i,j} = \{x_{i,(u_{k_1},v_{k_2}),j}\}, \forall i, j$;
5 **while** $q \leq Q$ **do**
6 perform randomized rounding on $\{X_{i,j}\}$ to get an integer solution $\mathbf{S}$ with *Algorithm* 2;
7 calculate objective T and validate all constraints in the original ILP with $\mathbf{S}$;
8 **if** $\mathbf{S}$ *is a feasible solution to the ILP* **then**
9 **if** $T \leq (1+\gamma) \cdot T_{LP}$ **then**
10 **break**;
11 **end**
12 **end**
13 $q = q + 1$;
14 **end**

Algorithm 2: Randomized Rounding

Input: SNT topology $G(V, E)$, set of vNF-SC requests $\mathbf{R}$, $\{X_{i,j}\}$ from the LP.
Output: An integer solution $\mathbf{S}$.

1 $\mathbf{S} = \emptyset$;
2 choose the value of p within $(0, 1)$ randomly;
3 **for** *each request* $R_i \in \mathbf{R}$ **do**
4 $t_1 = s_i$, $t_2 = o_u^+$, $p_1^{v,k} = 0$, $p_2^{v,k} = 0$;
5 **for** *each LAG* G_j *used by* R_i **do**
6 $F_1 = 0$;
7 **while** $F_1 = 0$ **do**
8 $F_2 = 0$;
9 **for** *each* $x_{i,(u_{k_1},v_{k_2}),j}$ *related to* G_j **do**
10 **if** $u = t_1$ ***and*** $k_1 = t_2$ **then**
11 $p_2^{u,k_1} = p_2^{u,k_1} + x_{i,(u_{k_1},v_{k_2}),j}$;
12 **if** $p_1^{u,k_1} < p \leq p_2^{u,k_1}$ **then**
13 **if** *link* (u_{k_1}, v_{k_2}) *has been chosen* **then**
14 **break**;
15 **end**
16 insert $x_{i,(u_{k_1},v_{k_2}),j} = 1$ in $\mathbf{S}$;
17 $t_1 = v$, $t_2 = k_2$, $F_2 = F_2 + 1$;
18 update $p_1^{t_1,t_2}$ and $p_2^{t_1,t_2}$;
19 **break**;
20 **end**
21 $p_1^{t_1,t_2} = p_2^{t_1,t_2}$;
22 **end**
23 **end**
24 **if** $F_2 > 0$ **then**
25 $F_1 = 0$;
26 **else**
27 $F_1 = 1$;
28 deploy the j-th vNF of SC_i on the t_2-th platform on SN t_1;
29 update the corresponding variable x and insert it in $\mathbf{S}$;
30 **end**
31 **end**
32 **end**
33 **end**
34 calculate $\{y_{i,l}^{v,k},\ \phi_m^{v,k}\}$ based on the x-based variables in $\mathbf{S}$ and insert them in $\mathbf{S}$;
35 **return**($\mathbf{S}$);

C. Time Complexity

The time complexity of *Algorithm* 2 is $O(|\mathbf{R}| \cdot \max_i(N_i) \cdot \max_{i,j}(|X_{i,j}|)^2)$. In *Algorithm* 1, solving the LP is known to be within polynomial time. Specifically, its complexity is $O(Z^{3.5} \cdot L)$ if we use the famous interior point method [53], where Z is the number of variables and L is the total number of bits of the input. Hence, the overall complexity of *Algorithm* 1 is $O(Q \cdot |\mathbf{R}| \cdot \max_i(N_i) \cdot \max_{i,j}(|X_{i,j}|)^2 + Z^{3.5} \cdot L)$. To this end, we can see that *Algorithm* 1 is a polynomial-time algorithm.

D. Approximation Ratio

Lemma 1. Algorithm *1 is an approximation algorithm for the original provisioning problem defined in the ILP, and its approximation ratio is upper-bounded by* $(1+\gamma)$.

Proof: After relaxing the ILP, we get the LP's solution (*i.e.*, T_{LP}) as a lower-bound of the optimal solution (*i.e.*, denoted as T_{ILP}). As the original optimization is a minimization problem, each feasible solution (*i.e.*, T) provided by *Algorithm* 1 sets an upper-bound on the optimal solution. Hence, we can calculate the approximation ratio of *Algorithm* 1 as

$$\eta = \frac{T}{T_{\text{ILP}}} \leq \frac{(1+\gamma) \cdot T_{\text{LP}}}{T_{\text{ILP}}} \leq \frac{(1+\gamma) \cdot T_{\text{ILP}}}{T_{\text{ILP}}} = 1 + \gamma, \qquad (20)$$

i.e., the approximation ratio will not exceed $(1+\gamma)$. ■

Lemma 2. *Due to the randomized rounding, the probability of* Algorithm *2 obtaining a feasible solution* $T \geq (1+\gamma) \cdot T_{ILP}$ *is upper-bounded by* $\exp(-\lambda \cdot \gamma^2)$, *where we have* $\lambda = \frac{\min_i(b_i)}{3 \cdot \max_i(b_i)}$.

Proof: In order to prove the probabilistic guarantee, we recast the algorithm in terms of random variables, and analyze the total bandwidth cost T_b in Eq. (5) first. For bounding T_b in *Algorithm* 2, we introduce the discrete random variable $Y_{i,u,k_1,v,k_2,j} \in \left\{0, \beta \cdot b_i \cdot d_{(u_{k_1},v_{k_2})}\right\}$. According to the principle of randomized rounding, we have the probability $P\left(Y_{i,u,k_1,v,k_2,j} = \beta \cdot b_i \cdot d_{(u_{k_1},v_{k_2})}\right) = x_{i,(u_{k_1},v_{k_2}),j}$. Hence, we can get the total bandwidth cost of one trial in *Algorithm* 2 as $\tilde{T}_b = \sum Y_{i,u,k_1,v,k_2,j}$, which makes the output of *Algorithm* 2 as $T_b = E(\tilde{T}_b)$. Next, we scale variables $Y_{i,u,k_1,v,k_2,j}$ within $[0,1]$, *i.e.*, $Y' = \frac{Y}{\beta \cdot \max\limits_i(b_i)}$, and have $\tilde{T}'_b = \sum Y'$. By applying the well-known Chernoff-Bound [54], we can get

$$P\left(\tilde{T}'_b > (1+\gamma) \cdot E(\tilde{T}'_b)\right) \leq \exp(-\gamma^2 \cdot \frac{E(\tilde{T}'_b)}{3}), \tag{21}$$

where $\frac{E(\tilde{T}'_b)}{3}$ is a part of the multiplicative form and provides a looser but more convenient bound [54]. Then, as the condition $T_b = E(\tilde{T}_b) \geq \beta \cdot \min\limits_i(b_i)$ always holds, we define

$$\lambda = \frac{\min\limits_i(b_i)}{3 \cdot \max\limits_i(b_i)}, \tag{22}$$

and finally get

$$P\left(\tilde{T}_b > (1+\gamma) \cdot T_b\right) \leq \exp(-\lambda \cdot \gamma^2), \tag{23}$$

which proves the probabilistic guarantee for the total bandwidth cost. Meanwhile, following the similar procedure, we can also prove the probabilistic guarantee for the total IT resource cost. Finally, as the output of *Algorithm* 2 provides an upper-bound on the optimal solution of the original problem (*i.e.*, T_{ILP}), we prove the probabilistic guarantee. ■

TABLE I
PARAMETERS OF VNF DEPLOYMENTS (ADAPTED FROM [19])

		vNF 1	*vNF* 2	*vNF* 3	*vNF* 4
Throughput (Gbps)	VM (b_m^U)	1.60	1.55	1.57	1.42
	Docker (b_m^D)	1.32	1.26	1.22	1.7
	SmartNIC (b_m^S)	10	10	10	10
Processing Latency (μs)	VM (r_m^U)	177	190	224	262
	Docker (r_m^D)	154.5	154.6	155.7	197
	SmartNIC (r_m^S)	110.2	110.7	110.9	111.7
Memory Usage (%)	VM ($\hat{c}_m^U$)	3.7	3.7	3.5	3.7
	Docker ($\hat{c}_m^D$)	0.003	0.003	0.002	0.01
	SmartNIC ($\hat{c}_m^S$)	26.25	22.75	26.25	23.83

VI. PERFORMANCE EVALUATION

In this section, we perform numerical simulations to evaluate the performance of our proposed approaches for the vNF-SC provisioning over heterogeneous NFV platforms.

A. Simulation Setup

In the simulations, we consider two SNT topologies, *i.e.*, the small-scale six-node topology in Fig. 4(a) and the large-scale NSFNET topology in Fig. 4(b). We consider $|M| = 4$ types of vNFs, and the parameters about their deployments on a VM/Docker/SmartNIC are in Table I, which are the experimental results measured in [19]. Specifically, we deployed the VMs and Dockers on Linux servers, each of which is a Lenovo ThinkSystem SR650 with 2.10 GHz Intel Xeon Silver 4110 CPU and 32 GB memory and runs Ubuntu 16.04, while we also equip Netronome Agilio SmartNICs on the servers to support vNFs. All the network interfaces on the servers were based on 10 GbE (*i.e.*, with a traffic processing throughput of 10 Gbps)[1]. Table I captures certain real differences when a vNF is operating over a VM/Docker/SmartNIC, while more practical differences will be considered in our future experimental studies. The number of vNFs required by each vNF-SC request is randomly chosen within $[1, 4]$. The cost coefficients are set as $\beta = 0.4$, $\alpha_m^U = 1$, $\alpha_m^D = 1.6$, and $\alpha_m^S = 1.76$, according to the latest realistic data [55, 56]. As programmable hardware is relatively expensive, we assume that the SNT with the six-node topology has at most one SmartNIC per SN, while each SN in the NSFNET topology can equip two at most.

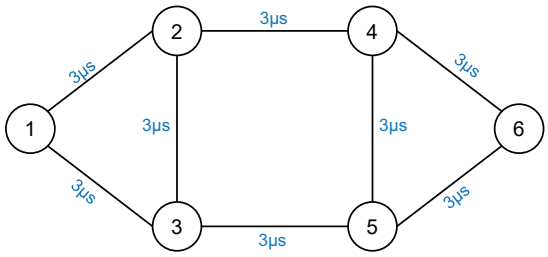
In order to emulate application-driven provisioning, the simulations consider three scenarios, whose QoS requirements are normal, large-bandwidth and low-latency, respectively. The bandwidth demand of each request in the normal and low-latency scenarios is randomly chosen from $[0.1, 0.3]$ Gbps, while for the large-bandwidth one, it is within $[0.5, 0.8]$ Gbps. Each request can tolerant an end-to-end latency randomly chosen within $[0.85, 5]$ msecs in the normal and large-bandwidth scenarios, while its requirement on end-to-end latency is within $[0.5, 0.8]$ msecs in the low-latency scenario. To ensure sufficient statistical accuracy, we run 10 independent simulations and average the results to get each data point. The simulations are carried out on a computer with 3.0 GHz Intel Core i5-7400 CPU and 8 GB memory, and the environment is MATLAB 2016a with GLPK toolbox and Gurobi v9.0.

B. Small-Scale Simulations

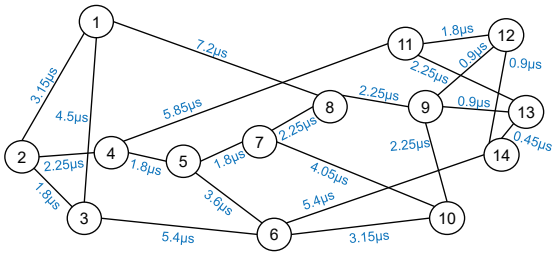
We first use the six-node topology to conduct small-scale simulations to compare the performance of our approximation algorithm to that of the ILP model. The maximum number of iterations (*i.e.*, the while-loop in *Algorithm* 1) is fixed as $Q = 10$, while in the simulations, the while-loop actually gets executed 7 times at most and roughly 3 times on average.

Table II lists the simulation results. We observe that the gap between the near-optimal results from *Algorithm* 1 and the exact ones from the ILP is very small. Specifically, the largest approximation ratio is $\gamma_{\max} = 0.0749$, while the average approximation ratio is $\bar{\gamma} = 0.0230$. We can also see that for the normal scenario in small scale, solving the ILP can take slightly longer time than the approximation algorithm. However, if we keep increasing the problem's scale or make it

[1]Theoretically speaking, the processing latency of a vNF should increase with the rate of the traffic being processed, especially when the traffic rate is approaching to the processing capacity of the vNF. However, in a practical network system, we will see packet drops when the traffic rate is close to the processing capacity of a vNF, which will severely affect the vNF's QoS. Hence, in [19], we measured each vNF's data processing throughput and latency in the situation that it is far from being stressed and there is no packet drop. This is actually the reason why the processing latencies in Table I are constant numbers, *i.e.*, the maximum traffic rate to a vNF is much smaller than its actual data processing capacity.

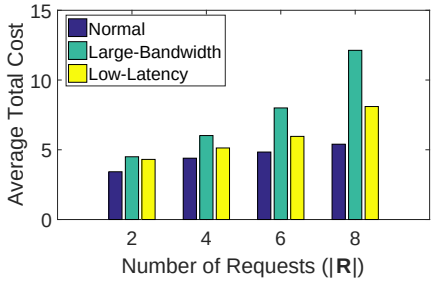


(a) Six-node topology

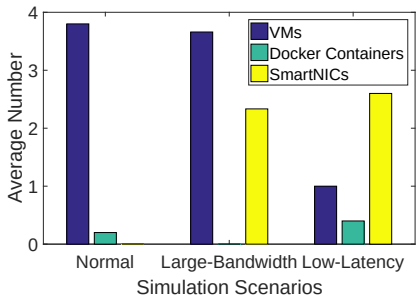


(b) NSFNET topology

Fig. 4. SNT topologies with propagation latencies marked in μs.



(a) vNF-SC deployment cost



(b) Device number for $|\mathbf{R}| = 8$

Fig. 5. Optimal results for application-driven vNF-SC provisioning over heterogeneous NFV platforms.

more complex (*e.g.*, the one in the large-bandwidth scenario, where more provisioning schemes have to be tried for each request to satisfy its bandwidth requirement), *Algorithm* 1 shows its high time-efficiency clearly. Specifically, *Algorithm* 1 runs much faster than the ILP in such cases, and its running time does not increase exponentially with the problem's scale. The results in Table II verify the effectiveness of our proposed approximation algorithm.

Fig. 5 shows the optimal results regarding the application-driven vNF-SC provisioning in the six-node topology. In Fig.

TABLE II
SIMULATION RESULTS WITH SIX-NODE TOPOLOGY

$\|R\|$		2	4	6	8
Normal Scenario					
ILP	Total Deployment Cost	3.40	4.41	4.88	5.40
	IT Resource Cost	3.16	3.85	4	4.12
	Bandwidth Cost	0.24	0.56	0.88	1.28
	Average running time (s)	0.41	1.12	3.05	70.05
Algorithm 1	Total Deployment Cost	3.40	4.41	4.88	5.44
	IT Resource Cost	3.16	3.85	4	4.15
	Bandwidth Cost	0.24	0.56	0.88	1.29
	Average running time (s)	0.31	0.55	1.03	2.08
Large-Bandwidth Scenario					
ILP	Total Deployment Cost	4.50	6.02	8.00	12.13
	IT Resource Cost	3.3	4.34	5.52	7.78
	Bandwidth Cost	1.20	1.68	2.48	4.35
	Average running time (s)	0.57	21.06	950.08	2581.75
Algorithm 1	Total Deployment Cost	4.50	6.27	8.24	12.51
	IT Resource Cost	3.3	4.67	5.76	8.68
	Bandwidth Cost	1.20	1.60	2.48	3.83
	Average running time (s)	0.49	0.99	1.48	3.90
Low-Latency Scenario					
ILP	Total Deployment Cost	4.31	5.13	5.96	8.10
	IT Resource Cost	3.97	4.57	4.96	6.22
	Bandwidth Cost	0.34	0.56	1.00	1.88
	Average running time (s)	1.12	17.45	27.32	103.52
Algorithm 1	Total Deployment Cost	4.35	5.20	6.37	8.70
	IT Resource Cost	4.03	4.64	5.52	7.36
	Bandwidth Cost	0.32	0.56	0.85	1.34
	Average running time (s)	0.42	0.96	1.49	2.97

5(a), we notice that when the number of requests is the same, the total deployment cost increases if we switch from the normal scenario to the large-bandwidth one or the low-latency one. This is because when the QoS requirement becomes more stringent, provisioning a vNF-SC request could involve more bandwidth or/and IT resource cost. The results in Table II confirm the analysis, and so do those in Fig. 5(b). For instance, Fig. 5(b) indicates that in the normal scenario, vNF-SC requests can be provisioned without using any SmartNICs, while the largest number of SmartNICs would be required to serve the requests in the low-latency scenario. Therefore, the results in Table II and Fig. 5 also suggest that by leveraging heterogeneous NFV platforms, we can provision vNF-SC requests with various QoS requirements more cost-effectively. This further justifies the motivation of this work.

C. Large-Scale Test

We then use the large-scale NSFNET topology to evaluate our proposal. This time, as the ILP has already become intractable, we only simulate our approximation algorithm (*i.e.*, *Algorithm* 1), and use it to provisioning different numbers of vNF-SCs to check its scalability. In the simulations, we assume that the maximum number of requests is $|\mathbf{R}| = 64$. This is because in practical cases, it would be rare for an SP to receive more than 64 vNF-SC requests simultaneously and provision them in a batch. Hence, if more requests come in at different time instants, the SP can simply apply our approximation algorithm multiple times instead of provisioning them in one batch. Moreover, in the large-scale simulations, we do

not distinguish the normal, large-bandwidth, and low-latency scenarios, but assume that vNF-SC requests with various QoS requirements distribute randomly in the pending requests. Therefore, the simulation scenario will be more practical.

We first evaluate the convergence performance of *Algorithm* 1 with the most stressful case (*i.e.*, it needs to jointly optimize the provisioning schemes of $|\mathbf{R}| = 64$ pending vNF-SC requests). Fig. 6 shows the results on convergence performance, where the result of the LP-relaxation gives a lower-bound and each feasible solution obtained in one iteration provides an upper-bound. We can see that for the most stressful case, *Algorithm* 1 converges after only 30 iterations to achieve an upper-bound on the approximation ratio as $1 + \gamma = 1.2629$.

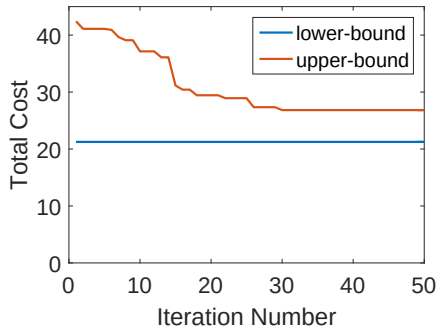
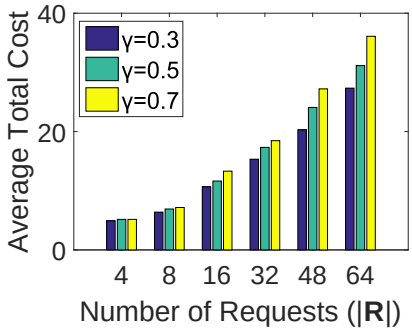


Fig. 6. Convergence performance of *Algorithm* 1 for serving $|\mathbf{R}| = 64$ vNF-SCs in NSFNET topology.

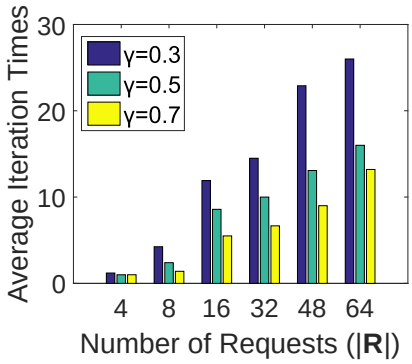
Then, we change the number of pending requests from 4 to 64, apply *Algorithm* 1 with $\gamma \in \{0.3, 0.5, 0.7\}$ to serve them, and record the average total deployment cost and the average number of iterations to achieve the selected γ. Fig. 7 shows the simulation results. As expected, the results in Fig. 7(a) suggest that with a smaller γ, the total deployment cost of vNF-SC requests can be further reduced with our approximation algorithm. Meanwhile, the tradeoff is that a smaller γ makes the algorithm run more iterations to obtain a qualified solution. Therefore, for our proposed approximation algorithm, we can adjust the value of γ to tackle the tradeoff between the solution's optimality and the running time. Nevertheless, in all the simulation scenarios, decreasing γ from 0.7 to 0.3 would not lead to an excessive increase of iterations, which confirms the scalability of our proposal. Table III lists the average running time taken by *Algorithm* 1, when γ is fixed as 0.3 and the number of requests in R changes from 4 to 64. It can be seen that when $|\mathbf{R}|$ increases from 4 to 64, the average time that *Algorithm* 1 takes to serve one request only increases from 1.56 to 3.26 seconds. This further confirms the time-efficiency and scalability of our proposal.

D. Comparative Evaluations

In addition to evaluating our proposal in different simulation scenarios, we should also compare it with state-of-art benchmarks. However, to the best of our knowledge, there is no existing algorithm that was designed to address our problem of application-driven provisioning of vNF-SCs over heterogeneous NFV platforms. Therefore, we try to adopt a



(a) vNF-SC deployment cost



(b) Iterations to achieve selected γ

Fig. 7. Simulation Results with NSFNET topology.

TABLE III
AVERAGE RUNNING TIME OF *Algorithm* 1 ($\gamma = 0.3$)

$\lvert\mathbf{R}\rvert$	Average Running Time (s)
4	6.22
8	15.59
16	33.07
32	81.18
48	136.01
64	208.76

benchmark from the studies whose backgrounds are similar to ours, and decide to use the approximation algorithm developed in [9], which is also based on LP relaxation with rounding and solves the problem of network function subgraph (NF-subgraph) provisioning. Here, an NF-subgraph can be understood as the merging result of several vNF-SCs. As the benchmark considers similar constraints and optimization objective as ours, we can apply minor modifications to adapt it to our problem. Meanwhile, since it treats NF-subgraphs as the basic virtual structures for service provisioning, we merge 4 vNF-SCs to compose each NF-subgraph[2].

We first compare *Algorithm* 1 and the benchmark in the SNT whose topology is the 14-node NSFNET, and the simulation parameters are the same as those in the previous subsection.

[2]Note that, if the number of vNF-SCs in each NF-subgraph is too large, it would be difficult for the benchmark to find the best provisioning scheme for the resulting NF-subgraph. Hence, for fair comparisons, we only include 4 vNF-SCs in each NF-subgraph. We also try to merge other small numbers of vNF-SCs as an NF-subgraph, and the results follow the same trend.

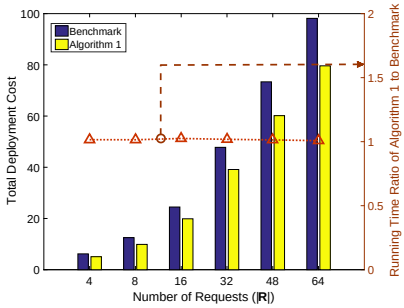


Fig. 8. Comparisons of Algorithm 1 and the benchmark with NSFNET topology.

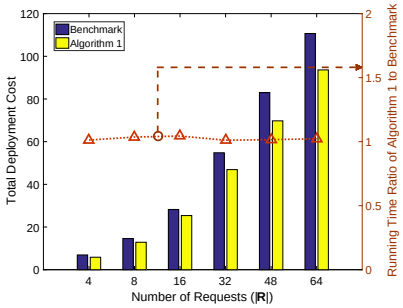


Fig. 10. Comparisons of Algorithm 1 and the benchmark with 16-node topology.

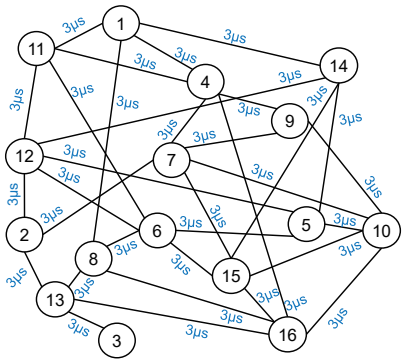


Fig. 9. 16-node SNT topology with propagation latencies marked in μs.

Fig. 8 compares the two algorithms in terms of the deployment cost of vNF-SCs and running time. Note that, to ensure fair comparisons, we terminate the two algorithms after similar numbers of iterations. We observe that *Algorithm* 1 outperforms the benchmark to provide lower deployment costs, while its running time is only slightly longer. This is because the benchmark does not specifically consider the heterogeneous NFV platforms to satisfy different QoS demands, or optimize the routing of each vNF-SC together with the placements of vNFs (*i.e.*, it first determines the placements of vNFs and then calculates the routing paths based on them). Meanwhile, as *Algorithm* 1 addresses a more sophisticated optimization, it takes slightly longer time to accomplish the task.

Next, we simulate the two algorithm with the random SNT topology in Fig. 9, which has 16 SNs, and a new type of vNFs (*i.e.*, *vNF* 5) is also introduced to compare the algorithms in more aspects. The parameters of *vNF* 5 are assumed to be the average values of the corresponding ones of *vNFs* 1-4 in Table I, and the other simulation parameters are still the same. Fig. 10 plots the simulation results of the two algorithms, which still have the similar trends as those in Fig. 8. The results in Figs. 8 and 10 verify the advantages of our proposal, and thus further justify the contributions of this work.

VII. Conclusion

In this paper, we studied application-driven provisioning of vNF-SCs over heterogeneous NFV platforms. An LAG-based approach was first introduced to model the provisioning problem. Then, based on the vNF-SC requests and their LAGs, we formulated a novel ILP model to optimize the application-driven provisioning of vNF-SCs, such that the total cost of vNF-SC deployment can be minimized while the QoS requirements of all the vNF-SCs are satisfied. To reduce the time complexity of problem solving, we designed an approximation algorithm based on LP relaxation with randomized rounding. Extensive simulations confirmed that with significantly improved time-efficiency, our proposed algorithm provides near-optimal solutions whose approximation ratios are bounded. Although the focus of this work is algorithm design but not system implementation, our proposal uses practical assumptions and thus can be implemented in the control plane of an SDN-based NFV framework (*e.g.*, the network service header (NSH) architecture [57]) to provide high-performance vNF-SC provisioning schemes. In our future work, we will try to verify this with experimental investigations.

Acknowledgments

This work was supported in part by the NSFC projects 61871357, 61771445 and 61701472, ZTE Research Fund PA-HQ-20190925001J-1, Zhejiang Lab Research Fund 2019LE0AB01, CAS Key Project (QYZDY-SSW-JSC003), and SPR Program of CAS (XDC02070300).

References

[1] Cisco Visual Networking Index: Forecast and Methodology, 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-datacenter networks," *IEEE Netw.*, vol. 29, pp. 36–42, Sept./Oct. 2015.

[3] M. Chiosi *et al.*, "Network functions virtualisation," 2012. [Online]. Available: https://portal.etsi.org/nfv/nfv_white_paper.pdf

[4] B. Han, V. Gopalakrishnan, L. Ji, and S. Lee, "Network function virtualization: Challenges and opportunities for innovations," *IEEE Commun. Mag.*, vol. 53, pp. 90–97, Feb. 2015.

[5] 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.

[6] A. Laghrissi and T. Taleb, "A survey on the placement of virtual resources and virtual network functions," *IEEE Commun. Surveys Tuts.*, pp. 1409–1434, Second Quarter 2018.

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

[8] B. Li, W. Lu, and Z. Zhu, "Deep-NFVOrch: Leveraging deep reinforcement learning to achieve adaptive vNF service chaining in EON-DCIs," *J. Opt. Commun. Netw.*, vol. 12, pp. A18–A27, Jan. 2020.

[9] D. Dietrich, A. Abujoda, A. Rizk, and P. Papadimitriou, "Multi-provider service chain embedding with Nestor," *IEEE Trans. Netw. Serv. Manag.*, vol. 14, pp. 91–105, Jan. 2017.

[10] 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.

[11] B. Li, W. Lu, S. Liu, and Z. Zhu, "Deep-learning-assisted network orchestration for on-demand and cost-effective vNF service chaining in inter-DC elastic optical networks," *J. Opt. Commun. Netw.*, vol. 10, pp. D29–D41, Oct. 2018.

[12] 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.

[13] 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.

[14] T. Kuo, B. Liou, K. Lin, and M. Tsai, "Deploying chains of virtual network functions: On the relation between link and server usage," *IEEE/ACM Trans. Netw.*, vol. 26, pp. 1562–1576, Aug. 2018.

[15] J. Anderson *et al.*, "Performance considerations of network functions virtualization using containers," in *Proc. of ICNC 2016*, pp. 1–7, Feb. 2016.

[16] C. Kachris, G. Sirakoulis, and D. Soudris, "Network function virtualization based on FPGAs: A framework for all-programmable network devices," *arXiv preprint arXiv:1406.0309*, 2014.

[17] Y. Le *et al.*, "UNO: Uniflying host and smart NIC offload for flexible packet processing," in *Proc. of ACM SoCC 2017*, pp. 506–519, Sept. 2017.

[18] 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.

[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] M. Bouet, T. Leguay, J.and Combe, and V. Conan, "Cost-based placement of vDPI functions in NFV infrastructures," *Int. J. Netw. Manag.*, vol. 25, pp. 490–506, Nov./Dec. 2015.

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

[22] "Network functions virtualisation (NFV): use cases," Tech. Rep., Oct. 2013. [Online]. Available: http://www.etsi.org/deliver/etsi_gs/nfv/001_099/001/01.01.01_60/gs_nfv001v010101p.pdf

[23] 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.

[24] 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.

[25] 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.

[26] L. Gong, Y. Wen, Z. Zhu, and T. Lee, "Toward profit-seeking virtual network embedding algorithm via global resource capacity," in *Proc. of INFOCOM 2014*, pp. 1–9, Apr. 2014.

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

[28] 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.

[29] 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.

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

[31] I. Jang *et al.*, "Optimal network resource utilization in service function chaining," in *Proc. of NetSoft 2016*, pp. 11–14, Jun. 2016.

[32] S. Draxler, H. Karl, and Z. Mann, "Jasper: Joint optimization of scaling, placement, and routing of virtual network services," *IEEE Trans. Netw. Serv. Manag.*, vol. 15, pp. 946–960, Sept. 2018.

[33] C. Mouradian *et al.*, "Application component placement in NFV-based hybrid cloud/fog systems with mobile fog nodes," *IEEE J. Sel. Areas Commun.*, vol. 37, pp. 1130–1143, May 2019.

[34] D. Bhamare *et al.*, "Optimal virtual network function placement in multi-cloud service function chaining architecture," *Comput. Commun.*, vol. 102, pp. 1–16, Apr. 2017.

[35] 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.

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

[37] 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.

[38] M. Xia *et al.*, "Network function placement for NFV chaining in packet/optical datacenters," *J. Lightw. Technol.*, vol. 33, pp. 1565–1570, Apr. 2015.

[39] W. Fang *et al.*, "Joint defragmentation of optical spectrum and IT resources in elastic optical datacenter interconnections," *J. Opt. Commun. Netw.*, vol. 7, pp. 314–324, Mar. 2015.

[40] X. Chen *et al.*, "Leveraging mixed-strategy gaming to realize incentive-driven VNF service chain provisioning in broker-based elastic optical inter-datacenter networks," *J. Opt. Commun. Netw.*, vol. 10, pp. A232–A240, Feb. 2018.

[41] X. Chen, Z. Zhu, R. Proietti, and B. Yoo, "On incentive-driven VNF service chaining in inter-datacenter elastic optical networks: A hierarchical game-theoretic mechanism," *IEEE Trans. Netw. Serv. Manag.*, vol. 16, pp. 1–12, Mar. 2019.

[42] R. Cohen, L. Lewin-Eytan, J. Naor, and D. Raz, "Near optimal placement of virtual network functions," in *Proc. of INFOCOM 2015*, pp. 1346–1354, Apr. 2015.

[43] Y. Li, L. Phan, and B. Loo, "Network functions virtualization with soft real-time guarantees," in *Proc. of INFOCOM 2016*, pp. 1–9, Apr. 2016.

[44] Y. Sang *et al.*, "Provably efficient algorithms for joint placement and allocation of virtual network functions," in *Proc. of INFOCOM 2017*, pp. 1–9, May 2017.

[45] J. Zhang, W. Wu, and J. Lui, "On the theory of function placement and chaining for network function virtualization," in *Proc. of ACM MobiHoc 2018*, pp. 91–100, Jun. 2018.

[46] I. Jang, D. Suh, S. Pack, and G. Dan, "Joint optimization of service function placement and flow distribution for service function chaining," *IEEE J. Sel. Areas Commun.*, vol. 35, pp. 2532–2541, Nov. 2017.

[47] H. Moens and F. De Turck, "VNF-P: A model for efficient placement of virtualized network functions," in *Proc. of CNSM 2014*, pp. 418–423, Nov. 2014.

[48] B. Yi, X. Wang, M. Huang, and A. Dong, "A multi-stage solution for NFV-enabled multicast over the hybrid infrastructure," *IEEE Commun. Lett.*, vol. 21, pp. 2061–2064, Sept. 2017.

[49] X. Liu, L. Gong, and Z. Zhu, "Design integrated RSA for multicast in elastic optical networks with a layered approach," in *Proc. of GLOBECOM 2013*, pp. 2346–2351, Dec. 2013.

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

[51] T. Cormen, C. Leiserson, R. Rivest, and C. Stein, *Introduction to Algorithms*. MIT press, 2009.

[52] D. Goldfarb and M. Todd, "Modifications and implementation of the ellipsoid algorithm for linear programming," *Math. Program.*, vol. 23, pp. 1–19, Dec. 1982.

[53] G. Strang, "Karmarkar's algorithm and its place in applied mathematics," *Math. Intell.*, vol. 9, pp. 4–10, Jan. 1987.

[54] D. Dubhashi and A. Panconesi, *Concentration of Measure for the Analysis of Randomized Algorithms*. Cambridge University Press, 2009.

[55] (2019) Amazon web services. [Online]. Available: https://aws.amazon.com/cn/ec2/pricing/reserved-instances/pricing/

[56] (2019) Colfax direct. [Online]. Available: http://www.colfaxdirect.com/store/pc/viewPrd.asp?idproduct=3017&idcategory=0

[57] J. Halpern and C. Pignataro, "Service function chaining (SFC) architecture," *RFC 7655*, Oct. 2015. [Online]. Available: https://tools.ietf.org/html/rfc7665.