Two-Phase Virtual Network Function Selection and Chaining Algorithm Based on Deep Learning in SDN/NFV-Enabled Networks

Jianing Pei, Peilin Hong, Kaiping Xue, *Senior Member, IEEE*, Defang Li, David S. L. Wei, *Senior Member, IEEE*, and Feng Wu, *Fellow, IEEE*

***Abstract*— With the advances of Software-Defined Networks (SDN) and Network Function Virtualization (NFV), Service Function Chain (SFC) has been becoming a popular paradigm to carry and complete network services. Such new computing and networking paradigm enables Virtual Network Functions (VNFs) to be placed in software entities/virtual machines over a network of physical equipments in elastic and flexible way with low capital and operation expenses. VNFs are chained together to steer traffic as needed. However, most of the existing traffic steering and routing path computation algorithms for SFC are complex, unscalable, and low time-efficiency. In this paper, we study the VNF Selection and Chaining Problem (VNF-SCP) in SDN/NFV-enabled networks. We formulate VNF-SCP as a Binary Integer Programming (BIP) model in order to compute routing path for each SFC Request (SFCR) with the minimum end-to-end delay. Then, a novel Deep Learning-based Two-Phase Algorithm (DL-TPA) is introduced, where VNF selection network and VNF chaining network are designed to achieve intelligent and efficient VNF selection and chaining for SFCRs. Performance evaluation shows that DL-TPA can achieve high prediction accuracy and time efficiency of routing path computation, and the overall network performance can be improved significantly.**

***Index Terms*— Software-defined networks, network function virtualization, VNF selection and chaining, routing path computation, deep learning.**

I. Introduction

NETWORK Functions (NFs) can be ubiquitously placed in enterprise networks to provide various services and enhance the performance, security and manageability [1], [2]. Traditional NFs for firewall, deep package inspection, intrusion detection, load balance, wide area network acceleration, and so on are mostly carried by specific physical equipments. Due to this tight coupling, Internet Service Providers (ISPs) have to spend expensive Capital Expense (CAPEX) to continuously purchase new physical equipments to satisfy increasing performance demands of users [3]. In addition, it is difficult for ISPs to timely configure, manage and optimize these NFs, which usually leads to long update cycle of network service and high Operation Expenses (OPEX) [4], [5].

Software-Defined Networks (SDN) and Network Function Virtualization (NFV) have emerged as a promising way to address the above mentioned limitations. SDN is a new networking paradigm which can separate control plane from data plane and conduct centralized management via SDN controllers [6], [7]. Meanwhile, NFV technology can decouple NFs from specific physical equipments based on virtualization technology and dynamically place Virtual Network Functions (VNFs) in appropriate locations of the network to provide specific services for users [8]. VNFs are realized in software and running on Commercial-Off-The-Shelf (COTS) devices, which can be managed by SDN controllers. Compared with traditional NFs carried by specific physical equipments, VNFs have strong potential in significantly reducing OPEX/CAPEX and enhancing service flexibility [2], [9].

Based on SDN/NFV technologies, Service Function Chain (SFC), standardized by Internet Engineering Task Force (IETF), defines a set of ordered or partially ordered VNFs, and for an SFC Request (SFCR), the traffic needs to be steered to traverse a sequence of specified VNFs in a predefined order [10], [11]. With the optimal VNF selection and chaining strategies, the performance and cost-effectiveness of SFC services can be improved [12]. For example, in Fig. 1, there are five types of VNFs in SDN/NFV-enabled networks and each type of VNF has multiple instances placed in different network locations. The parameters VNF_a^1, VNF_a^2 and VNF_a^3 respectively represent the first, second and third VNF instances of VNF_a. An SFCR starting from the ingress node *A* needs to sequentially traverse the instances of VNF_a, VNF_c, VNF_b, and VNF_d before reaching the egress node *J*. However, there usually exist many paths (*e.g.,* the dotted lines with different colors in Fig. 1) that can satisfy the predefined order of VNFs about this SFCR. Therefore, it is challenging to design optimal VNF selection and chaining strategies to steer the traffic of SFCRs.

Furthermore, SDN controllers conduct centralized path management for all OpenFlow switches, so they are easy to become a single bottleneck of performance. Therefore, based

Manuscript received April 15, 2019; revised December 3, 2019; accepted January 29, 2020. Date of publication April 9, 2020; date of current version May 21, 2020. This work was supported in part by the National Natural Science Foundation of China under Grant 61671420 and Grant 61972371, and in part by the Youth Innovation Promotion Association of the Chinese Academy of Sciences (CAS) under Grant 2016394. *(Corresponding author: Kaiping Xue.)*

Jianing Pei, Peilin Hong, Kaiping Xue, Defang Li, and Feng Wu are with the Department of Electronic Engineering and Information Science, University of Science and Technology of China, Hefei 230027, China (e-mail: jianingp@mail.ustc.edu.cn; plhong@ustc.edu.cn; kpxue@ustc.edu.cn; fengwu@ustc.edu.cn; ldf911@mail.ustc.edu.cn).

David S. L. Wei is with the Computer and Information Science Department, Fordham University, Bronx, NY 10458 USA (e-mail: wei@cis.fordham.edu).

Color versions of one or more of the figures in this article are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/JSAC.2020.2986592

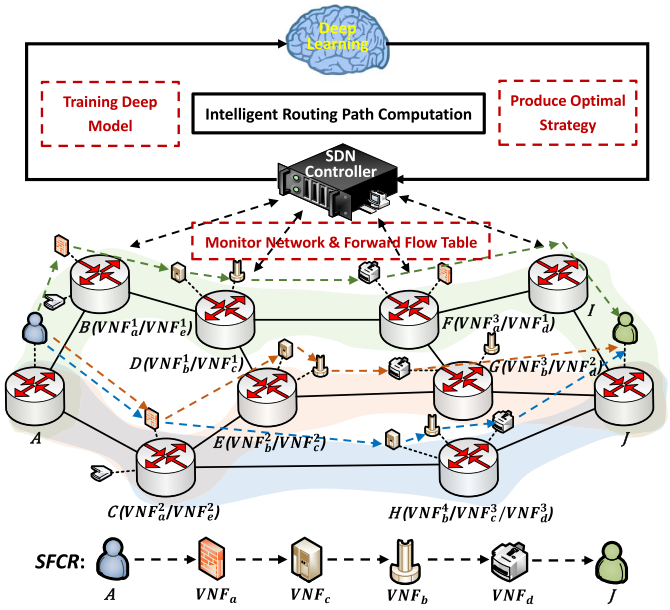


Fig. 1. VNF selection and chaining for SFCRs with deep learning technology in SDN/NFV-enabled networks.

on the basic SDN architecture, the functionality enhancement is not supposed to introduce unbearable overhead. However, in order to get the optimal strategies to steer the traffic of SFCRs, the existing rule-based routing algorithms developed for SDN controllers usually realize traffic path management based on massive iteration and computation, which incurs high computation overhead and low time-efficiency [13]. Thus, for SFCRs, ISPs should re-consider to introduce new routing path computation algorithms with low computational complexity and high time-efficiency so as to cope with tremendously increasing traffic.

Meanwhile, witnessing the breakthrough and significant performance gains, deep learning has obtained remarkable achievements in various domains, such as computer vision, natural language processing, self-driving and strategic game playing [14], [15]. As the traffic path management is related to network conditions, the hidden rules behind the routing path computation should be deeply mined to speed up the optimal path problem solving during traffic steering. Fortunately, due to powerful learning capacity and performance optimization in software and hardware [15], deep learning can efficiently discover and characterize the structural features of complex problems. Given large scale of training data and the objective of fine-grained feature extraction and classification, deep learning is more suitable than conventional machine learning algorithms to realize routing path computation for SFCRs [16], [17]. Additionally, based on deep learning, many network-related factors, parameters and metrics can be considered to design a fine-grained routing path computation strategy for SFCRs in a more intelligent and autonomous manner. Therefore, by introducing deep learning, it is hopeful to integrate intelligence with the network technology to avoid massive iteration and computation to conduct intelligent routing path computation, which would likely outperform traditional rule-based routing algorithms [18], [19].

As shown in Fig. 1, SDN and deep learning technologies are integrated to realize intelligent routing path computation for SFCRs. The network condition data can be timely collected by SDN controller and be used to train deep models. Then, the optimal strategies can be produced from deep models, thereby achieving efficient network management.

In this paper, we study the VNF Selection and Chaining Problem (VNF-SCP). As the network load in SDN/NFV-enabled networks changes dynamically, the routing strategies should be dynamically optimized according to real time network load to avoid bottlenecks and enhance the QoS of network and the QoE of users [6], [7]. By introducing deep learning to address VNF-SCP, the key technical problem is how to make the optimal VNF selection and chaining strategies for SFCRs with high time efficiency and low computation complexity. To solve VNF-SCP, we firstly formulate it as a Binary Integer Programming (BIP) model aiming to minimize the end-to-end delay for each SFCR. Then, we propose a novel Deep Learning-based Two-Phase Algorithm (DL-TPA) to solve this problem. The deep models of DL-TPA are constructed based on Deep Belief Networks (DBNs) and can achieve routing path computation for SFCRs in batches, which consists of two parts, respectively named as VNF selection network and VNF chaining network. Both supervised and unsupervised learning algorithms are used to train these deep models, where the training data are generated by running an optimal algorithm. The running process has two phases: 1) select the optimal VNF instances in VNF selection network; 2) use the VNF chaining network to concatenate the selected VNF instances and construct complete routing paths of SFCRs to satisfy predefined orders and constraints. The contributions of our work are listed below:

- We give a detailed analysis on VNF Selection and Chaining Problem (VNF-SCP) in SDN/NFV-enabled networks and formulate it as a BIP model aiming to minimize the end-to-end delay for each SFCR.
- We propose a deep learning-based two-phase VNF selection and chaining algorithm, DL-TPA, to solve this problem. In DL-TPA, based on available resources and different SFCRs, we generate training data by solving the BIP model with an optimal algorithm [20], [21]. We design two types of DBN networks, named as VNF selection network and VNF chaining network, and train them to solve VNF-SCP in two phases. Furthermore, given the relationship of SFCRs and network topology, we reduce the computation complexity of output layers of VNF selection network and VNF chaining network with the optimization of solution space size.
- We conduct a theoretical analysis on the effectiveness of DL-TPA and further construct a Tensorflow-based [22] simulation environment to verify the efficiency of our scheme.

The rest of this paper is organized as follows: Section II reviews the related works and introduces the technical background of DBN. We formulate VNF-SCP in Section III, then propose our DL-TPA algorithm in Section IV. The effectiveness of DL-TPA is analyzed and compared with

existing approaches in Section V. Finally, Section VI concludes the paper.

II. Related Works And Technical Background

A. Related Works

In recent years, traffic steering and routing path computation problem for SFCRs has become a hot issue in academia, and various solutions have been proposed [13], [20], [21], [23]–[28]. Considering the resource consumptions on links and nodes in distributed cloud environments, Mechtri *et al.* [13] proposed a novel eigendecomposition based approach to address the placement and chaining problems for SFCs. Yu *et al.* [20] and Dwaraki Wolf [21], respectively, proposed to compute routing paths by concatenating specified VNF instances in predefined orders with graph layering approaches. Jiao *et al.* [23] considered the VNF selection and traffic steering problem for SFC and formulated it as an Integer Linear Programming (ILP) model, where the objective is to maximize the network throughput in SDN/NFV-enabled environment. Huang *et al.* [24], [25] leveraged markov approximation technique to maximize the profit and cost-efficient utility. Cheng *et al.* [26] studied service chain instantiation problem and proposed a simulated annealing algorithm to get the optimal solution. Pei *et al.* [27] proposed to construct routing paths for SFCRs and achieved load balancing by considering multi-resource constraints and flow features. Furthermore, the commercial SDN controller, opendaylight, has supported four SFC scheduling algorithms including Random, Round Robin, Load Balance and Shortest Path [28]. However, all these mentioned solutions are rule-based and cannot achieve intelligent traffic steering for SFCRs, which usually incurs complex strategy design and low time-efficiency in routing path computation.

Faced with tremendous growth of network traffic and the declining profits of ISPs, deep learning technology appears to be a viable approach for efficient routing path computation with intelligent traffic management [12]. Mao *et al.* [16] proposed a deep learning architecture to achieve efficient routing path computation for high-speed core network. Kato *et al.* [19] presented a DBN based deep learning system and applied it to improve heterogeneous network traffic control. In order to simplify the design of routing strategies when considering exponentially increased traffic, Mao *et al.* [29] further proposed a tensor-based deep belief architecture to achieve fine-grained network traffic control. In our previous work [30], we proposed a hop-by-hop approach based on DBN to compute routing paths for SFCRs, but the number of deep models needed to be trained is proportional to the square of the number of switches and routers, which leads to flexibility and scalability problem in large-scale networks.

For the utilization of deep learning in routing path computation and traffic management, literatures [16], [19] and [29] also encounter the flexibility and scalability problems as that in [30]. Besides, they only consider the traffic routing problem in SDNs or conventional IP networks, which cannot fulfill the traffic steering and routing path computation for SFCRs in SDN/NFV-enabled environment. Different from the approaches mentioned above, our proposed scheme, DL-TPA, is a two-phase VNF selection and chaining algorithm based on deep learning technology. In DL-TPA, the VNF selection network and VNF chaining network are designed to compute the routing paths of SFCRs. The number of models needed to be trained in VNF selection network is proportional to the number of types of SFC services. And VNF chaining network only includes one deep model. Therefore, the DL-TPA is not sensitive to network scale, which makes DL-TPA efficiently overcome the flexibility and scalability problems encountered in literatures [16], [19], [29], [30]. In addition, the solution space optimization is also considered in VNF selection network and VNF chaining network to reduce the computation complexity, making DL-TPA achieve efficient routing path computation of SFCRs in large-scale networks.

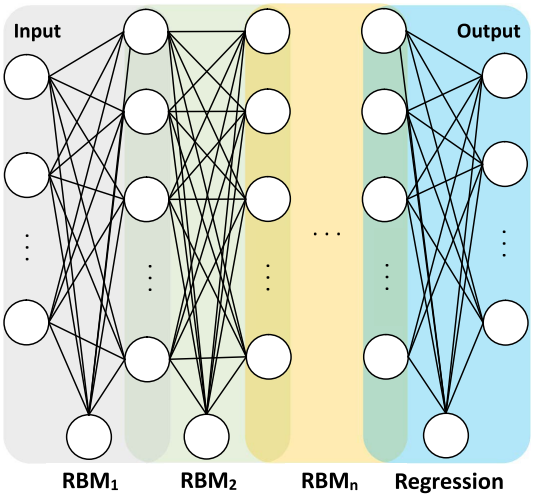


Fig. 2. Structure of DBN.

B. Deep Belief Network (DBN)

Among all deep learning models, DBN proposed by Hinton is regraded as the most common and effective approach to efficiently extract and learn the features from data. DBN consists of many layers of hidden causal variables, which is equivalent to a stacked Restricted Boltzmann Machine (RBM) model with a regression layer on the top [31]. Fig. 2 depicts the structure of DBN. The training process of DBN can be divided into two steps: 1) train RBMs with a fast greedy layer-wise unsupervised learning algorithm; 2) use training data to fine-tune the whole network with back-propagation algorithm [32]. Here, the first step aims to initialize parameters, which is helpful for the convergence of the fine-tuning process, and the second step aims to further optimize the parameters to fit training data.

An RBM can be described by a two-layer network. For an RBM shown in the upper box of Fig. 3, it consists of a visible layer $\boldsymbol{v}$ and a hidden layer $\boldsymbol{h}$. We use v_i and h_j to represent the i^{th} and j^{th} neurons of $\boldsymbol{v}$ and $\boldsymbol{h}$, respectively. The parameter set of an RBM is denoted as $\boldsymbol{\theta} = (\boldsymbol{w}, \boldsymbol{a}, \boldsymbol{b})$, where w_{ji} represents the weight between v_i and h_j. The biases of v_i and h_j are denoted as a_i and b_j, respectively. In addition, a log-likelihood

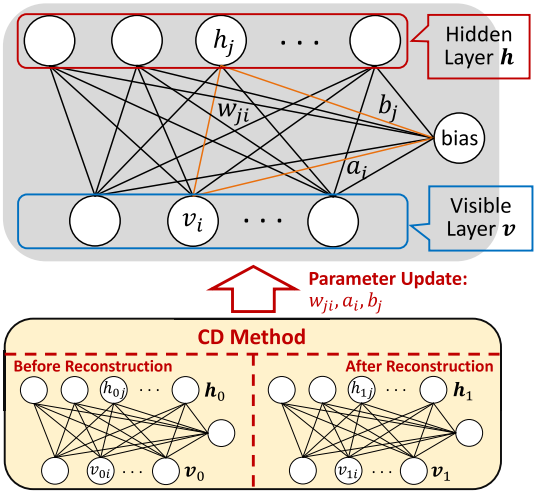


Fig. 3. Structure of RBM and CD Method.

function in Eq. (1) is used to model the training process of an RBM, where $\boldsymbol{x}$ stands for the labeled input data and its probability is represented as $P(\boldsymbol{v}|\boldsymbol{\theta})$:

$$\mathcal{L}(\boldsymbol{\theta}) = \sum_{\boldsymbol{x}} logP(\boldsymbol{v}|\boldsymbol{\theta}). \tag{1}$$

When training RBM, the objective is to maximize the log-likelihood function and obtain the optimal parameter $\boldsymbol{\theta}^*$ as:

$$\boldsymbol{\theta}^* = \arg\max_{\boldsymbol{\theta}} \mathcal{L}(\boldsymbol{\theta}) = \arg\max_{\boldsymbol{\theta}} \sum_{\boldsymbol{x}} logP(\boldsymbol{v}|\boldsymbol{\theta}). \tag{2}$$

In a binary RBM (the value of a neuron is 0 or 1), as there are no direct connections between neurons in the same layer, the conditional probability distributions that h_j and v_i are activated can be easily computed using Eq. (3) and Eq. (4), respectively, where $sigm(\cdot)$ stands for a sigmoid activation function. The parameters $N_{\boldsymbol{v}}$ and $N_{\boldsymbol{h}}$ represent the numbers of neurons of $\boldsymbol{v}$ and $\boldsymbol{h}$.

$$P(h_j = 1|\boldsymbol{v}) = sigm\left(\sum_{i=1}^{N_{\boldsymbol{v}}} w_{ji}v_i + b_j\right), \tag{3}$$

$$P(v_i = 1|\boldsymbol{h}) = sigm\left(\sum_{j=1}^{N_{\boldsymbol{h}}} w_{ji}h_j + a_i\right). \tag{4}$$

Once the states of visible neurons are fixed, the hidden neurons can be computed with Eq. (3). Similarly, after fixing the states of hidden neurons, we can compute the visible neurons with Eq. (4) as well. Here, the Contrastive Divergence (CD) method [31], [33] is used to complete these steps which optimize the parameters of an RBM. We depict the states of an RBM before and after reconstruction with CD method in the lower box of Fig. 3. The parameters $\boldsymbol{v_0}$, $\boldsymbol{h_0}$ and $\boldsymbol{v_1}$, $\boldsymbol{h_1}$ represent the visible layer and hidden layer of an RBM before and after reconstruction, respectively. In CD method, we first get $\boldsymbol{v_0}$ by inputting training data into visible layer, then compute hidden layer $\boldsymbol{h_0}$ based on Eq. (3). After that, according to hidden layer $\boldsymbol{h_0}$, we can reconstruct visible layer $\boldsymbol{v_1}$ according to Eq. (4), then reconstruct hidden layer $\boldsymbol{h_1}$ based on Eq. (3). We use v_{0i}, h_{0j} and v_{1i}, h_{1j} to indicate the i^{th} and j^{th} neurons of $\boldsymbol{v_0}$, $\boldsymbol{h_0}$ and $\boldsymbol{v_1}$, $\boldsymbol{h_1}$, respectively. The parameters of an RBM with CD method can be updated using Eqs. (5)-(7). Here, $N_{\boldsymbol{x}}$ denotes the total number of training data items. The parameter η^{rbm} represents the learning rate of training RBMs. $P(h_{0j} = 1|\boldsymbol{v_0})$ and $P(h_{1j} = 1|\boldsymbol{v_1})$ represent the possibilities that h_{0j} equals 1 and h_{1j} equals 1 according to $\boldsymbol{v_0}$ and $\boldsymbol{v_1}$, respectively.

$$a_i = a_i + \frac{\eta^{rbm}}{N_{\boldsymbol{x}}}(v_{0i} - v_{1i}), \tag{5}$$

$$b_j = b_j + \frac{\eta^{rbm}}{N_{\boldsymbol{x}}}[P(h_{0j} = 1|\boldsymbol{v_0}) - P(h_{1j} = 1|\boldsymbol{v_1})], \tag{6}$$

$$w_{ji} = w_{ji} + \frac{\eta^{rbm}}{N_{\boldsymbol{x}}}[P(h_{0j}=1|\boldsymbol{v_0})v_{0i} - P(h_{1j}=1|\boldsymbol{v_1})v_{1i}]. \tag{7}$$

As stated before, there are two steps to train DBN including unsupervised learning and supervised fine-tuning processes. In unsupervised learning process, first, we train one RBM each time from left to right in Fig. 2 with the steps stated above. If the training process of an RBM is finished, its hidden layer will be used as the visible layer of next RBM. And we repeat the training process until all the RBMs of DBN have been trained. In the supervised fine-tuning process, we treat the DBN as a neural network, and we input the training data into DBN and get the output. By comparing the output of DBN with the corresponding labels, we can get the error and run the back-propagation algorithm to adjust the parameters of the whole network.

III. Problem Formulation

A. System Model

In this paper, the physical network is considered as an undirected graph, $G = (\mathcal{V}, \mathcal{E})$. We use $u, v \in \mathcal{V}$ to represent two physical nodes, and $uv \in \mathcal{E}$ represents a physical link. There are a series of VNF instances placed in the network and $\mathcal{M}$ stands for the set of all VNF instances. The bandwidth capacity of link $uv \in \mathcal{E}$ is denoted as C_{uv}^{bw}. We denote C_u^{mem} as the memory capacity of $u \in \mathcal{V}$, and C_m^{cpu} indicates the CPU capacity that the VNF instance $m \in \mathcal{M}$ can apply from the corresponding node. The available resource ratios of link $uv \in \mathcal{E}$, node $u \in \mathcal{V}$, and VNF instance $m \in \mathcal{M}$ are symbolized as r_{uv}^{bw}, r_u^{mem} and r_m^{cpu}, respectively. All the symbols and variables of this section are listed in Table I.

As shown in Fig. 1, each SFCR consists of an ingress node, an egress node and a sequence of VNF requests. We use a directed service function graph, $\bar{G}_f = (\bar{\mathcal{V}}_f, \bar{\mathcal{E}}_f)$, to represent *SFCR*$_f$. In $\bar{G}_f$, $\bar{u}, \bar{v} \in \bar{\mathcal{V}}_f$ stand for two nodes and $\bar{u}\bar{v} \in \bar{\mathcal{E}}_f$ represents the link between nodes $\bar{u}$ and $\bar{v}$. We use S_f and T_f to represent the ingress and egress nodes of *SFCR*$_f$. For SFCRs, they need to consume network resources (*e.g.,* bandwidth, memory, CPU, *etc*) to be served in the network [5]. We define Ψ_f^{bw} as the bandwidth consumption of *SFCR*$_f$. Ψ_f^{mem} and Ψ_f^{cpu} are used to represent the consumptions of memory and CPU of *SFCR*$_f$. When dealing with *SFCR*$_f$, d_u, d_{uv}, and d_m denote the delays suffered in $u \in \mathcal{V}$, $uv \in \mathcal{E}$ and $m \in \mathcal{M}$, respectively. The maximum tolerable delay of *SFCR*$_f$ is symbolized as Ψ_f^{td}.

TABLE I
SYMBOLS AND VARIABLES OF BIP MODEL

Symbols and Variables	Description
Physical Network	
$G=(\mathcal{V},\mathcal{E})$	Physical network G with the sets of nodes $\mathcal{V}$ and links $\mathcal{E}$, $u,v\in\mathcal{V}$, $uv\in\mathcal{E}$.
C_{uv}^{bw} ,C_{u}^{mem}, C_{m}^{cpu}	Capacities of bandwidth, memory and CPU of link $uv\in\mathcal{E}$, node $u\in\mathcal{V}$ and VNF instance $m\in\mathcal{M}$.
r_{uv}^{bw}, r_{u}^{mem}, r_{m}^{cpu}	Available ratios of bandwidth, memory and CPU of link $uv\in\mathcal{E}$, node $u\in\mathcal{V}$ and VNF instance $m\in\mathcal{M}$.
d_{uv}, d_u, d_m	Delay on link $uv\in\mathcal{L}$, node $u\in\mathcal{V}$ and VNF instance $m\in\mathcal{M}$.
$\mathcal{M}$	Set of all the VNF instances, $m\in\mathcal{M}$.
Service Function Graph	
$\bar{G}_f=(\bar{\mathcal{V}}_f,\bar{\mathcal{E}}_f)$	Service function graph $\bar{G}_f$ with the sets of nodes $\bar{\mathcal{V}}_f$ and links $\bar{\mathcal{E}}_f$, $\bar{u},\bar{v}\in\bar{\mathcal{V}}_f$, $\bar{u}\bar{v}\in\bar{\mathcal{E}}_f$.
S_f, T_f	The ingress node and egress node of *SFCR*$_f$.
Ψ_f^{bw}, Ψ_f^{mem}, Ψ_f^{cpu}, Ψ_f^{td}	Consumptions of bandwidth, memory, CPU and the maximum tolerable delay of *SFCR*$_f$.
Binary Variables	
$z_{f,uv}^{\bar{u}\bar{v}}$	Whether $\bar{u}\bar{v}\in\mathcal{E}_f$ traverses link $uv\in\mathcal{E}$.
$z_{f,u}^{\bar{u}\bar{v}}$	Whether $\bar{u}\bar{v}\in\mathcal{E}_f$ traverses node $u\in\mathcal{V}$.
$z_{f,m}^{\bar{u}}$	Whether $\bar{u}\in\mathcal{V}_f$ is served by VNF instance $m\in\mathcal{M}$.
y_u^m	Whether VNF instance $m\in\mathcal{M}$ is placed in physical node $u\in\mathcal{V}$.

B. BIP Model

In this subsection, we formulate the VNF-SCP as a BIP model in detail.

To serve *SFCR*$_f$, the available resources in $uv\in\mathcal{E},u\in\mathcal{V}$ and $m\in\mathcal{M}$ should be sufficient as:

$$\sum_{\bar{u}\bar{v}\in\bar{\mathcal{E}}_f}\Psi_f^{bw}z_{f,uv}^{\bar{u}\bar{v}}\le r_{uv}^{bw}C_{uv}^{bw},\quad\forall uv\in\mathcal{E},\tag{8}$$

$$\sum_{\bar{u}\bar{v}\in\bar{\mathcal{E}}_f}\Psi_f^{mem}z_{f,u}^{\bar{u}\bar{v}}\le r_{u}^{mem}C_{u}^{mem},\quad\forall u\in\mathcal{V},\tag{9}$$

$$\sum_{\bar{u}\in\bar{\mathcal{V}}_f}\Psi_f^{cpu}z_{f,m}^{\bar{u}}\le r_{m}^{cpu}C_{m}^{cpu},\quad\forall m\in\mathcal{M}.\tag{10}$$

Here, the binary variables $z_{f,u}^{\bar{u}\bar{v}}$ and $z_{f,uv}^{\bar{u}\bar{v}}$ are used to indicate whether $\bar{u}\bar{v}\in\bar{\mathcal{E}}_f$ traverses node $u\in\mathcal{V}$ and link $uv\in\mathcal{E}$, respectively. The variables $z_{f,u}^{\bar{u}\bar{v}}$ and $z_{f,uv}^{\bar{u}\bar{v}}$ equal 1, if $\bar{u}\bar{v}\in\bar{\mathcal{E}}_f$ traverses the node $u\in\mathcal{V}$ and link $uv\in\mathcal{E}$, and 0 otherwise. Also, binary variable $z_{f,m}^{\bar{u}}$ is used to indicate whether $\bar{u}\in\bar{\mathcal{V}}_f$ is served by VNF instance $m\in\mathcal{M}$. $z_{f,m}^{\bar{u}}$ equals 1, if $\bar{u}\in\bar{\mathcal{V}}_f$ is served by VNF instance $m\in\mathcal{M}$, and 0 otherwise.

As shown in Eq. (11), the total end-to-end delay which consists of the delay of links, nodes and VNF instances in a path cannot exceed the maximum tolerable delay of *SFCR*$_f$:

$$\sum_{uv\in\mathcal{E}}\sum_{\bar{u}\bar{v}\in\bar{\mathcal{E}}_f}d_{uv}z_{f,uv}^{\bar{u}\bar{v}}+\sum_{u\in\mathcal{V}}\sum_{\bar{u}\bar{v}\in\bar{\mathcal{E}}_f}d_{u}z_{f,u}^{\bar{u}\bar{v}}+\sum_{m\in\mathcal{M}}\sum_{\bar{u}\in\bar{\mathcal{V}}_f}d_{m}z_{f,m}^{\bar{u}}\le\Psi_f^{td}.\tag{11}$$

Eq. (12) ensures that we can get a consecutive path of *SFCR*$_f$ by mapping the service function graph to physical network:

$$\sum_{v\in\mathcal{V}}\sum_{\bar{u}\bar{v}\in\bar{\mathcal{E}}_f}\left(z_{f,uv}^{\bar{u}\bar{v}}-z_{f,vu}^{\bar{u}\bar{v}}\right)=\begin{cases}1, & u=S_f,\\ -1, & u=T_f,\\ 0, & \text{otherwise.}\end{cases}\tag{12}$$

In Eq. (13), if a link $uv\in\mathcal{E}$ is selected, both its end nodes u and v must be selected as well:

$$z_{f,u}^{\bar{u}\bar{v}}z_{f,v}^{\bar{u}\bar{v}}=\begin{cases}1, & z_{f,uv}^{\bar{u}\bar{v}}=1,\ \forall u,v\in\mathcal{V},\forall uv\in\mathcal{E},\forall\bar{u}\bar{v}\in\bar{\mathcal{E}}_f,\\ 0, & \text{otherwise.}\end{cases}\tag{13}$$

We should ensure that each VNF instance $m\in\mathcal{M}$ can be only placed on one node as:

$$\sum_{m\in\mathcal{M}}y_u^m=1,\quad\forall u\in\mathcal{V},\tag{14}$$

where binary variable y_u^m is used to indicate whether VNF instance $m\in\mathcal{M}$ is placed on node $u\in\mathcal{V}$. And y_u^m equals 1, if $m\in\mathcal{M}$ is placed on $u\in\mathcal{V}$, and 0 otherwise.

For *SFCR*$_f$, the path should traverse all the nodes that have the selected VNF instances:

$$z_{f,u}^{\bar{u}\bar{v}}=1,\text{if } z_{f,m}^{\bar{u}}y_u^m=1,\quad\forall u\in\mathcal{V},\ \forall\bar{u}\in\bar{\mathcal{V}}_f,\ \forall\bar{u}\bar{v}\in\bar{\mathcal{E}}_f,\ \forall m\in\mathcal{M}.\tag{15}$$

Each VNF request $\bar{u}$ of *SFCR*$_f$ can be only served by one VNF instance, which can be ensured as:

$$\sum_{m\in\mathcal{M}}z_{f,m}^{\bar{u}}y_u^m\le 1,\quad\forall u\in\mathcal{V},\ \forall\bar{u}\in\bar{\mathcal{V}}_f.\tag{16}$$

As shown in Eq. (17), the objective of our work is to obtain the path of *SFCR*$_f$ which has the minimum end-to-end delay and can satisfy all the constraints in Eqs. (8)-(16).

$$\underset{z_{f,uv}^{\bar{u}\bar{v}},z_{f,u}^{\bar{u}\bar{v}},z_{f,m}^{\bar{u}}}{\textit{Minimize}}\ \sum_{uv\in\mathcal{E}}\sum_{\bar{u}\bar{v}\in\bar{\mathcal{E}}_f}d_{uv}z_{f,uv}^{\bar{u}\bar{v}}+\sum_{u\in\mathcal{V}}\sum_{\bar{u}\bar{v}\in\bar{\mathcal{E}}_f}d_{u}z_{f,u}^{\bar{u}\bar{v}}+\sum_{m\in\mathcal{M}}\sum_{\bar{u}\in\bar{\mathcal{V}}_f}d_{m}z_{f,m}^{\bar{u}},\tag{17}$$
$$s.t.\ Eqs.\ (8)-(16).$$

As we know there exist some algorithms, such as those shown in [20], [21], proposed to solve Eq. (17). However, all these rule-based algorithms suffer from massive iteration and computation to compute routing paths, and have difficulty in flexibility and scalability. In this paper, deep learning technology is used to solve this problem. We use these rule-based algorithms to generate training data by solving the BIP model. After model training, we can achieve intelligent routing path computation to replace rule-based algorithms and efficiently avoid the disadvantages above.

IV. DEEP LEARNING-BASED TWO-PHASE ALGORITHM FOR ROUTING PATH COMPUTATION OF SFCRS

In this section, we first give an overview of the proposed DL-TPA, then show the structures of VNF selection network and VNF chaining network. After that, we give a description

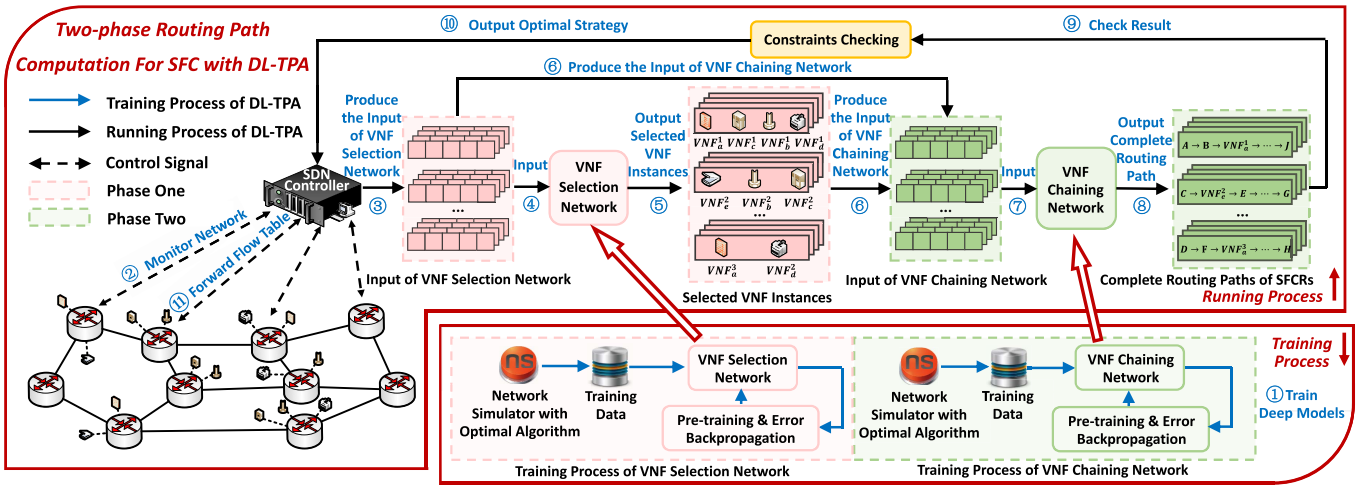


Fig. 4. Executing processes of intelligent routing path computation for SFC with DL-TPA.

on how to route the traffic of SFCRs in two phases with our intelligent approach. Finally, a detailed analysis on the computation complexity of DL-TPA is presented.

A. Overview of DL-TPA

Our proposed scheme is a deep learning-based two-phase VNF selection and chaining algorithm for SFCRs. It consists of two parts: VNF selection network and VNF chaining network. VNF selection network and VNF chaining network are both constructed based on DBNs and trained with the training data generated by running an optimal algorithm. The optimal algorithm used to generate training data is a rule-based algorithm based on the idea of graph layering. However, since it faces the problem of low-efficiency using the optimal algorithm to solve the BIP model, we propose DL-TPA, an intelligent algorithm based on deep learning, to achieve efficient routing path computation for SFCRs. Here, the optimal algorithm takes charge of the generation of training data which is used to train the VNF selection network and VNF chaining network in DL-TPA. The running process of DL-TPA solves VNF-SCP problem in two phases: 1) select the optimal VNF instances for SFCRs; 2) compute the paths to concatenate these selected VNF instances in predefined orders.

In DL-TPA, to achieve routing path computation for SFCRs, we firstly generate training data using the optimal algorithm, then train and run deep models, and finally check constraints. Fig. 4 shows the executing processes of DL-TPA. Here, the training process of DL-TPA is in lower right of the figure and the running process is in the upper left. Moreover, the pink part of Fig. 4 includes the training and running processes of VNF selection network and the training and running processes of VNF chaining network are illustrated in the green part. In the training process of step ①, a network simulator is constructed according to physical network and runs an optimal algorithm to generate training data by solving the BIP model. In training data, the input data include the information of SFCRs and network conditions, and the output data include the corresponding selected VNF instances and paths. Then, the training data are used to train the VNF selection network and VNF chaining network. In training process, the parameters of VNF selection network and VNF chaining network are optimized to fit training data with both unsupervised and supervised learning.

As for the running process of DL-TPA in Fig. 4, phase one including steps ③-⑤ is to select the optimal VNF instances for SFCRs. Here, step ② monitors the network using SDN controller and gets the information of SFCRs and network conditions. Next, the information gotten in step ② is formatted in step ③ and input into VNF selection network to obtain the optimal VNF instances in steps ④-⑤.

Given predefined orders, phase two including steps ⑥-⑧ aims to compute the paths to concatenate these selected VNF instances in phase one. In Fig. 4, step ⑥ formats the network conditions and the selected VNF instances output from VNF selection network in phase one. These formatted information is input into VNF chaining network in step ⑦. Step ⑧ computes the paths in VNF chaining network by concatenating these selected VNF instances in predefined orders.

After phase two, step ⑨ checks the complete routing paths with the constraints in Eqs. (8)-(16). If all these constraints are satisfied, these routing strategies are output in step ⑩, otherwise other candidate routing strategies will be checked and output. And we reject an SFCR, if there is no feasible routing strategy obtained to serve it or the iteration times reach a stopping threshold. Finally, the SDN controller forwards flow tables to the devices along these complete routing paths to steer the traffic of SFCRs in step ⑪.

B. Structure of VNF Selection Network

For different SFCRs, the VNF requests and pre-defined orders could be different. We define that all the SFCRs, with the same type of VNF requests and predefined order, belong to the same type of SFC service. For example, for two SFCRs both of which VNF requests and pre-defined orders are: $VNF_a \rightarrow VNF_b$, they belong to the same type of SFC services, and they belong to different SFC service, if their VNF requests or

TABLE II
SYMBOLS AND VARIABLES OF DL-TPA

Symbols and Variables	Description
VNF Selection Network	
k	The type of SFC service, $k = 1, 2, ..., K$.
$\Omega(k)$	Set of all the candidates of VNF instances that can probably be used by SFC service type k.
$\boldsymbol{x}_{s,k}, \widetilde{\boldsymbol{y}}_{s,k}$	Input and output of DBN k in VNF selection network.
$\boldsymbol{g}_f^{ing}, \boldsymbol{g}_f^{egr}$	Binary codes of the ingress and egress nodes of SFCR$_f$, and the length of binary code is q, $2^{q-1} \leq \lvert\mathcal{V}\rvert < 2^q$.
$\Psi_f^{bw*}, \Psi_f^{mem*}, \Psi_f^{cpu*}$	Normalization of bandwidth, memory and CPU consumptions of SFCR$_f$.
$M_{s,k}$	The number of VNF instance groups that can be used to serve SFC service type k.
$\boldsymbol{\beta}_{s,k}^L$	The output layer of DBN k in VNF selection network.
VNF Chaining Network	
$\boldsymbol{x}_c, \widetilde{\boldsymbol{y}}_c$	Input and output of VNF chaining network.
$\{s_1, s_2, ...\}$	A tuple including the ingress node, egress node and all the nodes that hold the selected VNF instances of SFCR$_f$.
$\boldsymbol{g}_f^{s_p}, \boldsymbol{g}_f^{s_{p+1}}$	Binary codes of (s_p, s_{p+1}), $s_p \in \{s_1, s_2, ...\}$, $p = 1, 2, ..., \lvert\bar{\mathcal{E}}_f\rvert$.
M_c	The number of paths that can be used to concatenate ingress nodes, egress nodes and selected VNF instances of SFCRs.
$\boldsymbol{\beta}_c^L$	The output layer of VNF chaining network.
DBN	
w_{ji}^l	The weight between neuron j in layer l and neuron i in layer $l-1$, $l = 2, 3, ..., L$, $i = 1, 2, ..., N_{l-1}$, $j = 1, 2, ...N_l$.
$\varphi_j^l, \delta_j^l, \gamma_j^l$	Weighted input, error and bias of neuron j in layer l.
$\boldsymbol{\beta}^l$	Output of layer l, β_i^l is the value of neuron i of $\boldsymbol{\beta}^l$.
$\boldsymbol{x}, \boldsymbol{y}$	Labeled input and output of DBN, and y_j means the value of j^{th} dimension of $\boldsymbol{y}$.
$N_{\boldsymbol{x}}$	Total number of training data items.
$sigm(\cdot)$	Sigmoid activation function.
$J\left(\boldsymbol{\theta}^{dbn}\right)$	Cross-entropy cost function of DBN model $\boldsymbol{\theta}^{dbn}$.

pre-defined orders are different. Then, in order to make the training process convenient and enhance the maintainability, we build a DBN for each type of SFC service in VNF selection network. Supposing that there are K types of SFC services provided to users, DBN k ($k = 1, \cdots, K$) is used to serve SFC service type k. In this section, we list all the symbols and variables in Table II.

In DL-TPA, it is important to characterize the input and output of DBNs in VNF selection network. As the available resources have influence on choosing the optimal VNF instances of SFCRs, the available resource ratios of nodes and links should be considered in the input of VNF selection network. Moreover, the available resources of VNF instances also have influence on routing path computation. As each kind of VNF has multiple instances placed in different network locations, we use $\Omega(k) \subseteq \mathcal{M}$ to denote the set of all the candidates of VNF instances that can probably be used by SFC service type k, and the available resource ratios of VNF instances $m \in \Omega(k)$ are included in the input of VNF selection network. In addition, users can set up SFCRs anywhere, and the resource consumption can be different. Therefore, the ingress and egress nodes and the resource consumptions of these SFCRs should be included in the input as well. The output of VNF selection network represents a set of groups of VNF instances. The output is defined as an one-hot vector, and each dimension represents a group of VNF instances that can serve this type of SFC service. Here, we define the group of VNF instances as VNF instance group. For example, both $\{VNF_a^1, VNF_c^1, VNF_b^1, VNF_d^1\}$ and $\{VNF_a^2, VNF_c^2, VNF_b^2, VNF_d^2\}$ are the VNF instance groups of the SFC service shown in Fig. 1. In the output, there is only one element that equals 1, which indicates that this VNF instance group is selected to serve the SFCR. All the information about the input and output data are generated by solving the BIP model using the optimal algorithm. According to the description above, for $SFCR_f$, the input and output are defined in Eqs. (18)-(19), where the vectors of $\boldsymbol{x}_{s,k}$ and $\widetilde{\boldsymbol{y}}_{s,k}$ represent the input and output of the k^{th} DBN of VNF selection network, respectively ($\forall uv \in \mathcal{E}$, $\forall u \in \mathcal{V}$, $\forall m \in \Omega(k)$).

$$\boldsymbol{x}_{s,k} = (\boldsymbol{g}_f^{ing}, \boldsymbol{g}_f^{egr}, \Psi_f^{bw*}, \Psi_f^{mem*}, \Psi_f^{cpu*}, r_{uv}^{bw}, \ldots, r_u^{mem}, \ldots, r_m^{cpu}, \ldots)^T, \tag{18}$$

$$\widetilde{\boldsymbol{y}}_{s,k} = (0, 0, 1, 0, \ldots, 0)^T. \tag{19}$$

Noting that the values of the neurons of DBN are in the range of $[0, 1]$, we encode $u \in \mathcal{V}$ based on binary coding. The binary codes of the ingress and egress nodes of $SFCR_f$ are denoted as $\boldsymbol{g}_f^{ing}$ and $\boldsymbol{g}_f^{egr}$. The length of the binary code q satisfies $2^{q-1} \leq |\mathcal{V}| < 2^q$. For example, in Fig. 1, the ingress node A and egress node J of $SFCR_f$ can be encoded as $\boldsymbol{g}_f^{ing} = (0, 0, 0, 1)$ and $\boldsymbol{g}_f^{egr} = (1, 0, 1, 0)$. The parameters $\Psi_f^{bw*}, \Psi_f^{mem*}, \Psi_f^{cpu*}$ represent the normalization of resource consumptions of $SFCR_f$ in links, nodes and VNF instances, respectively. Thus, the dimension of the input $\boldsymbol{x}_{s,k}$ equals $|\mathcal{V}| + |\mathcal{E}| + |\Omega(k)| + 2q + 3$. The dimension of $\widetilde{\boldsymbol{y}}_{s,k}$ is obtained according to the training data. If the number of VNF instance group that can be used to serve SFC service type k is $M_{s,k}$, the dimension of $\widetilde{\boldsymbol{y}}_{s,k}$ equals $M_{s,k}$

Since the normalization of resource consumptions Ψ_f^{bw*}, Ψ_f^{mem*}, Ψ_f^{cpu*} and available resource ratios r_{uv}^{bw}, r_u^{mem}, r_m^{cpu} are all continuous values, we replace the binary RBM$_1$ presented in Fig. 2 by real-valued neurons. Here, the Gaussian distribution is used to model the RBM$_1$ that consists of the input layer and the first hidden layer of a DBN. Then, for RBM$_1$, the conditional probability distribution in Eq. (4) is revised as:

$$P(v_i = 1|\boldsymbol{h}) = N\left(a_i + \sigma_i \sum_{j=1}^{N_{\boldsymbol{h}}} w_{ji} h_j, \sigma_i^2\right), \tag{20}$$

where σ_i is the standard deviation of v_i, and $N(\mu, \sigma^2)$ is Gaussian distribution.

Fig. 5 shows the structure of VNF selection network. The data formation layer aims to produce the input data $\boldsymbol{x}_{s,k}$ based on raw data. A softmax layer, which is always used to map

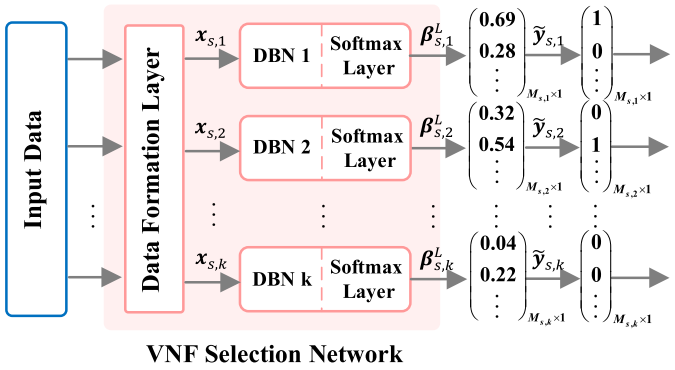


Fig. 5. Structure of VNF selection network.

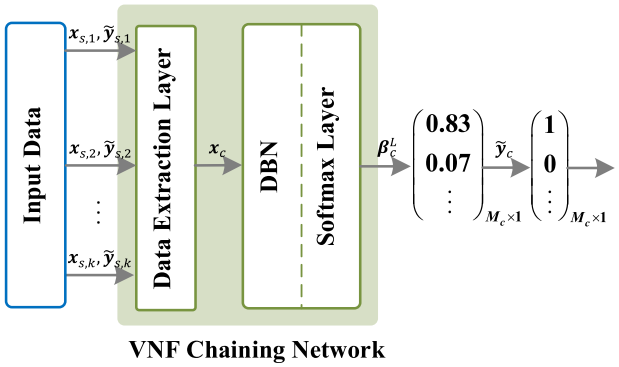


Fig. 6. Structure of VNF chaining network.

the values of neurons to a probability distribution, is set as the top layer of each DBN so as to perform supervised fine-tuning process on the whole deep model. To obtain the selected VNF instance group in VNF selection network, first, we get the output of the L^{th} layer of DBN k, $\boldsymbol{\beta}_{s,k}^{L}$, which represents the probability distributions of all the VNF instance groups with input data $\boldsymbol{x}_{s,k}$. Then, $\widetilde{\boldsymbol{y}}_{s,k}$ can be obtained by setting the element with the maximum probability in $\boldsymbol{\beta}_{s,k}^{L}$ as 1. After that, the VNF instance group whose corresponding value is 1 is output as the optimal solution to serve this SFCR.

C. Structure of VNF Chaining Network

After the process of VNF selection network, the VNF instance group of $SFCR_f$ can be selected. In this paper, we use a tuple $\{s_1, s_2, \ldots\}$ to represent the ingress node, egress node and all the nodes that hold the selected VNF instances. The number of elements in the tuple equals $|\bar{\mathcal{V}}_f|$. In the tuple, all the elements are arranged as the specified order of $SFCR_f$. For example, in Fig. 1, there are six elements in the tuple of $SFCR_f$. s_1 and s_6 represent the ingress node A and egress node J, respectively. Assuming that VNF selection network chooses the VNF instance group $\{VNF_a^1, VNF_c^1, VNF_b^1, VNF_d^1\}$ to serve $SFCR_f$, the parameters s_2, s_3, s_4, and s_5, respectively, represent the nodes B, D, D, F where these four VNF instances are placed.

The input of VNF chaining network includes the starting and ending points of a path, the bandwidth and memory consumptions and available resource ratios of links and nodes. As for $SFCR_f$ in Fig. 1, five paths are needed to be produced to chain the ingress and egress nodes and the selected VNF instances in predefined order. Supposing that the optimal VNF instance group output from VNF selection network is $\{VNF_a^1, VNF_c^1, VNF_b^1, VNF_d^1\}$ (green dotted lines in Fig. 1) which are placed on nodes B, D, D, F, respectively, the starting and ending points of these five paths are (A, B), (B, D), (D, D), (D, F) and (F, J). If the optimal paths with these five pairs of starting and ending points can be obtained, $SFCR_f$ can be served in the network.

The output of VNF chaining network represents a path set. It is also a one-hot vector where each element represents a path and there is only one element that equals 1. The path of which value in the output is 1 is chosen to chain the corresponding starting and ending points. For example, in Fig. 1, $D \rightarrow F$ and $D \rightarrow E \rightarrow G \rightarrow F$ represent two paths connecting (D, F), and they are included in the path set. In the output of VNF chaining network, if the value of the former path equals 1, the path $D \rightarrow F$ will be selected to steer the traffic of this SFCR. In Eq. (21) and Eq. (22), we denote the input and output of VNF chaining network as vectors $\boldsymbol{x}_c$ and $\widetilde{\boldsymbol{y}}_c$, respectively ($p = 1, 2, \ldots, |\bar{\mathcal{E}}_f|$, $\forall uv \in \mathcal{E}$, $\forall u \in \mathcal{V}$):

$$\boldsymbol{x}_c = \left(\boldsymbol{g}_f^{s_p}, \boldsymbol{g}_f^{s_{p+1}}, \Psi_f^{bw*}, \Psi_f^{mem*}, r_{uv}^{bw}, \ldots, r_u^{mem}, \ldots\right)^T, \quad (21)$$

$$\widetilde{\boldsymbol{y}}_c = (0, 1, 0, 0, \ldots, 0)^T, \quad (22)$$

where $\boldsymbol{g}_f^{s_p}$ and $\boldsymbol{g}_f^{s_{p+1}}$ represent the binary codes of the starting and ending points (s_p, s_{p+1}) of $SFCR_f$, $p = 1, 2, \ldots |\bar{\mathcal{E}}_f|$. The dimension of $\boldsymbol{x}_c$ equals $|\mathcal{V}|+|\mathcal{E}|+2q+2$. The dimension of $\widetilde{\boldsymbol{y}}_c$ is obtained from the training data. If there are M_c paths used to chain the ingress nodes, egress nodes and selected VNF instances of SFCRs, the dimension of $\widetilde{\boldsymbol{y}}_c$ is M_c.

VNF chaining network is a DBN with a softmax layer on the top. As the normalization of resource consumptions Ψ_f^{bw*} and Ψ_f^{mem*} and available resource ratios r_{uv}^{bw} and r_u^{mem} are continuous values, the RBM_1 that consists of the input layer and first hidden layer of VNF chaining network is modeled with Gaussian distribution. Its conditional probability distribution is the same as Eq. (20).

Fig. 6 shows the structure of VNF chaining network. The data extraction layer takes charge of collecting the input data in $\boldsymbol{x}_{s,k}$ and $\widetilde{\boldsymbol{y}}_{s,k}$, and produces input data $\boldsymbol{x}_c$. The output of softmax layer is denoted as $\boldsymbol{\beta}_c^L$ and it indicates the probability distribution of the path set based on $\boldsymbol{x}_c$. The parameter $\widetilde{\boldsymbol{y}}_c$ indicates the path chosen to chain the starting and ending points $(\boldsymbol{g}_f^{s_p}, \boldsymbol{g}_f^{s_{p+1}})$ in $\boldsymbol{x}_c$, which is obtained by setting the element with the maximum probability in $\boldsymbol{\beta}_c^L$ as 1.

D. Training Algorithm

In the training process of VNF selection network, we train a DBN for each type of SFC service. For VNF chaining network, we train a DBN model to achieve the path prediction. The training algorithm for a DBN is separated into two steps: firstly, we pre-train one RBM of a DBN each time with fast greedy layer-wise unsupervised learning algorithm. In this

step, the hidden neurons of a well trained RBM are used as the input of the next RBM. Secondly, the back-propagation algorithm is used to fine-tune the whole network.

The graph layering approach [20], [21] is used to generate the optimal paths of SFCRs. Then, the VNF instance group set can be obtained as the training data of VNF selection network, and we can also get the path set as the training data of VNF chaining network.

The supervised fine-tuning process aims to minimize the difference between the prediction and the labeled output. For each DBN of VNF selection network and VNF chaining network, the cross-entropy cost function with regularization penalty, as shown in Eq. (23), where the DBN model is denoted as $\boldsymbol{\theta}^{dbn}$, is used to describe the difference and avoid overfitting. The parameter $\boldsymbol{\beta}^L$ represents the output of softmax layer (in k^{th} DBN of VNF selection network, $\boldsymbol{\beta}^L = \boldsymbol{\beta}^L_{s,k}$; in the DBN of VNF chaining network, $\boldsymbol{\beta}^L = \boldsymbol{\beta}^L_c$), and w^l_{ji} indicates the weight between neuron j in layer l and neuron i in layer $l-1$. N_l denotes the number of neurons in layer l. $\boldsymbol{x}$ and $\boldsymbol{y}$ represent labeled input and output data. $N_{\boldsymbol{x}}$ indicates the total number of training data items and λ is a weighted parameter to control the two parts.

$$J\left(\boldsymbol{\theta}^{dbn}\right) = -\frac{1}{N_{\boldsymbol{x}}}\sum_{\boldsymbol{x}} \boldsymbol{y} log \boldsymbol{\beta}^L + \frac{\lambda}{2}\sum_{l=2}^{L}\sum_{j=1}^{N_l}\sum_{i=1}^{N_{l-1}}\left(w^l_{ji}\right)^2. \quad (23)$$

In fine-tuning process, we use φ^l_j to represent the weighted input of neuron j in layer l:

$$\varphi^l_j = \sum_{i=1}^{N_{l-1}} w^l_{ji}\beta^{l-1}_i + \gamma^l_j, \quad l = 2, \ldots, L. \quad (24)$$

where γ^l_j stands for the bias of neuron j of layer l and β^{l-1}_i denotes the value of i^{th} neuron of $\boldsymbol{\beta}^{l-1}$.

Then, we use δ^l_j to define the error of neuron j in layer l which can be used to optimize $\boldsymbol{\theta}^{dbn}$ in back-propagation algorithm. Eq. (25) shows the computation of δ^l_j, where $sigm'(\cdot)$ indicates the first derivative of sigmoid activation function and y_j means the value of j^{th} dimension of $\boldsymbol{y}$.

$$\delta^l_j = \begin{cases} \frac{1}{N_{\boldsymbol{x}}}\sum_{\boldsymbol{x}}\left(\beta^L_j - y_j\right), & l = L, \\ \sum_{i=1}^{N_l} w^{l+1}_{ji}\delta^{l+1}_i sigm'\left(\varphi^l_j\right), & \text{otherwise.} \end{cases} \quad (25)$$

Given Eqs. (24)-(25), we can update the parameters of $\boldsymbol{\theta}^{dbn}$ with the partial derivatives $\partial J\left(\boldsymbol{\theta}^{dbn}\right)/\partial w^l_{ji}$ and $\partial J\left(\boldsymbol{\theta}^{dbn}\right)/\partial \gamma^l_j$ based on Eq. (26) and Eq. (27), respectively.

$$\frac{\partial J\left(\boldsymbol{\theta}^{dbn}\right)}{\partial w^l_{ji}} = \beta^{l-1}_i\delta^l_j + \lambda w^l_{ji}, \quad (26)$$

$$\frac{\partial J\left(\boldsymbol{\theta}^{dbn}\right)}{\partial \gamma^l_j} = \delta^l_j. \quad (27)$$

The pseudo-code of training algorithms of DL-TPA is given in **Algorithms 1-4**. In **Algorithm 1**, firstly, training data are generated for VNF selection network and VNF chaining network in line 1, and recorded in $(\mathcal{X}_{s,k}, \mathcal{Y}_{s,k})$ and $(\mathcal{X}_c, \mathcal{Y}_c)$, respectively. Then, we train a DBN model for each type of SFC service in VNF selection network in lines 2-4 of **Algorithm 1**. Next, VNF chaining network is trained in line 5 of **Algorithm 1**. In the training process of DBN in **Algorithm 2**, lines 1-5 pre-train one RBM of a DBN each time. During the pre-training of RBM, first, we initialize related parameters in lines 1-3 of **Algorithm 3**. Then, fast greedy layer-wise unsupervised learning algorithm is used to train this RBM in lines 4-18 of **Algorithm 3**, where $\boldsymbol{x}^{(n)}$ stands for the n^{th} item of labeled input data. After that, back-propagation algorithm is executed to fine-tune the whole DBN in line 6 of **Algorithm 2**, and η^{bp} denotes its learning rate. **Algorithm 4** presents fine-tuning process. After the initialization of related parameters in lines 1-3 of **Algorithm 4**, the feedforward is performed in lines 4-20, where $x^{(n)}_j$ denotes the j^{th} dimension of $\boldsymbol{x}^{(n)}$. Then, the backpropagation is applied to optimize the parameters of DBN in lines 21-29 of **Algorithm 4**.

Algorithm 1 Training Algorithm of DL-TPA

Input: Learning rates: η^{rbm}, η^{bp}.
Output: VNF selection network: $\{\boldsymbol{\theta}^{dbn}_{s,k} | k = 1, \ldots, K\}$;
VNF chaining network: $\boldsymbol{\theta}^{dbn}_c$.
1: $(\mathcal{X}_{s,k}, \mathcal{Y}_{s,k})$, $(\mathcal{X}_c, \mathcal{Y}_c)$ $\leftarrow$ Generate training data with an optimal algorithm;
2: **for** $k = 1, \ldots, K$ **do**
3: $\boldsymbol{\theta}^{dbn}_{s,k} \leftarrow$ TrainDBN($(\mathcal{X}_{s,k}, \mathcal{Y}_{s,k})$, $\boldsymbol{\theta}^{dbn}_{s,k}$, η^{rbm}, η^{bp});
4: **end for**
5: $\boldsymbol{\theta}^{dbn}_c \leftarrow$ TrainDBN($(\mathcal{X}_c, \mathcal{Y}_c)$, $\boldsymbol{\theta}^{dbn}_c$, η^{rbm}, η^{bp});
6: **return** $\{\boldsymbol{\theta}^{dbn}_{s,k} | k = 1, \ldots, K\}$, $\boldsymbol{\theta}^{dbn}_c$;

Algorithm 2 TrainDBN

Input: Training data: $(\mathcal{X}, \mathcal{Y})$;
Learning rates: η^{rbm}, η^{bp}.
Output: DBN: $\boldsymbol{\theta}^{dbn}$.
1: $L \leftarrow$ Get the number of layers of DBN;
2: **for** $l = 1, \ldots, L-2$ **do**
3: $N_{\boldsymbol{v}}$, $N_{\boldsymbol{h}} \leftarrow$ Get the number of neurons of the l^{th} and $(l+1)^{th}$ layers, respectively;
4: $\boldsymbol{\theta}^{dbn} \leftarrow$ PreTrainRBM($\mathcal{X}$, $N_{\boldsymbol{v}}$, $N_{\boldsymbol{h}}$, η^{rbm});
5: **end for**
6: $\boldsymbol{\theta}^{dbn} \leftarrow$ FineTuneDBN($(\mathcal{X}, \mathcal{Y}), \boldsymbol{\theta}^{dbn}, \eta^{bp}$);
7: **return** $\boldsymbol{\theta}^{dbn}$;

E. Running Algorithm

In the running process, we first obtain the optimal VNF instance group from VNF selection network in the first phase, then get the paths to chain ingress node, egress node and those selected VNF instances using the VNF chaining network in the second phase.

The pseudo-code of the running process of DL-TPA is presented in **Algorithm 5**. Lines 1-5 of **Algorithm 5** initialize related parameters and get the optimal VNF instance group. Then, in lines 6-10 of **Algorithm 5**, we compute the optimal paths between the adjacent nodes in $\{s_1, s_2, \ldots\}$ in VNF

Algorithm 3 PreTrainRBM

Input: Training Data: $\mathcal{X}$;
The number of neurons of $\boldsymbol{v}$ and $\boldsymbol{h}$: $N_{\boldsymbol{v}}, N_{\boldsymbol{h}}$;
Learning rate: η^{rbm}.
Output: $\boldsymbol{w}$= $\{w_{ji}|i=1,\ldots,N_{\boldsymbol{v}}, j=1,\ldots,N_{\boldsymbol{h}}\}$;
$\boldsymbol{a}$= $\{a_i|i=1,\ldots,N_{\boldsymbol{v}}\}$;
$\boldsymbol{b}$= $\{b_j|j=1,\ldots,N_{\boldsymbol{h}}\}$.
1: **for** $i=1,\ldots,N_{\boldsymbol{v}}, j=1,\ldots,N_{\boldsymbol{h}}$ **do**
2: Initialize a_i, b_j, w_{ji} with small values randomly;
3: **end for**
4: **repeat**
5: Select training batch from $\mathcal{X}$;
6: **for** $n=1,\ldots,N_{\boldsymbol{x}}$ **do**
7: $\boldsymbol{v}_0 \leftarrow \boldsymbol{x}^{(n)}$;
8: Sample $h_{0j} \sim P(h_j|\boldsymbol{v_0})$ for $j=1,\ldots,N_{\boldsymbol{h}}$;
9: Sample $v_{1i} \sim P(v_i|\boldsymbol{h_0})$ for $i=1,\ldots,N_{\boldsymbol{v}}$;
10: Sample $h_{1j} \sim P(h_j|\boldsymbol{v_1})$ for $j=1,\ldots,N_{\boldsymbol{h}}$;
11: **for** $i=1,\ldots,N_{\boldsymbol{v}}, j=1,\ldots,N_{\boldsymbol{h}}$ **do**
12: $a_i \leftarrow a_i + \frac{\eta^{rbm}}{N_{\boldsymbol{x}}}(v_{0i}-v_{1i})$;
13: $b_j \leftarrow b_j + \frac{\eta^{rbm}}{N_{\boldsymbol{x}}}[P(h_{0j}=1|\boldsymbol{v}_0) - P(h_{1j}=1|\boldsymbol{v}_1)]$;
14: $w_{ji} \leftarrow w_{ji} + \frac{\eta^{rbm}}{N_{\boldsymbol{x}}}[P(h_{0j}=1|\boldsymbol{v}_0)v_{0i} - P(h_{1j}=1|\boldsymbol{v}_1)v_{1i}]$;
15: **end for**
16: **end for**
17: **until** Convergency;
18: **return** $\boldsymbol{w}$, $\boldsymbol{a}$, $\boldsymbol{b}$;

Algorithm 4 FineTuneDBN

Input: Training data: $(\mathcal{X}, \mathcal{Y})$;
Learning rate: η^{bp}.
Output: $\boldsymbol{w}$ = $\{w_{ji}^l|i=1,\ldots,N_{l-1}, j=1,\ldots,N_l, l=2,\ldots,L\}$;
$\boldsymbol{\gamma}$ = $\{\gamma_j^l|j=1,\ldots,N_l, l=2,\ldots,L\}$.
1: **for** $i=1,\ldots,N_{l-1}, j=1,\ldots,N_l, l=2,\ldots,L$ **do**
2: Initialize w_{ji}^l, γ_j^l with small values randomly;
3: **end for**
4: **repeat**
5: Select training batch from $(\mathcal{X}, \mathcal{Y})$;
6: **for** $n=1,\ldots,N_{\boldsymbol{x}}$ **do**
7: **for** $j=1,\ldots,N_1$ **do**
8: $\beta_j^1 \leftarrow x_j^{(n)}$;
9: **end for**
10: **for** $l=2,\ldots,L$ **do**
11: **for** $j=1,\ldots,N_l$ **do**
12: $\varphi_j^l \leftarrow \sum_{i=1}^{N_{l-1}} w_{ji}^l \beta_i^{l-1} + \gamma_j^l$;
13: **if** $l \leq L-1$ **then**
14: $\beta_j^l \leftarrow sigm\left(\varphi_j^l\right)$;
15: **else**
16: $\beta_j^l \leftarrow softmax\left(\varphi_j^l\right)$;
17: **end if**
18: **end for**
19: **end for**
20: **end for**
21: **for** $l=L,\ldots,2$ **do**
22: **for** $j=1,\ldots,N_l$ **do**
23: $\gamma_j^l \leftarrow \gamma_j^l - \frac{\eta^{bp}}{N_{\boldsymbol{x}}}\delta_j^l$;
24: **for** $i=1,\ldots,N_{l-1}$ **do**
25: $w_{ji}^l \leftarrow w_{ji}^l - \frac{\eta^{bp}}{N_{\boldsymbol{x}}}\left(\beta_i^{l-1}\delta_j^l + \lambda w_{ji}^l\right)$;
26: **end for**
27: **end for**
28: **end for**
29: **until** Convergency
30: **return** $\boldsymbol{w}$, $\boldsymbol{\gamma}$;

chaining network. After that, in lines 11-18 of **Algorithm 5**, we obtain the complete routing path Φ, and if Φ satisfies all the constraints in Eqs. (8)-(16), it is output to steer the traffic of $SFCR_f$. Otherwise, in line 15 of **Algorithm 5**, we iterate and check other routing paths to recompute Φ. We stop the running process of DL-TPA until there is no feasible routing path or the iteration times reach a stopping threshold.

F. Solution Space Optimization

In VNF selection network and VNF chaining network, output layers can be custom-made based on the information of SFCRs to optimize solution space sizes. With solution space optimization, it is helpful to enhance the predication accuracy and reduce the computation complexity.

In NFV environment, there exist many VNF instance groups that satisfy the predefined order of an SFCR. For example, in Fig. 1, there are respectively three VNF instances of VNF_a, VNF_c and VNF_d and four VNF instances of VNF_b. Therefore, there are $3 \times 3 \times 4 \times 3 = 108$ VNF instance groups that can serve the SFCR in Fig. 1. Nevertheless, only some of these VNF instance groups can serve as the optimal solutions. Moreover, during the training process, a series of discrete network conditions are sampled to train VNF selection network. Discrete training data cannot cover all the network conditions, which may lead to unreasonable solutions and the reduction of prediction accuracy. In order to solve these problems, we define a feasible VNF instance group set $\mathcal{S}_{s,k}$ for $\boldsymbol{x}_{s,k}$, and each VNF instance group in $\mathcal{S}_{s,k}$ represents the optimal solution of $\boldsymbol{x}_{s,k}$ in some network condition. If the information of $SFCR_f$ is input as $\boldsymbol{x}_{s,k}$, we can get the corresponding $\mathcal{S}_{s,k}$ by counting all the optimal VNF instance groups based on $\boldsymbol{g}_f^{ing}$ and $\boldsymbol{g}_f^{egr}$ from the training data.

Additionally, feasible path set is defined in VNF chaining network to avoid infeasible routing paths and reduce solution space size. In VNF chaining network, the pairs of starting and ending points determines the feasible paths of an SFCR. For example, if a pair of starting and ending points (A, B) is waiting to be chained, only the paths with starting point A and ending point B are feasible for this request. Then, we define the feasible path set $\mathcal{S}_c$ to include the paths with the same starting and ending points of $\boldsymbol{x}_c$, and it can be counted according to the training data as well. For the SFCR in Fig. 1, if VNF_a^1 and VNF_c^1 placed in B and D, respectively, are selected to be traversed by VNF selection network, $\mathcal{S}_c$ represents the paths with the same pair of starting and ending points (B, D).

Algorithm 5 Running Algorithm of DL-TPA

Input: VNF selection network: $\{\boldsymbol{\theta}_{s,k}^{dbn}|k=1,\ldots,K\}$;
VNF chaining network: $\boldsymbol{\theta}_{c}^{dbn}$;
Available resource ratios: $r_{uv}^{bw}, r_{u}^{mem}, r_{m}^{cpu}$;
Service function graph: $\bar{G}_f = (\bar{\mathcal{V}}_f, \bar{\mathcal{E}}_f)$;
Parameters of *SFCR*$_f$: Ψ_f^{bw*}, Ψ_f^{mem*}, Ψ_f^{cpu*}, Ψ_f^{td}.
Output: Complete routing path of *SFCR*$_f$: Φ.

```
Initialize routing path Φ ← ∅, maximum iteration times Π and current iteration times π ← 1;
k ← Get the type of SFCR_f;
x_{s,k} ←Produce x_{s,k} based on r_uv^bw, r_u^mem, r_m^cpu, Ψ_f^{bw*}, Ψ_f^{mem*}, Ψ_f^{cpu*}, Ḡ_f;
ỹ_{s,k} ← Input x_{s,k} into θ_{s,k}^{dbn};
Obtain {s_1, s_2, ....} of SFCR_f;
for p = 1, ..., |Ē_f| do
    x_c ← Produce x_c based on r_uv^bw, r_u^mem, Ψ_f^{bw*}, Ψ_f^{mem*}, s_p, s_{p+1};
    ỹ_c ← Input x_c into θ_c^{dbn};
    Φ[p] ← Obtain the optimal path between s_p and s_{p+1} from ỹ_c;
end for
repeat
    if Φ satisfies all the constraints then
        return Φ;
    else
        Φ ← Iterate and recompute Φ;
        π ← π + 1;
    end if
until π > Π;
return Rejected;
```

Let N_s and N_c represent the number of hidden neurons per hidden layer of a DBN in VNF selection network and VNF chaining network, respectively. Given the description above, assuming that *SFCR*$_f$ belongs to the k^{th} type of SFC service and there exist $M_{s,k} \gg |\mathcal{S}_{s,k}|$ and $M_c \gg |\mathcal{S}_c|$, then the computation complexity of the forward pass between the last hidden layer and the output layer can be reduced from $O(N_s M_{s,k} + N_c|\bar{\mathcal{E}}_f|M_c)$ to $O(N_s|\mathcal{S}_{s,k}| + N_c|\bar{\mathcal{E}}_f||\mathcal{S}_c|)$. Therefore, given the definition of feasible VNF instance group set and feasible path set, we can optimize the solution space size of VNF selection network and VNF chaining network based on the information of SFCRs, and make the training and running processes more effectively.

G. Complexity Analysis

The training process of DL-TPA runs off-line, and its computation complexity is proportional to the size of training data and training period. After off-line training, we can achieve intelligently routing path computation of SFCRs with trained models instead of running rule-based algorithms to solve the BIP model with massive iterations and computations. And we can make DL-TPA adapt to the changes of network topology (*e.g.,* add a node or remove a link) by generating new training data to update the VNF selection network and VNF chaining network.

In the feedforward of the running process of DL-TPA, the computation complexity is mainly related to the size of DBN models. We assume that the structure of the input and hidden layers are the same in all the DBNs of VNF selection network, and the DBN of VNF chaining network is also the same. Here, we use L_s and L_c to represent the numbers of hidden layers of the DBN in VNF selection network and VNF chaining network, respectively. In VNF selection network, computing the optimal VNF instance group of an SFCR runs in $O(N_s(|\mathcal{V}| + |\mathcal{E}| + |\Omega(k)| + 2q + L_s N_s + |\mathcal{S}_{s,k}|))$. In VNF chaining network, the computation complexity of chaining these selected VNF instances is $O(N_c|\bar{\mathcal{E}}_f|(|\mathcal{V}| + |\mathcal{E}| + 2q + L_c N_c + |\mathcal{S}_c|))$. Since checking constraints runs in $O(1)$ and q is a small value that equals $\lceil log_2|\mathcal{V}|\rceil$, in the running process of DL-TPA, the total computation complexity of routing path computation for an SFCR is $O(N_s(|\mathcal{V}|+|\mathcal{E}|+|\Omega(k)|+L_s N_s + |\mathcal{S}_{s,k}|) + N_c|\bar{\mathcal{E}}_f|(|\mathcal{V}| + |\mathcal{E}| + L_c N_c + |\mathcal{S}_c|))$.

V. Performance Evaluation

This section depicts the performance evaluation of DL-TPA in SDN/NFV-enabled networks. Both the VNF selection network and VNF chaining network are trained with Tensorflow-gpu 1.4 version [22]. We evaluated our approach on a computer with an Intel(R) Core(TM) i7-4790 CPU 3.60 @ GHz and a Nvidia GeForce GTX 1080Ti GPU.

A. Simulation Settings

In the simulation, the network topology that we use is a US carrier network [34], which consists of 60 nodes and 77 links. In the network, 10 nodes are selected as function nodes to place VNF instances and the other nodes are served as access nodes. There are 5 types of VNFs, and each function node is placed with 3-5 types of VNF instances. There are 5 types of SFC services, and SFCRs are randomly produced in access nodes. The bandwidth of each link is 3.5 Gbps. The memory capacities and CPU capacities of each node and VNF instance are 5 GB and 2000 MIPS, respectively [5]. For each SFCR, the traffic needs to traverse 3 types of VNFs before reaching the egress node. The resource consumptions of each SFCR in links, nodes and VNF instances are set as numbers randomly between (0, 10]. The maximum tolerable delay is set randomly from 50-100 *ms* [27]. The delays in links, nodes and VNF instances (denoted as d_{uv}, d_u and d_m) are computed according to Eqs. (28)-(30) [21], respectively.

$$d_{uv} = d_{uv}^{prop} + d_{uv}^{tx} + \frac{1 - r_{uv}^{bw}}{r_{uv}^{bw}} d_{uv}^{tx}, \quad \forall uv \in \mathcal{E}, \tag{28}$$

$$d_u = \frac{1 - r_u^{mem}}{r_u^{mem}} t_u^{proc}, \quad \forall u \in \mathcal{V}, \tag{29}$$

$$d_m = \frac{1 - r_m^{cpu}}{r_m^{cpu}} t_m^{proc}, \quad \forall m \in \mathcal{M}. \tag{30}$$

In Eq. (28), the first part d_{uv}^{prop} indicates the propagation delay of link $uv \in \mathcal{E}$ which is computed by the ratio of the length of link uv to the propagation speed of signals in that medium. The second and third parts represent the transmission delay and queuing delay. The transmission delay d_{uv}^{tx} is computed by dividing the bandwidth capacity of link

uv with the packet size. The queening delay is related to the load conditions and transmission delay. Eqs. (29)-(30) mean the processing delay in node $u \in \mathcal{V}$ and VNF instance $m \in \mathcal{M}$. The parameters t_u^{proc} and t_m^{proc} are set as 10 μs and 1000 μs [35], which denote the per-packet processing time in node $u \in \mathcal{V}$ and VNF instance $m \in \mathcal{M}$, respectively.

Note that, in the ideal case, the deep models should be trained based on published training data collected in a practical SDN/NFV-enabled environment. However, since SFC is a new service paradigm in the network, there are no such published training data currently available for us. Thus, we generate the training data referring to existing literatures [4], [27]. In the simulation, we construct a network simulator to generate training data and a SDN/NFV-enabled environment to evaluate the performance of DL-TPA. They are constructed independently, but their parameter settings are the same according to this subsection. As for the training data, first, we generate a set of SFCRs with the parameter settings above and get the optimal solution of each SFCR one by one by running an optimal algorithm named as graph layering algorithm until the available resources of network simulator is exhausted. Then, we remove all the SFCRs of previous set in the network simulator and repeat the previous steps until we collect enough training data. In the network simulator, a training data set consisting of about 2×10^7 items is generated to train each DBN in VNF selection network and VNF chaining network, respectively. And we also generate a testing data set including about 10^5 items to evaluate the performance of DL-TPA in the SDN/NFV-enabled environment. Moreover, for each DBN of VNF selection network, there are three hidden layers and each hidden layer has forty hidden neurons. For the DBN of VNF chaining network, it includes two hidden layers and each hidden layer has eighty hidden neurons.

Since the training data are generated with an optimal algorithm, the more likely the prediction results of DL-TPA and the optimal solution are, the shorter the end-to-end delays of SFCRs in Eq. (17) are obtained. Therefore, the accuracy is used as a performance indicator of DL-TPA. In VNF selection network and VNF chaining network, the accuracy of a DBN model is defined as follows:

$$ACC = \frac{1}{N_{\boldsymbol{x}}} \sum_{\boldsymbol{x}} \mathbb{I}(\boldsymbol{y} = \widetilde{\boldsymbol{y}}), \tag{31}$$

where $\widetilde{\boldsymbol{y}}$ denotes the prediction result based on input $\boldsymbol{x}$, and $\boldsymbol{y}$ indicates the labeled output. $N_{\boldsymbol{x}}$ stands for the total number of items of training data. $\mathbb{I}(\cdot)$ is an indicator function. If the condition of $\mathbb{I}(\cdot)$ is satisfied, it equals 1, and 0 otherwise. In addition, the epoch for the training of DBN is set as 2000, and each simulation is repeated for 10 times. λ in Eq. (23) is set as 10^{-4}. All other parameters in the training process that we set are according to [36].

B. Introduction of Compared Algorithms

In the simulation, we compare DL-TPA with graph layering algorithm [20], [21] and Eigendecomposition [13]. Graph layering algorithm is used as the optimal algorithm to generate training data. Eigendecomposition is a typical heuristic algorithm. And we aim to prove that DL-TPA can get near-optimal performance and perform more efficiently than typical heuristic algorithm. Before the introduction of the simulation, we give a brief description to these compared algorithms:

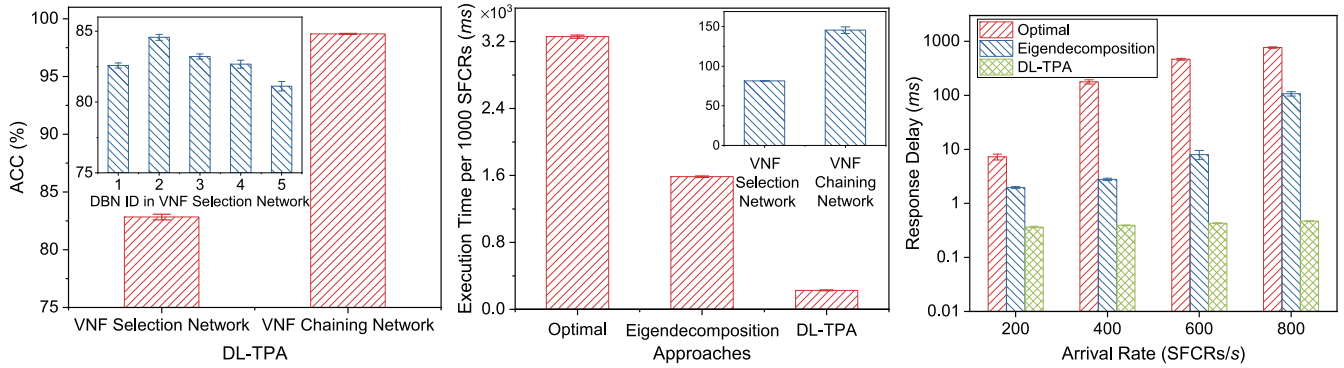
- Optimal: Delay is considered as the weight of each link. First, the optimal algorithm generates a layered graph that consists of serval copies of original network topologies, and the adjacent copies of layered graph are connected to satisfy the predefined order of an SFCR. Then the Dijkstra algorithm is executed in layered graph to compute the path with the minimum end-to-end delay for each SFCR.
- Eigendecomposition: The Umeyama's eigendecomposition approach is adopted and extended to achieve the optimal matching of an SFCR in network topology. First, Eigendecomposition generates an adjacent matrix for network topology using the widest-shortest path to calculate the weight of each element. An adjacent matrix is also generated for each SFCR according to its demand of resource consumption. Next, Eigendecomposition extends the adjacent matrix of SFCR to be with the same size of the network's. Then, the eigenvector matrixes of the two adjacent matrixes are computed. Afterwards, Eigendecomposition computes the conjugate matrixes for both eigenvector matrixes, respectively, and multiplies them together. Finally, Eigendecomposition constructs the routing path for an SFCR by choosing the locations with the maximum value in each row of the product.

C. Simulation Results

*1) **Prediction Accuracy**:* Fig. 7(a) shows the prediction accuracy of VNF selection network and VNF chaining network. In the figure, the average prediction accuracy of VNF selection network is about 82.8%, and is about 98.7% for VNF chaining network. As there are five different types of SFC services, the VNF selection network consists of five DBNs. The prediction accuracy of DBNs in the VNF selection network are 82.5%, 84.6%, 83.2%, 82.7%, and 81.1%, respectively. The reason why the prediction accuracy of VNF selection network is lower than that of the VNF chaining network is because the resource consumption in the VNF instances leads to more increment of delay [35]. Thus, VNF selection network needs to do more complex decisions to make the optimal selection of VNF instances in multi-instance environment.

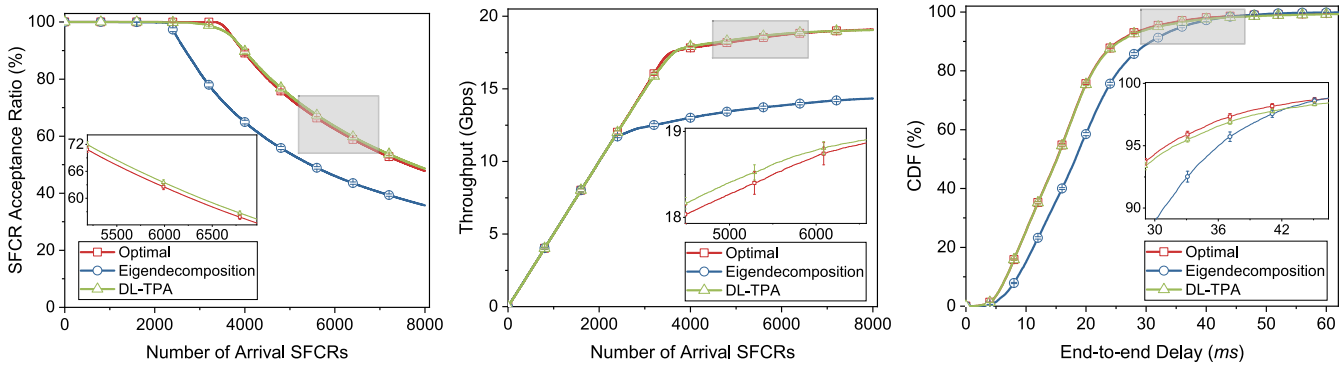
*2) **Comparison of Time Efficiency**:* Fig. 7(b) presents the execution time of three algorithms with 1000 SFCRs. According to the shown results, the optimal algorithm leads to the longest time consumption, which is about 3200 *ms*. Eigendecomposition costs about 1600 *ms*. DL-TPA performs the best, which leads to about 8× speed-up than Eigendecomposition and about 16× speed-up than the optimal algorithm (VNF selection network spends about 80 *ms*, and VNF chaining network accounts for about 140 *ms*).

Fig. 7(c) illustrates the response time under different SFCR arrival rates, and we count the response delay of SFCRs within 1000 *ms* period. The response delay indicates the time between the arrival of an SFCR and its routing path being obtained. So, the lower the response delay is, the higher time



(a) Prediction accuracy of DL-TPA (b) Execution time (c) Response delay

Fig. 7. Prediction accuracy of DL-TPA and the comparison of time efficiency (with 95% confidence intervals).



(a) SFCR acceptance ratio (b) Throughput (c) End-to-end delay

Fig. 8. Comparison of average SFCR acceptance ratio, throughput and CDF of end-to-end delay (with 95% confidence intervals).

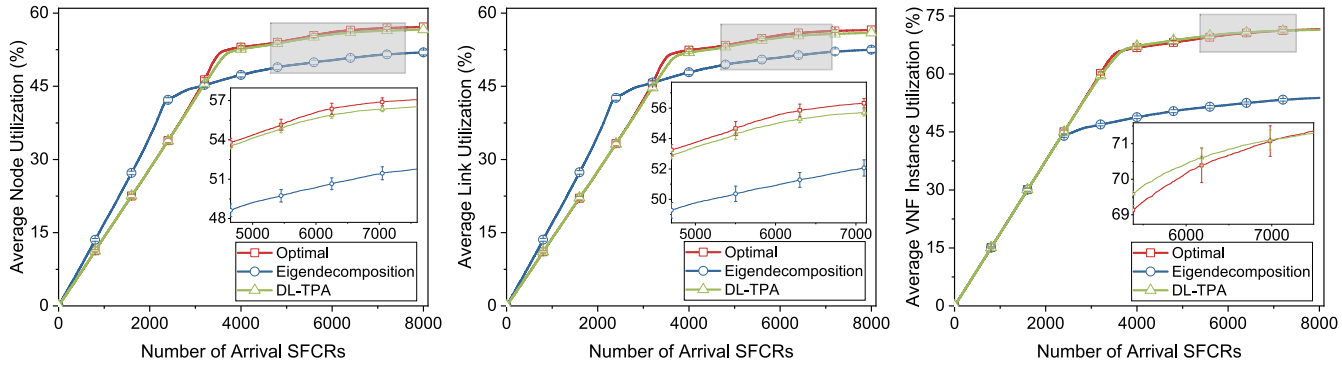
efficiency the routing algorithm is. In the figure, with the increment of SFCR arrival rates, the response delay of the optimal algorithm grows much faster than that of Eigendecomposition and DL-TPA. The response delay of Eigendecomposition grows sharply, when the arrival rate is bigger than 600 SFCRs/*s*. Compared with the optimal and Eigendecomposition algorithms, the response delay of DL-TPA is very low, which is only about 0.47 *ms*. Also, the response delay of DL-TPA changes slow with the increment of arrival rate from 200 SFCRs/*s* to 800 SFCRs/*s*. When the arrival rate is 800 SFCRs/*s*, the response delay of DL-TPA is about 1600× and 200× lower than that of the optimal algorithm and Eigendecomposition, respectively.

*3) **Comparison of Average SFCR Acceptance Ratio, Throughput and CDF of End-to-End Delay**:* Fig. 8(a) illustrates the average SFCR acceptance ratios of three algorithms. Average SFCR acceptance ratio reflects the SFCRs successfully served in the network accounting for the total ones. Due to effective feature extraction and learning, DL-TPA yields near optimal performance. When about 3200 SFCRs are served in the network, the average SFCR acceptance ratios of the optimal algorithm and DL-TPA start to decline, while the performance of Eigendecomposition declines after serving about 2300 SFCRs. The average SFCR acceptance ratio of Eigendecomposition is about 20% lower than that of the optimal algorithm and DL-TPA. This is because Eigendecomposition could not ensure to get the optimal VNF instances and routing paths for SFCRs. Moreover, the widest-shortest routing algorithm is used in Eigendecomposition to avoid bottleneck, which leads to longer routing paths and more resource consumption during the routing path computation.

Fig. 8(b) shows the average throughput of these three algorithms. Throughput indicates the total bandwidth consumption of SFCRs successfully served in the network. In this simulation, the optimal algorithm and DL-TPA get the highest performance, where the throughput of Eigendecomposition is about 5 Gbps lower than that of the other two algorithms.

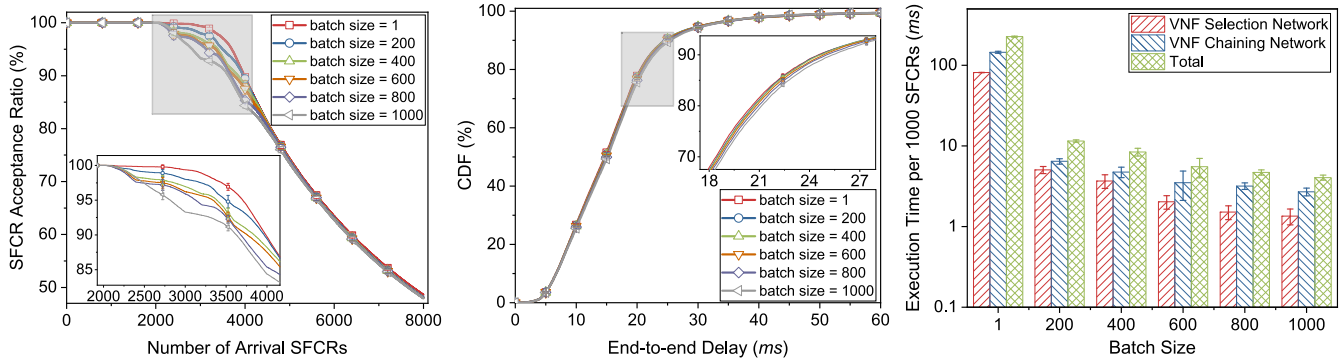
The CDF of end-to-end delay is presented in Fig. 8(c). The end-to-end delay reflects the total delay of the routing paths of SFCRs. In this simulation, the optimal algorithm gets the highest performance. Due to efficient VNF selection and chaining, DL-TPA can provide near optimal result. The delay distributions between 0-20 *ms* are about 75% for the optimal algorithm and DL-TPA, which is only about 60% for Eigendecomposition. As stated before, Eigendecomposition cannot ensure to obtain the optimal solutions and the widest-shortest routing algorithm is used to avoid network bottleneck. Thus, longer routing paths of SFCRs will be produced, which incurs longer end-to-end delay. Furthermore, given simulated results, we can get that DL-TPA can serve SFCRs with low-delay demands.

*4) **Comparison of Average Utilizations in Nodes, Links and VNF Instances**:* We show average resource utilizations in Fig. 9. The optimal algorithm obtains the best performance.



(a) Average node utilization (b) Average link utilization (c) Average VNF instance utilization

Fig. 9. Comparison of average utilizations in nodes, links and VNF instances (with 95% confidence intervals).



(a) SFCR acceptance ratio vs batch size (b) CDF of end-to-end delay vs batch size (c) Execution time vs batch size

Fig. 10. Comparison of average SFCR acceptance ratio, CDF of end-to-end delay and execution time vs batch size of SFCR (with 95% confidence intervals).

DL-TPA gets approximate performance of the optimal results, and the performance of Eigendecomposition is the worst. In Fig. 9(a)-9(b), due to longer routing paths and more resource consumption, Eigendecomposition needs to consume more resources than the optimal and DL-TPA, when receiving an equal number of SFCRs. So, with the number of arrival SFCRs between 0 and 3000, the resource utilizations of Eigendecompositionin in Fig. 9(a)-9(b) are higher than other two algorithms'. In Fig. 9(a)-9(c), compared with Eigendecomposition, the optimal and DL-TPA get about 5% higher performance in average resource utilizations of links and nodes, however the performance enhancement of them is about 18% in VNF utilization ratio, when the number of arrival SFCRs is more than 4000. Given this phenomenon, we can get that it is because the resource exhaustion in nodes and links that leads to the performance decline of Eigendecomposition.

*5) **Comparison of Average SFCR Acceptance Ratio, CDF of End-to-End Delay and Execution Time Under Different SFCR Batch Sizes**:* In the simulations above, the network conditions are collected in real time to make DL-TPA output the optimal strategies. However, collecting the network conditions in real time leads to frequent signaling interactions between control plane and data plane, which burdens the SDN controller with more load. Moreover, in deep learning, it is available to process samples in batches due to software performance optimization and GPU acceleration. So, routing SFCR one at each time reduces the execution efficiency of deep learning network, and it is not effective to deal with tremendous growth of network traffic. Then, in Fig. 10, we collect the network conditions periodically, and evaluate the performance of DL-TPA under different SFCR batch sizes in the simulations below.

Fig. 10(a) shows the average SFCR acceptance ratio of DL-TPA under different SFCR batch sizes. With the increment of SFCR batch size, the average acceptance ratio with 2000-4500 arrival SFCRs declines fast. This is because increasing the SFCR batch size leads to bigger changes in network conditions. Therefore, the network congestion is aggravated, because of imprecise evaluation of network conditions.

Fig. 10(b) describes the CDF of end-to-end delay under different SFCR batches. In the figure, the larger the SFCR batch size is, the longer end-to-end delay the paths of SFCRs have. For example, when the SFCR batch size is 200, the proportion of SFCRs with end-to-end delay being shorter than 20 *ms* is about 77%, and this proportion declines to 75%, when the SFCR batch size is 1000. As stated in Fig. 10(a), large SFCR batch size results in imprecise evaluation of network conditions. The imprecise evaluation of network conditions aggravates network congestion which leads to lengthening the end-to-end delay of SFCRs.

The average execution time is evaluated with 1000 SFCRs under different SFCR batch sizes. In this simulation, when dealing with SFCR one at each time (batch size equals 1),

the execution time is about 220 *ms* (VNF selection network accounts for about 80 *ms* and VNF chaining network spends about 140 *ms*). When the SFCR batch size is 200, the execution time of DL-TPA is about 12 *ms* (VNF selection network accounts for about 5 *ms* and VNF chaining network accounts for about 7 *ms*). When the SFCR batch size becomes 1000, the execution time with 1000 SFCRs declines to about 4 *ms* (VNF selection network accounts for about 1 *ms* and VNF chaining network spends about 3 *ms*). Combined with Fig. 10(a) and Fig. 10(b), we can get a good tradeoff between the network performance (*e.g.,* average SFCR acceptance ratio and end-to-end delay) and the execution time of DL-TPA with SFCR batch size of no larger than 200.

VI. Conclusion

In this paper, to achieve efficient routing path computation of SFCRs, we studied VNF-SCP in SDN/NFV-enabled networks. This problem was formulated as a BIP model aiming to minimize the end-to-end delay for each SFCR. Then, a novel two-phase VNF selection and chaining algorithm based on deep learning technology, DL-TPA, was proposed to solve VNF-SCP. In DL-TPA, VNF selection network and VNF chaining network were designed to achieve the optimal selection and chaining of VNF instances. In addition, feasible VNF instance group set and path set were defined to reduce solution space sizes of VNF selection network and VNF chaining network, respectively. Performance evaluation showed that DL-TPA can get high prediction accuracy of optimal paths for SFCRs and high network performance in terms of SFCR acceptance ratio, throughput, end-to-end delay and resource utilization. Also, DL-TPA can significantly enhance the time efficiency of routing path computation for SFCR compared with existing rule-based algorithms.

In the future work, we intend to extend our work in a number of ways. We plan to implement the DL-TPA in a real SDN/NFV-enabled network and conduct more comprehensive performance evaluation. We plan to further improve the scalability and feasibility of DL-TPA to make it adaptable to different types of network topologies simultaneously. We also plan to study the resource allocation in SDN/NFV-enabled networks with deep learning technology in the future.

References

[1] J. Sherry, S. Hasan, C. Scott, A. Krishnamurthy, S. Ratnasamy, and V. Sekar, "Making middleboxes someone else's problem: Network processing as a cloud service," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 42, no. 4, pp. 13–24, Sep. 2012.

[2] R. Mijumbi, J. Serrat, J.-L. Gorricho, N. Bouten, F. De Turck, and R. Boutaba, "Network function virtualization: State-of-the-art and research challenges," *IEEE Commun. Surveys Tuts.*, vol. 18, no. 1, pp. 236–262, 1st Quart., 2016.

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

[4] F. Bari, S. R. Chowdhury, R. Ahmed, R. Boutaba, and O. C. M. B. Duarte, "Orchestrating virtualized network functions," *IEEE Trans. Netw. Service Manage.*, vol. 13, no. 4, pp. 725–739, Dec. 2016.

[5] J. Pei, P. Hong, K. Xue, and D. Li, "Efficiently embedding service function chains with dynamic virtual network function placement in geo-distributed cloud system," *IEEE Trans. Parallel Distrib. Syst.*, vol. 30, no. 10, pp. 2179–2192, Oct. 2019.

[6] M. Karakus and A. Durresi, "Quality of Service (QoS) in software defined networking (SDN): A survey," *J. Netw. Comput. Appl.*, vol. 80, pp. 200–218, Feb. 2017.

[7] I. F. Akyildiz, A. Lee, P. Wang, M. Luo, and W. Chou, "Research challenges for traffic engineering in software defined networks," *IEEE Netw.*, vol. 30, no. 3, pp. 52–58, May 2016.

[8] A. Laghrissi and T. Taleb, "A survey on the placement of virtual resources and virtual network functions," *IEEE Commun. Surveys Tuts.*, vol. 21, no. 2, pp. 1409–1434, 2nd Quart., 2019.

[9] D. Li, P. Hong, K. Xue, and J. Pei, "Virtual network function placement considering resource optimization and SFC requests in cloud datacenter," *IEEE Trans. Parallel Distrib. Syst.*, vol. 29, no. 7, pp. 1664–1677, Jul. 2018.

[10] H. Hantouti, N. Benamar, T. Taleb, and A. Laghrissi, "Traffic steering for service function chaining," *IEEE Commun. Surveys Tuts.*, vol. 21, no. 1, pp. 487–507, 1st Quart., 2019.

[11] D. Li, P. Hong, K. Xue, and J. Pei, "Virtual network function placement and resource optimization in NFV and edge computing enabled networks," *Comput. Netw.*, vol. 152, pp. 12–24, Apr. 2019.

[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] M. Mechtri, C. Ghribi, and D. Zeghlache, "A scalable algorithm for the placement of service function chains," *IEEE Trans. Netw. Service Manage.*, vol. 13, no. 3, pp. 533–546, Sep. 2016.

[14] D. Silver *et al.*, "Mastering the game of go with deep neural networks and tree search," *Nature*, vol. 529, no. 7587, pp. 484–489, Jan. 2016.

[15] G. Litjens *et al.*, "A survey on deep learning in medical image analysis," *Med. Image Anal.*, vol. 42, pp. 60–88, Dec. 2017.

[16] B. Mao *et al.*, "Routing or computing? The paradigm shift towards intelligent computer network packet transmission based on deep learning," *IEEE Trans. Comput.*, vol. 66, no. 11, pp. 1946–1960, Nov. 2017.

[17] J. Pei, P. Hong, M. Pan, J. Liu, and J. Zhou, "Optimal VNF placement via deep reinforcement learning in SDN/NFV-enabled networks," *IEEE J. Sel. Areas Commun.*, vol. 38, no. 2, pp. 263–278, Feb. 2020, doi: 10.1109/JSAC.2019.2959181.

[18] Z. Fadlullah *et al.*, "State-of-the-art deep learning: Evolving machine intelligence toward tomorrow's intelligent network traffic control systems," *IEEE Commun. Surveys Tuts.*, vol. 19, no. 4, pp. 2432–2455, 4th Quart., 2017.

[19] N. Kato *et al.*, "The deep learning vision for heterogeneous network traffic control: Proposal, challenges, and future perspective," *IEEE Wireless Commun.*, vol. 24, no. 3, pp. 146–153, Jun. 2017.

[20] R. Yu, G. Xue, and X. Zhang, "QoS-aware and reliable traffic steering for service function chaining in mobile networks," *IEEE J. Sel. Areas Commun.*, vol. 35, no. 11, pp. 2522–2531, Nov. 2017.

[21] A. Dwaraki and T. Wolf, "Adaptive service-chain routing for virtual network functions in software-defined networks," in *Proc. Workshop Hot Topics Middleboxes Netw. Function Virtualization (HotMIddlebox)*, 2016, pp. 32–37.

[22] *Tensorflow*. Accessed: Nov. 25, 2019. [Online]. Available: https://www.tensorflow.org/

[23] S. Jiao, X. Zhang, S. Yu, X. Song, and Z. Xu, "Joint virtual network function selection and traffic steering in telecom networks," in *Proc. IEEE Global Commun. Conf. (GLOBECOM)*, Dec. 2017, pp. 1–7.

[24] H. Huang, S. Guo, J. Wu, and J. Li, "Service chaining for hybrid network function," *IEEE Trans. Cloud Comput.*, vol. 7, no. 4, pp. 1082–1094, Oct. 2019.

[25] H. Huang, P. Li, S. Guo, W. Liang, and K. Wang, "Near-optimal deployment of service chains by exploiting correlations between network functions," *IEEE Trans. Cloud Comput.*, early access, Dec. 6, 62017, doi: 10.1109/TCC.2017.2780165.

[26] G. Cheng, H. Chen, H. Hu, Z. Wang, and J. Lan, "Enabling network function combination via service chain instantiation," *Comput. Netw.*, vol. 92, pp. 396–407, Dec. 2015.

[27] J. Pei, P. Hong, K. Xue, and D. Li, "Resource aware routing for service function chains in SDN and NFV-enabled network," *IEEE Trans. Services Comput.*, early access, Jun. 22, 2019, doi: 10.1109/TSC.2018.2849712.

[28] *Opendaylight*. Accessed: Nov. 25, 2019. [Online]. Available: https://media.readthedocs.org/pdf/opendaylight/latest/opendaylight.pdf

[29] B. Mao *et al.*, "A tensor based deep learning technique for intelligent packet routing," in *Proc. IEEE Global Commun. Conf. (GLOBECOM)*, Dec. 2017, pp. 1–6.

[30] J. Pei, P. Hong, and D. Li, "Virtual network function selection and chaining based on deep learning in SDN and NFV-enabled networks," in *Proc. IEEE Int. Conf. Commun. Workshops (ICC Workshops)*, May 2018, pp. 1–6.
[31] Y. Bengio, "Learning deep architectures for AI," *Found. Trends Mach. Learn.*, vol. 2, no. 1, pp. 1–127, 2009.
[32] G. E. Hinton, S. Osindero, and Y.-W. Teh, "A fast learning algorithm for deep belief nets," *Neural Comput.*, vol. 18, no. 7, pp. 1527–1554, Jul. 2006.
[33] G. E. Hinton, "Training products of experts by minimizing contrastive divergence," *Neural Comput.*, vol. 14, no. 8, pp. 1771–1800, Aug. 2002.
[34] *Optical Network Design and Planning*. Accessed: Nov. 25, 2019. [Online]. Available: http://www.monarchna.com/topology.html
[35] R. Ramaswamy, N. Weng, and T. Wolf, "Characterizing network processing delay," in *Proc. IEEE Global Telecommun. Conf. (GLOBECOM)*, Nov./Dec. 2004, pp. 1629–1634.
[36] G. E. Hinton, "A practical guide to training restricted Boltzmann machines," in *Neural Networks: Tricks of the Trade*, G. Montavon, G. Orr, and K.-R. Müller, Eds. Berlin, Germany: Springer, pp. 599–619.

Jianing Pei received the B.S. degree from the Department of Information and Electrical Engineering (IEE), China University of Mining and Technology (CUMT), in 2015. He is pursuing the Ph.D. degree with the University of Science and Technology of China (USTC), with his advisor Peilin Hong. His research interests include software-defined networks, network function virtualization, network resource orchestration and management, and machine learning algorithms.

Peilin Hong received the B.S. and M.S. degrees from the Department of Electronic Engineering and Information Science (EEIS), University of Science and Technology of China (USTC), in 1983 and 1986, respectively. She is currently a Professor and an Advisor of the Ph.D. candidates with the Department of EEIS, USTC. She has published two books and over 100 academic articles in several journals and conference proceedings. Her research interests include next-generation Internet, policy control, IP QoS, and information security.

Kaiping Xue (Senior Member, IEEE) received the bachelor's degree from the Department of Information Security, University of Science and Technology of China (USTC), in 2003, and the Ph.D. degree from the Department of Electronic Engineering and Information Science (EEIS), USTC, in 2007. From May 2012 to May 2013, he was a Post-Doctoral Researcher with the Department of Electrical and Computer Engineering, University of Florida. He is currently an Associate Professor with the Department of EEIS and the School of Cyber Security, USTC. He has authored and coauthored more than 80 technical articles in the areas of communication networks and network security. His research interests include next-generation Internet, distributed networks, and network security. He is an IET Fellow. His work won best paper awards in the IEEE MSN 2017, the IEEE HotICN 2019, and Best Paper Runner-Up Award in the IEEE MASS 2018. He serves on the Editorial Board of several journals, including the IEEE TRANSACTIONS ON WIRELESS COMMUNICATIONS (TWC), the IEEE TRANSACTIONS ON NETWORK AND SERVICE MANAGEMENT (TNSM), *Ad Hoc Networks*, IEEE ACCESS and *China Communications*. He has also served as a Guest Editor of the IEEE JOURNAL ON SELECTED AREAS IN COMMUNICATIONS (JSAC) and a Lead Guest Editor of the *IEEE Communications Magazine*. He is serving as the Program Co-Chair for the IEEE IWCMC 2020 and SIGSAC@TURC 2020.

Defang Li was born in 1991. He received the B.S. and Ph.D. degrees from the Department of Electronic Engineering and Information Science (EEIS), University of Science and Technology of China (USTC), in 2014 and 2019, respectively. His research interests include SDN, NFV, network resource orchestration and management, and cloud computing.

David S. L. Wei (Senior Member, IEEE) received the Ph.D. degree in computer and information science from the University of Pennsylvania in 1991. From May 1993 to August 1997, he was on the Faculty of Computer Science and Engineering, University of Aizu, Japan, as an Associate Professor and then a Full Professor. He is currently a Full Professor with the Computer and Information Science Department, Fordham University. He has authored and coauthored more than 120 technical articles in the areas of distributed and parallel processing, wireless networks and mobile computing, optical networks, peer-to-peer communications, cognitive radio networks, big data, cloud computing, and the IoT in various archival journals and conference proceedings. He served on the program committee. He was a session chair for several reputed international conferences. He was a Lead Guest Editor of the IEEE JOURNAL ON SELECTED AREAS IN COMMUNICATIONS for the special issue on Mobile Computing and Networking and the IEEE JOURNAL ON SELECTED AREAS IN COMMUNICATIONS for the special issue on Networking Challenges in Cloud Computing Systems and Applications, and a Guest Editor of the IEEE JOURNAL ON SELECTED AREAS IN COMMUNICATIONS for the special issue on Peer-to-Peer Communications and Applications, a Lead Guest Editor of IEEE Transactions on Cloud Computing for the special issue on Cloud Security, the IEEE TRANSACTIONS ON BIG DATA for the special issue on Trustworthiness in Big Data and Cloud Computing Systems, and the IEEE TRANSACTIONS ON BIG DATA for the special issue on Edge Analytics in the Internet of Things. He also served as an Associate Editor of the IEEE TRANSACTIONS ON CLOUD COMPUTING from 2014 to 2018 and JOURNAL OF CIRCUITS, SYSTEMS AND COMPUTERS from 2013 to 2018. He is presently an Editor of IEEE JOURNAL ON SELECTED AREAS in Communications for the Series on Network Softwarization & Enablers and a Lead Guest Editor of the IEEE JOURNAL ON SELECTED AREAS IN COMMUNICATIONS for the special issue on Leveraging Machine Learning in SDN/NFV-based Networks. His current research interests include cloud and edge computing, the IoT, big data, machine learning, and cognitive radio networks.

Feng Wu (Fellow, IEEE) received the B.S. degree in electrical engineering from Xidian University in 1992, and the M.S. and Ph.D. degrees in computer science from the Harbin Institute of Technology, in 1996 and 1999, respectively. He is currently a Professor with the University of Science and Technology of China (USTC). Before that, he was a Principle Researcher and Research Manager with Microsoft Research Asia. His research interests include computational photography, image and video compression, media communication, and media analysis and synthesis. He has authored or coauthored over 200 high quality articles. As a coauthor, he received the Best Paper Award from the IEEE TRANSACTIONS ON CIRCUITS AND SYSTEMS FOR VIDEO TECHNOLOGY 2009, PCM 2008, and SPIE VCIP 2007. His research interests include multimedia communications, image and video processing, and artificial intelligence. He serves as an Associate Editor for the IEEE TRANSACTIONS ON CIRCUITS AND SYSTEM FOR VIDEO TECHNOLOGY, the IEEE TRANSACTIONS ON MULTIMEDIA, and several other International journals. He received the IEEE Circuits and Systems Society 2012 Best Associate Editor Award. He also served as the TPC Chair for MMSP 2011, VCIP 2010 and PCM 2009, and the Special Sessions Chair for ICME 2010 and ISCAS 2013