Service Function Chain Reconfiguration in 5G Core Networks Using Deep Learning

Mehdi Setayesh and Vincent W.S. Wong
Department of Electrical and Computer Engineering, The University of British Columbia, Vancouver, Canada
email: {setayeshm, vincentw}@ece.ubc.ca

***Abstract*—Software-defined networking (SDN) and network functions virtualization (NFV) enable service providers to accommodate diversified service requests in the fifth generation (5G) core networks. Given the time-varying traffic demand of the service requests, it is crucial for service providers to embed the service function chains (SFCs) of the service requests in the network to support load balancing, and to minimize the reconfiguration overhead due to virtual network functions (VNFs) migration while satisfying their quality of service (QoS) requirements. In this paper, we study a delay-aware VNF migration problem for embedding SFCs in a network with limited processing resource capacity for NFV-enabled nodes. We formulate it as a mixed-integer nonlinear optimization problem. We decompose this problem into two subproblems for stateful VNF mapping and allocation of processing resources, where the second subproblem is a convex optimization problem. To solve the first subproblem, we propose an algorithm based on deep neural network (DNN) with attention mechanism for learning the stochastic policy of a near-optimal VNF mapping. Simulation results show that our proposed algorithm provides a solution which is very close to the optimal solution obtained by solving a mixed-integer quadratically constrained programming problem.**

I. Introduction

The fifth generation (5G) communication networks are envisioned to support multiple network services with diverse quality of service (QoS) requirements. Software-defined networking (SDN) and network functions virtualization (NFV) are two enablers to provide a flexible and programmable architecture for 5G networks [1]. SDN provides flexible network management using logically centralized control module. NFV facilitates the deployment of the virtualized network functions (VNFs) over NFV-enabled nodes (e.g., commodity servers) [2]. Leveraging SDN/NFV technologies, different network functions (e.g., firewall, deep packet inspection (DPI), intrusion detection system (IDS)) can be deployed as modular software components on NFV-enabled nodes and can be chained together to provide a network service in the form of a service function chain (SFC) [3].

A service request, in the form of SFC, has its own QoS requirement in terms of end-to-end delay and average data rate. VNFs running in virtual machines (VMs) are referred to as VNF instances (VNFIs). The physical links and switches, which are used to connect two VNFs based on the SFC sequences, are referred to as tunnels. In the planning phase of a virtual network, the placement of VNFIs on the NFV-enabled nodes, the processing resource capacity of VNFIs, the tunnels connecting VNFs, and the bandwidth of the links along the tunnels are determined by the service provider in order to embed SFCs in the virtual network and to meet the QoS requirements of the service requests.

Given the predetermined VNFIs and tunnels, there are major challenges for embedding SFCs in the operation phase of the virtual network. First, the service requests can arrive at different points of time. Hence, the service provider should dynamically embed the SFCs. Second, the virtual network resources can be shared among multiple SFCs. Moreover, the traffic arrival rate of service requests fluctuates over time, which may lead to the QoS degradation for service requests due to insufficient processing resource for VNFs, as well as bandwidth for the tunnels [4], [5].

To tackle the aforementioned challenges, SFC reconfiguration has received considerable attention recently. Two types of SFC reconfiguration have been proposed in the literature: VNF mapping and VNF migration [6]. In VNF mapping, a new mapping of VNFs on VNFIs with their required processing resources is considered to satisfy the QoS requirements of the service requests. In VNF migration, a stateful VNF mapping on VNFIs is considered. Zhang *et al.* [7] modeled the VNF mapping problem as a mixed binary linear program subject to the constraints of the SFCs data rate requirement and the link capacity. Tajiki *et al.* [8] proposed heuristic algorithms to jointly manage the VNF placement and routing to enable energy-aware SFC embedding.

To preserve the continuity of the existing flows for the previously embedded service requests and prevent any service interruption, VNF migration approach is standardized in [9]. Li *et al.* in [6] proposed two heuristic algorithms to solve a SFC reconfiguration problem with the objective of minimizing the end-to-end latency for all the affected services after the migration process. Qu *et al.* in [10] formulated a delay-aware flow migration problem by considering the trade-off between load balancing among the VNFIs and the required reconfiguration overhead. However, in [6] and [10], the effect of arrival/departure of service requests on VNF migration is not considered. Specifically, a VNF migration problem is required to be solved when a new service request arrives. To mitigate service quality degradation due to the service interruption, it is important to solve the VNF migration problem by a low time complexity algorithm for obtaining a near-optimal solution.

To tackle these challenges, in this paper, we study the VNF migration problem for embedding SFCs in a virtual network. We propose an algorithm using deep neural network (DNN) with attention mechanism [11] to minimize the number of

migrations and achieve load balancing by jointly optimizing the VNF mapping and processing resource allocation over a virtual network of VNFIs. Our contributions are as follows:

- Considering the arrival of new service requests and time-varying nature of service requests traffic load, we model the joint stateful VNF mapping and processing resource allocation in a delay-aware VNF migration problem as a mixed-integer nonlinear programming problem.
- We decompose the VNF migration problem into two subproblems: VNF mapping subproblem and processing resource allocation subproblem, where the former is a combinatorial optimization problem and the latter is a convex optimization problem.
- We propose an algorithm using DNN with attention based mechanism for solving the VNF mapping subproblem. In particular, we employ a DNN module as an encoder to learn the high-dimensional representations of service requests packet arrival rate and previously executed VNF mapping solution. We employ another DNN module as a decoder, which takes the representations as input and determines the stochastic policy for VNF mapping.
- Simulation results show that our proposed algorithm achieves near-optimal performance, with an optimality gap of 4.5% on average and a lower runtime when compared with the optimal solution based on solving a mixed-integer quadratically constrained programming (MIQCP) problem.

This paper is organized as follows. The system model is described in Section II. The problem formulation is presented in Section III. Our proposed algorithm is given in Section IV. In Section V, we evaluate the performance of the proposed algorithm. Section VI concludes the paper.

II. System Model

Consider a 5G core network shown in Fig. 1 that consists of directed links and SDN-enabled nodes, including switches and NFV-enabled nodes. Switches have only forwarding capability to direct traffic from their incoming links to the outgoing links. Those switches, which are at the edge of the network, are considered as access nodes. NFV-enabled nodes have both forwarding and processing capabilities. Given the limited processing capacity of NFV-enabled nodes, different VNFIs are placed on each NFV-enabled node. A service provider aims to support multiple network services in the form of SFCs in the network. Specifically, the service provider should have the SFC reconfiguration ability to embed the newly arrived service requests in the network and handle the traffic load fluctuation of the service requests.

Let $\mathcal{R}$ denote the set of service requests supported by the service provider. Each service request can arrive or depart the network at different point of time. Let $s^{(r)}$ and $d^{(r)}$ denote the source and destination nodes of service request $r \in \mathcal{R}$, respectively. The source and destination nodes are the access nodes. Given all possible combinations of source nodes and destination nodes, the service provider can obtain the total number of service requests (i.e., $|\mathcal{R}|$). Let $\Pi^{(r)}$ denote the SFC

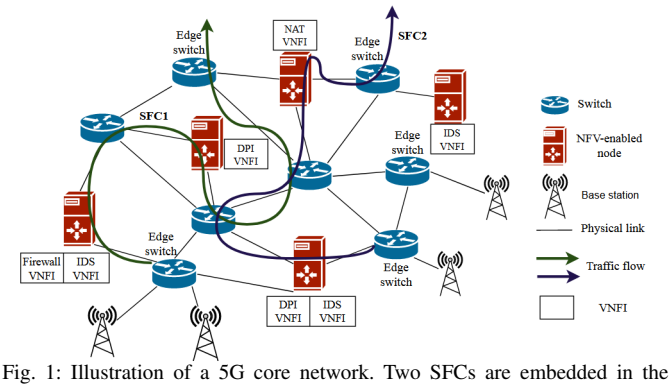


Fig. 1: Illustration of a 5G core network. Two SFCs are embedded in the network using the predetermined VNFIs.

of request $r \in \mathcal{R}$. $\Pi^{(r)}$ contains a sequence of VNFs through which request r should pass in the network. We denote the set of VNF indices in SFC of request r by $\mathcal{N}^{(r)} = \{1, \ldots, N^{(r)}\}$. VNF index $n \in \mathcal{N}^{(r)}$ in $\Pi^{(r)}$ is denoted as $\pi_n^{(r)}$ and we have $\Pi^{(r)} = (\pi_1^{(r)} \rightarrow \pi_2^{(r)} \rightarrow \cdots \rightarrow \pi_{N^{(r)}}^{(r)})$. We use $\pi_0^{(r)}$ and $\pi_{N^{(r)}+1}^{(r)}$ as two dummy VNFs located, respectively, at source node $s^{(r)}$ and destination node $d^{(r)}$. Let $\mathcal{F}$ denote the set of all VNF types (e.g., firewall, IDS) supported by the service provider. We assume that at most one VNF of type $f \in \mathcal{F}$ exists in each SFC $\Pi^{(r)}$. Each service request r has an average end-to-end delay requirement $D^{(r)}$ and an average arrival rate $\lambda^{(r)}$ in packets per second. Since the source node of each request r is an access node, traffic flow of request r is the aggregate traffic of the users sending their traffic through the base stations to the edge switch $s^{(r)}$ and requiring the same SFC $\Pi^{(r)}$ between the same source and destination nodes [12]. Hence, $\lambda^{(r)}$ can fluctuate over time due to new users' subscription or mobility.

Let $\mathcal{E}$ denote the set of physical links in the network. We denote the bandwidth of link $e \in \mathcal{E}$ by B_e in bits per second. We also define a virtual link as a directed logical link between two consecutive VNFs (i.e., from $\pi_n^{(r)}$ to $\pi_{n+1}^{(r)}$, $n \in \{0\} \cup \mathcal{N}^{(r)}$). Each VNF is embedded to a single NFV-enabled node in the physical network. Each virtual link can be embedded to multiple physical links with SDN switches among them.

In practical systems, VNFs are operated in VMs, known as VNFIs. Multiple VNFIs, each with different type $f \in \mathcal{F}$, can be mapped to one NFV-enabled node. Also, multiple VNFs with type f, each belongs to different service request $r \in \mathcal{R}$, can share the processing resource capacity of one VNFI with the same type. Let $G(\mathcal{I} \cup \mathcal{A}, \mathcal{L})$ denote a directed graph, where $\mathcal{I}$ and $\mathcal{A}$ denote the set of VNFI nodes and access nodes, respectively, and $\mathcal{L}$ is a set of logical links between the nodes. Let $\mathcal{I}_n^{(r)} \subseteq \mathcal{I}$ denote the subset of VNFI nodes that have the same type as VNF $\pi_n^{(r)}$, $n \in \mathcal{N}^{(r)}$ for service request $r \in \mathcal{R}$. We denote the processing resource capacity of VNFI $i \in \mathcal{I}$ by C_i in cycle/s. Let $P_i^{(r)}$ denote the processing density (in cycle/packet) of VNFI i for service request r. The processing density depends on the service type and VNF type.

There is a directed logical link between two nodes $i, j \in \mathcal{I} \cup \mathcal{A}$ if there is at least one virtual link between the VNFI/access node, which is mapped to node i, and the

VNFI/access node, which is mapped to node j. We define an $|\mathcal{I} \cup \mathcal{A}| \times |\mathcal{I} \cup \mathcal{A}|$ matrix $\boldsymbol{L}$, where $l_{ij} = 1$ if there is a logical link from node i to j, and $l_{ij} = 0$ otherwise. For routing traffic from node i to node j, the service provider can use the available mapping of the virtual links to the physical links (i.e., predetermined tunnels) if $l_{ij} = 1$. Otherwise, given the capacity of the physical links (i.e., $B_e,\ e \in \mathcal{E}$), the service provider can solve a routing problem to find the paths connecting node i and node j. Let $t^{(r),\text{trans}}$ and t_{ij}^{prop} denote, respectively, the transmission delay for service request r and propagation delay along the path from i to j. In this work, we ignore the switch processing delay, which is negligible in comparison with other delays in the network.

Due to the time-varying nature of the service requests traffic load, $\lambda^{(r)}$ changes over time. If $\lambda^{(r)}$ changes in a way that degrades the QoS for service request r or reduces the resource utilization of the VNFIs used by service request r, then the service provider should consider SFC reconfiguration by solving a VNF migration problem. The service provider can consider SFC reconfiguration for embedding a newly arrived service request. Thus, the service provider may need to release some idle resources, embed the new service requests, and consider a stateful flow migration for the existing service requests in the reconfiguration process [4], [5].

III. Problem Formulation

Two sets of overhead can be considered for reconfiguration of the existing service requests. The first set of overhead is required due to the stateful mapping of a VNF with type $f \in \mathcal{F}$ for a service request from one VNFI node to another VNFI node. The second set of overhead is required when additional logical links are required for rerouting the traffic. The service provider aims to keep the reconfiguration overhead at a low level by reducing the number of stateful VNF migrations and decreasing the number of required extra logical links. The service provider aims to minimize the maximum processing resource utilization ratio among all the VNFIs in the network in order to balance the load among different VNFIs.

We use the binary decision variable $x_i^{(r)}$ to indicate whether VNFI $i \in \mathcal{I}_n^{(r)}$ is chosen for embedding VNF index $n \in \mathcal{N}^{(r)}$ in SFC of service request $r \in \mathcal{R}$ (i.e., $x_i^{(r)} = 1$) or not (i.e., $x_i^{(r)} = 0$). Each VNF $\pi_n^{(r)}$ in SFC of service request r should be mapped to exactly one VNFI of the same type. We have

$$\textstyle\sum_{i \in \mathcal{I}_n^{(r)}} x_i^{(r)} = \mathbb{1}\big(\lambda^{(r)} > 0\big),\ \ n \in \mathcal{N}^{(r)},\ r \in \mathcal{R}, \quad (1)$$

where $\mathbb{1}(\cdot)$ is an indicator function.

Let $\mu_i^{(r)}$ denote the processing rate (in packets per second) which is allocated to service request $r \in \mathcal{R}$ at VNFI $i \in \mathcal{I}$. Considering the processing resource capacity C_i and the processing density $P_i^{(r)}$, we have the following constraint:

$$\lambda^{(r)} < \mu_i^{(r)} \le C_i / P_i^{(r)},\ \ i \in \mathcal{I},\ r \in \mathcal{R}, \quad (2)$$

where the lower bound in (2) is required for the VNFI node queue to be stable. We use variable η to denote the maximum processing resource utilization ratio. Hence, We have

$$\textstyle\sum_{r \in \mathcal{R}} x_i^{(r)} P_i^{(r)} \mu_i^{(r)} \le \eta\, C_i,\ \ i \in \mathcal{I}, \quad (3a)$$

$$0 \le \eta \le 1. \quad (3b)$$

Let $t_i^{(r),\text{VNFI}}$ denote the average delay for service request $r \in \mathcal{R}$ in VNFI node $i \in \mathcal{I}$. Assuming that the traffic arrival rate and packet processing time follow Poisson and exponential distributions, respectively, the processing system is an M/M/1 queue. We have

$$t_i^{(r),\text{VNFI}} = \frac{1}{\mu_i^{(r)} - \lambda^{(r)} + \epsilon},\ \ i \in \mathcal{I},\ r \in \mathcal{R}, \quad (4)$$

where $0 < \epsilon \ll 1$ is a constant. Considering the average end-to-end delay requirement for each service request $r \in \mathcal{R}$, we have the following constraint:

$$t^{(r),\text{trans}} + \textstyle\sum_{n \in \{0\} \cup \mathcal{N}^{(r)}} \sum_{i \in \mathcal{I}_n^{(r)}} \sum_{j \in \mathcal{I}_{n+1}^{(r)}} x_i^{(r)} x_j^{(r)} t_{ij}^{\text{prop}}$$
$$+ \textstyle\sum_{n \in \mathcal{N}^{(r)}} \sum_{i \in \mathcal{I}_n^{(r)}} x_i^{(r)} t_i^{(r),\text{VNFI}} \le D^{(r)},\ \ r \in \mathcal{R}, \quad (5)$$

where $\mathcal{I}_0^{(r)}, \mathcal{I}_{N^{(r)}+1}^{(r)} \subset \mathcal{A}$ denote the set of source and destination nodes for service request $r \in \mathcal{R}$. We set $x_i^{(r)} = 1$ for $i \in \mathcal{I}_0^{(r)}, \mathcal{I}_{N^{(r)}+1}^{(r)}$. We define binary variable $y_{ij}^{(r)}$ to indicate whether a directed logical link is required to be created from node i to node j, $i, j \in \mathcal{I} \cup \mathcal{A}$ for service request $r \in \mathcal{R}$ (i.e., $y_{ij}^{(r)} = 1$) or not (i.e., $y_{ij}^{(r)} = 0$). We have

$$y_{ij}^{(r)} = \begin{cases} \mathbb{1}(l_{ij} = 0)\mathbb{1}\Big(x_i^{(r)} = 1\Big)\mathbb{1}\Big(x_j^{(r)} = 1\Big), \\ \qquad i \in \mathcal{I}_n^{(r)},\ j \in \mathcal{I}_{n+1}^{(r)},\ n \in \{0\} \cup \mathcal{N}^{(r)},\ r \in \mathcal{R}, \\ 0, \quad \text{otherwise}, \end{cases} \quad (6)$$

The objective function of the SFC reconfiguration problem is as follows:

$$f^{\text{obj}} = \alpha_1 \eta + \alpha_2 \textstyle\sum_{r \in \mathcal{R}} \sum_{n \in \mathcal{N}^{(r)}} \sum_{i, j \in \mathcal{I}_n^{(r)},\, i \ne j} x_i^{(r)} x_{j,0}^{(r)}$$
$$+ \alpha_3 \textstyle\sum_{r \in \mathcal{R}} \sum_{i,j \in \mathcal{I} \cup \mathcal{A}} y_{ij}^{(r)}, \quad (7)$$

where α_1, α_2, and α_3 are the priority weights, which are used to control the tradeoff between the load among the VNFIs and the reconfiguration overhead. In particular, α_1, α_2, and α_3 are coefficients for maximum loading factor η among all VNFIs, number of VNF migrations ν, and the number of additional logical links, respectively. $x_{j,0}^{(r)}$ is a known parameter that indicates the state before reconfiguration. That is, it shows whether the VNFI $j \in \mathcal{I}_n^{(r)}$ was chosen for embedding VNF index $n \in \mathcal{N}^{(r)}$ in SFC of service request r before reconfiguration (i.e., $x_{j,0}^{(r)} = 1$) or not (i.e., $x_{j,0}^{(r)} = 0$). For the newly arrived service requests, we set $x_{j,0}^{(r)} = 0$. The service provider aims to solve the following optimization problem:

$$\underset{\eta,\, x_i^{(r)},\, \mu_i^{(r)},\, i \in \mathcal{I},\, r \in \mathcal{R}}{\text{minimize}} \quad f^{\text{obj}} \quad (8)$$
$$\text{subject to} \quad \text{constraints (1)}-\text{(6)}.$$

Problem (8) is a mixed-integer nonlinear program, which is NP-hard and difficult to solve. Note that the interruption due to service request can be detrimental if an algorithm with a

high computational complexity is used to find a new reconfiguration. In the next section, we propose a low computational complexity algorithm using attention-based DNNs to find a near-optimal solution for problem (8).

IV. Proposed Algorithm

We decompose problem (8) into two subproblems for obtaining the VNF mapping (i.e., obtaining $x_i^{(r)}$) and processing resource allocation (i.e., obtaining η and $\mu_i^{(r)}$), respectively. Given the processing resource allocation, we have the following VNF mapping subproblem:

$$\underset{x_i^{(r)},\, i\in\mathcal{I},\, r\in\mathcal{R}}{\text{minimize}} \quad f^{\text{obj}} \quad \text{subject to constraints (1) and (6).} \tag{9}$$

Given the VNF mapping, we have the following processing resource allocation subproblem:

$$\underset{\eta,\, \mu_i^{(r)},\, i\in\mathcal{I},\, r\in\mathcal{R}}{\text{minimize}} \quad f^{\text{obj}} \quad \text{subject to constraints (2)}-\text{(5).} \tag{10}$$

Since subproblem (10) is a convex optimization problem, its optimal solution can be obtained efficiently. However, subproblem (9) is a combinatorial optimization problem, which is NP-hard [11]. DNN with attention mechanism is a powerful tool for finding a near-optimal solution for combinatorial optimization problems [11], [13]. Hence, to solve subproblem (9), we design encoder and decoder DNNs for obtaining the stochastic policy to efficiently determine the mapping of VNFs on VNFI nodes after the training phase of the DNN modules.

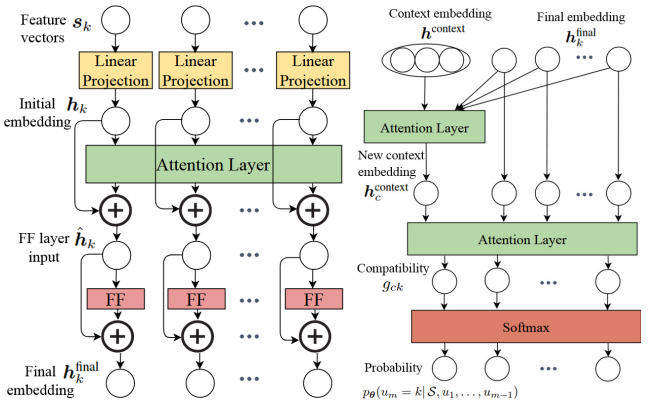
A. Stochastic Policy and Encoder

For the training phase of the algorithm, we need training data, which consists of different instances of the VNF mapping problem. Let $\mathcal{K} = \left\{(i,\, r)\,|\, n\in\mathcal{N}^{(r)},\, i\in\mathcal{I}_n^{(r)},\, r\in\mathcal{R}\right\}$ denote the set of all possible service request-VNFI node pairs in the network. We denote the feature vector of a possible service request-VNFI node pair $k\in\mathcal{K}$ by $\boldsymbol{s}_k = \left(\lambda^{(r)},\, x_{i,0}^{(r)}\right)$. To show a problem instance, we use set $\mathcal{S} = \{\boldsymbol{s}_k\,|\,k\in\mathcal{K}\}$. For each problem instance, we aim to find a selection of possible service request-VNFI node pairs $\mathcal{U} = \{u_1,\ldots,u_M\}\subseteq\mathcal{K}$, where $M = \sum_{\{r\in\mathcal{R}\,|\,\lambda^{(r)}>0\}}|\mathcal{N}^{(r)}|$. We set $x_i^{(r)} = 1$ for the pair $i\in\mathcal{I}$ and $r\in\mathcal{R}$, which is related to each $u_m\in\mathcal{U}$. Let $p(\mathcal{U}\,|\,\mathcal{S})$ denote the stochastic policy of selecting a solution $\mathcal{U}$ given the problem instance $\mathcal{S}$. Using the attention-based DNN model, we can factorize $p(\mathcal{U}\,|\,\mathcal{S})$ and parameterize it by $\boldsymbol{\theta}$ as:

$$p_{\boldsymbol{\theta}}(\mathcal{U}\,|\,\mathcal{S}) = \textstyle\prod_{m=1}^{M} p_{\boldsymbol{\theta}}(u_m\,|\,\mathcal{S}, u_1,\ldots,u_{m-1}). \tag{11}$$

The encoder module produces embeddings (i.e., high-dimensional representations) of the feature vector $\boldsymbol{s}_k$ for all the service request-VNFI node pairs in a problem instance $\mathcal{S}$. Given the encoder embeddings, the decoder module produces the sequence $\mathcal{U}$ using (11).

Fig. 2(a) shows an illustration of DNN structure for the encoder module. The 2-dimensional feature vectors are the inputs of the encoder. A linear projection layer with learnable parameters $\boldsymbol{W}_{\text{en}}^{s}\in\mathbb{R}^{d_h\times 2}$ and $\boldsymbol{b}_{\text{en}}^{s}\in\mathbb{R}^{d_h\times 1}$ is used to compute the initial d_h-dimensional embeddings of the inputs.



(a) Encoder module (b) Decoder module

Fig. 2: Illustration of the DNN structure for (a) the encoder module and (b) the decoder module.

Let $\boldsymbol{h}_k = \boldsymbol{W}_{\text{en}}^{s}\boldsymbol{s}_k + \boldsymbol{b}_{\text{en}}^{s}$ denote the initial embedding of the feature vector $\boldsymbol{s}_k$. The initial embeddings are the inputs of the attention layer. Attention mechanism is a weighted message passing algorithm between the inputs of the attention layer in order to highlight the relevant parts of the input [11]. For obtaining the message that an initial embedding receives from the other initial embeddings, at first, three vectors, which are named query $\boldsymbol{q}_k = \boldsymbol{W}_{\text{en}}^{q}\boldsymbol{h}_k$, key $\boldsymbol{\kappa}_k = \boldsymbol{W}_{\text{en}}^{\kappa}\boldsymbol{h}_k$, and value $\boldsymbol{v}_k = \boldsymbol{W}_{\text{en}}^{v}\boldsymbol{h}_k$, are computed for each initial embedding $\boldsymbol{h}_k$, where $\boldsymbol{W}_{\text{en}}^{q}\in\mathbb{R}^{d_\kappa\times d_h}$, $\boldsymbol{W}_{\text{en}}^{\kappa}\in\mathbb{R}^{d_\kappa\times d_h}$, and $\boldsymbol{W}_{\text{en}}^{v}\in\mathbb{R}^{d_h\times d_h}$ are the learnable parameters. Then, the compatibility $g_{kl}\in\mathbb{R}$ is computed between the query $\boldsymbol{q}_k$ of $\boldsymbol{h}_k$ and the key $\boldsymbol{\kappa}_l$ of another initial embedding $\boldsymbol{h}_l$ as $g_{kl} = \frac{\boldsymbol{q}_k\cdot\boldsymbol{\kappa}_l}{\sqrt{d_\kappa}}$. The attention weight $a_{kl}\in[0,1]$, which is the weight of the message value sent by each initial embedding $\boldsymbol{h}_l$ to initial embedding $\boldsymbol{h}_k$, can be obtained using a softmax function as $a_{kl} = \frac{e^{g_{kl}}}{\sum_{l'\in\mathcal{K}} e^{g_{kl'}}}$. The output of the attention layer is the sum of the weighted message value received by each initial embedding $\boldsymbol{h}_k$ from the other initial embeddings. We have $\boldsymbol{h}'_k = \sum_{l\in\mathcal{K}} a_{kl}\boldsymbol{v}_l$.

Using a skip connection [11] in our encoder, we obtain the input of the next layer, which is a fully connected feed-forward (FF) layer. Let $\hat{\boldsymbol{h}}_k = \boldsymbol{h}_k + \boldsymbol{h}'_k$ denote the input of the FF layer. For the FF layer, we use one hidden layer with dimension d_f. Let $\boldsymbol{W}^{f}$ and $\tilde{\boldsymbol{h}}_k$ denote, respectively, the learnable parameters and the output of the FF layer. We denote the output of the encoder module by $\boldsymbol{h}_k^{\text{final}}$. Using a skip connection, the final embedding corresponding to each input feature vector $\boldsymbol{s}_k$ is computed as $\boldsymbol{h}_k^{\text{final}} = \hat{\boldsymbol{h}}_k + \tilde{\boldsymbol{h}}_k$.

B. Decoder

Fig. 2(b) shows the decoder structure. The decoder module is invoked M times. At each iteration, it computes a probability distribution over the service request r and VNFI node i pairs, whose selection is possible but they are not selected yet. That is, at each iteration m, the output is the probability of selecting a possible index k given the indices $u_1,\ldots,u_{m-1}$, which are selected in the previous iterations, i.e., $p_{\boldsymbol{\theta}}(u_m = k\,|\,\mathcal{S}, u_1,\ldots,u_{m-1})$. The input of the decoder has two parts. The first part is the output of the encoder, i.e., the final embeddings $\boldsymbol{h}_k^{\text{final}}$, which are fixed for all M iterations. The

second part, which is called context embedding $\boldsymbol{h}^{\text{context}}$, is obtained as $\boldsymbol{h}^{\text{context}} = \left[\boldsymbol{h}^{\text{mean}}, \boldsymbol{h}^{\text{final}}_{u_{m-1}}, M-m+1\right]$, where $\boldsymbol{h}^{\text{mean}}$ is the mean of the final embeddings, $\boldsymbol{h}^{\text{final}}_{u_{m-1}}$ is the final embedding of the previously selected service request-VNFI node pair, and $M-m+1$ helps the decoder to determine the number of remaining selections. At $m=1$, we set $\boldsymbol{h}^{\text{final}}_{u_0} = 0$.

The first layer in the decoder is an attention layer. At each iteration m, we mask all the final embeddings with $\lambda^{(r)} = 0$ in their corresponding feature vector. We mask the final embeddings corresponding to the previously selected indices u_l, $l = \{0, \ldots, m-1\}$. Each u_l specifies the selected pair of service request r and VNFI node i. Given constraint (1), we mask all the other embeddings which correspond to the pairs of the same request r and the other VNFI nodes with the same type of VNFI node i. We consider $\boldsymbol{W}^{q}_{\text{de},1} \in \mathbb{R}^{d_\kappa \times (2d_h+1)}$, $\boldsymbol{W}^{\kappa}_{\text{de},1} \in \mathbb{R}^{d_\kappa \times d_h}$, and $\boldsymbol{W}^{v}_{\text{de},1} \in \mathbb{R}^{d_h \times d_h}$ as the learnable parameters for the attention layer in the first layer of the decoder DNN. The second layer of the decoder is another attention layer. We use the same masking procedure as the one in the first attention layer of the decoder. The outputs of this layer are the probabilities over possible indices k, i.e., $p_{\boldsymbol{\theta}}(u_m = k \,|\, \mathcal{S}, u_1, \ldots, u_{m-1})$. Using the learnable parameters $\boldsymbol{W}^{q}_{\text{de},2} \in \mathbb{R}^{d_\kappa \times d_h}$ and $\boldsymbol{W}^{\kappa}_{\text{de},2} \in \mathbb{R}^{d_\kappa \times d_h}$, the compatibility $g_{ck} \in \mathbb{R}$ of the query $\boldsymbol{q}^{\text{context}}_c$ of $\boldsymbol{h}^{\text{context}}_c$ with the key $\boldsymbol{\kappa}^{\text{final}}_k$ of $\boldsymbol{h}^{\text{final}}_k$ is computed as $g_{ck} = \beta \tanh\left(\frac{\boldsymbol{q}^{\text{context}}_c \cdot \boldsymbol{\kappa}^{\text{final}}_k}{\sqrt{d_\kappa}}\right)$, where the $\tanh$ function is used to clip the result within $[-\beta, \beta]$. Then, using the softmax function, we have the final probabilities as follows:

$$p_{\boldsymbol{\theta}}(u_m = k \,|\, \mathcal{S}, u_1, \ldots, u_{m-1}) = \frac{e^{g_{ck}}}{\sum_{k' \in \mathcal{K}} e^{g_{ck'}}}, \tag{12}$$

where $\boldsymbol{\theta}$ is the collection of all learnable parameters in both encoder and decoder modules.

C. Training Algorithm

Based on encoder and decoder modules, given a problem instance $\mathcal{S}$ as input, we obtain a probability distribution $p_{\boldsymbol{\theta}}(\mathcal{U} \,|\, \mathcal{S})$, from which we can sample to determine each VNF in the SFC of the service requests should be mapped to which VNFI node, i.e., where $x^{(r)}_i$, $i \in \mathcal{I}$, $r \in \mathcal{R}$ should be equal to one. Considering the objective function in (7), we define $\mathcal{L}(\boldsymbol{\theta} \,|\, \mathcal{S}) = \mathbb{E}_{p_{\boldsymbol{\theta}}(\mathcal{U} \,|\, \mathcal{S})}[f(\mathcal{U})]$ as loss function for training our model and obtaining the optimal learnable parameters $\boldsymbol{\theta}$, where $f(\mathcal{U})$ is the objective value of problem (10). We minimize $\mathcal{L}(\boldsymbol{\theta} \,|\, \mathcal{S})$ by using Adam optimizer [14], and use the REINFORCE gradient estimator as follows:

$$\nabla \mathcal{L}(\boldsymbol{\theta} \,|\, \mathcal{S}) = \mathbb{E}_{p_{\boldsymbol{\theta}}(\mathcal{U} \,|\, \mathcal{S})}\left[f(\mathcal{U}) \nabla \log p_{\boldsymbol{\theta}}(\mathcal{U} \,|\, \mathcal{S})\right]. \tag{13}$$

The training algorithm is shown in Algorithm 1. The computation complexity of the online execution of the proposed algorithm for each SFC reconfiguration problem is $O(|\mathcal{K}| + C_{\text{opt}})$, where C_{opt} is the computational complexity of solving convex problem (10), which has a polynomial time complexity.

Algorithm 1 Training Algorithm for Stateful VNF Mapping

1: Set the number of epochs and batch size B. Initialize $\boldsymbol{\theta}$.
2: **for** each epoch **do**
3: Consider B different problem instances $\mathcal{S}_B$.
4: **for** each $\mathcal{S} \in \mathcal{S}_B$ **do**
5: Feed the feature vectors in $\mathcal{S}$ into the DNN modules and obtain the solution $\mathcal{U}$ using $p_{\boldsymbol{\theta}}(\mathcal{U} \,|\, \mathcal{S})$.
6: Determine $f(\mathcal{U})$ by solving problem (10).
7: Determine $\nabla \mathcal{L}(\boldsymbol{\theta} \,|\, \mathcal{S})$ based on (13).
8: **end for**
9: Determine the aggregate gradient over the batch as $\nabla \mathcal{L}(\boldsymbol{\theta} \,|\, \mathcal{S}_B) := \sum_{\mathcal{S} \in \mathcal{S}_B} \nabla \mathcal{L}(\boldsymbol{\theta} \,|\, \mathcal{S})$
10: Update $\boldsymbol{\theta}$ using Adam optimizer [14].
11: **end for**
12: Outputs are the learned parameters $\boldsymbol{\theta}$.

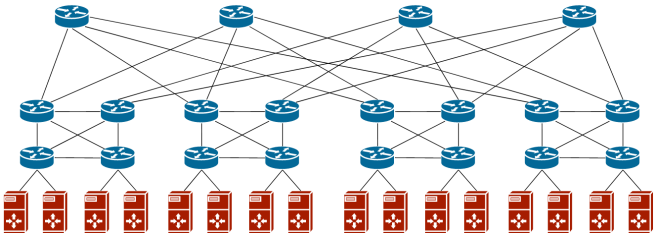


Fig. 3: 4-ary fat tree topology.

V. Performance Evaluation

We consider a 4-ary fat-tree topology with 16 NFV-enabled nodes and 20 SDN switches as shown in Fig. 3. There are four VNF types, i.e., $\mathcal{F} = \{f_1, f_2, f_3, f_4\}$. There is a VNFI for each VNF type on each NFV-enabled node. Thus, the network has 64 VNFI nodes. We set the processing rate for each VNFI to be 1000 packet/s. The propagation delay of the physical links are chosen uniformly between 2 and 5 ms. There are four service requests: SFC1 ($f_2 \to f_3 \to f_4$), SFC2 and SFC3 ($f_1 \to f_2 \to f_3 \to f_4$), and SFC4 ($f_1 \to f_2 \to f_4$). One of them can be selected as the departure service request or the newly arrived service request at the reconfiguration time. For the existing service requests at the reconfiguration time, we set the average arrival rate between $[200, 400]$ packet/s, and the delay requirement $D^{(r)}$ is equal to 45 ms. In Algorithm 1, we set hyperparameters $d_h = d_\kappa = 128$, $\beta = 10$, and $B = 512$. We set the number of epochs to be 5000. For generating the problem instances, given the logical links, we randomly map the VNFs in the SFC of each service request to the VNFI nodes to obtain $x^{(r)}_{i,0}$. We implement our algorithm in Python 3.7 using PyTorch library and MOSEK optimization solver on a PC with Nvidia GeForce RTX 2070 GPU, 2.9 GHz Intel(R) Core(TM) i7-10700 CPU processor and 32 GB memory. For performance comparison, we transform problem (8) into an MIQCP problem using the approach proposed in [10] and solve it by the Gurobi optimization solver.

In Fig. 4, we compare the performance of Algorithm 1 with MIQCP. The priority factor α_1 varies from 0 to 0.8. We set $\alpha_2 = 0.8 - \alpha_1$ and $\alpha_3 = 0.2$. The results are obtained by averaging over 50 realizations in which SFC1, SFC2, and SFC3 share the same VNFI for VNF f_2, SFC1 and SFC3 share the same VNFI for VNF f_4, and SFC2 and SFC4 share another VNFI for VNF f_4, before the reconfiguration time. Results

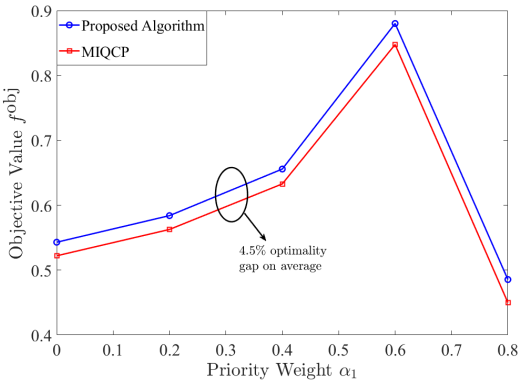


Fig. 4: The objective value versus the priority weight α_1.

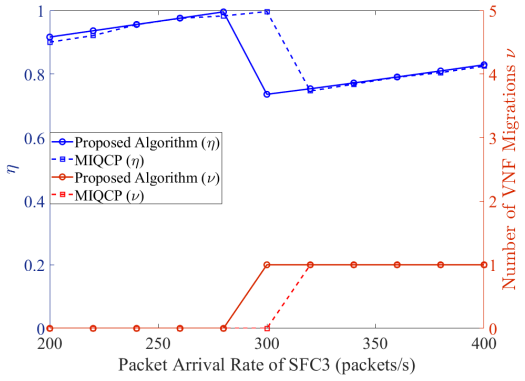


Fig. 5: The maximum loading factor η and the number of VNF migrations ν versus the packet arrival rate for SFC3. We set $\lambda^{(1)} = 350$ packet/s and $\lambda^{(2)} = 250$ packet/s. We set $\alpha_1 = \alpha_2 = 0.4$ and $\alpha_3 = 0.2$.

show that there is on average 4.5% optimality gap between the objective value obtained from our proposed algorithm and the optimal value obtained by solving the MIQCP problem. Note that the runtime for the online execution of our proposed algorithm including solving the convex optimization problem (10) is only 13 ms, whereas solving the MIQCP problem has an exponential time complexity with the runtime of 3.65 s.

In Fig. 5, we investigate the impact of the arrival rate of SFC3 on η and ν when SFC1, SFC2, and SFC3 share the processing resource of one VNFI for VNF f_2 before the reconfiguration time. We consider SFC4 as the newly arrived service request with $\lambda^{(4)} = 400$ packet/s at the reconfiguration time. When the packet arrival rate of SFC3 increases, the maximum loading factor η also increases in order to guarantee the end-to-end delay for SFCs. When η approaches 1, one of the VNFs on the shared VNFI migrates to another VNFI. After the migration, the value of η decreases and the end-to-end delay for SFCs is satisfied. Also, our algorithm maps the VNFs of the SFC4 to the VNFIs which are not shared among the other SFCs. Hence, there is no increase in the values of η and ν due to the newly arrived SFC4.

VI. Conclusion

In this paper, we proposed an algorithm to solve a SFC reconfiguration problem in SDN/NFV enabled 5G core networks. By considering the end-to-end delay for service requests, the limited processing resource capacity of VNFIs, and the order of the VNFs in the SFCs of the service requests, we formulated the joint stateful VNF mapping and processing resource allocation for service requests as a mixed-integer nonlinear optimization problem. To obtain a near-optimal solution for such an NP-hard optimization problem, we decomposed the problem into two subproblems. We proposed an algorithm based on DNN with attention mechanism to solve the first subproblem for obtaining the VNF mapping. Given the solution of the first subproblem, a convex optimization problem is solved in the second subproblem to obtain the processing resource allocation. Through simulations, we have shown that our proposed algorithm can achieve a solution that is close to the optimal solution obtained by transforming the SFC reconfiguration problem into an MIQCP problem. For future work, we will consider the service downtime due to the VNF migration operation, as well as the techniques by which the scalability of the proposed scheme can be improved.

Acknowledgement

This work was supported by Rogers Communications Canada Inc.

References

[1] J. Ordonez-Lucena, P. Ameigeiras, D. Lopez, J. J. Ramos-Munoz, J. Lorca, and J. Folgueira, "Network slicing for 5G with SDN/NFV: Concepts, architectures, and challenges," *IEEE Commun. Mag.*, vol. 55, no. 5, pp. 80–87, May 2017.

[2] W. Zhuang, Q. Ye, F. Lyu, N. Cheng, and J. Ren, "SDN/NFV-empowered future IoV with enhanced communication, computing, and caching," *Proc. of the IEEE*, vol. 108, no. 2, pp. 274–291, Feb. 2019.

[3] J. Halpern and C. Pignataro, "Service function chaining (SFC) architecture," *IETF RFC 7665*, Oct. 2015.

[4] G. Wang, G. Feng, T. Q. Quek, S. Qin, R. Wen, and W. Tan, "Reconfiguration in network slicing – Optimizing the profit and performance," *IEEE Trans. Netw. Service Manag.*, vol. 16, no. 2, pp. 591–605, Jun. 2019.

[5] X. Fei, F. Liu, H. Xu, and H. Jin, "Adaptive VNF scaling and flow routing with proactive demand prediction," in *Proc. of IEEE Int'l Conf. on Computer Commun. (INFOCOM)*, Honolulu, HI, Apr. 2018.

[6] B. Li, B. Cheng, X. Liu, M. Wang, Y. Yue, and J. Chen, "Joint resource optimization and delay-aware virtual network function migration in data center networks," *IEEE Trans. Netw. Service Manag.*, 2021.

[7] N. Zhang, Y.-F. Liu, H. Farmanbar, T.-H. Chang, M. Hong, and Z.-Q. Luo, "Network slicing for service-oriented networks under resource constraints," *IEEE J. Sel. Areas Commun.*, vol. 35, no. 11, pp. 2512–2521, Nov. 2017.

[8] M. M. Tajiki, S. Salsano, L. Chiaraviglio, M. Shojafar, and B. Akbari, "Joint energy efficient and QoS-aware path allocation and VNF placement for service function chaining," *IEEE Trans. Netw. Service Manag.*, vol. 16, no. 1, pp. 374–388, Mar. 2019.

[9] ETSI GS NFV-REL 006 V3.1.1, "Network Functions Virtualisation (NFV) Release 3," Feb. 2018.

[10] K. Qu, W. Zhuang, Q. Ye, X. Shen, X. Li, and J. Rao, "Dynamic flow migration for embedded services in SDN/NFV-enabled 5G core networks," *IEEE Trans. on Commun.*, vol. 68, no. 4, pp. 2394–2408, Apr. 2020.

[11] W. Kool, H. Van Hoof, and M. Welling, "Attention, learn to solve routing problems!" in *Proc. of Int'l Conf. Learn. Representations (ICLR)*, New Orleans, LA, May 2019.

[12] 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.

[13] C. He, Y. Hu, Y. Chen, and B. Zeng, "Joint power allocation and channel assignment for NOMA with deep reinforcement learning," *IEEE J. Sel. Areas Commun.*, vol. 37, no. 10, pp. 2200–2210, Oct. 2019.

[14] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *Proc. of Int'l Conf. Learning Representations (ICLR)*, San Diego, CA, May 2015.