OctoMap: Supporting Service Function Chaining via Supervised Learning and Online Contextual Bandit

Aziza Alzadjali* Maria Mushtaq† Flavio Esposito† Claudio Fiandrino‡ Jitender Deogun*

*University of Nebraska-Lincoln, USA †Saint Louis University, USA ‡IMDEA Networks Institute, Spain

2021 IEEE 7th International Conference on Network Softwarization (NetSoft) | 978-1-6654-0522-5/21/$31.00 ©2021 IEEE | DOI: 10.1109/NetSoft51509.2021.9492660

***Abstract*—Network Function Virtualization (NFV) replaces physical middleboxes with elastic Virtual Network Functions (VNFs). Those VNFs need to be instantiated, and their resources dynamically scaled to meet application and traffic fluctuation requirements. Despite recent extensive research, deciding how to map virtual resources optimally to the underlying infrastructure remains practically a challenge. Existing approaches mostly assign fixed resources to each VNF instance, and transfer virtual flows using a single physical path, without prior knowledge of traffic patterns and available bandwidth. Such resource binding strategies lead to suboptimal physical link utilization. We advance the state of the art in this regard by presenting *OctoMap*, a system designed to support with learning theory any chain embedding algorithm. OctoMap utilizes a Convolution Neural Network for traffic prediction and provisioning, and a contextual multi-armed bandit algorithm to solve the online VNF chain embedding problem. We show the performance benefits of OctoMap with a trace-driven simulation campaign using publicly available datasets. In particular, we show how *OctoMap* reduces the costs of provisioning network services under node and link constraints, comparing different predictors and different multi-armed bandit policies.**

I. Introduction

Processing network traffic requires different network function services such as firewalls, load balancing, and deep packet inspection. These operations can be performed by routing packets through several middleboxes. Network Function Virtualization (NFV) allows an agile deployment of network functions on general platforms using virtualization techniques, often replacing traditional dedicated hardware. Virtual Network Functions (VNFs) are chained by virtual links in a predefined sequence, forming a Service Function Chain (SFC). Many elements of such chains (virtual nodes, or VNFs) are deployed on data center nodes. The chain is then mapped to those servers through the data center physical links to deliver the required traffic flow services [1]. The VNF chain mapping is an NP-hard problem of finding a hosting physical node for each virtual network function, and at least one loop-free physical path for every virtual link connecting the VNF nodes [2]. A Software-Defined Network (SDN) controller is often integrated with the NFV system to control and steer traffic. The dynamic nature of traffic loads hinders the efficiency of both instantiation and maintenance of such chain mappings, resulting in a suboptimal data center and network resource utilization.

The need for better chain embedding prediction systems. Several solutions have been proposed to handle such chain (re)mapping under dynamic traffic loads. Some recent examples include [3]–[6]. Most of these solutions considered fixed resource requirements and assumed a (restrictive) linear relation between VNFs and the allocations of the underlying resources [7], [8]. However, both the complexity and the dynamics of data center traffic call for a higher degree of automation and an efficient decision making process [9]. To this aim, a proactive i.e., online virtual resource optimization strategy could prove very effective for chain embedding decisions, meeting the fluctuating network and service demands while minimizing costs and maximizing the utility.

The decision of how to map chain requests to underlying resources can have a significant performance impact on network utilization and data center maintenance costs. When traffic loads are dynamic, optimizing the chain mapping is challenging without looking at traffic patterns ahead [10], [11]. Fortunately, modern advances in machine learning have enabled accurate predictions of short-term network requests given sufficient historical data [12]. Moreover, deep reinforcement learning has widely demonstrated its effectiveness on several network management problems, including chain mapping [13]–[15]. However, these algorithms are often expensive to train, and their convergence is slow. The reason is that they do not efficiently minimize the regret, i.e., the long term difference in total reward between the exploitation and exploration phases. All the approaches that use deep reinforcement learning to solve the chain mapping problem have inherently this drawback.

Aside from the drawback of using a plain deep reinforcement learning, two critical challenges in solving the chain mapping problem are (i) the dynamic nature of future traffic demands that need to be processed by the chain, and (ii) the partial knowledge of the traffic demands currently flowing in the data center network fabric.

Our Contributions. In this paper, we design and implement a system able to handle the above-explained challenges in an efficient online framework, using learning theory yet minimizing regrets. Specifically, we first propose a geometric program [16] to model the optimization of a set of service chain embedding variables. To minimize the total cost of provisioning all chain requests, we propose *OctoMap*, a system whose design is based around two subsequent prediction steps: first, to handle the dynamic nature of future traffic demands we use supervised learning; second, to address the lack of accurate knowledge of the current states, we use an online *contextual bandit algorithm*. *OctoMap* exploits the available

978-1-6654-0522-5/21/$31.00 ©2021 IEEE

Authorized licensed use limited to: SAINT LOUIS UNIVERSITY. Downloaded on October 05,2021 at 03:36:09 UTC from IEEE Xplore. Restrictions apply.

network information collected from experience while exploring knowledge about the network bandwidth and CPU utilization. Our system can efficiently handle higher dimensional features, extract traffic features, and collect updated network utilization from a logically centralized controller. *OctoMap* decides on the embedding of interconnected NFV chains using states from the underlying network infrastructure: CPU and links bandwidth. Our data-driven simulations are based on real data center traffic and show how *OctoMap* can cope with the traffic dynamics. In our performance evaluation, we used a harm-based approach [17] to quantify the throughput harm caused to the system due to the suboptimal chain allocation and traffic peaks. Harm matrix is more practical and handles a wider range of quality metrics than the traditional Jain's fairness index. Our experiments showed that *OctoMap* did not exceed 50% harm compared to the baseline algorithms.

Paper Organization. The rest of this paper is organized as follows. Our contribution with respect to the most relevant work is discussed in Section II. Section III covers details of the problem formulation while in Section IV we present our system. The performance evaluation is reported in Section V and finally, in Section VI we draw our conclusions.

II. Related Work

Service Function Chain Mapping Algorithms. In the past few years, many studies have tackled the NP-hard [18] chain mapping problem, in data centers or wide area networks. Several heuristics have been proposed to cope with the large scale nature of this problem [19]. Some studies addressed this graph matching problem by utilizing multiple links and multiple VNFs to balance the load on the underlying infrastructure [20], [21]. The authors in [22] instead designed an online scaling chain deployment algorithm across data centers to minimize delays. Moreover, G. Sun et al. [23] proposed a heuristic algorithm to effectively deploy SFCs in parallel by dividing dynamic arrival service requests into multiple subflows and instantiating the same number of chains for data centers. None of those studies consider the continuous network load variations in their solutions, and the efficiency that contextual bandit algorithms have.

The problem of chain mapping is closely related to the Virtual Network Embedding (VNE) problem which involves mapping virtual nodes and links to physical nodes and links in the given substrate network [19], [24]. As our approach, several heuristics and approximation algorithms address the online virtual network embedding problem. Some, e.g. [25] propose to do so with an integer linear program whose objective is to minimize the resource consumption and load balancing. Similar to our contribution, NeuroViNE [26] aims to improve VNE algorithms by leveraging the artificial neural network.

Machine learning for Service Function Chaining. Approaches that share similarities with ours are those applying reinforcement learning for the chain mapping problem. Chen et al. [27] proposed a QoS/QoE-aware SFC framework using reinforcement learning. They included a lightweight QoS over LLDP scheme to collect QoS Information from underlying switches and a Deep Q Network-based chain algorithm for maximizing the reinforcement learning rewards. The actions in deep reinforcement learning algorithms change the state of the environment to achieve an optimal final target; this is not required for addressing the chain mapping decisions, where the taken actions does not affect the network state and the "contexts", $i.e.$, the states. Surprisingly, very few research studies are centered on the dynamic mapping of chains with variable network traffic flows. The closest to our work was proposed by Wang et al. [28], which employs a multi-armed bandit (MAB) for VNF placement. They first used an online algorithm called ski-rental to compute the NFV instances to be deployed. By contrast, we use a CNN predictor for this purpose. They then used a traditional MAB algorithm to decide the VNF placement instead of the contextual bandit and do not account for the variation of traffic flows and physical network constraints, which *OctoMap* does. Unlike prior work in this area, we exploit the online contextual bandit algorithm to support embedding decisions for dynamic SFCs onto a substrate network. Schneider et al. [12] proposed a machine learning approach to allocate virtual resources in NFV systems dynamically. Nevertheless, the authors represented temporal dynamics that updates the environment only if there are changes in network topology and flow on the network, whereas we employ predicted traffic flows to compute required resources to be mapped to each requested chain. The use of a CNN to forecast future traffic load is key to our design since previous traffic conditions are likely to have a significant impact on future service traffic. This is known as *long-range dependence* (LRD), which can be quantified by correlation among network traffic sequences [29]. Statistical methods are unable to capture the LRD and non-linearity of network traffic due to strong theoretical assumptions and the simplistic implementation of the conventional network traffic prediction. Although Support Vector Machine models are robust, large-scale network traffic prediction is slow and requires powerful computational machines [30]. Deep learning models such as Recurrent Neural Network (RNN) have gained superiority over traditional machine learning models, especially in the context of time-series data. Specifically, Long Short Term Memory (LSTM), a class of RNN, have been proven successful in *i)* accurately predicting LTE uplink throughput to promote sophisticated video rate adaptation techniques that significantly improve user experience [31] and *ii)* dynamically predicting traffic load to optimize the resource allocation in network slicing context [32] and 5G core functions [33].

III. Model and Problem Definition

In this section, we define the online (service function) chain embedding problem using geometric programming. Our design predicts the network load to obtain future knowledge of the chain demands. We model our problem as an online network utility maximization, where the unknown inputs are the number and size of the chain service requests as shown in Figure 1.

We consider a set of packets arriving over a time horizon $\mathcal{T}$. We assume that such traffic demands need to be routed across

Authorized licensed use limited to: SAINT LOUIS UNIVERSITY. Downloaded on October 05,2021 at 03:36:09 UTC from IEEE Xplore. Restrictions apply.

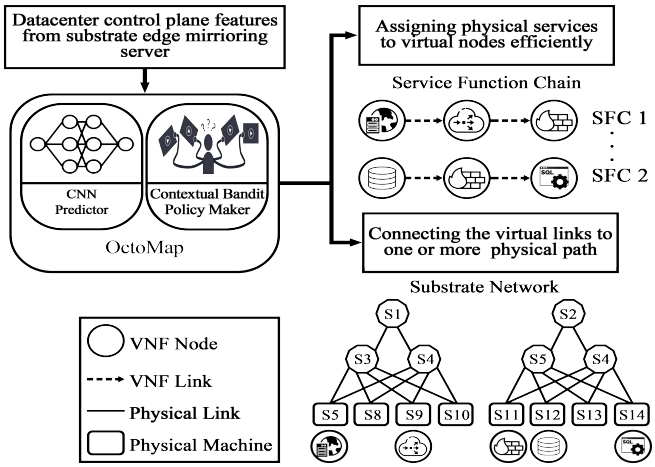


Fig. 1. The proposed supervised online Service Function Chain embedding system: *OctoMap*. A Convolutional Neural Network predicts the incoming packets length and converts them to chain demands; the contextual bandit policy maker decides the chain embedding and connects virtual functions (nodes) to the physical machines using multiple links.

at least one chain of network functions. We model the number of chain requests R from the incoming traffic volume. The optimization problem seeks an optimal allocation of physical hosting nodes and bandwidth link resources to map to a chain. From R, we then extract information about the service types along with CPU and bandwidth demands. We then determine the embedding for virtual nodes (virtual network function instances) and virtual links. For the nodes, we map the requests to the underlying physical machines so that the node resources are allocated. For the links, we attempt to find the number of physical links to be used for transmitting all service demands in R. As typical for these problems [19], we consider the consumption on the virtual CPU as the node (computing) resource constraint and the residual link bandwidth capacity as transmission resource constraint.

We assume that the set of $\mathcal{P}$ packets at time $t \in \mathcal{T}$ is proportional to the chain request demand, as in [5]. We model the chain in a k-port fat-tree topology, for ease of notation denoted as a directed graph $G = (V, E)$, where V is the set of physical hosting machines, and E denotes the physical link set in the network topology. Elements in G are associated with a capacity constraint $C = (C_v, C_e)$, where C_v specifies the hardware capacity of the physical nodes and C_e the bandwidth resource capacity of the physical links. Among the subset of all physical resources potentially available to map a node of the chain, we seek a non-empty subset of all available hosting source and destination nodes, along with traffic transmission links, if available. The time to probe the search space for this problem may have polynomial or exponential complexity, depending on the number of physical nodes and links [34]–[36].

Let $r_i \in R$ be a subset of i service chain requests. From r_i, we can extract the service types along with CPU and bandwidth demands of r_i. Each set of packets $\mathcal{P}$ used to form the chain mapping requests in R generate a utility level denoted as U_{pr}, it represents the translation of packet sets mapped to requested services in the chain. We denote x_{pr} as the binary variable, set to one if the incoming flows/packets is bound to chain request r, and zero otherwise. n^V_{vr} and l^E_{er} are binary variables that are unitary if virtual node v or virtual link e, respectively, have been assigned to chain request r, and zero otherwise. The overall objective function is then, for each period $\mathcal{T}$:

$$\begin{aligned} &\max_R \sum_{p\in\mathcal{P}} \sum_{r\in R} x_{pr} n^V_{vr} l^E_{er} U_{pr}, \\ &x_{pr}, n^V_{vr}, l^E_{er} \in \{0,1\}. \end{aligned} \tag{1}$$

Note that equation (1) is a *posynomial* function [37], *i.e.*, it contains a product of decision variables. We assume a fixed number of chain requests r common to each request type. Each request r is associated with resource usage over the data center. If a chain request is mapped to the physical machines s and d, then this embedding generates costs of CPU load on those machines and packet processing load on the network elements (e.g., network interface cards, switches) along the paths between s and d. Such CPU and processing costs should be considered to determine the optimal SFC embedding decisions.

We denote the CPU usage as c^v_{pr}, and the total CPU capacity of machine v as C_v. This ensures that for each $v \in V$ at each time $t \in \mathcal{T}$, the following constraint is imposed:

$$\sum_{p\in\mathcal{P}(t)} \sum_{r\in R} x_{pr} c^v_{pr} \leq C_v, \ \forall v \tag{2}$$

The second constraint set of the mapping process concerns the link usage, i.e., the cost of processing the traffic that request r generates across each portion of the network that interconnects the single SFC instances allocated to specific machines. Each packet p belonging to r consumes the bandwidth available over the set of links E. We denote this cost as c^e_{pr} and the total bandwidth as C_e. For each link $e \in E$ and at each time $t \in \mathcal{T}$, the following constraint must hold:

$$\sum_{p\in\mathcal{P}(t)} \sum_{r\in R} x_{pr} c^e_{pr} \leq C_e, \ \forall e \tag{3}$$

The embedding is optimal if the utility U_{pr} in equation (1) is maximized, subject to the resource constraints (2) and (3). Our goal is to map the chain online in a cost-efficient way and evaluate our performance on the optimization problem. In our online learning approach, at each time $t \in \mathcal{T}$, the system reads a subset of dynamically predicted $\mathcal{P}$ and optimize accordingly. Finding such optimal chain mapping is NP-hard [38]. The NFV chain mapping can indeed be re-conducted to a graph matching problem where the hosting graph is a physical network with constraints and the NFV is a virtual network with linear topology. *OctoMap* uses an efficient heuristic algorithm to balance the tradeoff between the predefined metrics: CPU and link bandwidth costs of the deployed VNF instances.

IV. OctoMap for Service Function Chain Embedding Decision

In this section, we present *OctoMap*, an online machine learning-based chain embedding algorithm to assist in NFVs mapping. Our design principle is based on the simultaneous

Authorized licensed use limited to: SAINT LOUIS UNIVERSITY. Downloaded on October 05,2021 at 03:36:09 UTC from IEEE Xplore. Restrictions apply.

Algorithm 1: *OctoMap:* Optimal nodes and links selection for users SFC requests using contextual multi-armed bandit based on softmax explorer policy.

Input: The predicted network demand, the λ hyperparameters to control exploration
Output: The SFC mapping of nodes and path for each set of SFC requests R
repeat
Collecting the contexts x data: SFC requests demand obtained from predictor,
for *each successive set of SFC mapping requests* R *within time* t*, with contexts* x_k^t **do**
  Sample action a with softmax policy function
  Update λ
Until obtained reward r_k^t, update observation $\{x^t, r_k^t\}$ to the history for arm k, update learning oracle
return The optimal nodes and links number for every R based on an online trained and updated data.

optimization of both node and link assignments. Our system uses first supervised learning on the network data statistics to train, predict, and map live traffic to chain requests. Then we use a multi-armed bandit to converge on the embedding decisions for nodes and links. Figure 1 shows the workflow of the proposed solution.

OctoMap solves the online chain embedding problem by predicting the number of chain requests to be deployed on the substrate network in advance. *OctoMap* uses the predicted normalized traffic volume to generate the chain requests and assign the corresponding resources for each chain. Section V covers the details of this process. With the forecasts of the future chain requests, *OctoMap* maps the request onto source and destination servers and selects multiple physical links from source to destination for each virtual link. Such embedding fulfills all resource capacity constraints: each physical link should have enough bandwidth, and each physical machine enough computing resources to satisfy the chain request demands. Being an online algorithm, *OctoMap* acquires CPU and bandwidth usage statistics from the network controller in (tunable) one-minute batches, $\mathcal{T}$. Then, it applies the sequential multi-armed bandit. Algorithm 1 details the operational workflow of OctoMap.

The goal of *OctoMap* is to use machine learning to support the SFC embedding in NFV systems optimally. We first use a CNN-trained model to predict and model the VNF resource requirements. The prediction is then passed to the contextual bandit policy to improve the dynamic chain embedding decision. The learning strategy is essential to solve the SFC nodes and links selection problem defined in Section III and to properly provision resources to VNFs tailored to scale dynamically.

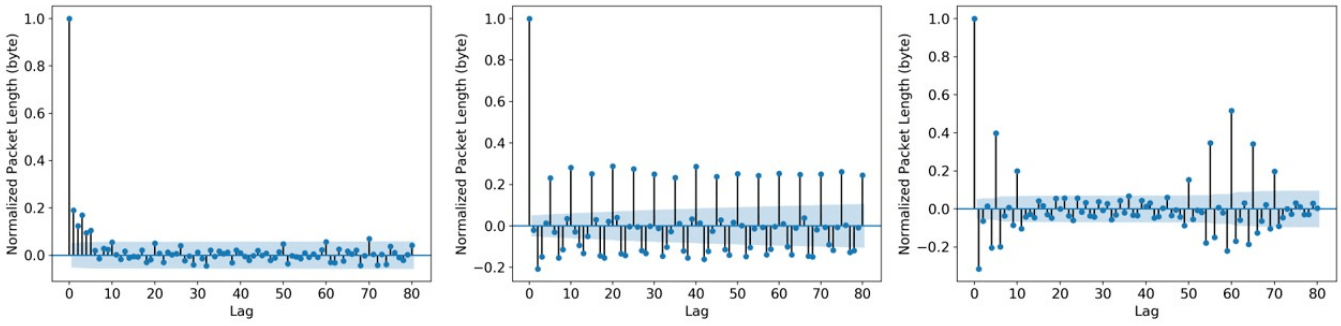
A. Convolutional Neural Network for traffic Load Prediction

For efficient service function chain embedding decision, we introduced a traffic load predictor to enhance the online multi-armed contextual bandit algorithm decisions. Instead of developing a new SFC mapping algorithm, we focus on how to model the embedding, deciding the required resources per SFC request based on the predicted traffic volume. A range of ML predictors exists for network traffic prediction. Specifically, *OctoMap* uses CNN [39] to forecast the incoming data center traffic load. Although CNNs are commonly used for 2D inputs (e.g., images), they are successfully employed in 1D function predictions, too. The alternating convolutional structure of CNN allows recognition of specific repeating patterns in the data, which makes it a good candidate to forecast time series values based on historic observation. CNN models can effectively exploit the temporal nature of the data [40] [41] and easily capture structures of non-linearity present in time series data. Moreover, retraining CNN models for network traffic forecast becomes feasible with the introduction of modern GPUs. These factors make CNN models suitable for real-time scenarios. This predicted traffic load is then used to modulate the SFC requests and their resource demands. The true VNF resource requirements linearly increase with traffic load, assigning fixed resources based on maximum expected traffic leads to over allocation. *OctoMap* uses a real Facebook data center dataset to train the CNN and approximates the SFC requests based on the traffic load.

To train the CNN model, we used the available dataset of Facebook traffic obtained from the Altoona data center [9]. The traces belong to three production unit deployment clusters, each cluster consists of three types of requests: database, web, and big data requests types in cluster-A, B, and C respectively. To estimate the service function chain demands, we extracted and aggregated the packet length and service types. The network traffic traces have temporal structure reflecting patterns in time series data. We approached the network traffic prediction as a supervised learning problem by transforming data into sequences to predict the next minute packet length p_{t+1} one step ahead. Choosing the best number of consecutive packet sizes $p_{t-l}, \ldots, p_{t-1}, p_t$, known as lag, is a hyperparameter tuning step. We applied the auto-correlation function (ACF) to measure the degree of similarity between observations of the time series over successive t lags for each cluster (see Figure 2). Large spikes above the 95% confidence band indicate a significant correlation among the observations at a given lag. For *OctoMap* predictor training, lag values 70, 40, and 60 for clusters A, B, and C were selected respectively. Note that the Facebook dataset does not incorporate network functions, it just predicts traffic load and thus improves the NFV resource allocation process.

CNN Prediction Evaluation: The CNN model was trained on the normalized packet length feature of data from each cluster to predict the data center network traffic load one minute ahead of actual traffic. The trained model was tested over 100 minutes and the trained inference model was used to predict the packet length of the next minute. Figure 3 plots the testing residual error of actual and predicted packet length values for each cluster. For clusters A and C, 97% of the testing errors are below 0.1, and for cluster B around 90% of the testing, the

Authorized licensed use limited to: SAINT LOUIS UNIVERSITY. Downloaded on October 05,2021 at 03:36:09 UTC from IEEE Xplore. Restrictions apply.



(a) Facebook Cluster A – Database Queries (b) Facebook Cluster B – Web Traffic (c) Facebook Cluster C – Hadoop based Traffic

Fig. 2. Autocorrelation function applied on packet length time series of network traces of Facebook data center clusters with band of 95% confidence: (a) cluster-A, (b) cluster-B, and (c) cluster-C. This analysis suggest best lag value to consider for the prediction model (CNN) for supervised training based on highest correlation for each clusters

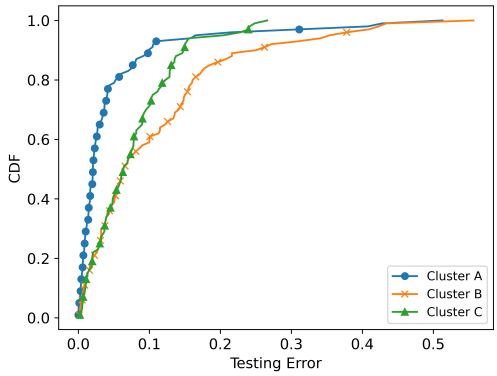


Fig. 3. The residual error for the testing dataset of actual and predicted packet length values for each cluster of Facebook Data center. The error is 90% below 0.2 for all clusters. The predicted values used to modulate the SFC nodes and links requirement and resource demand to assist contextual bandit algorithm in decision making

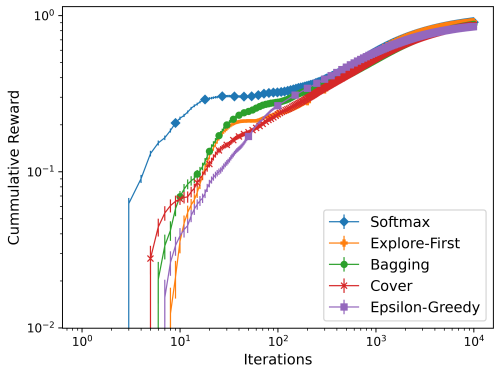


Fig. 4. Contextual bandit cumulative rewards calculated for each policy over 10k iterations. The Softmax policy reaches good cumulative reward values faster than others.

error is below 0.2.

B. Contextual Multi-Armed Bandits for SFC Embedding

Since *OctoMap* uses contextual Multi-Armed Bandits (MAB) [42], [43] approach to VNF mapping decisions, in this Section we give some background of this approach. Multi-armed bandits is a simple and powerful framework for algorithms that make decisions overtime in unstable network conditions. The algorithm can choose from K possible actions (arms) during T rounds. The algorithm chooses an arm in each round and collects a reward. After each round, the algorithm only observes the reward for the arm it chose but have no reward knowledge about the other arms that could have been chosen. Thus, the algorithm needs to acquire new information by trying out different arms, this process is called exploration. If an algorithm always chooses one arm, it would not know if any other arm is better. Therefore, the exploration and exploitation tradeoff is key to make optimal impending decisions based on the available information. In a nutshell, the objective of the algorithm is to learn which is the best arms while not spending too much time exploring [43]. The MAB problem can be formulated as a semi Markov decision process. At each round T, it selects an action $i \in \{1, k\}$, then observes the reward r_i that the world chose during the learning process. The reward is independently picked from a fixed distribution depending on the chosen arm, but not known to the algorithm. The contextual MAB also observes context information x used to determine which action to select. This property helps in making decisions under uncertainty by balancing between the exploration and exploitation of several agents with the help of a greedy policy π that only takes into account short term effects. In the exploration stage, the policy tries different actions to learn the given reward for each of them. While in exploitation, it takes decisions based on collected and learned information in each period to maximize an immediate reward. The context affects how a reward is associated with each action: as contexts change, the model learns to adapt its action choices accordingly. A good policy efficiently maps the best actions with varying contexts. In contextual bandits, the learner observes some contexts repeatedly, then it picks one action and observes the reward or cost associated with the chosen action. Contextual MAB uses some contexts to aid online decision making, which makes it a good candidate for dynamic environments with rapidly changing statistics and a limited set of available actions. We adopt the contextual bandit to find the optimal physical nodes and links mapping for SFC requests under data center networks as an active online learner. In the following, we explain how typical contextual bandit concepts of context, agent, arm (action), and reward are used for the SFC mapping problem.

Contexts: The contexts module is responsible for parsing network status information from the data center controller server, obtaining the contextual features that are continuously observed by the agent. In *OctoMap*, the contexts $x_k^t : t =$

Authorized licensed use limited to: SAINT LOUIS UNIVERSITY. Downloaded on October 05,2021 at 03:36:09 UTC from IEEE Xplore. Restrictions apply.

$rounds,\ k = arms$ represent the dynamic data center traffic load, CPU consumption, and link bandwidth utilization of the physical network topology.

Agent: The agent performs the learning and recommends the actions. In *OctoMap*, the agent is responsible for making decisions of physical nodes assignment and number of links to be used for packets transmission, picked based on the online collection of substrate links status data from the edge monitoring server. In a contextual bandit algorithm, the policy entity makes those decisions by taking contexts as input and returning an action. The goal is to find a policy that maximizes the average reward over a sequence of interactions. ***Arms or Actions:*** The arms in the bandit setting indicate how an agent responds to the observed context. *OctoMap* defines two sets of arms, one for the number of physical machines, and one for the number of physical links. Such a setting fulfills the mapping demand of incoming SFC requests. Therefore, there are two sequential arm selection processes based on the same observed context. Specifically, in each round T, the agent must choose the best arm a_n^T for node assignment, and the best arm a_l^T for the number of links selection. Given this output, *OctoMap* finds the most adequate candidate set sizes among all feasible physical nodes and links for hosting the VNFs which reduce the rejection of the SFC placement requests within a given execution time and directly affects the quality of service and experience.

Reward: The reward is the overall benefit that an agent wants to maximize in the long term. *OctoMap* assigns stochastic binary rewards $r_k^t \in \{0, 1\}$, which evolves linearly for each arm selection over time, and measures cumulative rewards throughout the rounds. If the agent tries to deploy an SFC request in a server with insufficient resources, the reward is equal to zero. When the SFC request is accepted and all VFNs required are allocated, the reward is equal to one. *OctoMap* enforces the agent to take those actions that provide the highest rewards.

$$Cumulative\ Reward = \sum_{t=1}^{T} r_k \tag{4}$$

Contextual Bandit Policies Evaluation: The policy maker is implemented in the agent, and it defines a mapping from the observed contexts of the network environment to the action to be taken in those contexts setup, policy $\pi : (\ context\ x) \ \mapsto (\ action\ a)$. The policy helps the agent to choose actions based on contexts by finding a good decision rule for selecting the action. We use policies that involve machine learning algorithm as an oracle to learn from past information and choose optimal actions based on learned response patterns. In *OctoMap*, this oracle is a Stochastic Gradient Descent (SGD) optimizer. To determine which policy to use, we compare five different contextual bandit exploration policies: softmax explorer (Boltzmann), explore-first, bagging explorer, online cover, and epsilon-greedy. Figure 4 illustrates the cumulative reward behavior for all policies concerning the training time. We found that the contextual softmax explorer is the overall best policy in terms of performance and learning speed. In softmax explorer, the probability of choosing an arm is proportional to an exponential function of the empirical mean of the reward of that arm. It predicts an action along with a prediction score reporting the quality of the selected action, then creates a distribution proportional to $exp(\lambda \cdot score(x, a))$. Actions with higher average rewards are picked with higher probability. To provide controlled exploration based on the uncertainty during learning, we follow a similar approach of [44] by separating the learning rates of each arm. The algorithm for softmax explorer is defined as follows: Given initial empirical mean rewards $r_1(0), ..., r_k(0)$, then

$$p_i(t+1) = \frac{\mathrm{e}^{u_i(t)/\tau}}{\sum_{j=1}^{K} u_i(t)/\tau}, \quad i = 1, ..., K, \tag{5}$$

where $p_i(t+1)$ is the probability of choosing arm i in round $(t+1)$, and τ controls the choice; as $\tau \mapsto \infty$. Softmax explorer algorithm can guarantee a distribution-independent regret bound of order $\sqrt{KT} \cdot \log K$ [44]. Figure 4 also shows that the softmax explorer policy learns and converges faster. Therefore, the rest of the evaluation experiments are carried out using the softmax explorer policy.

V. OctoMap Performance Evaluation

A. Simulation Setup

To evaluate and quantify the performance of *OctoMap*, we setup k-port fat-trees data center topology as the substrate network with scale $k = [4, 16]$. Fat-tree topologies are commonly used in data centers for their properties of low diameter and high bisection bandwidth (respectively the maximum distance between any pair of servers - in several hops, - and the maximum bandwidth that can be supported when the topology is cut in exactly two parts) [45]. *OctoMap* uses dynamic flow scheduling to handle the cost of the chain which is particularly effective with topologies like fat-tree with high bisection bandwidth, i.e., multiple links available between any given pair of hosts allowing for such dynamic flow scheduling. Therefore, although we assess the performance of *OctoMap* in fat-tree topologies, we expect *OctoMap* to provide comparable results with other network topologies designed to support high bisection bandwidth like Jupiter from Google [46]. The nodes CPU capacity C_v is set to 90% per node, and the links bandwidth capacity C_e is set to 10 GB per link. SFC requests R were modulated bi-linearly based on predicted network load. Actual simulated chain requests $R = \sum_k x_k y_k$, where x is the predicted traffic, k is the number of nodes and links, and y is a fixed approximated constant for each requested service type given in our dataset: database, web request, and big data processing services. Based on this bi-linear mapping assumption, we calculated R between $[50, 400]$ in the 4-port fat-tree, and $[200, 1600]$ in the 16-port topology. The nodes CPU usage c_{pr}^r and link bandwidth usage c_p^e were calculated based on R with values falling within $[1, 90]$ and $[1, 10]$ respectively. Based on this setup, the contextual bandit algorithm calculates the costs and assign rewards to each action accordingly. Our simulations are implemented using Vowpal Wabbit [47] toolkit as a cost-sensitive contextual bandit algorithm trained with

Authorized licensed use limited to: SAINT LOUIS UNIVERSITY. Downloaded on October 05,2021 at 03:36:09 UTC from IEEE Xplore. Restrictions apply.

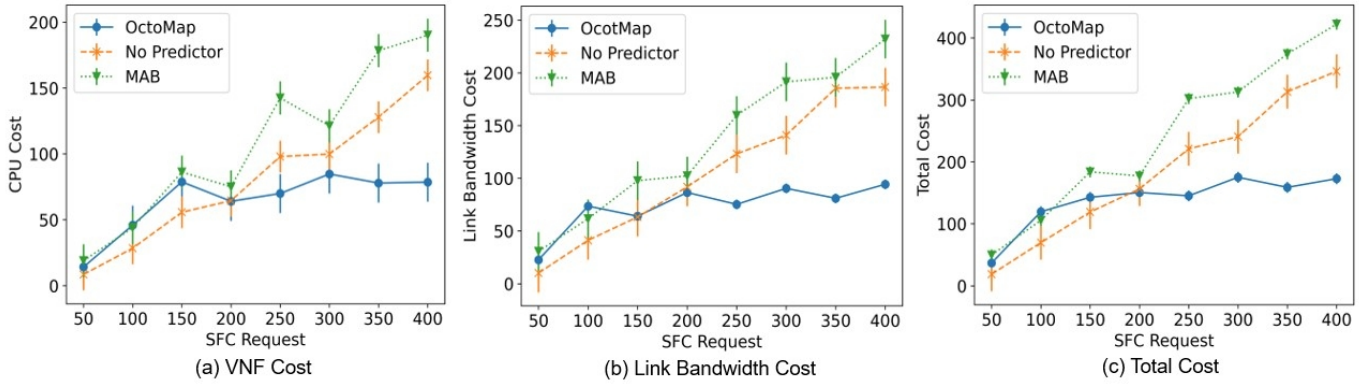


Fig. 5. 4-k fat-tree data center simulations of full *OctoMap* framework, *OctoMap* without the predictor module, and MAB algorithm without contexts (a) CPU cost for mapping the SFC virtual nodes to the physical machines. (b) Links Bandwidth cost for transmitting the SFC requested service through the physical links. (c) Total cost of VNF and link bandwidth. All costs calculated up to 95% confidence level. *OctoMap* achieves less overall cost compared to the baseline.

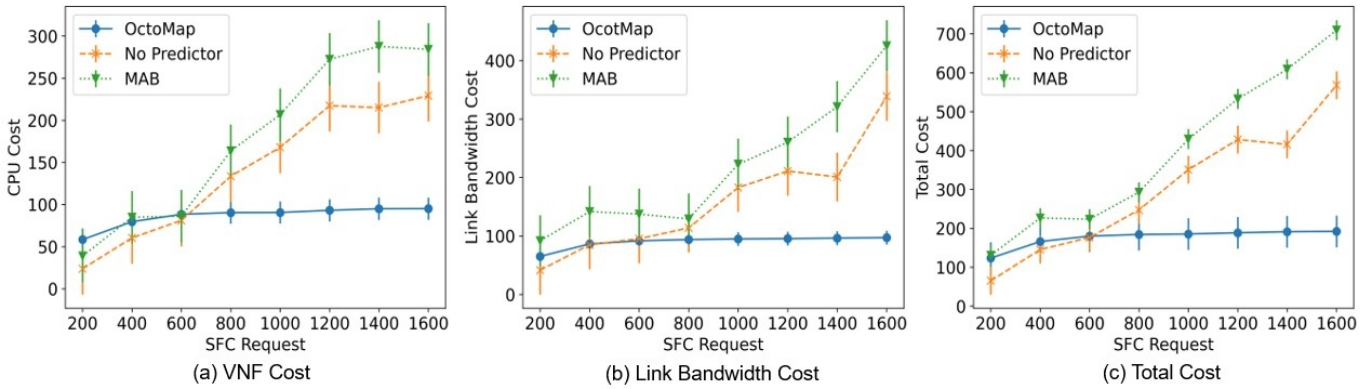


Fig. 6. 16-k fat-tree data center simulations of full *OctoMap* framework, *OctoMap* without the predictor module, and MAB algorithm without contexts (a) CPU cost for mapping the SFC virtual nodes to the required physical machines. (b) Links Bandwidth cost for transmitting the SFC requested service through the physical links. (c) Total cost of VNF and link bandwidth. All costs calculated up to 95% confidence level. *OctoMap* achieves less overall cost compared to the baseline.

online updates. It has several powerful features, including online learning speed and execution time, scalability to a big dataset, and features the pairing option to allow learning based on correlated features. We set the softmax policy hyperparameter $\lambda = 0.5$ for our evaluation purpose, this allows equal amount of exploration and exploitation.

B. Baseline Algorithms

We compare the performance of *OctoMap* with and without the prediction model using the contextual bandit with the active learning softmax policy. We compare our approach with the most similar online SFC mapping work [28], where the authors used an online algorithm to predict the requested service chains, and then used Multi-Armed Bandits (MAB) to map the VNF instances and thus minimize the congestion in a data center network.

C. SFC Embedding Decisions Evaluation

We evaluate the SFC embedding decision through cost generated if we want to map R chain requests, the overall cost contains the physical machines CPU cost and physical links bandwidth cost. The no-prediction baseline is simulated within the same framework and optimization objective as given in equation (1) but with disabling the CNN predictor to show the effect if we do not consider the dynamic traffic load and SFC requests incoming the data center network. The MAB baseline is simulated using the same *OctoMap* framework and topologies except with using contexts-less learning.

We evaluated the amount of throughput harm done by embedding selected network links for SFC requests. Unlike Jain's fairness index, the *harm* matrix handles a wider range of quality metrics than the traditional. To measure the throughput harm, we exploit the definition of [17] that defines the harm as $\frac{y-x}{y}$. In our setting, x is the total throughput capacity, and y is the capacity that a given resource mapping algorithm utilizes for SFC requests. The amount of harm reflects how well the algorithm distributes the SFC demands within the available data center links. The normal values should be placed on a $[0, 1]$ scale, where 1 is the maximum value of harm and 0 represents no harm.

CPU cost: Figures 5(a) and 6(a) depict the physical machines CPU cost with various SFC requests on both 4-port and 16-port fat-tree topologies, respectively. The trend of the CPU cost vs. SFC requests proves that *OctoMap* managed to map all the 400 SFC requests in small scale data center situation while keeping the cost below the overall CPU cost limit of 100. The no-predictor baseline exceeded that limit after 300 and 800 concurrent SFC requests on 4-port and 16-port networks. This implies that the network's capability to handle more SFC

Authorized licensed use limited to: SAINT LOUIS UNIVERSITY. Downloaded on October 05,2021 at 03:36:09 UTC from IEEE Xplore. Restrictions apply.

requests efficiently is limited if no traffic prediction is made in advance. The MAB baseline performed comparably similar to the no-predictor scenario since it does not consider the network context - nodes and links utilization from the data center network - when making decisions. Using MAP, this network could handle 250 requests at a time on the 4-port network and around 800 on the 16-port network. After that amount of requests, it starts a dramatic increase of CPU cost for the utilized servers. This is because the network contexts are not considered during the MAB embedding decision, causing non-optimal results. *OctoMap* manages to map more chain requests in small scale data center situation optimally in a more distributed and cost-effective way. However, *OctoMap* results in higher costs than the rest of the algorithms for a low number of SFC requests since it does not yet have enough data to take the best decision.

Link bandwidth cost: Figures 5(b) and 6(b) depict the physical link bandwidth cost with various SFC requests on both 4-port and 16-port fat-tree topologies, respectively. It is conspicuous that both no-predictor and MAB baselines map up to 200 and 800 concurrent SFCs in 4-port and 16-port based data center networks, respectively. After that limit, those approaches exceeded the bandwidth cost above 100, leading to higher unnecessary bandwidth consumption. *OctoMap* uses about 50% less bandwidth cost compared to both baselines in 4-port fat-tree topology and around 30% less bandwidth consumption on the larger topology. This effect is due to the active learning in the contextual bandit algorithm for the selection of links during the mapping process.

Total cost: Figures 5(c) and 6(c) shows the total cost of 4-port and 16-port fat-tree topologies, respectively. It is the sum of CPU and link bandwidth cost. Clearly, *OctoMap* consumes less overall cost compared with the baselines; approximately 42% less cost on smaller data center, and about 33% less cost on the larger 16-port data center topology.

Our approach achieves less cost, even for a larger scale data center topology since it had advanced knowledge of the required chain types and amount of resources. The objective of this work is not to optimize the SFC mapping but to implement a suitable platform to assist the mapping decisions. The simulation costs results presented are compared with existing work [48], [49], confirming the validity of *OctoMap* in providing chain mapping decisions.

Throughput Harm: Figure 7 compares the amount of throughput harm done by *OctoMap* and the baseline algorithms. Some of the no predictor baseline harm values are dropping beyond 0. This reflects an under-utilization of the network resources during SFC mapping decisions. This fact is due to the SFC demands unawareness when the predictor module is excluded from the MAP decision framework. The MAB baseline keeps the harm level between 0 and 1 on the smaller simulated topology while it lies between 0.5 and 1 (harmful) on the larger simulated network topology. This is because although the MAB algorithm learns from past decisions, it does not consider the current network status while deciding the SFC mapping actions. In the case of *OctoMap*, the amount of harm

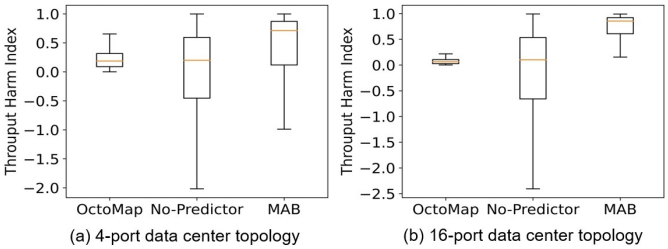


Fig. 7. Throughput harm caused by *OctoMap*, no-predictor baseline, and MAB baseline on (a) the 4-port data center topology, and (b) the 16-port data center topology. *OctoMap* caused less harm to the throughput of the small and larger data center networks.

lay between 0 (harmless) and did not exceed 0.5 for both the 4-port and 16-port data center topologies. This implies that *OctoMap* distributed the SFC requests among all the available data center links optimally by taking both the future predicted traffic and the current network load into consideration during the SFC mapping decisions.

VI. Conclusion

Service function chain mapping facilitates the complex virtual network function process by assisting in embedding virtual to physical nodes and links of all users' requested services. Dealing with the network uncertainties and with the nature of highly fluctuating traffic, it is important to provide a smooth flow of data while avoiding over or under-utilization of network resources. This can be achieved through bandwidth prediction of future traffic and active online consideration of network utilization to aid in traffic mapping using machine learning models. In this paper, we proposed *OctoMap*, a framework to provide cost-efficient chain embedding decisions. To utilize all the available network resources, *OctoMap* facilitates the embedding of SFC by selecting the best number of nodes and links based on future expected VNF and SFCs requests. It has two main components: the predictor and the contextual bandit policymaker modules. The predictor employs traces from the Facebook data center to predict future traffic loads. The contextual bandit module considers both predicted traffic along with other network constraints, including CPU capacity and link bandwidth, to make optimal embedding decisions for the requested SFC services. Those decisions support the parallel transmission of packets in data centers and improve throughput and network utilization. The simulation results indicated the effectiveness of the proposed learning approach. Our evaluation showed that *OctoMap* can serve the maximum amount of SFC requests and yet managed to save around 50% and 30% of total incurred costs of small 4-port and larger 16-port data center topologies, respectively.

VII. Acknowledgment

This work has been supported by NSF under Award Numbers CNS1647084, CNS1836906, and CNS1908574, and by an international travel grant from the GENI Project Office and Boston University, under NSF collaborative agreement CNS-1536090. The work of Aziza Alzadjali was conducted while at Saint Louis University. Dr. Fiandrino's work is supported by the Juan de la Cierva grant (IJC2019-039885-I).

Authorized licensed use limited to: SAINT LOUIS UNIVERSITY. Downloaded on October 05,2021 at 03:36:09 UTC from IEEE Xplore. Restrictions apply.

REFERENCES

[1] J. Halpern, C. Pignataro *et al.*, "Service function chaining (SFC) architecture," in *RFC 7665*, 2015.

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

[3] J. Fan, C. Guan, Y. Zhao, and C. Qiao, "Availability-aware mapping of service function chains," in *Proc. of IEEE INFOCOM*, 2017, pp. 1–9.

[4] A. AbdelSalam, F. Clad, C. Filsfils, S. Salsano, G. Siracusano, and L. Veltri, "Implementation of virtual network function chaining through segment routing in a linux-based nfv infrastructure," in *Proc. of IEEE NetSoft*, 2017, pp. 1–5.

[5] S. Lange, H.-G. Kim, S.-Y. Jeong, H. Choi, J.-H. Yoo, and J. W.-K. Hong, "Predicting VNF deployment decisions under dynamically changing network conditions," in *Proc. of IEEE CNSM*, 2019, pp. 1–9.

[6] F. Esposito, M. Mushtaq, M. Berno, G. Davoli, D. Borsatti, W. Cerroni, and M. Rossi, "Necklace: An architecture for distributed and robust service function chains with guarantees," *IEEE Transactions on Network and Service Management*, pp. 1–1, 2020.

[7] M. C. Luizelli, L. R. Bays, L. S. Buriol, M. P. Barcellos, and L. P. Gaspary, "Piecing together the NFV provisioning puzzle: Efficient placement and chaining of virtual network functions," in *Proc. of IFIP/IEEE IM*, 2015, pp. 98–106.

[8] R. Gau, "Optimal traffic engineering and placement of virtual machines in SDNs with service chaining," in *Proc. of NetSoft*, 2017, pp. 1–9.

[9] A. Roy, H. Zeng, J. Bagga, G. Porter, and A. C. Snoeren, "Inside the social network's (datacenter) network," in *Proc. of ACM SIGCOMM*, 2015, pp. 123–137.

[10] T. Benson, A. Anand, A. Akella, and M. Zhang, "Understanding data center traffic characteristics," *ACM SIGCOMM Computer Communication Review*, vol. 40, no. 1, pp. 92–99, 2010.

[11] T. Benson, A. Anand, A. Akella, and M. Zhang, "MicroTE: Fine grained traffic engineering for data centers," in *Proc. of ACM CoNEXT*, 2011, pp. 1–12.

[12] S. B. Schneider, N. P. Satheeschandran, M. Peuster, and H. Karl, "Machine learning for dynamic resource allocation in network function virtualization," in *Proc. of NetSoft*, 2020.

[13] R. Chen, H. Lu, Y. Lu, and J. Liu, "MSDF: A deep reinforcement learning framework for service function chain migration," in *Proc. of IEEE WCNC*, 2020, pp. 1–6.

[14] P. T. A. Quang, Y. Hadjadj-Aoul, and A. Outtagarts, "A deep reinforcement learning approach for vnf forwarding graph embedding," *IEEE Transactions on Network and Service Management*, vol. 16, no. 4, pp. 1318–1331, 2019.

[15] N. Jalodia, S. Henna, and A. Davy, "Deep reinforcement learning for topology-aware VNF resource prediction in nfv environments," in *Proc. of IEEE NFV-SDN*, 11 2019, pp. 1–5.

[16] E. L. Peterson, "Geometric programming," in *Advances in Geometric Programming*. Springer, 1980, pp. 31–94.

[17] R. Ware, M. K. Mukerjee, S. Seshan, and J. Sherry, "Beyond Jain's fairness index: Setting the bar for the deployment of congestion control algorithms," in *Proc. of ACM HotNets*, 2019, pp. 17–24.

[18] B. N. Chun and A. Vahdat, "Workload and failure characterization on a large-scale federated testbed," 2003.

[19] F. Esposito, I. Matta, and V. Ishakian, "Slice embedding solutions for distributed service architectures," *ACM Comput. Surv.*, vol. 46, no. 1, Jul. 2013. [Online]. Available: https://doi.org/10.1145/2522968.2522974

[20] A. Engelmann and A. Jukan, "A reliability study of parallelized VNF chaining," in *Proc. of IEEE ICC*, 2018, pp. 1–6.

[21] T.-M. Pham, S. Fdida, H. T. T. Binh *et al.*, "Online load balancing for network functions virtualization," in *Proc. of IEEE ICC*, 2017, pp. 1–6.

[22] Y. Jia, C. Wu, Z. Li, F. Le, and A. Liu, "Online scaling of NFV service chains across geo-distributed datacenters," *IEEE/ACM Transactions on Networking*, vol. 26, no. 2, pp. 699–710, 2018.

[23] G. Sun, Z. Chen, H. Yu, X. Du, and M. Guizani, "Online parallelized service function chain orchestration in data center networks," *IEEE Access*, vol. 7, pp. 100 147–100 161, 2019.

[24] A. Fischer, J. F. Botero, M. T. Beck, H. De Meer, and X. Hesselbach, "Virtual network embedding: A survey," *IEEE Communications Surveys & Tutorials*, vol. 15, no. 4, pp. 1888–1906, 2013.

[25] M. Melo, S. Sargento, U. Killat, A. Timm-Giel, and J. Carapinha, "Optimal virtual network embedding: Node-link formulation," *IEEE Transactions on Network and Service Management*, vol. 10, no. 4, pp. 356–368, 2013.

[26] A. Blenk, P. Kalmbach, J. Zerwas, M. Jarschel, S. Schmid, and W. Kellerer, "Neurovine: A neural preprocessor for your virtual network embedding algorithm," in *Proc. of IEEE INFOCOM*, 2018, pp. 405–413.

[27] X. Chen, Z. Li, Y. Zhang, R. Long, H. Yu, X. Du, and M. Guizani, "Reinforcement learning–based QoS/QoE-aware service function chaining in software-driven 5G slices," *Transactions on Emerging Telecommunications Technologies*, vol. 29, no. 11, p. e3477, 2018.

[28] X. Wang, C. Wu, F. Le, and F. C. Lau, "Online learning-assisted VNF service chain scaling with network uncertainties," in *Proc. of IEEE CLOUD*, 2017, pp. 205–213.

[29] J. Beran, Y. Feng, S. Ghosh, and R. Kulik, *Long-Memory Processes*. Springer, 2016.

[30] P. Bermolen and D. Rossi, "Support vector regression for link load prediction," *Computer Networks*, vol. 53, no. 2, pp. 191–201, 2009.

[31] J. Lee, S. Lee, J. Lee, S. D. Sathyanarayana, H. Lim, J. Lee, X. Zhu, S. Ramakrishnan, D. Grunwald, K. Lee, and S. Ha, "PERCEIVE: Deep learning-based cellular uplink prediction using real-time scheduling patterns," in *Proc. of ACM MobiSys*, 2020, p. 377–390.

[32] A. Scalingi, F. Esposito, W. Muhammad, and A. Pescapé, "Scalable provisioning of virtual network functions via supervised learning," in *Proc. of IEEE NetSoft*, 2019, pp. 423–431.

[33] C. Fiandrino, C. Zhang, P. Patras, A. Banchs, and J. Widmer, "A machine learning-based framework for optimizing the operation of future networks," *IEEE Communications Magazine*, vol. 58, no. 6, 2020.

[34] D. Chemodanov, P. Calyam, F. Esposito, and A. Sukhov, "A general constrained shortest path approach for virtual path embedding," in *Proc. of IEEE LANMAN*, June 2016, pp. 1–7.

[35] D. Chemodanov, P. Calyam, and F. Esposito, "A near optimal reliable composition approach for Geo-Distributed Latency-Sensitive service chains," in *Proc. of IEEE INFOCOM*, Apr. 2019, pp. 1792–1800.

[36] D. Chemodanov, P. Calyam, F. Esposito, R. McGarvey, K. Palaniappan, and A. Pescapé, "A near optimal reliable orchestration approach for geo-distributed latency-sensitive SFCs," *IEEE Transactions on Network Science and Engineering*, vol. 7, no. 4, pp. 2730–2745, 2020.

[37] S. Boyd, S. P. Boyd, and L. Vandenberghe, *Convex optimization*. Cambridge university press, 2004.

[38] F. Esposito, "Catena: A distributed architecture for robust service function chain instantiation with guarantees," in *Proc. of IEEE NetSoft*, 2017, pp. 1–9.

[39] S. Kiranyaz, O. Avci, O. Abdeljaber, T. Ince, M. Gabbouj, and D. J. Inman, "1D convolutional neural networks and applications: A survey," *Mechanical Systems and Signal Processing*, vol. 151, p. 107398, 2021.

[40] X. Cao, Y. Zhong, Y. Zhou, J. Wang, C. Zhu, and W. Zhang, "Interactive temporal recurrent convolution network for traffic prediction in data centers," *IEEE Access*, vol. 6, pp. 5276–5289, 2017.

[41] A. Mozo, B. Ordozgoiti, and S. Gómez-Canaval, "Forecasting short-term data center network traffic load with convolutional neural networks," *PLOS one*, vol. 13, no. 2, p. e0191939, 2018.

[42] L. Li, W. Chu, J. Langford, and R. E. Schapire, "A contextual-bandit approach to personalized news article recommendation," in *Proc. of ACM WWW*, 2010, pp. 661–670.

[43] A. Slivkins, "Introduction to multi-armed bandits," *arXiv preprint arXiv:1904.07272*, 2019.

[44] N. Cesa-Bianchi, C. Gentile, G. Lugosi, and G. Neu, "Boltzmann exploration done right," in *Advances in neural information processing systems*, 2017, pp. 6284–6293.

[45] J. A. Aroca and A. F. Anta, "Bisection (band)width of product networks with application to data centers," *IEEE Transactions on Parallel and Distributed Systems*, vol. 25, no. 3, pp. 570–580, 2014.

[46] A. Singh, J. Ong, and et. al., "Jupiter rising: A decade of clos topologies and centralized control in Google's datacenter network," in *Proc. of ACM SIGCOMM*, 2015, p. 183–197.

[47] L. John, S. Alex, , and L. Lihong, "Vowpal wabbit," Available from https://hunch.net/ vw/.

[48] X. Zhong, Y. Wang, and X. Qiu, "Cost-aware service function chaining with reliability guarantees in NFV-enabled inter-DC network," in *Proc. of IFIP/IEEE IM*, 2019, pp. 304–311.

[49] G. Sun, Z. Chen, H. Yu, X. Du, and M. Guizani, "Online parallelized service function chain orchestration in data center networks," *IEEE Access*, vol. 7, pp. 100 147–100 161, 2019.

Authorized licensed use limited to: SAINT LOUIS UNIVERSITY. Downloaded on October 05,2021 at 03:36:09 UTC from IEEE Xplore. Restrictions apply.