IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received June 24, 2019, accepted July 10, 2019, date of publication July 16, 2019, date of current version August 5, 2019.

Digital Object Identifier 10.1109/ACCESS.2019.2929260

Elastic Virtual Network Function Orchestration Policy Based on Workload Prediction

YUNJIE GU, YUXIANG HU, YUEHANG DING, JIE LU, AND JICHAO XIE
National Digital Switching System Engineering and Technological Research and Development Center, Zhengzhou 450002, China

Corresponding author: Yunjie Gu (lizardwhite@163.com)

This work was supported in part by the National Key Research and Development Plan under Grant 2017YFB0803204, in part by the National Natural Science Foundation of China under Grant 61521003 and Grant 61872382, and in part by the Research and Development Program in Key Areas of Guangdong Province under Grant 2018B010113001.

ABSTRACT By separating network functions from hardware-dependent middleboxes, network function virtualization (NFV) is expected to lead to significant cost reduction and the flexibility improvement in network management. Elastic orchestration of virtual network functions (VNF) is a key factor to achieve NFV goals. However, most existing VNF orchestration researches are limited to offline policy, ignoring the dynamic characteristics of the workload. To reduce the operational expenditure of NFV providers, this paper proposes an Elastic Virtual Network Function Orchestration (EVNFO) policy based on workload prediction. We adapt the online learning algorithm for predicting the flows rate of service function chains (SFC), which can help to obtain the VNF scaling decision. We further design the online instance provisioning strategy (OIPS) to accomplish the deployment of VNF instances according to the decision. The simulation proves that EVNFO can provide good performance with dynamic resource provision. The throughput of VNF is improved by 11.1%–22.9%, and the total operational expenditure can be reduced by 13.8% compared with other online approaches.

INDEX TERMS Service function chain, scaling, elastic orchestrating, online learning.

I. INTRODUCTION

Traditionally, enterprise network ubiquitously deploys hardware dedicated middleboxes, such as firewalls, network address translators and proxies to offer network service. Although middleboxes are able to handle heavy traffic loads, they are expensive and inflexible to scale [1]. Network Function Virtualization (NFV) aims to implement network functions as software on standard virtualized platforms. In contrast with middleboxes, the deployment of virtual network functions(VNFs) in a virtualized environment is more flexible [2]. Virtual network functions are typically sequenced into service function chains (SFCs) to provide practical network service [3], e.g. the service function chain "Firewall -IDS -Proxy" is commonly deployed in an on-premise datacenter for access control.

In terms of NFV providers, cost minimization is among the most important priorities [2]. To fulfil this goal, a significant strategy is to provide VNF in an elastic way, which means dynamically scaling the provision of VNF instances according to the demand of time-varying workload. With the precondition of sufficient computing resource for workload processing, instances with low throughput should be released to cut down the unnecessary running cost. Existing VNF orchestration strategies have favorable effect on minimization cost by optimizing resource management through the practice and exploration. However, most of the proposed strategies concentrated on the offline optimal VNF deployment problem, which are based on assumption that the flows rate of all SFC requests are always constant values. Those strategies may be difficult to put into real scenario, because the dynamic nature of an NFV system has been ignored. A few studies notice the importance of dynamic instances provision for the workload fluctuation and design the reactive schemes according to demands that already arrived. But an unneglectable delay is typically incurred in copying VM images and launching VNF instances, so that the service quality may be jeopardized during the adjustment period [4].

A proactive VNF provision approach is much more appreciated by NFV providers, because it is benefit to both the

The associate editor coordinating the review of this manuscript and approving it for publication was Ruilong Deng.

resource cost minimization and service quality assurance. If the upcoming workload can be predicted, the scaling of VNF instances can be executed in advance so that the delay of deployment has no influence on network services. This study focuses on online elastic virtual network function orchestration. For NFV providers that rent virtual resource in datacenter, we aim to minimize their operational expenditure by provision and scaling the instances of VNF in a proactive way. The challenge to enable this vision can be described as follows: (i) To acquire a proactive VNF scaling decision, how we can predict the flow rate of upcoming SFC. (ii) How to deploy VNF instances to serve the flows according to the scaling decision.

We design an algorithm called Elastic Virtual Network Function Orchestration policy (EVNFO) based on workload prediction to solve the above problems. Firstly, we aggregate the flow rates of homogeneity service function chains to remove the influence of SFC requests' start time and life-cycle [5]. To help NFV providers make proactive scaling decisions, we introduce online learning into the Evolution Prediction algorithm of Flow Rate (EPFR), which can predict the upcoming workload with the minimized error. Due to the amount of VNF instances are determined by time-varying workload of NFV system, fine-grained VNF scaling decision can be obtained in advance by calculating the increment of each kind of VNF instances based on the workload we predicted.

Next, in the online elastic VNF orchestration stage, the deployment of VNF instances and the routing of new arrival SFC requests need to be addressed. We use Online Instance Provision Strategy (OIPS) to determine where to place the newly launched instances. A virtual resources network which represents the distribution of available VNF instances will be constructed. Based on this network, the path of SFC requests with minimum traffic forwarding cost can be selected by Viterbi algorithm. For the redundant instances need to be removed, we carefully design a release mechanism to cut down the running cost and the deployment cost.

An overview of our contributions is presented as follows:

i) We introduce an online learning algorithm to predict the upcoming aggregated flow rate, which performs better than other algorithms according to the simulation;

ii) We design an online instance provision strategy to orchestrate VNF instances;

iii) We introduce a release mechanism for redundant instances to cut down the running cost and the deployment cost.

The rest of the paper is organized as follows:

Section II discusses related work. Section III describes an optimization formulation of online elastic virtual network function orchestration, towards an operational expenditure minimization target for NFV providers. Section IV introduces online learning into the Evolution Prediction algorithm of Flow Rate(EPFR) and obtains the fine-grained VNF scaling decision at each time slot. We also propose the online instance provision strategy in this section. Section V presents the trace-driven simulation to confirm that the EVNFO algorithm can help NFV providers reduce total operational expenditure in a proactive way. Section VI concludes the paper.

II. RELATED WORK

Interest on NFV starts from the white paper published by telecommunication operators that introduced virtualized network functions running on commodity hardware [6]. After IETF drafts highlighted the broad relationship between service function chain and NFV [7], a large number of researches concentrated on the optimal placement of VNF, which has been proved to be an NP-hard problem [8]. Algorithms designed for VNF placement are mainly devoted to optimizing resource allocation. VNF-P [9] presented an optimization model for VNF placement and designed a fast heuristic algorithm in a hybrid environment where dedicated hardware supplied part of network functions. Bari *et al.* [10] formulated the resource allocation problem in to integer linear program (ILP) and presented a multi-objective heuristic algorithm for VNF placement optimization in large scale network. Yuan *et al.* [11] proposed an approach to the problem of pooling deployment of VNF instance in virtual EPC network, which achieves fine-grained management and overall scheduling of node resources. Jang *et al.* [12] proposed a multi-objective optimization problem to maximize the acceptable SFC flow rate and to minimize the energy cost. Ghaznavi *et al.* [13] investigated distributed service function chain and selected appropriate VNF instances from typical VNF offerings. Cohen *et al.* [14] proposed heuristic algorithms for VNF placement across geo-distributed datacenters, to minimize the VNF setup costs, and presented rigorous analysis. Addis *et al.* [15] proposed a service function chain and VNF placement model and devised a mixed integer linear programming (MILP) formulation.

Those above literatures have favorable effect on minimization cost by optimizing resource management through the practice and exploration. However, most of them deal with the static or one-time VNF placement, ignoring the dynamic nature of SFC flow rate and its scaling demand. The scaling is a fundamental task that allows addressing performance variations in Network Functions Virtualization, which can be categorized into two classes: horizontal scaling and vertical scaling [16]. Both mechanisms can help to optimize the usage of the resource while ensure the acceptance ratio of SFC requests. Recently, there have been a few studies on dynamic VNF provision and its scaling. Arteaga *et al.* [16] proposed for NFV an adaptive scaling approach based on reinforcement learning and Gaussian Processes to carry out an improvement scaling policy. But this method might not work well with various service function chains. Wang *et al.* [17] targeted dynamic VNF instances provision for enterprise services. The algorithm takes server capacities and flow rate into consideration, but the strategy is reactive in nature and might cause SLA violation when the providers can not spin up new instances on time. There are also a few works concentrate on robust VNF placement optimization algorithms that take into account demand uncertainty. Fei *et al.* [18] employed online algorithm to predict the upcoming flow rate and derive the VNF instances with adaptive processing capacities. Zhang *et al.* [19] designed an online algorithm to purchase short/long term VMs for virtual resource scaling. The strategy achieved a good performance guarantee by both theoretical analysis and simulation under realistic settings. Bilal *et al.* [20] used time series to predict future resource usage to sustain true elasticity in NFV. Simulation showed that the obtained results can efficiently handle elasticity in virtualized networks.

III. PROBLEM MODEL

In this section we establish the mathematical model for each element in NFV system. We make a careful analysis for factors that influence the operational expenditure and formally define the optimization objective.

A. SYSTEM MODEL

1) PHYSICAL NETWORK

NFV providers purchases virtual resource from distributed cloud datacenter network, which can be modelled as an undirected graph $G = (N, E, A_n, A_e)$ Where N and E are, respectively, denote the sets of physical nodes and links. $A_n = \{C_n^r, L_n\}$ is the properties of node n. C_n^r represents the total amount of physical resources, $r \in R = \{cpu, ram, hdd\}$ is the set of resource types, including CPU, RAM and HDD. L_n represents the location information of node n. $A_e = \{B_e, \delta_e\}$ is the properties of link e. B_e and δ_e denote the bandwidth capability and forwarding cost respectively.

2) VIRTUAL NETWORK FUNCTION (VNF)

The available VNF types are given in a set I, such as Firewall, IDS. The properties of VNF can be denoted as $A_i = \{C_i^r, B_i, \varphi_i, \phi_i, \lambda_i\}$. Each type of VNF has a specific amount of required physical resource C_i^r, and possesses similar processing capacity B_i. We suppose different SFC can share one VNF instances until its processing capability is exhausted. Let φ_i represent the cost of running one i type VNF instance per time slot, and ϕ_i denote the cost of VNF instance deployment, respectively. It's worth noting that flow rate might vary after the processing of VNF: firewall may drop some packets because of security policies violation, VPN may increase the data size for encapsulation. Considering this factor, we use λ_i to represent the gain/drop factor of VNF i, which represents the change of flow rate after being processed by the type- i VNF.

3) SERVICE FUNCTION CHAIN(SFC)

The set of SFC is denoted by J. We use five element array $A^j = \{s_j, o_j, \beta_j, \alpha_j(t), \tau_j\}$ to describe the properties of SFC in detail. $s_j/o_j \in N$ can be the source/destination node and β_j represent the VNF sequence of chain j, such as *Firewall* $\rightarrow$ *IDS* $\rightarrow$ *Proxy*. The flow of j must be processed by those functions in given order. Let T be the total number of running

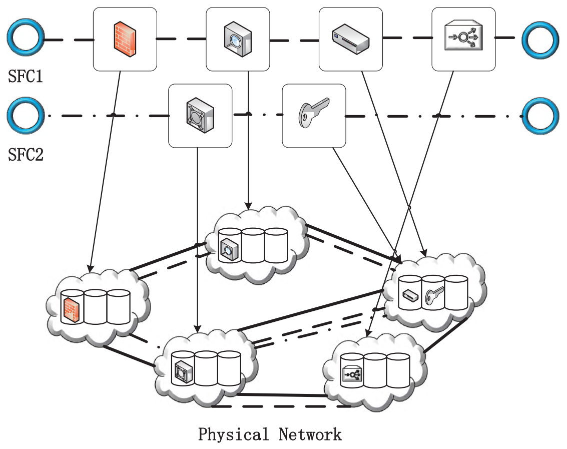


FIGURE 1. VNF orchestration in physical network.

time of NFV system. $t = \{1, 2, 3, \cdots, T\}$ is discrete time series with equal interval Δt. $\alpha_j(t)$ represents the flow rate of SFC j in t. Suppose the lifecycle of SFC j is τ_j and the start time is t_0, then $\alpha_j(t) > 0$ when $t_0 \le t \le t_0 + \tau_j$, on the contrary, $\alpha_j(t) = 0$ when $t \in [0, t_0) \cup (t_0 + \tau_j, T]$.

Fig.1 shows the process of VNF orchestration in physical network. According to the embedding relationship between VNF and physical node, we can define the variable $x_i(t)$ to represent the total amount of i type VNF instances at time t. $x_i^n(t)$ is the number of i type VNF instances deployed on node n in t. The binary variable $x_{i_k}^n(t)$ represents whether VNF instance k of i type is deployed on node n in t. $\upsilon_j^{e(u,v)}(t)$ represents whether link e between node u, v is in path of SFC j. $\upsilon_j^{i_k}(t)$ denotes whether SFC j is processed by VNF instance i_k.

$$x_{i_k}^n(t) = \begin{cases} 1, & \textit{instance } k \textit{ is deployed on } n \\ 0, & \textit{otherwise} \end{cases} \tag{1}$$

$$x_i^n(t) = \sum_{k \in x_i(t)} x_{i_k}^n(t) \tag{2}$$

$$\upsilon_j^{e(u,v)}(t) = \begin{cases} 1, & j \textit{ pass link } e(u,v) \\ 0, & \textit{otherwise} \end{cases} \tag{3}$$

$$\upsilon_j^{i_k}(t) = \begin{cases} 1, & j \textit{ is processed by VNFinstance } i_k \\ 0, & \textit{otherwise} \end{cases} \tag{4}$$

B. OBJECTIVES

Online elastic virtual network function orchestration policy focuses on minimize NFV provider's operational expenditure by provisioning and scaling the instances of VNF in a proactive way. In this work, we consider the total operational expenditure to be composed of the following four components:

1) VNF INSTANCES RUNNING COST

Without loss of generality, we suppose that the running cost of VNF instances can be decided by the amount of VNF and the corresponding operating cost φ_i. The overall VNF instances running cost is:

$$R_{total}(t) = \sum_{t\in T}\sum_{n\in N}\sum_{i\in I}\phi_i x_i^n(t) = \sum_{t\in T}\sum_{i\in I}\phi_i x_i(t) \tag{5}$$

2) VNF DEPLOYMENT COST

Launching a new VNF instance need to transferring a VM image, booting it and attaching it on devices. It is worth noting that we consider VNF migration cost within the deployment cost. Suppose there is an i type VNF instance on node n_1 at time $t-1$, this instance migrates from node n_1 to n_2 at time t. Although the total amount of instances in NFV system is not changed, the VNF deployment cost is increased because VNF migration causes another deployment operation on node n_2. The total VNF deployment cost can be expressed as:

$$D_{total}(t) = \sum_{t\in T}\sum_{n\in N}\sum_{i\in I}\varphi_i[x_i^n(t) - x_i^n(t-1)]^+ \tag{6}$$

The notation $[.]^+$ is defined as $[f(x)]^+ = \max\{f(x), 0\}$.

The deployment cost caused by adding new instances can be expressed as:

$$D_{deploy}(t) = \sum_{t\in T}\sum_{i\in I}\varphi_i[x_i(t) - x_i(t-1)]^+ \tag{7}$$

So the deployment cost caused by VNF migration can be computed as:

$$\begin{aligned} D_{migrate}(t) &= D_{total}(t) - D_{deploy}(t) \\ &= \varphi_i\sum_{t\in T}\sum_{i\in I}\{\sum_{n\in N}[x_i^n(t) - x_i^n(t-1)]^+ \\ &\quad - [x_i(t) - x_i(t-1)]^+\} \end{aligned} \tag{8}$$

3) BACKUP FACILITIES COST

To improve the availability of NFV system, providers always maintain a small amount of on-premise facilities such as backup VMs running in private server cluster. When insufficient VNF provision leads to traffic flows unserved, the backup VM instances then can be used for running VNF to process the unserved flows [19]. The backup facilities are maintained by NFV providers instead of the public cloud, which lead to the backup facilities cost:

$$U(t) = \sum_{t\in T}\sum_{j\in J}\psi\alpha_j^{loss}(t) \tag{9}$$

ψ denotes the process cost of private server cluster. $\alpha_j^{\mathrm{loss}}(t)$ represents the unserved flow rate.

4) TRAFFIC FORWARDING COST

Traffic forwarding cost occurs because of leasing cost of transit links. We assume that the flow rate of SFC j is $\alpha_j^e(t)$ when it pass along the link e and the forwarding cost is δ_e, then the total traffic forwarding cost is:

$$F(t) = \sum_{t\in T}\sum_{j\in J}\sum_{e(u,v)\in E}\delta_e\upsilon_j^{e(u,v)}\alpha_j^e(t) \tag{10}$$

Now, we can compute the total operational expenditure of the NFV system as follows:

$$\begin{aligned} C(t) &= R_{total}(t) + D_{total}(t) + U(t) + F(t) \\ &= \sum_{t\in T}\{\sum_{i\in I}\{\phi_i x_i(t) + \sum_{n\in N}\varphi_i[x_i^n(t) - x_i^n(t-1)]^+\} \\ &\quad + \sum_{j\in J}\{\psi\alpha_j^{loss}(t) + \sum_{e(u,v)\in E}\delta_e\upsilon_j^{e(u,v)}\alpha_j^e(t)\}\} \end{aligned} \tag{11}$$

NFV providers target to minimize $C(t)$, subject to:

$$\sum_{i\in I}x_i^n(t)C_i^r \le C_n^r, \quad \forall n\in N,\ r\in R,\ t\in T \tag{12}$$

$$\sum_{j\in J}\upsilon_j^{e(u,v)}\alpha_j^e(t) \le B_e, \quad \forall e\in E,\ t\in T \tag{13}$$

$$\sum_{n\in N}x_i^n(t) = x_i(t), \quad \forall i\in I,\ t\in T \tag{14}$$

$$\sum_{n\in N}x_{i_k}^n(t) = 1, \quad \forall n\in N, t\in T,\ 0\le k\le x_i(t) \tag{15}$$

$$\sum_{v\in N}\upsilon_j^{e(w,v)}\alpha_j(t) - \sum_{u\in N}\upsilon_j^{e(u,w)}\alpha_j(t) = 0, \quad w\in N\backslash\{s_j, o_j\} \tag{16}$$

Constraint (13) ensures that the physical resource capability of nodes would not be violated by the deployed VNF instances. Constraint (14) guarantees that bandwidth capabilities of links are not violated by accumulative flow rate. Constraint (15) describes the relationship between $x_i(t)$ and $x_i^n(t)$. Constraint (16) ensures that one instance can only be deployed on one physical node. Constraint (17) represents the requirement of flow conservation.

IV. ONLINE ELASTIC VIRTUAL NETWORK FUNCTION ORCHESTRATION POLICY BASED ON WORKLOAD PREDICTION

We design EVNFO, a proactive algorithm called Elastic Virtual Network Function Orchestration policy based on workload Prediction. First we introduce the Evolution Prediction algorithm of Flow Rate (EPFR), which can help us predict the upcoming flow rate and obtain the fine-grained VNF scaling decision at each time slot. By provisioning exactly amount of VNF instances according to the scaling decision, we can reduce $\{U(t), R(t)\}$. After that we use Online Instance Provisioning Strategy(OIPS) to determine where to place those instances and how to route new coming traffic in order to reduce $\{D_{\mathrm{migrate}}(t), D_{\mathrm{deploy}}(t), F(t)\}$.

A. SFC FLOW RATE EVOLUTION PREDICTION

The flow rate of a single SFC is difficult to predict. Because SFC can be launched by users whenever and wherever possible, causing its ruleless start time and uncertain lifecycle. To overcome this challenge, we aggregate the flow rates of service function chains whose source/destination nodes and VNF sequences are the same. The aggregated flow rate has statistical characteristics by removing the influence of SFC's start time and lifecycle.

Definition 1: Homogeneous SFC (HFSC). $\forall j_1, j_2 \in J$, if $\{s_{j_1}, o_{j_1}, \beta_{j_1}\} = \{s_{j_2}, o_{j_2}, \beta_{j_2}\}$, we regard j_1, j_2 as a homogeneous SFC $\bar{j}$, the aggregated flow rate of this HSFC is $\alpha_{\bar{j}}(t) = \sum_{j \in \bar{j}} \alpha_j(t)$, $\bar{J}$ is the set of HSFC.

Suppose the aggregated flow rate of HSFC $\bar{J}$ we predict at time $t+1$ is $\alpha^*_{\bar{j}}(t+1)$ and the real value is $\alpha_{\bar{j}}(t+1)$. If $\alpha^*_{\bar{j}}(t+1) < \alpha_{\bar{j}}(t+1)$, NFV providers may provision insufficient instances, causing part of SFC traffic flows unserved and adding to $U(t)$. If $\alpha^*_{\bar{j}}(t+1) > \alpha_{\bar{j}}(t+1)$, redundancy instances in physical network will lead to more $R(t)$. So accuracy is very important when we predict the aggregated flow rate.Also, timeliness is a significant characteristic of online algorithm, because providers need to scale VNF instances according to prediction results just in time. Inspired by literature [18], [19],we introduce an online learning method called FTRL- Proximal [22]. The FTRL-Proximal algorithm is being broadly applied and, recently has become prominent in networking and cloud computing applications, including the design of dynamic capacity planning, load shifting and demand response for data centers, also the ad click-through rates prediction [22], [24].

1) ONLINE CONVEX OPTIMIZATION AND EVOLUTION PREDICTION ALGORITHM OF FLOW RATE (EPFR)

After aggregating flow rates, we assume the maximum value of $\alpha_{\bar{j}}(t)$ is $\alpha^{\max}_{\bar{j}}$ based on prior experience, then the predicted value will be $\alpha^*_{\bar{j}}(t) \in [0, \alpha^{\max}_{\bar{j}}]$ for each time slot. We target to minimize the prediction error between $\alpha_{\bar{j}}(t)$ and $\alpha^*_{\bar{j}}(t)$, so the loss function can be constructed as:

$$f_t[\alpha^*_{\bar{j}}(t)] = [\alpha^*_{\bar{j}}(t) - \alpha_{\bar{j}}(t)]^2, \quad \forall \bar{j} \in J \tag{17}$$

Intuitively, we could obtain the predicted value $\alpha^*_{\bar{j}}(t+1)$ by minimizing the cumulative loss so far. The FTL algorithm [22] represent the most natural learning rules that use the value which has minimal loss on all past rounds, which can be expressed by:

$$\alpha^*_{\bar{j}}(t+1) = \arg\min_{\alpha^w_{\bar{j}}} \sum_{s=1}^{t} f_s(\alpha^w_{\bar{j}}), \quad \alpha^w_{\bar{j}} \in [0, \alpha^{\max}_{\bar{j}}] \tag{18}$$

Although FTL is easy to understand, it cannot guarantee low regret for all scenarios. Especially when the value of $\alpha_{\bar{j}}(t)$ in the adjacent time slot shifts drastically from round to round, the prediction will be unstable [23]. So FTL won't be a good choice for online prediction because of the traffic fluctuation. FTRL-Proximal is a natural modification of FTL where we minimize the loss on all past rounds plus a regularization term $R(\alpha^w_{\bar{j}})$, which can help to stabilize the solution.

$$\alpha^*_{\bar{j}}(t+1) = \arg\min_{\alpha^w_{\bar{j}}} \{\sum_{s=1}^{t} f_s(\alpha^w_{\bar{j}}) + R(\alpha^w_{\bar{j}})\} \tag{19}$$

To solve the convex optimization problem in an effective way, we use surrogate loss function to simplify $f_t[\alpha^*_{\bar{j}}(t)]$. Let $g_{\bar{j}t} = \partial f_t[\alpha^*_{\bar{j}}(t)] = 2[\alpha^*_{\bar{j}}(t) - \alpha_{\bar{j}}(t)]$ represent the sub-gradient of $f_t[\alpha^*_{\bar{j}}(t)]$, then $f_s[\alpha^w_{\bar{j}}(t)]$ in (20) can be rewritten as $g_{\bar{j}s}\alpha^w_{\bar{j}}(t+1)$. In order to limit the distance between two adjacent predicted results, we use regularization functions $R(\alpha^w_{\bar{j}}) = \frac{\sum_{s=1}^{t}\sigma_t}{2}||\alpha^w_{\bar{j}} - \alpha^*_{\bar{j}}(t)||_2^2$, where $\sigma_t = \frac{1}{\eta_t} - \frac{1}{\eta_{t-1}}$ is the update strategy of η_t. The learning rate $\eta_t = \frac{\theta}{G\sqrt{t}}$ is a decreasing sequence over time and $\theta = \alpha^{\max}_{\bar{j}}$, $||g_{\bar{j}t}|| \leq G$. So the flow rate evolution prediction strategy(EPFR) based on FTRL-Proximal can be expressed as:

$$f_{EPFR}(\alpha^w_{\bar{j}}) = \sum_{s=1}^{t} g_{\bar{j}s}\alpha^w_{\bar{j}} + \frac{1}{2}\sum_{s=1}^{t}\sigma_s||\alpha^w_{\bar{j}} - \alpha^*_{\bar{j}}(s)||_2^2,$$
$$\alpha^*_{\bar{j}}(t+1) = \arg\min_{\alpha^w_{\bar{j}}}\{f_{EPFR}(\alpha^w_{\bar{j}})\}, \quad \alpha^w_{\bar{j}} \in \alpha^*_{\bar{j}}, \forall \bar{j} \in J \tag{20}$$

The predicted value $\alpha^*_{\bar{j}}(t+1)$ can be easily obtained by calculating extreme point of convex function (20).

2) REGRET ANALYSIS FOR EPFR

We can analyze the effectiveness of online prediction algorithm EPFR by calculating the regret upper bound. Comparing with the offline optimal static solution $\alpha^{\text{static}}_{\bar{j}}$ [24], the regret function can be computed as:

$$reg_T = \sum_{t=1}^{T} f_t[\alpha^*_{\bar{j}}(t)] - \sum_{t=1}^{T} f_t[\alpha^{static}_{\bar{j}}], \quad \forall \bar{j} \in J \tag{21}$$

According to literature [25], [26], the upper bound of EPFR is:

$$reg_T \leq R(\alpha^w_{\bar{j}}) + \frac{1}{2}\sum_{t=1}^{T}\frac{1}{\sigma_{1:t}}||g_t||_2^2 = \frac{G\alpha^{\max}_{\bar{j}}(3\sqrt{T} - 1)}{2} \tag{22}$$

This upper bound is sublinear with T, which can ensure the effectiveness of online prediction [23].

3) VNF SCALING DECISION

NFV providers need to make VNF scaling decisions in advance to process the upcoming traffic according to the prediction results. As introduced in [16], VNF scaling is triggered by the increase/decrease of workload. The scaling mechanisms include the horizontal scaling (add/remove VNF instances) and vertical scaling (reconfigure the capacity/size of existing virtualized resources). We use horizontal scaling instead of vertical scaling, because vertical scaling need to change compute or memory resources on-the-fly, which requires rebooting the system causing SLA violations [18]. The overall flows rate coming towards i type VNF instances is $\sum_{\bar{j} \in J} \bar{\lambda}^{\bar{j}}_i \alpha^*_{\bar{j}}(t+1)$, where $\bar{\lambda}^{\bar{j}}_i$ is the cumulative gain/drop factor before the flows of $\bar{j}$ reach i type VNF instances. $\bar{\lambda}^{\bar{j}}_i$ can be computed as: $\bar{\lambda}^{\bar{j}}_i = \prod_{s=0}^{k-1} \lambda_s$,where k is the position of VNF i

in VNF sequence $\beta_{\bar{j}}$, $\lambda_0 = 1$. Constraint (23) ensures that the amount of VNF instances in NFV system can serve total incoming traffic rate at any time:

$$x_i(t+1)B_i \geq \sum\nolimits_{\bar{j}\in\bar{J}} \bar{\lambda}_i^{\bar{j}} \alpha_{\bar{j}}^*(t+1), \quad \forall t \in T,\ i \in I \tag{23}$$

Intuitively, the optimal number of instances can be calculated as:

$$\tilde{x}_i(t+1) = \frac{\sum\limits_{\bar{j}\in\bar{J}(t+1)} \bar{\lambda}_i^{\bar{j}} \alpha_{\bar{j}}^*(t+1)}{B_i} \tag{24}$$

But $\tilde{x}_i(t+1)$ may be not an integer. We formulate a rounding scheme which can ensure that the expectation of instances number $E[x_i(t+1)]$ equals to $\tilde{x}_i(t+1)$ in the long run:

$$F(\tilde{x}_i(t+1)) = \begin{cases} P\left[x_i(t+1) = \lceil \tilde{x}_i(t+1) \rceil\right] \\ = 1 + \lfloor \tilde{x}_i(t+1) \rfloor - \tilde{x}_i(t+1) \\ P\left[x_i(t+1) = \lfloor \tilde{x}_i(t+1) \rfloor\right] \\ = \tilde{x}_i(t+1) - \lfloor \tilde{x}_i(t+1) \rfloor \end{cases} \tag{25}$$

After rounding, $x_i(t+1)$ can be regarded as the optimal number of VNF instances for each type. The VNF scaling decision can be easily obtained by calculating the increments $\Delta x_i(t+1) = x_i(t+1) - x_i(t)$ for each VNF type. If $\Delta x_i(t+1) > 0$, the supply of instances now will fail to meet the traffic processing demand in $t+1$, providers should add to the amount of instances for i type VNF. On the contrary, if $\Delta x_i(t+1) < 0$, the amount of i type VNF instances should be reduced.

B. ONLINE INSTANCE PROVISIONING STRATEGY (OIPS)

Based on the VNF scaling decision $\Delta x_i(t+1)$, OIPS should accomplish the deployment of VNF instances and the construction of paths for upcoming flows in advance. OIPS can be divided into two stages that interact on each other. The first stage is the construction process of virtual resources network. The algorithm determines the location to deploy newly launched instances according to the residual node resource and real-time updating SFC path. In addition, the redundant instances need to be released. Finally, a virtual resources network representing the distribution of available VNF instances can be constructed in this way. In the second stage, a path construction strategy for new arrival SFC is proposed, which is based on Viterbi algorithm with the assistance of virtual resources graph.

To make a brief description, several definitions should be clarified:

Definition 2: Node CPU utilization and VNF instance throughput. Noting that CPU is always the scarce physical resource in the NFV system, we define $W_n(t)$ as the CPU utilization, which represents the CPU cores occupancy rate of node n. $\bar{W}_N(t)$ is the average value of $W_n(t)$ for all nodes in N. Also, we define $W_{i_k}(t)$ as the throughput of i type VNF instance k. $\bar{W}_{VNF}(t)$ denotes the average throughput of all VNF instances.

$$W_n(t) = \frac{\sum\limits_{i\in I} x_i^n(t) C_i^{cpu}}{C_n^{cpu}}, \quad \bar{W}_N(t) = \frac{\sum\limits_{n\in N} W_n(t)}{N} \tag{26}$$

$$W_{i_k}(t) = \frac{\sum\limits_{j\in J} \upsilon_j^{i_k}(t)\alpha_j(t)}{B_i}, \quad \bar{W}_{VNF}(t) = \frac{\sum\limits_{i\in I}\sum\limits_{k\in x_i(t)} W_{i_k}(t)}{\sum\limits_{i\in I} x_i(t)} \tag{27}$$

Definition 3: New arrival SFC. For the aggregated flow rate of HSFC, if $\Delta\alpha_{\bar{j}}^*(t+1) = \alpha_{\bar{j}}^*(t+1) - \alpha_{\bar{j}}(t) > 0$, the predicted increment $\Delta\alpha_{\bar{j}}^*(t+1)$ will be regarded as a new arrival SFC at $t+1$. Let $\Delta\bar{j}(t+1)$ denote this new arrival SFC. $\Delta\bar{J}(t+1)$ is the set of $\Delta\bar{j}(t+1)$ at $t+1$. The path of $\Delta\bar{j}(t+1)$ in $t+1$ can be denoted by $\text{Path}(\Delta\bar{j}(t+1))$.

Definition 4: Node-VNF Betweenness Centrality (NBW). The betweenness centrality of node n for i type VNF can be computed as:

$$NBW_n^i(t) = \sum_{\Delta\bar{j}\in\Delta\bar{J}(t)} \sum_{k\in x_i(t)} \upsilon_{\Delta\bar{j}}^{i_k} x_{i_k}^n, \quad \forall i \in I,\ n \in N \tag{28}$$

The value of $\text{NBW}_n^i(t)$ equals to the number of $\text{Path}(\Delta\bar{j}(t))$, whose VNF sequence has i type VNF, traversing node n. $\text{NBW}_n^i(t)$ represents the i type VNF demand intensity of node n.

Definition 5: Node-VNF Attractiveness intensity (NA). $\text{NA}_n^i(t)$ denotes the attractiveness intensity of node n for i type VNF. NFV providers tends to deploy VNF on the node which has low CPU utilization and high i type VNF demand intensity, so $\text{NA}_n^i(t)$ is determined by the CPU utilization $W_n(t)$ and betweenness centrality $\text{NBW}_n^i(t)$ of n:

$$NA_n^i(t) = \frac{\bar{W}_N(t) NBW_n^i(t)}{W_n(t)} \tag{29}$$

1) THE CONSTRUCTION OF VIRTUAL RESOURCES NETWORK

When $\Delta x_i(t+1) > 0$, providers should add to the amount of i type VNF instances. The deployment location of new VNF instances should be directed by Node-VNF Attractiveness intensity, because instances always tend to be deployed on the node with high i type VNF demand intensity but low CPU utilization. In order to achieve above requirements, OIPS calculates the value of $\text{NBW}_n^i(t)$ per time slot according to path of SFC constructed at the last time slot. Combining with the CPU utilization $W_n(t)$ of each node, we get the value of $\text{NA}_n^i(t)$ (lines11-12). The node which has higher $\text{NA}_n^i(t)$ and sufficient resource capability (lines13-16) will get a higher priority when we deploy the new VNF instances. The residual resource and the value of $\text{NBW}_n^i(t)$ should be updated after we record the deployment location using map (i_k, n) (lines17-21). In this way, newly deployed instances can respond to the time-varying workload. The value of $D_{\text{migrate}}(t)$ will be reduced.

When $\Delta x_i(t+1) < 0$, redundant instances need to be released. But if the number of instances that we remove at each time slot equals to $\Delta x_i(t+1)$, the total amount of instances will oscillate because of workload short-term fluctuations, which will increase the cost of VNF deployment by adding/removing instances too frequently. In order to avoid this problem, we select $\Delta x_i(t+1)$ instances with the lowest throughput $W_{i_k}(t)$, mark them as idle state instead of removing them at once and reroute the flows processed by them (lines8-9). Although an idle instance can not process the SFC request, its energy consumption is much lower than a running one. For idle instances, they won't be removed until the deadline of buffer time comes. If an idle instance is required during that buffer time, we could simply turn its idle state into running state without causing any deployment cost. According to Ski-Rental algorithm [27], we generate buffer time t_{buff} for each idle instance according to the following distribution:

$$P\{t_{buff}=s\}=(\frac{\eta_i-1}{\eta_i})^{\eta_i-s}\frac{1}{\eta_i(1-(1-\eta_i{}^{-1})^{\eta_i})},\quad \eta_i=\frac{\varphi_i}{\phi_i} \tag{30}$$

It can be proved that the sum of $D_{\text{total}}(t)$ and $R_{\text{total}}(t)$ obtained by this online strategy is at most $e/e-1$ times than the offline optimal solution [27].

After new VNF instances deployment and redundant instances removing, a Virtual Resources Network (VRN) which represents the available VNF instances can be constructed. $G_{\text{VRN}}(t+1) = (N_{\text{VNF}}(t+1), E_{\text{res}}(t+1))$, where $N^i_{\text{VNF}}(t+1)$ consists of $\Delta x^n_i(t+1)$ and $x^{res}_{ni}(t)$. $\Delta x^n_i(t+1)$ represents newly launched VNF instances in node n, and $x^{res}_{ni}(t)$ represents the instances that have already been deployed in node n, whose residual processing capability is not empty. $e_{\text{res}}(t+1)$ denotes the residual bandwidth capability in link e. The algorithm of VRN construction is given in Alg.1. The complexity of Alg.1. is $O(I \cdot \bar{J} \cdot k) = O(n^2 \cdot k)$, where $k = \max\{\Delta x_i(t+1)\}$.

Algorithm 1 The Construction of Virtual Resources Network

```
Input: $G = (N, E)$, $\Delta x_i(t+1)$, Path($\Delta\bar{J}(t)$)
Output: $G_{\text{VRN}}(t+1) = (N_{\text{VNF}}(t+1), E_{\text{res}}(t+1))$
(1) initialization: $NBW^i_n = 0$, map($I, N$)=null
(2) gather information of $x^n_i(t)$, $x^{res}_{ni}(t)$, $B^e_{\bar{j}}(t)$ from physical
    network
(3) remove idle instances whose $t_{\text{buff}}$ reach deadline
(4) for $n$: $N$ do
(5)     calculate $W_n(t)$, $\bar{W}_N(t)$ according to $x^n_i(t)$, $C^{\text{cpu}}_i$
(6) for $i$: $I$ do
(7)     if $\Delta x_i(t+1) < 0$
(8)         mark $\Delta x_i(t+1)$ instances with lowest $W_{i_k}(t)$
            as idle state
(9)         reroute flows processed by idle instances,
            generate $t_{\text{buff}}$
(10)    if $\Delta x_i(t+1) > 0$
(11)        calculate NBW$^i_n$ according to Path($\Delta\bar{J}(t)$)
(12)        update NA$^i_n(t)$, initialize $k = 0$, $K = 0$
(13)        while( $k < \Delta x_i(t+1)$) do
(14)            $n = \max(NA^i_n(t))$
(15)            if $[C^{\text{cpu}}_n - \sum_{i\in I} x^n_i(t) \cdot C^{\text{cpu}}_i] > C^{\text{cpu}}_i$
(16)                deploy $i_k$ or recover an idle
                    instance on $n$
(17)                recorde the deployment location
                    by map($i_k, n$)
(18)            update residual resources and $W_n(t)$
(19)            $K =$ Path($\Delta\bar{j}(t)$) though $n$
(20)            for node ($m$) of $K$ do
(21)                NBW$^i_m - 1$
(22)            $k++$
(23)            recalculate NA$^i_n(t)$
(24) calculate $\Delta x^n_i(t+1)$ according to map($I, N$)
(25) $n^i_{\text{VNF}}(t+1) = \Delta x^n_i(t+1) + x^{\text{res}}_{ni}(t)$
(26) $e_{\text{res}}(t+1) = B_e - \sum_{\bar{j}\in\bar{J}(t)} v^e_{\bar{j}} \alpha^e_{\bar{j}}(t)$
(27) $G_{\text{VRN}}(t+1) = (N_{\text{VNF}}(t+1), E_{\text{res}}(t+1))$
```

2) PATH CONSTRUCTION STRATEGY FOR NEW ARRIVAL SFC

Based on the virtual resources network, the path for each new arrival SFC can be constructed by Viterbi algorithm [10]. In order to reduce the total forwarding cost, we sort the set of new arrival SFC $\Delta\bar{J}(t+1)$ according to the value of their flow rates $\Delta\alpha^*_{\bar{j}}(t+1)$ and construct path for SFC whose flow rate is largest first. The process of path construction is illustrated in Fig 2. For a new arrival SFC $\Delta\bar{j}$: {Firewall $\rightarrow$ Encryption $\rightarrow$ IDS}, we first construct a multi-stage graph of $\Delta\bar{j}$ with the help of virtual resources graph. The first/last layer of multi-stage graph represents the position of source/destination node of $\Delta\bar{j}$, denoted by $X_{0,1}$, $X_{4,5}$. And the middle layers denote the distribution of available instances in virtual resources graph, such as firewall is distributed in node 2,3,4, so first middle layer contains $X_{1,2}, X_{1,3}, X_{1,4}$. Note that the subscripts of stage X represent the count of layer and the position of related available VNF instances, respectively. The weight of link represents the forwarding cost ratio.

Now, we traverse multi-stage graph starting at stage $X_{0,1}$. According to Viterbi algorithm, if we come to $X_{2,4}$ (Encryption instance on node 4), three paths between $X_{0,1}$ and $X_{2,4}$ are available. And the path $X_{0,1} \rightarrow X_{1,3} \rightarrow X_{2,4}$ has the lowest forwarding cost. So we save a pointer $\pi_{4,3}$ to mark this selection. $\pi_{2,2}$ can also be obtained in this way for $X_{2,2}$. The iterative process will be continued until we reach the destination node $X_{4,5}$, then we can follow the pointer to construct $Path_{\Delta\bar{j}} = \{1 \rightarrow 3 \rightarrow 4 \rightarrow 4 \rightarrow 5\}$, which means the path of $\Delta\bar{j}$ will pass firewall instance in node 3, Encryption and IDS in node 4. The complexity of path construction strategy is $O(N^2 \cdot |\beta(i)|)$.

V. PERFORMANCE EVALUATION

A. SIMULATION SETUP

We simulate an NFV system providing service for thousands of minutes. Each time slot is 5 minutes long. For service

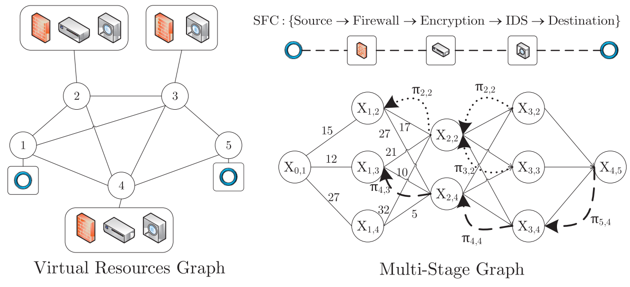


FIGURE 2. Multi-stage graph of SFC and path construction.

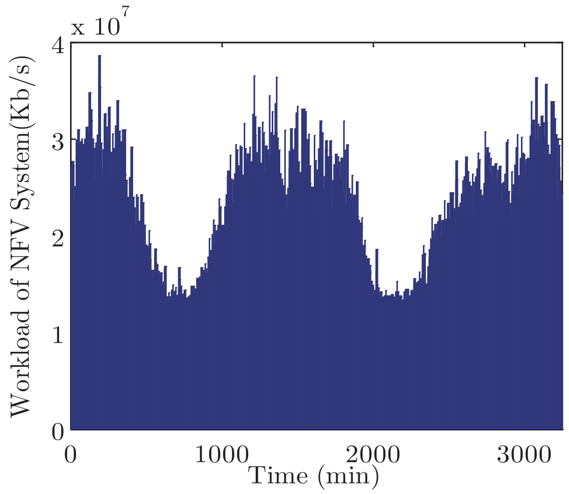


FIGURE 3. The workload of NFV system.

function chains, we use YouTube traces from the UMass campus network [28] to simulate their flow rates. Although there is no public dataset to represent SFC traffic, we can use streaming video traffic dataset to do simulation because the content popularity and the diurnal characteristics can lead to the strong demand of VNF scaling, which is the most important point of our research. Each request in trace has a timestamp, a lifecycle, source/destination IP, the capture flow rate and other information. We regard each request as a service function chain containing 2-4 VNFs generated randomly form table.1. For each time slot, we aggregate the flow rates of homogeneity service function chains whose source/destination IP and VNF sequences are the same. We simulate 462 HSFCs in total in this way. From Fig.3 we can see, the workload is high and has obvious fluctuations in the period of 0-400min, 1100-1800min, 2700min-3300min. The peak-to-mean ratio(PMR) is 1.64. For the topology of physical network, we use Geant Network [10] which has 22 nodes and 36 links. 12 of those nodes are equipped with 96-core CPU, 120GB RAM and 2000GB HDD. Another 10 nodes only act as forwarding nodes. The bandwidth of each link is 3Gbps.

The parameters of NFV system are listed as follows: The cost of operating one i type VNF instance per time slot is proportionate to the number of CPU cores that it requires. In this paper, we set $\phi_i = 0.6C_i^{\text{CPU}}$, $\phi_{idle} = 0$. According to literature [29], [30], VNF deployment cost is decided by its operating cost, so we set $\varphi_i = 6\phi_i$. The process cost of server cluster (backup facilities) ψ is three times larger than average VNF operating cost, which means $\psi = 3\sum\phi_i/\sum B_i$. The weight of link ω_e is generated randomly between 36 and 200, and the forwarding cost $\delta_e = 3.6265\omega_e \times 10^{-11}/Kbps$.

TABLE 1. VNF main parameter.

VNF Type	CPU (core)	RAM (MB)	Processing Capacity	Gain/Drop Factor λ_i
Firewall	4	4000	600Mbps	0.9
Encryption	4	4000	480Mbps	1.2
IDS	8	8000	600Mbps	0.8
NAT	2	2000	400Mbps	1

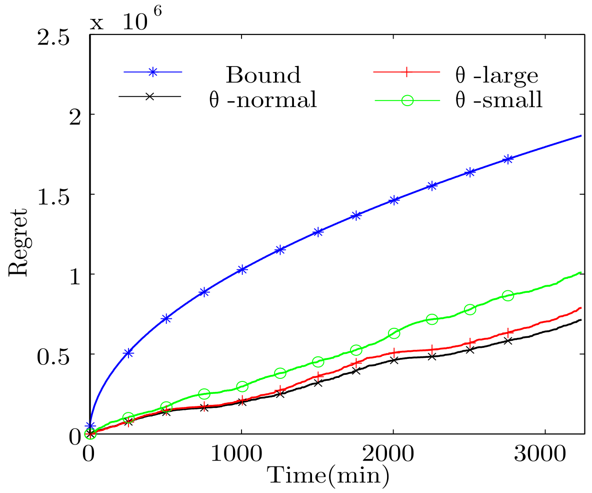


FIGURE 4. Regret of EPFR with different θ.

B. EFFECTIVENESS OF FLOW RATE EVOLUTION PREDICTION

We evaluate the effectiveness of flow rate evolution prediction by analyzing the regret function. First we compare the regret by enlarge/narrow the value of parameter θ in learning rate η_t to check whether inaccurate estimation of θ will cause serious influence in prediction. For comparison, we use $\theta_{\text{normal}} = \alpha_j^{\max}$, $\theta_{\text{small}} = 0.25\alpha_j^{\max}$ and $\theta_{\text{large}} = 4\alpha_j^{\max}$ in computing η_t. Fig.4 illustrates when $\theta = \alpha_j^{\max}$, the effectiveness of flow rate evolution prediction gets the lowest regret value. In addition, although enlarge/narrow the value of θ will add to the value of regret, all regrets are always much smaller than the theoretical upper bound obtained in section 4.1.2. This explains EPFR is robust to the inaccurate estimation of θ.

Fig.5 shows the comparison of $\lim\limits_{T\to\infty}\frac{\text{reg}_T}{T}$ for different online learning algorithm as well as the limit of the theoretical upper bound for EPFR. We can see that EPFR satisfies the request of sublinear increasing, and it is more effective than other online algorithms. FTL algorithm is hard to converge because of the loss of regular term, which means it can not be used to predict the flow rate. The effectiveness of OGD algorithm [31] is inferior to EPFR because it uses the constant

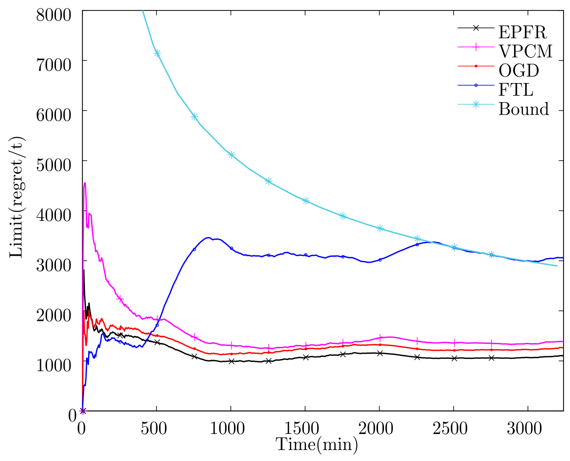


FIGURE 5. Comparison of different online algorithms.

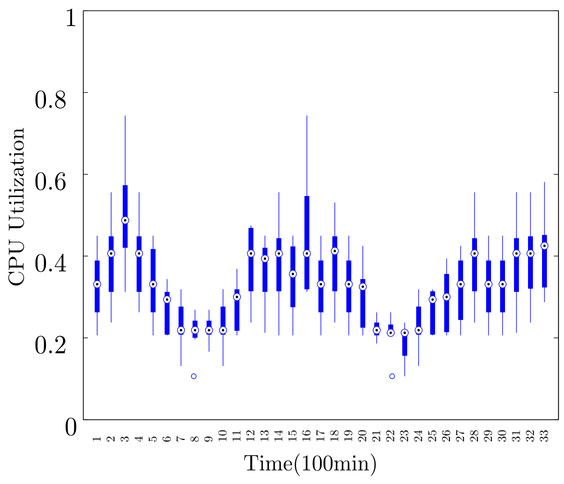


FIGURE 6. The distribution of nodes' CPU utilization.

learning rate. The loss function of VPCM algorithm [18] is $f_t[\alpha_j^*(t)] = \left|\alpha_j^*(t) - \alpha_{\bar{j}}(t)\right|$ so that the sub-gradient of $f_t[\alpha_j^*(t)]$ is identically equal to ± 1. When using surrogate loss function to simplify $f_t[\alpha_j^*(t)]$, VPCM can only use ± 1 to adjust the loss no matter how large the distance between the predicted value $\alpha_j^*(t)$ and real value $\alpha_{\bar{j}}(t)$ is. So VPCM's performance is worse than EPFR.

C. PERFORMANCE OF ONLINE ELASTIC VNF ORCHESTRATION

We first evaluate the performance of elastic orchestration by observing the distribution of CPU utilization $W_n(t)$ for nodes with resource. Fig.6 shows the distribution of $W_n(t)$ each 100 minutes. We can observe that the distribution of $W_n(t)$ has the feature of diurnal variation, and this variation is consistent with change of workload shown in Fig.3. This phenomenon proves that the amount of VNF instances can scale with the requirement of workload in time. Moreover, few abnormal points of the boxplot illustrate that OIPS tries to balance the traffic load by giving preference to nodes with low CPU occupancy rate when deploying newly launched instance.

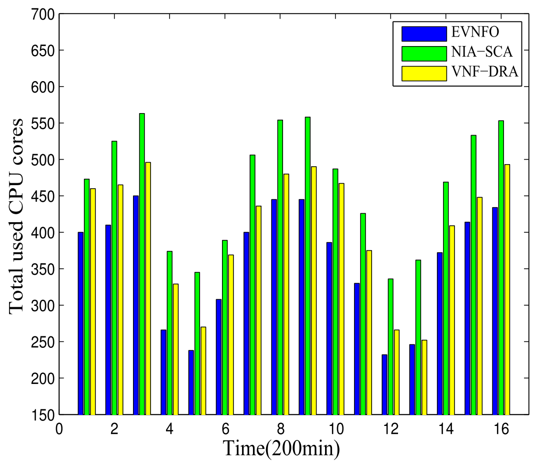


FIGURE 7. The total amount of CPU cores required.

The reason why abnormal points appear is that nearly no traffic request is launched by those corresponding nodes at that time.

We compare elastic orchestration policy (EVNFO) with other VNF deployment approaches such as NIA-SCA [18] and VNF-DRA [32]. Both algorithms have the same goal with EVNFO (reduces the OPEX). NIA-SCA includes two independent subprograms. The algorithm NIA is based on a variant of bin packing problem to minimize the volumes of the nodes used. Then the algorithm SCA routes the arriving flows for the purpose of balancing the use of bandwidth. VNF-DRA deploys VNF instances by multi-layer graph and releasing the redundant VNF instance with low throughput for the purpose of cost reducing.

Fig 7 shows the total amount of CPU cores required when processing the same workload using different online algorithms. According to Table 1, the amount of CPU cores can indirectly represent the total number of VNF instances. EVNFO can adjust the number of CPU cores with workload sensitively, because EVNFO obtains the fine-grained VNF scaling decision in each time slot by predicting the upcoming workload using EPFR. NIA-SCA can also scale the amount of VNF instances with the time-varying workload, but its performance is not as good as EVNFO. Because the redundant instances can only be removed when they are completely not used, however there is no mechanism to avoid new traffic being routed into them. For VNF-DRA algorithm, VNF instance will be defined as a redundant one at once when its utilization was lower than the threshold. Those redundant VNF instances cannot accept any new SFC request, and they won't be released until all the SFC requests processed by them have expired. In the meantime, new instances must be deployed to process the new requests. So the total amount of CPU cores is large than EVNFO especially when the workload of NFV system is decreasing. Fig.8 shows the average throughput for VNF instances. We can observe that EVNFO keeps the VNF average throughput at a high level. VNF-DRA cannot perform as well as EVNFO. For NIA-SCA algorithm, the value of may drop clearly when the workload

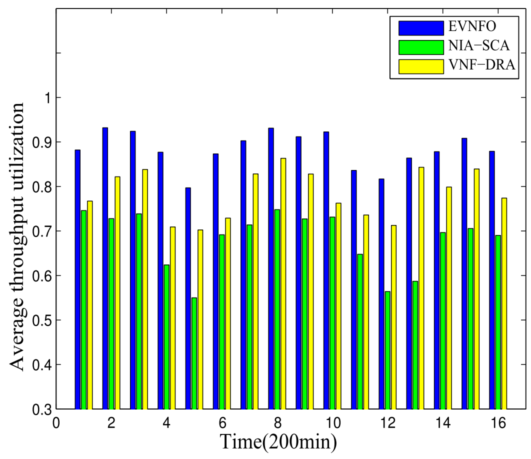


FIGURE 8. Average throughput for VNF instances.

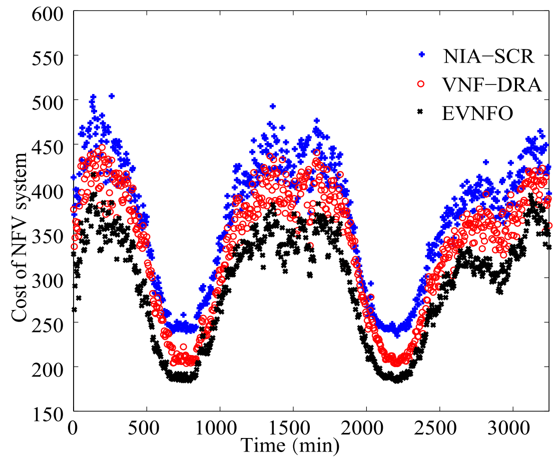


FIGURE 9. Cost of the NFV system each time slot.

is decreasing. The average throughput of VNF can be improved by 11.1% (compared with VNF-DRA) or 22.9% (compared with NIA-SCA) when we use EVNFO.

D. OVERALL COST UNDER DIFFERENT ALGORITHM

We compare the operational expenditure of NFV system under different online algorithm. Fig.9 shows the cost of the NFV system each time slot. We can see that EVNFO has the best performance in cost reduction. For NIA-SCA algorithm, the cost is large because the subprogram NIA assigns newly launched VNF instances based on best-fit first decreasing loading heuristic, which aims to minimize the volumes of the nodes used. Though this improves the utilization, SFC requests may need to extend their path to reach the shared nodes, which leads to high traffic forwarding cost. VNF-DRA algorithm will face to high operational expenditure when the workload is decreasing because the redundant instances also lead to running cost when processing the residual SFC requests. Fig.10 shows that total operational expenditure of the NFV system can be reduced by 13.8% at least when we use EVNFO.

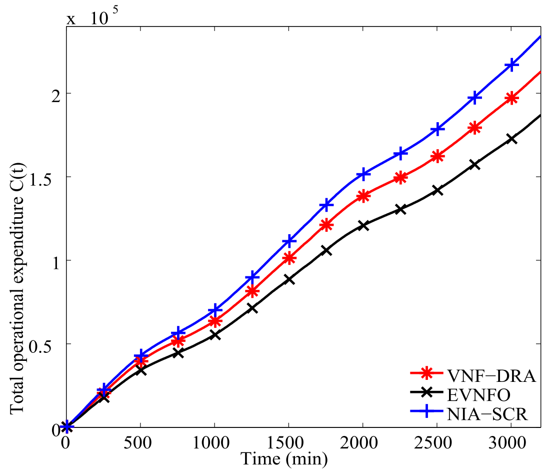


FIGURE 10. Total operational expenditure $C(t)$.

VI. CONCLUSION

With the introduction of NFV, the flexibility and efficiency of network management are improved obviously. For the NFV service providers who rent virtual recourse in datacenter, we aim to minimize their cost by provisioning and scaling the instances of VNF in a proactive way. We first use flow rate evolution prediction algorithm to obtain the VNF scaling decision for the next time slot. Then the deployment of newly launched instances and the routing of upcoming traffic can be determined by OIPS algorithm. The simulation shows that by effective flow rate evolution prediction, our proposed algorithm enhances the matching characteristics between virtual resource supply and workload change. The total operational expenditure of providers can be reduced obviously compared with other solutions.

REFERENCES

[1] A. Gember-Jacobson, R. Viswanathan, C. Prakash, R. Grandl, J. Khalid, S. Das, and A. Akella, "OpenNF: Enabling innovation in network function control," in *Proc. ACM SIGCOMM Comput. Commun. Rev.*, Aug. 2014, pp. 163–174.
[2] S. Mehraghdam, M. Keller, and H. Karl, "Specifying and placing chains of virtual network functions," in *Proc. IEEE 3rd Int. Conf. Cloud Netw. (CloudNet)*, Oct. 2014, pp. 7–13.
[3] E. P. Quinn and E. T. Nadeau. (2015). *Problem Statement for Service Function Chaining, document IETF RFC-Informational*. [Online]. Available: https://datatracker.ietf.org/doc/rfc7498/
[4] M. Ghaznavi, A. Khan, N. Shahriar, K. Alsubhi, R. Ahmed, and R. Boutaba, "Elastic virtual network function placement," in *Proc. IEEE 4th Int. Conf. Cloud Netw. (CloudNet)*, Oct. 2015, pp. 255–260.
[5] W. Ma, O. Sandoval, J. Beltran, D. Pan, and N. Pissinou, "Traffic aware placement of interdependent NFV middleboxes," in *Proc. IEEE Conf. Comput. Commun.*, May 2017, pp. 1–9.
[6] "Network functions virtualization," Netw. Funct. Ind. Specification Group, White Paper, Oct. 2013.
[7] M. T. Surendra *et al.*, *Service Function Chaining Use Cases in Data Centers*, document, Working Draft, IETF Secretariat, Internet-Draft draft-ietf-sfcdc-use-cases-06, Aug. 2017. [Online]. Available: https://datatracker.ietf.org/doc/draft-ietf-sfc-dc-use-cases
[8] R. Mijumbi, S. Hasija, S. Davy, A. Davy, B. Jennings, and R. Boutaba, "A connectionist approach to dynamic resource management for virtualised network functions," in *Proc. 12th Int. Conf. Netw. Service Manage. (CNSM)*, Oct./Nov. 2016, pp. 1–9.
[9] H. Moens and F. De Turck, "VNF-P: A model for efficient placement of virtualized network functions," in *Proc. 10th Int. Conf. Netw. Service Manage. (CNSM) and Workshop (CNSM)*, Nov. 2014, pp. 1–6.

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

[11] Q. Yuan, H. Tang, Y. Zhao, X. Wang, "An approach for virtual network function deployment based on pooling in vEPC," *IEICE Trans. Commun.*, vol. E101B, no. 6, pp. 1398–1410, 2018.

[12] I. Jang, D. Suh, S. Pack, and G. Dán, "Joint optimization of service function placement and flow distribution for service function chaining," *IEEE J. Sel. Areas Commun.*, vol. 35, no. 11, pp. 2532–2541, Nov. 2017.

[13] M. Ghaznavi, N. Shahriar, S. Kamali, R. Ahmed, and R. Boutaba, "Distributed service function chaining," *IEEE J. Sel. Areas Commun.*, vol. 35, no. 11, pp. 2479–2489, Nov. 2017.

[14] R. Cohen, L. Lewin-Eytan, J. S. Naor, and D. Raz, "Near optimal placement of virtual network functions," in *Proc. IEEE Conf. Comput. Commun. (INFOCOM)*, Apr./May 2015, pp. 1346–1354.

[15] B. Addis, D. Belabed, M. Bouet, and S. Secci,, "Virtual network functions placement and routing optimization," in *Proc. IEEE 4th Int. Conf. Cloud Netw. (CloudNet)*, Oct. 2015, pp. 171–177.

[16] C. H. T. Arteaga, F. Rissoi, and O. M. C. Rendon, "An adaptive scaling mechanism for managing performance variations in network functions virtualization: A case study in an NFV-based EPC," in *Proc. 13th Int. Conf. Netw. Service Manage. (CNSM)*, Tokyo, Japan, Nov. 2017, pp. 1–7.

[17] X. Wang, C. Wu, F. Le, A. Liu, Z. Li, and F. Lau, "Online VNF scaling in datacenters," in *Proc. IEEE 9th Int. Conf. Cloud Comput. (CLOUD)*, Jun./Jul. 2016, pp. 140–147.

[18] X. Fei, F. Liu, H. Xu, and H. Jin, "Adaptive VNF scaling and flow routing with proactive demand prediction," in *Proc. IEEE Conf. Comput. Commun.*, Apr. 2018, pp. 486–494. doi: 10.1109/INFOCOM.2018.8486320.

[19] X. Zhang, C. Wu, Z. Li, and F. C. M. Lau, "Proactive VNF provisioning with multi-timescale cloud resources: Fusing online learning and online optimization," in *Proc. IEEE Conf. Comput. Commun.*, May 2017, pp. 1–9. doi: 10.1109/INFOCOM.2017.8057118.

[20] A. Bilal, T. Tarik, A. Vajda, and B. Miloud, "Dynamic cloud resource scheduling in virtualized 5g mobile systems," in *Proc. IEEE Global Commun. Conf. (GLOBECOM)*, Dec. 2016, pp. 1–6.

[21] X. Fei. F. Liu, H. Xu, and H. Jin, "Towards load-balanced VNF assignment in geo-distributed NFV infrastructure," in *Proc. IEEE/ACM 25th Int. Symp. Quality Service (IWQoS)*, Jun. 2017, pp. 1–10.

[22] H. B. McMahan, G. Holt, D. Sculley, M. Young, D. Ebner, J. Grady, L. Nie, T. Phillips, E. Davydov, D. Golovin, and S. Chikkerur, "Ad click prediction: A view from the trenches," in *Proc. 19th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, Chicago, IL, USA, Aug. 2013, pp. 1222–1230.

[23] S. Shalev-Shwartz, "Online learning and online convex optimization," *Found. Trends Mach. Learn.*, vol. 4, no. 2, pp. 107–194, Mar. 2012.

[24] N. Chen, A. Agarwal, A. Wierman, S. Barman, and L. L. Andrew, "Online convex optimization using predictions," *ACM SIGMETRICS Perform. Eval. Rev.*, vol. 43, no. 1, pp. 191–204, Jun. 2015.

[25] H. B. McMahan, "A survey of algorithms and analysis for adaptive online learning," *J. Mach. Learn. Res.*, vol. 18, no. 1, pp. 3117–3166, Jan. 2017.

[26] N. Chen, J. Comden, Z. Liu, A. Gandhi, and A. Wierman, "Using predictions in online optimization: Looking forward with an eye on the past," *ACM SIGMETRICS Perform. Eval. Rev.*, vol. 44, no. 1, pp. 193–206, Jun. 2016.

[27] A. R. Karlin, M. S. Manasse, L. A. McGeoch, and S. Owicki, "Competitive randomized algorithms for nonuniform problems," *Algorithmica*, vol. 11, no. 6, pp. 542–571, Jun. 1994.

[28] M. Zink, K. Suh, Y. Gu, and J. Kurose, "Characteristics of YouTube network traffic at a campus network—Measurements, models, and implications," *Comput. Netw.*, vol. 53, no. 4, pp. 501–514, Mar. 2009.

[29] M. Lin, A. Wierman, L. L. Andrew, and E. Thereska, "Dynamic right-sizing for power-proportional data centers," *IEEE/ACM Trans. Netw.*, vol. 21, no. 5, pp. 1378–1391, Oct. 2013.

[30] M. Mechtri, C. Ghribi, and D. Zeghlache, "VNF placement and chaining in distributed cloud," in *Proc. IEEE 9th Int. Conf. Cloud Comput. (CLOUD)*, Jun./Jul. 2016, pp. 376–383. doi: 10.1109/CLOUD.2016.0057.

[31] M. Zinkevich, "Online convex programming and generalized infinitesimal gradient ascent,"Ì in *Proc. Int. Conf. Mach. Learn. (ICML)*, 2003, pp. 928–936.

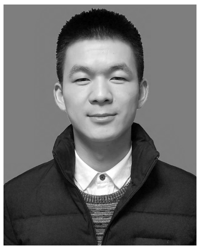
[32] J. Pei, P. Hong, K. Xue, and D. Li, "Efficiently embedding service function chains with dynamic virtual network function placement in geo-distributed cloud system," *IEEE Trans. Parallel Distrib. Syst.*, to be published.

YUNJIE GU received the B.S. degree from the Harbin University of Science and Technology, in 2017. He is currently pursuing the M.S. degree with the National Digital Switching System Engineering and Technological Research and Development Center. His research interests include network function virtualization, software defined networks, and future networks.

YUXIANG HU is currently an Assistant Professor with the National Digital Switching System Engineering and Technological Research and Development Center (NDSC), China. His research interests mainly include network security, routing protocols, and future networks.

YUEHANG DING received the B.S. degree from the Harbin University of Science and Technology, in 2017. She is currently pursuing the M.S. degree with the National Digital Switching System Engineering and Technological Research and Development Center.

JIE LU received the bachelor's degree from Zhejiang University, in 2016. He is currently pursuing the master's degree with the National Digital Switching System Engineering and Technological Research and Development Center, Zhengzhou, China. His research interests include software defined networking, network management, and network security.



JICHAO XIE received the B.E. degree from the Nanjing University of Science and Technology. He is currently pursuing the M.S. degree in communication and information system with the National Digital Switching System Engineering and Technological Research and Development Center, Zhengzhou, China. His research interests include cyber security, software defined networking, and network function virtualization.