Forecast-Assisted NFV Service Chain Deployment based on Affiliation-Aware vNF Placement

Quanying Sun, Ping Lu, Wei Lu, Zuqing Zhu†
School of Information Science and Technology, University of Science and Technology of China, Hefei, China
†Email: {zqzhu}@ieee.org

***Abstract*—This paper studies the problem of service chain (SC) deployment. Specifically, we try to place virtual network functions (vNFs) on network nodes and connect the vNFs in sequence through link mapping. We start with the offline problem. An integer linear programming (ILP) model is formulated to minimize the total SC deployment cost. With the ILP, we prove that the offline problem is $\mathcal{NP}$-hard and propose a time-efficient heuristic based on affiliation-aware vNF placement. Then, we move to the online problem, and design a forecast-assisted online SC deployment algorithm that includes the prediction of future vNF requirements. Simulation results show that the online algorithms can reduce the blocking probability of SC requests and increase the service provider's profit from SC deployment effectively.**

***Index Terms*—Network Function Virtualization, Service Chain Deployment, vNF Placement, Link Mapping, Forecast.**

I. Introduction

With the fast evolution of the Internet, service providers are facing the challenge to deploy new network services quickly and cost-effectively. This motivates the research and development on network function virtualization (NFV) [1], which leverages generic servers, switches and storage to realize the virtual network functions (vNFs) for replacing special-purpose network elements [2, 3]. Hence, network elements such as gateway, firewall, and load balancer can take the software form and become convenient and inexpensive to be deployed, maintained, and upgraded. Meanwhile, by forwarding data traffic through a sequence of vNFs, one can realize service chaining and further enrich the functionality of NFV [4–6].

Fig. 1 shows an example of NFV service chaining. We consider three types of vNFs, and there are two service chains (SCs) in Fig. 1(a): *SC* 1 originates from *Node* 2, and its data traffic needs to be processed by *vNFs* 1 and 2 in sequence and sent to *Node* 4; *SC* 2 originates from *Node* 1, and its data traffic needs to be processed by *vNFs* 1 and 3 in sequence and sent to *Node* 6. To realize an SC deployment, we need 1) deploy the requested vNFs on nodes, and 2) connect the vNFs in sequence through link mapping, considering the resource constraints on nodes and links. Meanwhile, to save the SC deployment cost, we can consider to share vNF(s) among SCs. In Fig. 1(b), as both *SCs* 1 and 2 require *vNF* 1, they can share *vNF* 1 on *Node* 3 by using the plotted SC deployment schemes. Moreover, in the dynamic network environment where SCs need to be deployed on demand, the service provisioning scheme would be even more sophisticated.

Previously, a few studies have addressed the system architecture to realize NFV [7, 8]. However, they did not consider

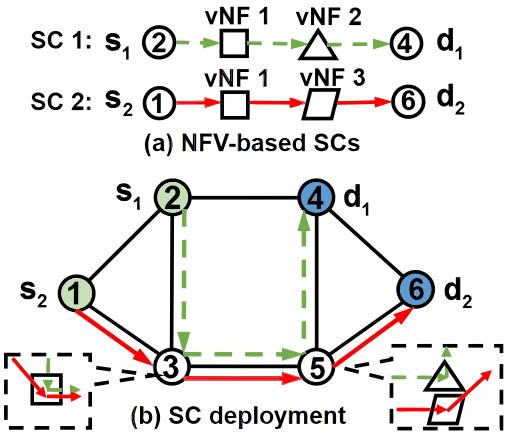


Fig. 1. SC deployment with vNF placement and link mapping.

the algorithm for optimizing the resource management for SC deployment. Kim *et al.* [9] studied how to realize SCs with pre-allocated vNFs and leveraged the genetic algorithm for optimization, but they did not take vNF placement into consideration. The authors of [10] designed an approximation algorithm based on the generalized assignment problem to tackle the problem of vNF placement. However, they did not address the problem from the perspective of SC deployment, *i.e.*, the link mapping to connect vNFs was absent.

In this paper, we study the problem of SC deployment. We start with the offline version of the problem in which all the SC requests are known in advance. An integer linear programming (ILP) model is first formulated to minimize the total resource cost related to vNF placement and link mapping. With the ILP model, we prove that the offline problem is $\mathcal{NP}$-hard and propose a time-efficient heuristic algorithm based on affiliation-aware vNF placement. Then we move to the online version of the problem where the SCs can be deployed and torn down dynamically. To reduce both the resource cost and setup delay of SCs, we design a forecast-assisted online SC deployment algorithm. With this algorithm, service providers can pre-allocate vNFs at a coarse time granularity and then fine-tune the vNF placement at each service provisioning time. Hence, frequent vNF placement adjustments can be avoided and both the operational complexity and setup delay of SCs can be reduced.

The rest of the paper is organized as follows. Section II

describes the SC deployment problem. We address the offline version of SC deployment in Section III, and the online version in Section IV. Finally, Section V summarizes the paper.

II. Problem Description

The network topology is modeled as $G = (V, E)$, where V is the node set and E is the link set. For each node $v \in V$, the amount of IT resources is C_v. For each link $e \in E$, its bandwidth capacity is B_e. The set of vNFs that are supported in the network is M, *i.e.*, $|M|$ types of vNFs can be instantiated on each node $v \in V$. For an $m \in M$ type vNF, it consumes $\hat{c}_m$ IT resources and can at most process $\hat{b}_m$ data traffic in terms of bandwidth units. Note that, although service chaining can greatly enrich the functionality of NFV, the actual form of an SC (*i.e.*, the vNF sequence in it) usually is not arbitrary. We assume that the SCs can only take N forms, and the vNF sequence in an n type SC is denoted as ψ_n. For instance, if $\psi_1 = \langle$*vNF 1, vNF 3*$\rangle$, all the $n = 1$ type SCs take the form *s→vNF 1→vNF 3→d*, where $s, d \in V$ are the source and destination nodes. Here, we use $|\psi_n|$ to represent the number of vNFs in an n type SC. An SC request is denoted as $R_j = \{s_j, d_j, b_j, n_j\}$, where s_j and d_j are the source and destination nodes, b_j is the bandwidth requirement, and n_j is the SC type.

To realize an SC deployment, we need 1) deploy the requested vNFs on nodes, and 2) connect the vNFs in sequence. Hence, the cost of an SC deployment consists of 1) the vNF placement cost, and 2) the bandwidth cost. To save the total SC deployment cost, we consider to share vNFs among SCs and avoid bandwidth consumption by deploying the adjacent vNFs in an SC on the same node since the communication between them becomes intra-node.

III. Offline NFV Service Chain Deployment

A. ILP Formulation

Notations:

- $\mathcal{I}$: the upper-bound number of any type of vNFs that can be deployed in the network, *i.e.*, $\mathcal{I} = \left\lfloor \frac{\sum_{v \in V} C_v}{\min_{m \in M} (\hat{c}_m)} \right\rfloor$.
- P: the pre-calculated path set that consists of K shortest paths between each source-destination pair in $G(V, E)$.
- α_m: the cost of IT resources used for deploying an m type vNF.
- β: the cost of per bandwidth unit on a link.
- $f_{l,n,m}$: the boolean flag that equals 1 if the l-th vNF in an n type SC is an m type vNF, and 0 otherwise.

Variables:

- $x_v^{m,i}$: the boolean variable that equals 1 if the i-th m type vNF is deployed on node v, and 0 otherwise.
- $y_{j,l}^{m,i}$: the boolean variable that equals 1 if the l-th vNF in the SC of R_j uses the i-th m type vNF, and 0 otherwise.
- $z_{j,l}^{p}$: the boolean variable that equals 1 if the l-th link in the SC of R_j uses path $p \in P$, and 0 otherwise.
- ξ_v: the total vNF placement cost.
- ξ_b: the total bandwidth cost.

Objective:

We aim to minimize the total SC deployment cost as:

$$\text{Minimize} \quad (\xi_v + \xi_b), \tag{1}$$

where ξ_v and ξ_b are calculated as:

$$\begin{aligned} \xi_v &= \sum_m \alpha_m \cdot \left(\sum_{i=1}^{\mathcal{I}} \sum_v x_v^{m,i} \right), \\ \xi_b &= \sum_{j,p} \beta \cdot |p| \cdot \left(b_j \cdot \sum_l z_{j,l}^{p} \right). \end{aligned} \tag{2}$$

Constraints:

$$\sum_v x_v^{m,i} \leq 1, \quad \forall m,\ i. \tag{3}$$

Eq. (3) ensures that each m type vNF is deployed only once.

$$\sum_m \left(\hat{c}_m \cdot \sum_{i=1}^{\mathcal{I}} x_v^{m,i} \right) \leq C_v, \quad \forall v \in V. \tag{4}$$

Eq. (4) ensures that each node's IT resources would not be overused during vNF placement.

$$\sum_{i=1}^{\mathcal{I}} \sum_{l=1}^{|\psi_{n_j}|} y_{j,l}^{m,i} = \sum_{l=1}^{|\psi_{n_j}|} f_{l,n_j,m}, \quad \forall j,\ m. \tag{5}$$

Eq. (5) ensures that we provision/reuse each requested vNF one and only one time when serving R_j.

$$\sum_j \left(b_j \cdot \sum_{l=1}^{|\psi_{n_j}|} y_{j,l}^{m,i} \right) \leq \sum_v x_v^{m,i} \cdot \hat{b}_m, \quad \forall m,\ i. \tag{6}$$

Eq. (6) ensures that each vNF's traffic processing capacity would not be overused.

$$\sum_{j,l} b_j \cdot \sum_{\{p: e \in p\}} z_{j,l}^{p} \leq B_e, \quad \forall e \in E. \tag{7}$$

Eq. (7) ensures that each link's bandwidth capacity would not be overused.

$$\sum_{\{p: p_{s_j,v} \in P\}} z_{j,1}^{p} \geq \left(y_{j,1}^{m,i} + x_v^{m,i} \right) \cdot f_{1,n_j,m} - 1, \quad \forall j,\ m,\ v,\ i, \tag{8}$$

$$\begin{aligned} \sum_{\{p: p_{u,v} \in P\}} z_{j,l}^{p} \geq{}& \left(y_{j,(l-1)}^{m_1,i_1} + x_u^{m_1,i_1} \right) \cdot f_{(l-1),n_j,m_1} \\ &+ \left(y_{j,l}^{m_2,i_2} + x_v^{m_2,i_2} \right) \cdot f_{l,n_j,m_2} - 3, \\ \{l : 1 < l \leq |\psi_{n_j}|\},\ \forall j,\ \forall u, v \in V,\ \forall m_1,\ m_2 \in M,\ \forall i_1, i_2, \end{aligned} \tag{9}$$

$$\begin{aligned} \sum_{\{p: p_{v,d_j} \in P\}} z_{j,|\psi_{n_j}|+1}^{p} \geq{}& \left(y_{j,|\psi_{n_j}|}^{m,i} + x_v^{m,i} \right) \cdot f_{|\psi_{n_j}|,n_j,m} - 1, \\ &\forall j,\ m,\ v,\ i. \end{aligned} \tag{10}$$

Eqs. (8)-(10) ensure that, for each R_j, its requested vNFs are connected in the right way. Here, $p_{u,v} \in P$ denotes a pre-calculated path from u to v.

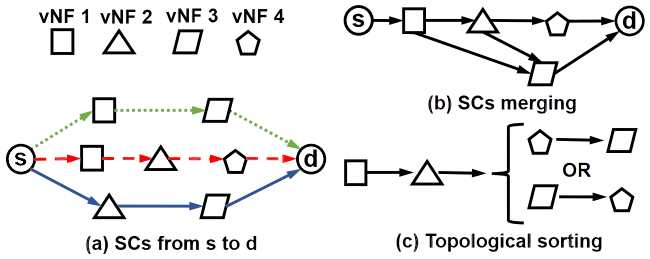


Fig. 2. Example on affiliation-aware vNF placement.

B. Complexity Analysis

Theorem. *The offline SC deployment problem is $\mathcal{NP}$-hard.*

Proof: We prove the $\mathcal{NP}$-hardness of the offline SC deployment problem by restriction, *i.e.*, restricting away certain aspects of the original problem until a known $\mathcal{NP}$-hard appears [11]. We first apply the restriction that $B_e = +\infty$ and $\beta = 0$, which means that the total bandwidth cost ξ_b is irrelevant in the optimization. Then, the problem becomes to minimize the vNF placement cost ξ_v. Next, we apply the restriction to guarantee that the available IT resources on all nodes in V are equal. If we consider the nodes as bins with a fixed capacity and the requested vNFs as items with different sizes, the offline SC deployment problem is transformed into the general case of the bin packing problem, which is known to be $\mathcal{NP}$-hard [11]. Hence, we prove that it is $\mathcal{NP}$-hard. ■

C. SC Deployment with Affiliation-Aware vNF Placement

To solve the offline problem, we propose a time-efficient heuristic as shown in *Algorithm* 1, the main idea of which is to place vNFs considering their affiliation in requested SCs. *Lines* 1-4 are for the initialization. Basically, we merge the SCs based on their source-destination pairs, and generate a vNF graph $\Psi_{s,d}(B_{s,d})$ to cover all the SCs whose source-destination pair is s-d, where $B_{s,d}$ is the total bandwidth demand. Figs. 2(a) and (b) show an example on merging SCs to generate a vNF graph. As shown in Fig. 2(a), there are three SCs from s to d. To obtain their vNF graph, we merge the same type vNFs together in Fig. 2(b). Then, we sort $\{\Psi_{s,d}(B_{s,d}), \forall s, d\}$ in the descending order of $B_{s,d}$ in *Line* 5. The for-loop that covers *Lines* 6-34 tries to deploy the vNF graphs one by one. *Line* 7 performs topological sorting [12] on vNF graph $\Psi_{s,d}(B_{s,d})$ to order the vNFs as a sequence ψ, *i.e.*, obtaining the vNFs' affiliation information. As shown in Fig. 2(c), topological sorting orders vNFs such that for each direct link *vNF* A$\rightarrow$*vNF* B in the vNF graph generated in Fig. 2(b), *vNF* A comes before *vNF* B. The for-loop covering *Lines* 8-28 tries to use the K pre-calculated shortest paths in P to connect the vNF sequence ψ. The rationale behind this is to minimize the bandwidth cost. For each path candidate that has enough bandwidth resources to satisfy $B_{s,d}$, as shown in *Lines* 9-24, we try to greedily place/reuse vNFs in ψ in sequence on each node along the path, and by doing so, we can encourage the sharing of vNFs and save the vNF placement cost. Then, if the vNF sequence ψ can be successfully deployed along path $p_{s,d}$, *Line* 14 commits the vNF placement result and *Line* 15 marks the successful flag as $flag = 1$. Otherwise, we proceed to try the next shortest path. If we cannot deploy the vNF sequence ψ in whole on any of the paths $p_{s,d} \in P$, as shown in *Lines* 29-33, we will try to split the SCs into two groups. *Line* 30 finds the SC that has the smallest bandwidth demand in $\Psi_{s,d}(B_{s,d})$, and treats it as a new vNF graph $\Psi'_{s,d}(B'_{s,d})$, while *Line* 31 treats the remaining SCs in $\Psi_{s,d}(B_{s,d})$ as another

Algorithm 1: SC Deployment with Affiliation-aware vNF Placement

```
1  for all s-d pairs in G(V,E) do
2      find all the SC requests that s_j = s and d_j = d;
3      merge the SCs to generate a vNF graph Ψ_{s,d}(B_{s,d});
4  end
5  sort {Ψ_{s,d}(B_{s,d}), ∀s,d} in descending order of B_{s,d};
6  for each Ψ_{s,d}(B_{s,d}) in sorted order do
7      perform topological sorting on Ψ_{s,d}(B_{s,d}) to get the
       vNF sequence ψ;
8      for each p_{s,d} ∈ P that has enough bandwidth
       resources to satisfy B_{s,d} from the shortest do
9          flag = 0, v = s;
10         while flag = 0 OR v ≠ NULL do
11             place/reuse vNFs in ψ in sequence greedily
               on v, and record the result temporarily;
12             remove the placed vNFs from ψ temporarily;
13             if ψ = ∅ then
14                 commit the vNF placement result so far;
15                 flag = 1;
16             else
17                 if v = d then
18                     v = NULL;
19                     restore ψ and network status;
20                 else
21                     assign v as the next node on p_{s,d};
22                 end
23             end
24         end
25         if flag = 1 then
26             break;
27         end
28     end
29     if flag = 0 then
30         treat the SC with the smallest bandwidth demand
           in Ψ_{s,d}(B_{s,d}) as a new vNF graph Ψ'_{s,d}(B'_{s,d});
31         treat the rest SCs in Ψ_{s,d}(B_{s,d}) as another new
           vNF graph Ψ''_{s,d}(B''_{s,d});
32         replace Ψ_{s,d}(B_{s,d}) with Ψ'_{s,d}(B'_{s,d}) and
           Ψ''_{s,d}(B''_{s,d}) and sort {Ψ_{s,d}(B_{s,d}), ∀s,d} in
           descending order of B_{s,d} again;
33     end
34 end
```

new vNF graph $\Psi^{''}_{s,d}(B^{''}_{s,d})$. We then replace $\Psi_{s,d}(B_{s,d})$ with $\Psi^{'}_{s,d}(B^{'}_{s,d})$ and $\Psi^{''}_{s,d}(B^{''}_{s,d})$ and sort $\{\Psi_{s,d}(B_{s,d}), \forall s, d\}$ in the descending order of $B_{s,d}$ again. The smaller vNF graphs will be processed in subsequent loops. Note that, as we consider the offline version of SC deployment, we assume that the total IT and bandwidth resources in the network are enough to accommodate all the SCs, and thus we do not consider the blocking of SC requests in *Algorithm* 1.

Complexity analysis: The computational complexity of *Algorithm* 1 is $O(J \cdot |V|^2 \cdot \left[M + K \cdot |V| \cdot \max_n(|\psi_n|)\right])$, where J is total number of SC requests and $|\cdot|$ gets the size of a set.

D. Performance Evaluation

We run simulations to evaluate the performance of the proposed algorithms, *i.e.*, the ILP and the SC deployment with affiliation-aware vNF placement (AaP). Here, we also design a benchmark algorithm that deploys SCs without using the affiliation-aware vNF placement (NAaP). Instead of merging the SCs into vNF graphs and using topological sorting to assist the SC deployment, NAaP just sorts the SCs in descending order of their bandwidth demands and then deploys them with the similar procedure as *Lines* 6-28 in *Algorithm* 1.

We first use a small-scale six-node topology as shown in Fig. 1(b) to evaluate ILP, AaP and NAaP. Here, each node has C_v = 100 units of IT resources, each link has $B_e = 200$ Gbps bandwidth, $|M| = 4$ types of vNFs are considered, and the SCs can take $N = 3$ forms. Each type of vNF consumes IT resources within $[0.4, 1]$ units (based on the results in [9]), while its traffic processing capacity is within $[20, 40]$ Gbps. Each type of SC includes 2 to 4 vNFs and the bandwidth demand is randomly selected within $[1, 5]$ Gbps. We set $\beta = 0.01$ and $\alpha_m \in [1, 1.2]$ according to the discussion in [13]. For each data point, we run 200 independent simulations and average the results. Table I shows the simulation results. We observe that, in terms of three kinds of costs, ILP provides the lowest values, followed by AaP and NAaP. However, ILP needs the longest time to serve the SC requests, and with the increase of the number of SCs, its average running time increases significantly, while AaP's average running time increases with the number of SCs linearly. Meanwhile, compared with NAaP, AaP achieves lower values in terms of three kinds of costs, and consumes less time as well. This is because, by considering the affiliation information of vNFs with the help of vNF graphs and topological sorting, AaP can effectively reduce the vNF placement cost, the bandwidth cost, and thus the SC deployment cost.

We then evaluate the performance of AaP and NAaP in the NSFNET topology that consists of 14 nodes and 22 links. We assume that there are $|M| = 10$ types of vNFs, and the SCs take $N = 5$ forms, each of which includes 2 to 7 vNFs. We also scale up other simulation parameters related to IT and bandwidth resources accordingly. Fig. 3 shows the simulation results on three kinds of costs. Similar to the results in Table I, AaP has lower SC deployment cost, vNF placement cost and bandwidth cost than NAaP, but the difference between these two becomes more obvious in the large-scale topology.

IV. Online NFV Service Chain Deployment

In practical situations, the SC requests come and leave dynamically and may be blocked due to resource insufficiency. With certain minor modifications to handle request blocking, *Algorithm* 1 is still applicable to solve online SC deployment. At each service provisioning time, we first free the resources allocated to expired requests, and then use *Algorithm* 1 to handle all the pending requests and mark one as blocked if it cannot be served with the available resources in the network. However, we have to point out that the cost model of vNF placement should be changed in online SC deployment. Considering that on-demand vNF placement is usually complicated and costly, service providers may prefer to pre-allocate certain vNFs at a coarse time granularity (*i.e.*, service maintenance time) and then fine-tune the vNF placement at each service provisioning time. With these considerations, we modify the cost for placing an m type vNF as:

$$\alpha_m = \begin{cases} \alpha_{m,T} + \widehat{\alpha_m} \cdot (t_e - t_s), & \text{if deployed at } t_s = k \cdot T, \\ \alpha_{m,\Delta} + \widehat{\alpha_m} \cdot (t_e - t_s), & \text{otherwise,} \end{cases} \tag{11}$$

where $\alpha_{m,T}$ is the initial cost if the vNF is deployed at a service maintenance time (*i.e.*, separated by a coarse time period T), $\widehat{\alpha_m}$ is the cost per time unit, t_s and t_e are the time instants for vNF placement and removal, k is an integer, and $\alpha_{m,\Delta}$ is the initial cost if the vNF is placed on-demand. Normally, we have $\alpha_{m,\Delta} - \alpha_{m,T} > \widehat{\alpha_m} \cdot T$ to ensure that the cost saving is positive if we pre-allocate a vNF correctly. Since there might be request blocking in the dynamic provisioning, we have to consider the revenue from serving the SCs. Otherwise, minimizing the total resource cost would simply lead to block all the SC requests. We assume that the service provider gets a revenue of $\eta_j \cdot b_j$ after provisioning an SC request R_j. For online SC deployment, we try to maximize the service provider's profit, which equals the total revenue minus the total SC deployment cost.

A. Forecast-Assisted Online SC Deployment

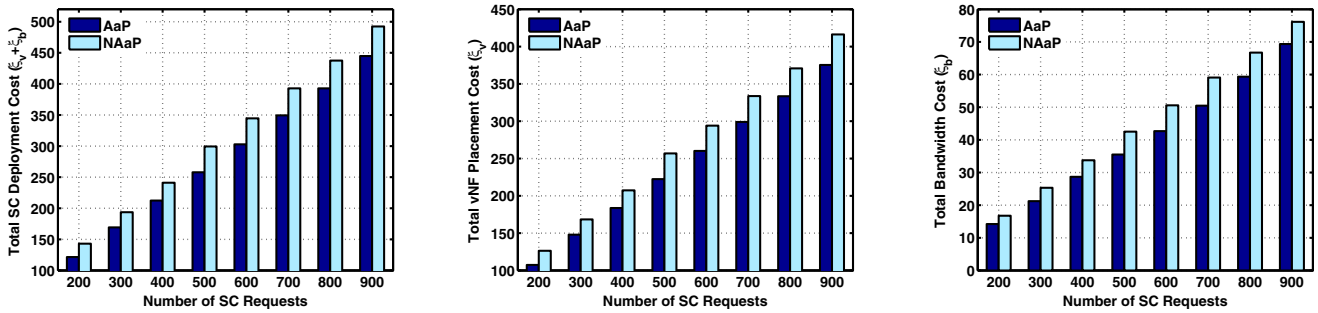
Algorithm 2: Forecast-assisted Online SC Deployment

```
1  while the network is operational do
2      free resources allocated to expired SCs;
3      apply Algorithm 1 to serve pending SC requests;
4      if t = k · T then
5          for each m ∈ M do
6              predict g_m leveraging the standard
               Fourier-Series-based prediction method;
7              if g_m > 0 then
8                  apply Algorithm 3 to pre-allocate vNFs;
9              end
10         end
11     end
12 end
```

To pre-allocate vNFs correctly, the service provider needs to forecast the requirements on vNFs precisely at each service

TABLE I
OFFLINE SC DEPLOYMENT WITH SIX-NODE TOPOLOGY

Number of SC Requests	Total SC Deployment Cost ($\xi_v + \xi_b$)			Total vNF Placement Cost (ξ_v)			Total Bandwidth Cost (ξ_b)			Average Running Time (Seconds)		
	ILP	AaP	NAaP	ILP	AaP	NAaP	ILP	AaP	NAaP	ILP	AaP	NAaP
50	19.30	21.56	23.07	16.24	18.59	20.06	2.96	2.97	3.01	61.40	3.37e-3	5.46e-3
60	20.80	25.49	25.97	17.30	21.88	22.36	3.50	3.61	3.61	86.80	4.74e-3	5.91e-3
70	24.86	28.68	29.86	20.70	24.48	25.66	4.16	4.20	4.20	393.54	5.00e-3	7.10e-3
80	27.50	32.49	33.03	22.80	27.69	28.20	4.70	4.80	4.83	1.11e+3	7.35e-3	7.38e-3
90	32.44	36.11	37.02	27.10	30.73	31.57	5.34	5.38	5.45	2.54e+3	8.13e-3	8.64e-3
100	35.48	40.27	40.92	29.44	34.23	34.88	6.04	6.04	6.04	2.11e+4	9.26e-3	9.37e-3



(a) Total SC deployment cost (b) Total vNF placement cost (c) Total bandwidth cost

Fig. 3. Offline SC deployment with NSFNET topology.

maintenance time (*i.e.*, at $t = k \cdot T$). Basically, it needs to know: 1) whether more vNFs of each type will be needed in the next period T? and 2) if yes, how many more vNFs of each type would be needed? Hence, we propose a forecast-assisted online SC deployment algorithm based on *Algorithm* 1 to predict future vNF requirements and pre-allocate vNFs accordingly at each service maintenance time. Then, at each service provisioning time within T, the algorithm fine-tunes vNF placement based on the actual incoming SC requests. *Algorithm* 2 provides the overall procedure of the forecast-assisted online SC deployment algorithm. Basically, at each service provisioning time, we first free resources allocated to expired SCs and then apply *Algorithm* 1 to serve pending SC requests as shown in *Lines* 2-3. If it is also a service maintenance time, *Lines* 4-11 perform prediction and pre-allocate each type of vNFs accordingly. Here, for the m type vNF, we leverage the standard Fourier-Series-based prediction method [14] and use the historical demands on the m type vNF as the inputs to predict the number of more vNFs needed in the next T, which is denoted as g_m. The pre-allocation of each type of vNFs is achieved with *Algorithm* 3.

Algorithm 3 tries to place new vNFs in the way such that the affiliation of the vNFs in supported SC types is considered. As all the supported SC types are known, we propose to quantify the affiliation of the vNFs with $\rho_{m,m'}$, which is the number of times that an m' type vNF is adjacent to an m type vNF in all the supported SC types. Fig. 4 shows an example on how to get $\{\rho_{m,m'}, \forall m, m'\}$. Note that, if $m = m'$, $\rho_{m,m'}$ represents the number of times that an m type vNF appears in all the supported SC types. Then, based on $\rho_{m,m'}$, we define a metric to quantify the preference of pre-allocating a new m type vNF on node $v \in V$ as:

$$\zeta_{m,v} = \sum_{\{u:u\neq v\}} \frac{\sum\limits_{m',i} x_u^{m',i} \cdot \gamma_{m',i} \cdot \rho_{m,m'}}{hops(u,v)}, \quad \forall m,\ v, \qquad (12)$$

where variable $x_u^{m',i}$ uses the same definition as that in Section III-A, $\gamma_{m',i}$ is the current utilization of the i-th m' type vNF in terms of traffic processing, and $hops(u,v)$ returns the hop-count of the shortest path between u and v. *Lines* 1-4 order the nodes based on $\zeta_{m,v}$. *Lines* 5-12 pre-allocate vNFs on the sorted nodes until all the needed vNFs have been placed.

Algorithm 3: Pre-allocation of New vNFs

```
1  for each node v ∈ V do
2  |  calculate ζ_{m,v} with Eq. (12);
3  end
4  sort nodes in V in descending order of ζ_{m,v};
5  for loop = 1 to g_m do
6  |  for each node v ∈ V in sorted order do
7  |  |  if v can accommodate an m type vNF AND
   |  |  g_m > 0 then
8  |  |  |  pre-allocate a new m type vNF on v;
9  |  |  |  g_m = g_m − 1;
10 |  |  end
11 |  end
12 end
```

B. Performance Evaluation

We use the NSFNET topology to evaluate the proposed algorithms for online SC deployment. Here, we assume that

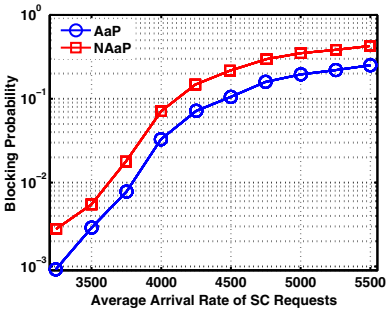


Fig. 5. Blocking probability.

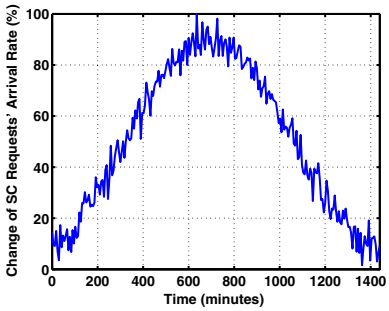


Fig. 6. Traffic fluctuation.

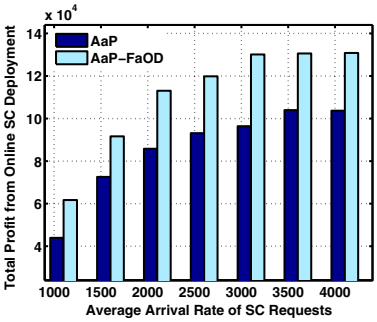
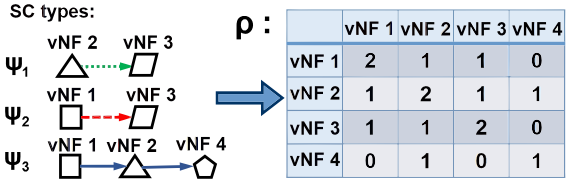


Fig. 7. Service provider's profit.



	vNF 1	vNF 2	vNF 3	vNF 4
vNF 1	2	1	1	0
vNF 2	1	2	1	1
vNF 3	1	1	2	0
vNF 4	0	1	0	1

Fig. 4. Example on quantifying the affiliation of vNFs with $\rho_{m,m'}$.

the service provisioning time is spaced by $t = 6$ minutes and the service maintenance period is $T = 60$ minutes. The revenue factor is $\eta_j \in [0.10, 0.12]$ for a deployed SC to serve per Gbps traffic, while the cost factors are set as $\alpha_{m,T} \in [0.5, 0.6]$ and $\alpha_{m,\Delta} = 8$, based on the discussion in [13]. All the other parameters are the same as those used for NSFNET in Section III-D.

We first generate the SC requests according to the Poisson traffic model, where the SC arrival rate is fixed so we can ignore the effect of prediction. Fig. 5 shows the results on blocking probability. We observe that, compared with NAaP, AaP can reduce the blocking probability by two orders of magnitude, which verifies that AaP can make better use of network resources again. Then, we generate the SC requests according to the traffic fluctuation in Fig. 6, where the online SC requests' arrival rate changes according to the percentage distribution. In the simulations, we use the same traffic fluctuation but try different time-average SC arrival rates. Since SC arrival rate is time-varying, we perform prediction and pre-allocation of vNFs using the proposed forecast-assisted online SC deployment algorithm (FaOD). Fig. 7 compares the performance of AaP-FaOD to that of AaP only (without prediction and pre-allocation of vNFs), in terms of service provider's profit. It is promising to notice that, AaP-FaOD brings much higher profit to the service provider than AaP. This verifies the effectiveness of prediction and pre-allocation of vNFs in AaP-FaOD.

V. Conclusion

This paper studied the SC deployment problem. We started with the offline version of the problem and proposed an ILP model and a time-efficient heuristic based on affiliation-aware vNF placement. Then, we moved to the online version of the problem, extended the heuristic algorithm to include the prediction on future vNF requirements, and designed a forecast-assisted online SC deployment algorithm. Simulation results showed that the online algorithms could reduce the blocking probability of SC requests and increase the service provider's profit from SC deployment effectively.

Acknowledgments

This work was supported in part by NSFC Project 61371117, Fundamental Research Funds for the Central Universities (WK2100060010), Natural Science Research Project for Universities in Anhui (KJ2014ZD38), and Strategic Priority Research Program of the CAS (XDA06010302).

References

[1] "Network functions virtualization (NFV)," Jan. 2012. [Online]. Available: https://portal.etsi.org/portal/server.pt/community/NFV/367
[2] B. Han *et al.*, "Network function virtualization: Challenges and opportunities for innovations," *IEEE Commun. Mag.*, vol. 53, pp. 90–97, Feb. 2015.
[3] M. Zeng, W. Fang, and Z. Zhu, "Orchestrating tree-type VNF forwarding graphs in inter-DC elastic optical networks," *J. Lightw. Technol.*, vol. 34, pp. 3330–3341, Jul. 2016.
[4] W. Fang *et al.*, "Joint defragmentation of optical spectrum and IT resources in elastic optical datacenter interconnections," *J. Opt. Commun. Netw.*, vol. 7, pp. 314–324, Mar. 2015.
[5] B. Addis *et al.*, "Virtual network functions placement and routing optimization," in *Proc. of CloudNet 2015*, pp. 171–177, Oct. 2015.
[6] W. Fang *et al.*, "Joint spectrum and IT resource allocation for efficient vNF service chaining in inter-datacenter elastic optical networks," *IEEE Commun. Lett., in Press*, 2016.
[7] T. Wood *et al.*, "Toward a software-based network: integrating software defined networking and network function virtualization," *IEEE Netw.*, vol. 29, pp. 36–41, May 2015.
[8] P. Lu *et al.*, "Highly-efficient data migration and backup for big data applications in elastic optical inter-datacenter networks," *IEEE Netw.*, vol. 29, pp. 36–42, Sept./Oct. 2015.
[9] T. Kim *et al.*, "A QoS assured network service chaining algorithm in network function virtualization architecture," in *Proc. of IEEE/ACM CCGrid 2015*, pp. 1221–1224, May 2015.
[10] R. Cohen *et al.*, "Near optimal placement of virtual network functions," in *Proc. of INFOCOM 2015*, pp. 1346–1354, Apr. 2015.
[11] M. Garey and D. Johnson, *Computers and Intractability: a Guide to the Theory of NP-Completeness*. W. H. Freeman & Co. New York, 1979.
[12] M. Zhang, C. You, H. Jiang, and Z. Zhu, "Dynamic and adaptive bandwidth defragmentation in spectrum-sliced elastic optical networks with time-varying traffic," *J. Lightw. Technol.*, vol. 32, pp. 1014–1023, Mar. 2014.
[13] "Amazon web services," Mar. 2016. [Online]. Available: https://aws.amazon.com/cn/ec2/pricing/
[14] A. Fumi *et al.*, "Fourier analysis for demand forecasting in a fashion company," *Int. J. Eng. Bus. Manag.*, vol. 5, pp. 1–10, Aug. 2013.