Network Function Consolidation in Service Function Chaining Orchestration

Tao Wen, Hongfang Yu, Gang Sun, Liu Liu
Key Lab of Optical Fiber Sensing and Communications, Ministry of Education
University of Electronic Science and Technology of China

***Abstract*—Network Function Virtualization (NFV) is an emerging technology proposed to improve service provisioning in telecommunication industry. The flexibility of NFV enables network operator to dynamically deploy Virtual Network Functions (VNFs) according to real-time requirements. However, a large amount of VNFs with low utilization may be deployed in network due to dynamic arrival and departure of services and excessive deployment of VNFs. In this paper, we analyse this phenomenon and model it as the Network Function Consolidation (NFC) problem, followed by an Integer Linear Programming (ILP) formulation. We also design a greedy based heuristic to solve large scale cases. We conduct simulations on both real-world and random generated network topologies, and the results show that we can reduce 32% number of VNFs in average.**

***Keywords*—Network Function Virtualization; Service Function Chaining; Network Function Consolidation.**

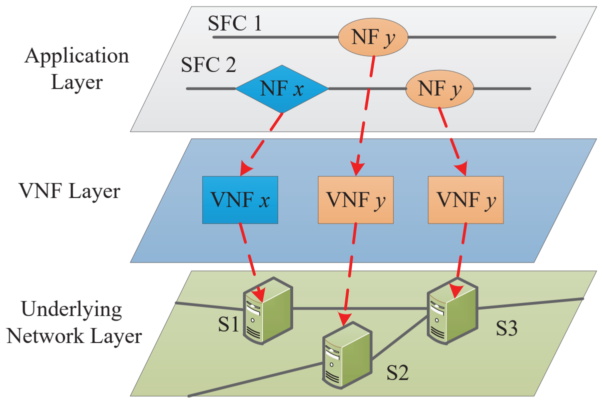

Fig. 1. NFV Service Provisioning Model

I. Introduction

Network Functions Virtualization (NFV) [1] is an emerging technology aiming to revolutionize network architecture and operation. According to [2, 3], the service provisioning model of NFV can be divided into three layer: *application layer*, *Virtual Network Function* (VNF) *layer* and *underlying network layer*, as shown in Fig.1. NFV provides services to end-users in application layer. The services in NFV are always refer to Service Function Chainings (SFCs) [2], which need to traverse several Network Functions (NFs) of different types with strict order, *e.g.*, a SFC may need to pass an intrusion detection system, then a proxy and finally a firewall [4]. The NFs are undertaken by VNFs in VNF layer, where VNFs are Virtual Machines (VMs) executing software of NFs. By leveraging virtualization technologies, the VNFs can be instantiated dynamically on commodity servers (*e.g.*, x86 based systems) in the underlying network layer.

NFV offers a great deal of flexibility to network operators in SFC orchestration [3, 5]. However, the flexibility also brings a new challenge in VNF deployment. As the instantiation of VNF is easy and the services arrive & depart dynamically, there may be a large number of VNFs deployed across the network even if they are seldom used [5]. These sprawl VNFs will increase the overhead of NFV system significantly in terms of two aspects. First, VNFs are executed by VMs, which have high resource consumption to run their own operating systems. Second, as entities existing in the network, VNFs need to be maintained during their life cycles [3, 6]. To alleviate this problem, we can consolidate NFs based on type and reconfigure the same type NFs in fewer VNFs, *e.g.*, we can reconfigure the NFs of type y in SFC 1 & 2 in one VNF deployed on S3 in Fig.1.

However, previous work about SFC orchestration do not consider this problem [7]–[9]. [7] proposes a greed mapping algorithm framework to do online SFC orchestration, [8] models SFC orchestration problem with multi-stage graph and designs a Viterbi based algorithm to solve it. [9] enhances the scalability of SFC online orchestration in its solution. All of them are proposed to solve orchestration problem of new arrived SFCs. Even we can modify their algorithms with NF consolidation objective, those SFCs already existing in the network can not be covered. Furthermore, it is infeasible to orchestrate the existing SFCs again for this way may bring unnecessary reconfiguration.

Moreover, VM consolidation [10] is a similar idea to NF consolidation. It is proposed to consolidate VMs on the physical machines with low utilization. However, there are two differences between NF and VM consolidation. First, NFs & VNFs have specific types thus NF consolidation is type based, whereas VMs do not have type feature. Second, SFCs typically have end-to-end performance requirements (*e.g.*, latency) which must be considered in NF consolidation,

Corresponding author: Tao Wen (winterwentao@gmail.com)

This work was supported in part by the 973 Program under Grant No. 2013CB329103, the 863 Program under Grant No. 2015AA015702 and 2015AA016102, the National Natural Science Foundation of China under Grant No. 61271171, 61271165, and 61571098, and the Ministry of Education–China Mobile Research Fund under Grant No. MCM20130131, China Postdoctoral Science Foundation (2015M570778)

while VM consolidation only needs to satisfy the performance requirements between directly connected VMs. These two aspects, which are not necessary to be considered in VM consolidation, make NF consolidation more challenging.

Above all, it is necessary to study NF consolidation in SFC orchestration. In this paper, we refer to this problem as the *Network Function Consolidation* (NFC) problem and propose a heuristic to solve it. The heuristic is a greedy based algorithm to find out the NF reconfiguration scheme with largest decrement of the number of VNFs. The main contribution of our work can be summarized as follows:

- To the best of our knowledge, we are the first to discuss and identify NFC problem in SFC orchestration.
- We formulate NFC by Integer Linear Programming (ILP) and use CPLEX [11] to find its optimal solutions for small scale cases.
- We design a heuristic algorithm and evaluate it by simulations on both real-world and random generated network topologies. The results show that:1) In small scale cases, we can reach 88% of optimal solutions. 2) In large scale cases, we can achieve 32% VNF decrement by NF consolidation.

The paper is organized as follows. We define NFC and formulate it as an ILP problem in Section II. We design a heuristic to solve NFC and analyse its computation complexity in Section III, followed by simulation results presented in Section IV. We review the related work of this paper in V and conclude the paper in Section VI.

II. Problem Statement

In this section, we define NFC in Section II-A, followed by formulation of NFC as an ILP problem in Section II-B.

A. Problem Definition

In NFC, we are given an underlying network, NF type specifications and a set of SFCs provisioned in the network with their current orchestration scheme. The current orchestration scheme refers to all the NFs configuration of SFCs in the network before NF consolidation.

Let $G = (V, E)$ denote the underlying network, where V and E represent node and link set of network, respectively. Each node $v \in V$ indicates a router or the commodity server attached to it. Let w_v denote the resource capacity of commodity server, and 0 means v is a router without attached commodity server. Each link $(m, n) \in E$ is configured with bandwidth w_{mn} and latency l_{mn}.

Let F denote NF type set can be provisioned in the network. Let d_f denote the server resource consumption to instantiate a VNF of type $f \in F$.

Let C denote SFC set provisioned in the network, where each SFC $c \in C$ is characterized by an end-to-end latency requirement q_c, and an order list of NF demands. For concision of following formulation, we insert ingress node of c at the beginning of the order list, and egress node at the end. Let l_c denote the length of this order list, and c_k the k-th NF on SFC c. Further, let d_c^k denote the server resource demand of c_k, and $d_c^{k,k+1}$ denote the bandwidth demand of traffic segment

TABLE I
Notations used in the paper

Notation	Description
V	The set of underlying network nodes.
E	The set of underlying network links.
F	The set of NF types.
C	The set of SFCs.
w_v	The server resource capacity of node v.
w_{mn}	The bandwidth capacity of link (m, n).
l_{mn}	The latency of link (m, n).
d_f	The server resource consumption to instantiate a VNF of type f.
q_c	The maximum latency that SFC c can tolerate.
l_c	The length of SFC c.
c_k	k-th NF in SFC c.
t_{ck}^f	1 if c_k is type of f, 0 otherwise.
d_c^k	The server resource demand of c_k.
$d_c^{k,k+1}$	The bandwidth demand between c_k and c_{k+1}.
x_{ck}^v	1 if c_k is configured on node v, 0 otherwise.
$\hat{x}_{ck}^v$	The value of x_{ck}^v before consolidation.
y_v^f	1 if VNF of type f is instantiated on node v, 0 otherwise.
$p_{cmn}^{k,k+1}$	1 if the traffic segment (c_k, c_{k+1}) on SFC c pass through link (m, n), 0 otherwise.

$(c_k,\ c_{k+1})$ between c_k and c_{k+1}. Besides, we define a binary parameter, $t_{ck}^f \in \{0, 1\}$, to indicate the types of NFs in SFCs:

$$t_{ck}^f = \begin{cases} 1 & \text{if } c_k \text{ is of type } f \in F \\ 0 & \text{otherwise} \end{cases}$$

Since ingress and egress nodes are not included in F, their t_{ck}^f are equal to 0, $\forall f \in F$.

In NFC, our purpose is to do NF consolidation with two objectives:

- minimizing total number of VNFs.
- minimizing total reconfiguration number of NFs.

The first objective is directly to optimize the NFC, while the second objective is to minimize unnecessary reconfiguration in NF consolidation.

B. Problem Formulation

For clear presentation, we summarize the notations used in the following formulation in Table I.

First, We define the decision binary variable x_{ck}^v to represent the configuration of NFs:

$$x_{ck}^v = \begin{cases} 1 & \text{if } c_k \text{ is assigned on node } v \\ 0 & \text{otherwise} \end{cases}$$

Let $\hat{x}_{cv}^i$ denote the value of x_{cv}^i before consolidation.

Next, we define another binary variable y_v^f to represent whether node v instantiates a VNF of type f, and y_v^f is equal to 1 if v does. y_v^f is not a decision variable, since it can be derived from x_{ck}^v as follow:

$$y_v^f = 1, \text{ if } \sum_{c \in C} \sum_{k=2}^{l_c - 1} x_{ck}^v t_{ck}^f \geq 1, \forall v \in V, f \in F \tag{1}$$

We assume that all the NFs of type f configured on node v are undertaken by one VNF. Thus y_v^f also represents the number of VNF of type f on v.

We need to make sure every NF is assigned to exactly one VNF. This constraint is represented as follows:

$$\sum_{v\in V} x^{v}_{ck} = 1, \forall c \in C, k \in (2, l_c - 1) \tag{2}$$

Here, we define the second decision variable to represent the mapping of traffic segment between NFs to physical links.

$$p^{k,k+1}_{cmn} = \begin{cases} 1 & \text{if } (c_k, c_{k+1}) \text{ pass through link } (m,n) \\ 0 & \text{otherwise} \end{cases}$$

We also have to guarantee that the assigned NFs do not violate the capacity constraints of nodes and links. These constraints are represented as follows:

$$\sum_{c\in C}\sum_{k=2}^{l_c-1} x^{v}_{ck} d^{k}_{c} + \sum_{f\in F} y^{f}_{v} d_f \le w_v, \forall v \in V \tag{3}$$

$$\sum_{c\in C}\sum_{k=1}^{l_c-1} p^{k,k+1}_{cmn} d^{k,k+1}_{c} \le w_{mn}, \forall (m,n) \in E \tag{4}$$

Furthermore, the flow conservation constraints need to be satisfied in our problem, for each traffic segment of every SFC. We present the constraints as follows:

$$\sum_{n\in\Omega(m)} (p^{k,k+1}_{cmn} - p^{k,k+1}_{cnm}) = x^{m}_{c,k} - x^{m}_{c,k+1}, \quad \forall v \in V, c \in C, k \in (1, l_c - 1) \tag{5}$$

Specifically, if c_k and c_{k+1} are assigned to same node, the traffic segment (c_k, c_{k+1}) is completed within the node. And this situation is also covered by (5).

Finally, we must ensure the end-to-end latency of each SFC does not exceed its tolerable value. The latency of SFC consists of link and node part. Due to the node resource guarantee of each VNF in (3), the node part latency of each SFC can be regarded as a constant. Therefore, we deduct it in advance and only consider the link part in the constraints:

$$\sum_{(m,n)\in E}\sum_{k=1}^{l_c-1} p^{k,k+1}_{cmn} l_{mn} \le q_c, \forall v \in V \tag{6}$$

As mentioned before, we consider two objective functions in our research:

- Minimize total number of VNF:

$$\text{minimize} \sum_{v\in V}\sum_{f\in F} y^{f}_{v} \tag{7}$$

- Minimize total reconfiguration number of NFs in SFCs:

$$\text{minimize} \sum_{c\in C}\sum_{i\in F}\sum_{v\in V} x^{i}_{cv}(1 - \hat{x}^{i}_{cv}) \tag{8}$$

NFC is a NP-hard problem. To easily prove it, we construct a simple case of NFC that can be reduced to a well-known NP-hard problem. In this case, there is only one SFC c, whose tolerable latency value is larger than l_c times of maximum simple path latency in network. In addition, capacity of each node in network can exactly hold one instance in F. In this case, NFC can be reduced to the *Virtual Network Embedding* problem, which has been proved to be NP-hard [12]. Therefore, NFC is also a NP-hard problem, and we design a heuristic to solve it in Section III.

Algorithm 1: Greedy Network Function Consolidation (GNFC)

```
   Input: $G = (V, E)$, $F$ and current configuration scheme.
   Output: $G$ under new configuration scheme.
1  Sort $V$ in descending order of its available capacity;
2  Create set $F_v \leftarrow F, \forall v \in V$;
3  for each $f \in F$ do
4      Create set $S_f \leftarrow \{c_k | t^{f}_{ck} = 1, \forall c \in C, k \in (2, l_c - 1)\}$;
5      Sort $S_f$ in ascending order of $d^{k}_{c}$;
6  while $\bigcup_{v\in V}(F_v \neq \varnothing)$ do
7      $v \leftarrow getFirstNode(V)$;
8      $(S_{fmax}, F_v) \leftarrow LCSS(G, v, F_v, S_f, \forall f \in F)$;
9      if $S_{fmax} \neq \varnothing$ then
10         $G \leftarrow G.reconfigure(S_{fmax}, v)$;
11         $S_f \leftarrow S_f.remove(S_{fmax})$;
12         Sort $V$ in descending order of available capacity;
13     else
14         $V \leftarrow V.remove(v)$;
15 return $G$;
```

III. Algorithm Design

Since NFC is NP-hard, we propose a heuristic, Greedy Network Function Consolidation (GNFC), to solve it. GNFC is greedy based algorithm, try to find the reconfiguration scheme with maximum decrement in VNF number. We describe the main processing of GNFC in Section III-A, followed by details of its key component in Section III-B. In addition, we analyse the complexity of GNFC in III-C.

A. Greedy Network Function Consolidation

In order to make GNFC practical, we design it based on the following three basic ideas.

- NF reconfiguration has its own cost in network operation, therefore we must avoid unnecessary reconfiguration. To meet this, we try to achieve the largest VNF decrement in every iteration of GNFC. GNFC will be terminated once no more VNFs can be shutdown.
- NF types provisioned in network can be quite huge, but the frequently used types are limited. Due to our objective, it is a appropriate strategy to only handle frequently used types.
- The source and destination nodes of SFC are fixed. Therefore, there are limited paths for each SFC satisfying its end-to-end latency requirement. We calculate a latency satisfied path set for each SFC in advance, which can reduce the consolidation searching space of each NF.

GNFC is shown as Algorithm 1. In initialization step, we introduce three types of sets. First, we sort V of network in descending order of its available capacity. Second, we create a NF set S_f for each type of NF $f \in F$, and sort S_f in ascending

Algorithm 2: Largest Candidate Set Selection (LCSS)

```
Input: $G = (V, E)$, current configuration scheme, $v$, $F_v$
       and $S_f, \forall f \in F$.
Output: $S_{fmax}$, $F_v$.
1  for each $f \in F_v$ do
2      $candSet_f \leftarrow \varnothing$;
3      $count_f \leftarrow 0$;
4      $avaCap_f \leftarrow v.remainCapacity - d_f * (1 - y_v^f)$;
5      for each $c_k \in S_f$ do
6          if $avaCap_f \geq d_c^k$ then
7              if find satisfied path for $x_c k^v = 1$ then
8                  $avaCap_f \leftarrow avaCap_f - d_c^k$;
9                  $count_f \leftarrow count + 1$;
10                 $candSet_f \leftarrow candSet_f \cup c_k$;
11         else
12             break;
13     if $count_f = 0$ then
14         $F_v \leftarrow F_v.remove(f)$;
15 $fmax \leftarrow argmax\{count_f\}$;
16 $S_{fmax} \leftarrow candSet_f max$;
17 $F_v \leftarrow F_v.remove(fmax)$;
18 return $(S_{fmax}, F_v)$;
```

order of NF server resource demand on this type. Third, we create a NF type F_v set for each $v \in V$, and initialize it with all type $f \in F$.

The following lines $6 \rightarrow 14$ are main loop of GNFC. In each iteration, we find out a NF candidate set S_{fmax} with type f, which can be assigned to the node with largest available capacity (lines $7 \rightarrow 8$). Thus we can enable a VNF of type f and reconfigure all NFs in S_{fmax} to this new VNF (line 12). Once it done, we remove type f from F_v, S_{fmax} from S_f, and resort V (lines $9 \rightarrow 12$). If no candidate set can be found for v, it will be removed from V (line 14). When $F_v = \varnothing, \forall v \in V$, the main loop will be terminated (line 6). Specifically, the function *LCSS()* in line 8 is used to find the largest NF candidate set in number for v, and the detail strategy of *LCSS()* will be discussed in next subsection.

B. Largest Candidate Set Selection

To find the largest NF candidate set in number for node v in GNFC, we propose Largest Candidate Set Selection (LCSS), as shown in Algorithm 2. The strategy is to find the set which can bring largest VNF amount decrement.

In LCSS, we first select the largest NF candidate set of each NF type for node v (lines $1 \rightarrow 14$). In each iteration, we identify $candSet_f$ to store fit NFs, and $count_f$ to count NFs for type f. We use $avaCap_f$ to represent the capacity available for f on node v(lines $2 \rightarrow 4$). If v do not enable VNF of type f, resource consumption to instantiate a VNF of type f will be deducted in advance (line 4). Since S_f has been sorted, we just need to take out the NFs in S_f in order to get the largest candidate set of type f for v (lines $5 \rightarrow 12$). For each NF inserted to $candSet_f$, both node and link resource requirements must to be satisfied (lines $6 \rightarrow 7$). To reduce complexity of LCSS, we calculate a latency satisfied path set for each SFC in advance, so the judgement in line 7 is only to search capacity satisfied path in this set. Once the available capacity of v is exhausted or all NFs in S_f have been tried, the selection of type f will be terminated.

After iterations for all types, we choose $candSet_f$ with the maximum $count_f$ as the candidate set for node v (lines $15 \rightarrow 16$). The types f with maximum $count_f$ or $count_f = 0$ will be removed from F_v, since no more NF candidate set of these types can be found for v (lines $14\&17$).

C. Complexity Analysis

We used k, m and n to represent the number of NF types, physical nodes and NFs, respectively. The most time-consuming computation of Algorithm 1 is the main loop in GNFC. As each iteration will kick out at least one type from NF type set of one node, the loop executes at most $k * m$ times. In each iteration, the most time-consuming computation causes by Algorithm 2. The complexity for Algorithm 2 is $O(kn)$, so the overall complexity of GNFC is $O(mnk^2)$. As explained in Section III-A, we only consider frequently used NF types, which means k can be regarded as a constant. In this case, complexity of GNFC is $O(mn)$.

IV. Simulation

A. Simulation Setup

1) Topology Dataset: We use both real-world and random generated topologies to evaluate our algorithm: (i) *Autonomous System 1755* (AS1755), *Autonomous System 1239* (AS1239) and *Autonomous System 3257* (AS3257) from Rocketfuel topology dataset [13]. (ii) *Random Graph 1* (RG-1, 50 nodes, 75 links, degrees of all nodes are 3) and *Random Graph 2* (RG-2, 100 nodes, 289 links, power law degree distribution) generated by NetworkX, which is a well-known python lib. The node and link capacities are generated randomly in $(1000 \backsim 2000)$ and $(500 \backsim 1000)$, respectively.

2) VNF & SFC Datasets: The type number of VNFs is set as 20, and VNF inherent server resource consumption is randomly generated in range $(10 \backsim 30)$. SFCs is generated by randomly choosing source and destination from node set. The server resource demands of NFs and the bandwidth demands between NFs are randomly generated in range $(10 \backsim 100)$.

3) Original Orchestration Scheme: We modify the Greedy Function Mapping (GFM) algorithm proposed in [7] as the online algorithm to generate original NF configuration scheme, for GFM can be easy modified with objective of NF consolidation. GFM We control the arrival and departure rate of SFCs to generate different input cases in the following simulations.

B. Evaluation Metrics

We use two evaluation metrics to show the performance of our algorithm: *1) VNF Decrement Ratio*: the ratio of number of VNFs reduced in processing; and *2) NF Reconfiguration Ratio*: the ratio of NFs need to be reconfigured in processing.

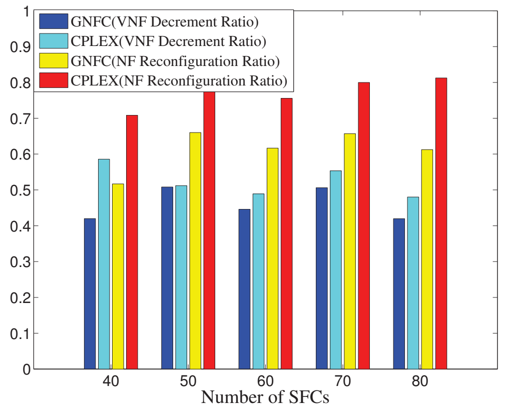

Fig. 2. Performance Comparison between CPLEX and GNFC

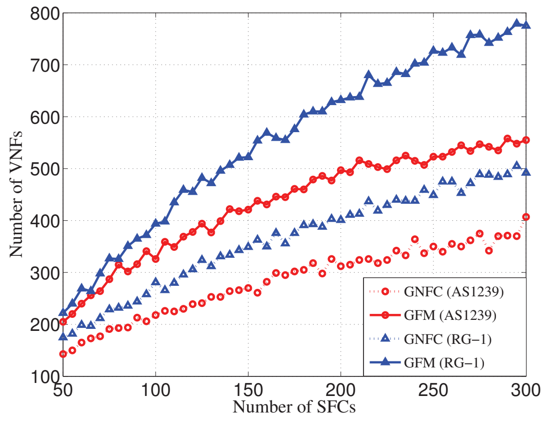

Fig. 3. Number of VNFs under GFM & GNFC with Different Number of SFCs

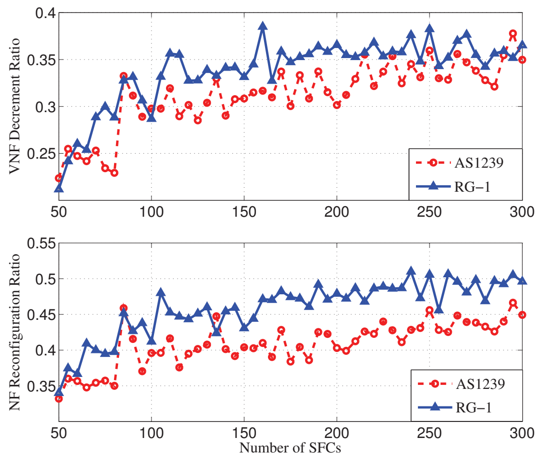

Fig. 4. GNFC Performance with Different Number of SFCs

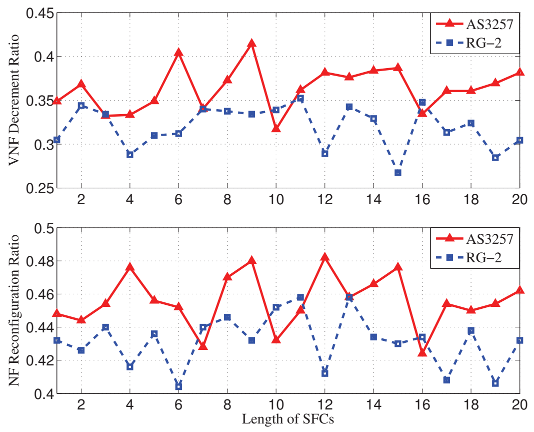

Fig. 5. GNFC Performance with Different Length of SFCs

C. *Simulation Result and Analysis*

We use CPLEX to solve ILP formulation of NFC and python to realize our algorithm, respectively. Since there is no similar work with NF consolidation purpose, we compare with CPLEX in small scale cases. In large scale cases, we use the modified GFM as the baseline to evaluate our algorithm in several network topologies.

1) Comparison with CPLEX: We use AS1755 as the network topologies to compare our algorithm with CPLEX, for no result can be got from CPLEX in acceptable time for larger cases. The comparison results are shown in Fig.2. The VNF decrement ratios of our algorithm can reach 88% of optimal solutions of CPLEX in average. While, our reconfiguration ratios are 79% of the optimal solutions. Our algorithm has strict capacity judgement to make sure that each step of reconfiguration does not cause traffic loss. Therefore, we cannot consolidate all the NFs possible to do consolidation. With changing of SFCs in network, the performance of our algorithm remain stable. Note that reduce one VNF need to reconfigure at least one NF. This is the reason why reconfiguration ratios are always higher than decrement ratios.

2) Performance in Multi-Networks: We evaluate our algorithm in the rest topologies under three scenarios:

a) First, we change SFCs number in network and fix the length of SFCs as 5. As shown in Fig.3, the number of VNFs increases at a near-linear trend. Besides, it increases faster in RG-1, which means that inefficient phenomenon of VNFs is more serious in large topology. The decrement number of VNFs caused by GNFC increases with the growth of the number of SFCs, which is reasonable in practice.

Furthermore, we display the VNF decrement ratios and NF reconfiguration ratios of GNFC under this scenario in Fig.4. The results indicate that VNF decrement ratios have an increasing trend with the increase of number of SFCs. It means inefficient phenomenon of VNFs is more serious with more SFCs. Besides, we can find that the reconfiguration ratios (44% in average) are match to VNF decrement ratios (32% in average) in GNFC. This verifies that our algorithm is efficient in terms of avoiding unnecessary NF reconfiguration.

b) Second, we vary the length of SFCs in range $(1 \backsim 20)$ and fix the number of SFCs at 200. The simulation results in Fig 5 show that two performance metrics of GNFC are very stable without clear trend. The reconfiguration ratios (43% in average) are also match to VNF decrement ratios (33% in average) under this scenario.

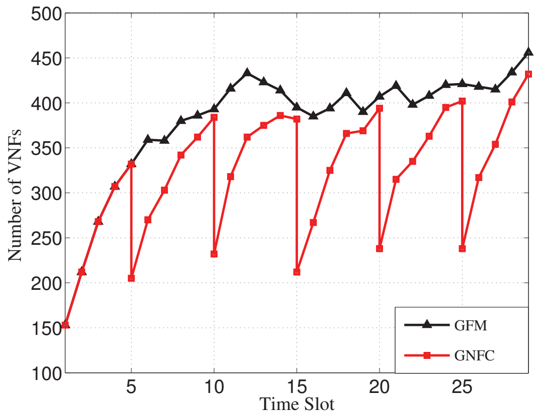

Fig. 6. Performance to Reduce Number of VNF over Time

c) Third, we simulate a real-world scenario with dynamic arrival and departure of SFCs in a period of time. In this scenario, we execute NFC periodically. As shown in Fig. 6, we can observe several significant drops of number of VNFs, which are due to execution of GNFC . During intervals of GNFC executions, the number of VNFs increases fast, indicating that dynamic arrival and departure of SFCs will bring serious inefficient phenomenon of VNFs.

V. Related Work

SFC orchestration, as a key problem in NFV, has received much attention in recent work [6–9, 14–16]. The main ideas in these work can be classified into two categories: The first category focuses on VNF deployment problem in SFC orchestration, such as [6, 14]. [14] models NFV location problem by combination of facility location problem and the generalized assignment problem. [6] proposes a problem about VNF deployment with minimal resources consumption. While both approaches solve the VNF deployment problem, it ignored end-to-end performance requirements of SFCs.

The second category considers both VNF deployment and the end-to-end performance requirements, such as [7–9, 15, 16]. [15] decouples SFC Orchestration into nested VM embedding problems. However, SFC is an end-to-end communication service, rather than a many-to-many embedding problem as in the context of virtual machine embeddings. [7] proposes a greed function mapping algorithm framework to do online SFC orchestration, but the algorithm must know VNF placement scheme in advance. [16] combines routing and VNF placement, but there is no efficient solution in [16]. Besides, capacity of VNF is fixed in [16], which is not reasonable in the virtualization context. [9] proposes an online orchestration solution of SFCs with well scalability. [8] deploys nested bin packing to formulate SFC orchestration problems, and proposed a Viterbi based algorithm with near optimal performance. However, both of [9] and [8] do not consider NF consolidation to reduce the number of VNFs.

Besides, VM consolidation [10] is a similar idea to ours. However, there are two differences between them. First, VMs do not have type feature, whereas NFs have. Second, there is no SFC-like service needs to be satisfied in VM consolidation.

VI. Conclusion

In this paper, we discuss the phenomenon that inefficient VNFs may sprawl across network in SFC orchestration. We propose the *Network Function Consolidation* (NFC) problem. In the NFC, we focus on the optimization of NF Consolidation to reduce the number of VNFs. We formulate the NFC as an ILP problem and prove that it is a NP-hard Problem. To address it, we design a greedy based heuristic, GNFC, to maximize the decrement of VNF number by reconfiguring NFs in SFCs. Our simulation results show that we can reduce 32% VNFs in average.

References

[1] M. Chiosi, D. Clarke, P. Willis, A. Reid, J. Feger, M. Bugenhagen, W. Khan, M. Fargano, C. Cui, H. Denf *et al.*, "Network functions virtualisation: An introduction, benefits, enablers, challenges and call for action," in *SDN and OpenFlow World Congress*, 2012, pp. 22–24.

[2] P. Quinn and T. Nadeau, "Problem Statement for Service Function Chaining," Internet Requests for Comments, RFC Editor, RFC 7498, April 2015.

[3] R. Mijumbi, J. Serrat, J. Gorricho, N. Bouten, F. De Turck, and R. Boutaba, "Network function virtualization: State-of-the-art and research challenges," *Communications Surveys Tutorials, IEEE*, Accepted for Publication September 2015.

[4] Z. A. Qazi, C.-C. Tu, L. Chiang, R. Miao, V. Sekar, and M. Yu, "Simplefying middlebox policy enforcement using sdn," in *ACM SIGCOMM Computer Communication Review*, vol. 43, no. 4. ACM, 2013, pp. 27–38.

[5] B. Han, V. Gopalakrishnan, L. Ji, and S. Lee, "Network function virtualization: Challenges and opportunities for innovations," *Communications Magazine, IEEE*, vol. 53, no. 2, 2015, pp. 90–97.

[6] X. Li and C. Qian, "The virtual network function placement problem," in *Computer Communications Workshops (INFOCOM WKSHPS), 2015 IEEE Conference on*, April 2015, pp. 69–70.

[7] R. Mijumbi, J. Serrat, J.-L. Gorricho, N. Bouten, F. De Turck, and S. Davy, "Design and evaluation of algorithms for mapping and scheduling of virtual network functions," in *Network Softwarization (NetSoft), 2015 1st IEEE Conference on*, 2015, pp. 1–9.

[8] M. Bari, S. R. Chowdhury, R. Ahmed, and R. Boutaba, "On orchestrating virtual network functions in nfv," *arXiv preprint arXiv:1503.06377v2*, 2015.

[9] S. Mehraghdam, M. Keller, and H. Karl, "Specifying and placing chains of virtual network functions," in *Cloud Networking (CloudNet), 2014 IEEE 3rd International Conference on*. IEEE, 2014, pp. 7–13.

[10] M. Wang, X. Meng, and L. Zhang, "Consolidating virtual machines with dynamic bandwidth demand in data centers," in *INFOCOM, 2011 Proceedings IEEE*. IEEE, 2011, pp. 71–75.

[11] IBM. www.ibm.com/software/commerce/optimization/cplex-optimizer/.

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

[13] N. Spring, R. Mahajan, and D. Wetherall, "Measuring isp topologies with rocketfuel," in *ACM SIGCOMM Computer Communication Review*, vol. 32, no. 4. ACM, 2002, pp. 133–145.

[14] R. Cohen, L. Lewin-Eytan, J. S. Naor, and D. Raz, "Near optimal placement of virtual network functions," in *Computer Communications (INFOCOM), 2015 IEEE Conference on*. IEEE, 2015, pp. 1346–1354.

[15] H. Moens and F. De Turck, "Vnf-p: A model for efficient placement of virtualized network functions," in *Network and Service Management (CNSM), 2014 10th International Conference on*. IEEE, 2014, pp. 418–423.

[16] B. Addis, D. Belabed, M. Bouet, and S. Secci, "Virtual network functions placement and routing optimization," in *Cloud Networking(CloudNet), 2015 IEEE 4th International Conference on*. IEEE, 2015.