

Cost-aware Placement and Chaining of Service Function Chain with VNF Instance Sharing

Hantao Guo*, Ying Wang*, Zifan Li[†], Xuesong Qiu*, Hengbin An*, Peng yu*, Ningcheng Yuan*

* *State Key Laboratory of Networking and Switching Technology*

Beijing University of Posts and Telecommunications

Beijing, China

[†]*State Grid Information & Telecommunication Branch*

Email:{guohantao, Wangy}@bupt.edu.cn

Abstract—Network Function Virtualization (NFV) is an important shift in telecommunication service provisioning. It enables the decoupling of network element functions and dedicated hardware devices. How to economically place and chain Virtual Network Functions (VNFs) according to the requirements of Service Functions Chains (SFCs) are the challenges for NFV orchestration. In this paper, we consider the offline deployment issue from the perspective of sharing VNF instance to improve resource utilization and reduce total placement costs. Firstly, we generalize the problem as a Facility Location Problem and propose a Mixed Integer Linear Programming (MILP) model. Besides, our model can be dynamically configured according to the different deployment preferences. Then we propose a heuristic algorithm based on the Steiner Tree Problem and Markov Decision Process (MDP). We evaluate our heuristic algorithm by comparing with the optimal solution of MILP and a classic graph based algorithm. The results show that the difference of the deployment costs between our algorithm and the optimal solution is less than 3%. However, the execution time can be significantly reduced by 57.4%.

Index Terms—Network Function Virtualization, NFV Orchestration, SFC Placement.

I. INTRODUCTION

In the traditional network, the physical intermediate devices are the main support of the underlying network, including various network functions such as information exchange, routing, firewall and network proxy. However, with more and more tenants publishing their applications in the cloud, it is necessary for operators to spend a lot of money on dedicated network devices to meet the service requirements. Network Function Virtualization (NFV) is proposed, which leads to simplification of the network structure, significant reductions in Operating Expenses (OPEX) and Capital Expenses (CAPEX) and facilitation of deploying new services with increased agility and faster time-to-value [1].

As an important shift in telecommunication service provisioning, NFV has drawn significant attention from both academia and industry. It integrates Network Functions (NFs) as Virtual Network Functions (VNFs) into industry standard servers, switches and storage hardware. By abstracting functions and decoupling software and hardware, NFs no longer depend on dedicated hardware. Besides, with NFV, networks

and resources can be centrally managed and automatically configured, which can achieve automatic deployment, scaling, fault isolation and self-healing based on the ever-changing business needs.

NFV orchestration is the process of automating the deployment and provisioning of multiple network software components. It involves deploying, instantiating and managing a composition of VNFs that form a Service Function Chain (SFC) [2]. How to place SFC requests properly in NFV infrastructure while ensuring low instance deployment cost, low forwarding bandwidth cost and high resource utilization is an NP-hard problem [3]. Most of the current deployment researches focus on the forwarding bandwidth cost [4] or VNF instance deployment cost [5]. Ye et al. [4] combine the different types of VNFs into a multi-function VNF to save the forwarding bandwidth among them. Tastevin et al. [5] propose a graph based algorithm to reduce the instance deployment cost.

Current works usually consider that the deployment process of VNFs in each SFC is independent, which means the VNF instances deployed in previous SFCs will not be reused in present SFC. However, with the proposal of a new VNF scheduling model based on the min-plus algebra theory [6], it is possible to share the deployed VNF instances among different services. Based on this scheduling method, if we consider the deployment of SFC from the point of view that the deployed VNF instances can be shared simultaneously between the same type of VNF from different SFC, the following advantages will be brought:

- Reuse the existing VNF instances, which avoids the time of instances initialization and enables faster service provisioning;
- Utilize the exclusive idle resource of VNF instance, which reduces internal resource fragmentation and improves resource utilization;
- Avoid the creation of redundant VNF instances, which reduces the VNF instance cost.

However, VNF instance sharing is also a double-edged sword. Some SFCs may not be deployed on the shortest path, which leads to the great increase in forwarding cost.

Therefore, in the scenario with VNF instance sharing, the main challenges are how to effectively select the Points of Presence (PoPs) to deploy shared VNF instances and minimize the total deployment costs. This article focuses on minimizing the total deployment costs and improving the resource utilization. The main contributions of this paper are as follows:

- (1) Proposing a new placement and chaining problem model based on the VNF instance sharing, which improves resource utilization and minimize the deployment costs.
- (2) Realizing the dynamic configuration of the deployment preference. By introducing a dynamic weight in the objective model, it can balance the benefit of VNF instance sharing with the penalty of forwarding path growth.
- (3) Proposing a heuristic algorithm based on the Steiner Tree and Markov Decision Process. The solution can be obtained more effectively by our method and the deployment costs are only 3% worse than the optimal solution.

The rest of this paper is organized as follows. Section II reviews the related work in the area of NFV orchestration. Section III describes our Service Functions Chaining Problem and proposes a Mixed Integer Linear Programming (MILP) formulation. Section IV presents the heuristic algorithm we proposed. Simulation results are presented in Section V. Section VI summarizes the paper.

II. RELATED WORK

According to the optimization objectives, past studies in the NFV orchestration can be reviewed with regard to three aspects: resource consumption and utilization, delay as well as reliability.

In terms of resource consumption and utilization, most papers adopt the strategy of node deployment cost first. Luizelli et al. [7] formalize the virtual network function placement and chaining problem as an Integer Linear Programming (ILP) model and propose a heuristic algorithm to solve it. Bari et al. [8] take the path length, the potential cost of opening a new PoP and the penalty cost due to Service Level Objective (SLO) violations into account when deploying and propose a heuristic algorithm based on the Viterbi algorithm. On the basis of [7], [8], Tastevin N et al. [5] propose a graph approach to place VNFs. However, the extension of forwarding path is still inevitable, which may lead to the increase of forwarding bandwidth consumption, especially when the network is large. Mijumbi, Rashid, et al. [9] propose a Graph Neural Network (GNN)-based algorithm which exploits Virtual Network Function Forwarding Graph (VNF-FG) topology information to predict future resource requirements for each Virtual Network Function Component (VNFC). And there are also some strategies that focus on bandwidth costs. F. Wang et al. [10] propose a novel elastic VNF placement strategy. It solves the problem of bandwidth waste due to the unpredictable placing location of VNF caused by the ever-changing workload. As we mentioned before, Ye et al. [4] propose a closed-loop with critical mapping feedback algorithm to minimize the forwarding consumption in the substrate network. In order to

improve the resource utilization, Yi, Bo et al. [6] propose a new VNF scheduling model. By scheduling the execution time slot for the traffic of each SFC traversing the VNF Instance, the resource utilization is improved and the resource fragmentation is reduced.

In terms of optimizing the service delay, Alleg [11] introduce a placement and chaining VNF model based on a flexible resource allocation approach. They take propagation delay and processing delay into account when searching for the optimal solution. And [12], [13] focus on the impact of the sequential order of scheduling between different VNFs on end-to-end delay.

In terms of reliability, Fan J et al. [14] propose an online algorithm to guarantee the required high reliability while minimizing the physical resources consumption. Qu et al. [15] establish a reliability-aware and delay-constrained routing optimization framework for NFV-enabled datacenter networks.

The focus of this paper is still on the resource consumption and utilization of the SFC placement. Under the condition of satisfying the resources constraint, this paper proposes a placement strategy based on sharing VNF instances and considering both node cost and bandwidth cost.

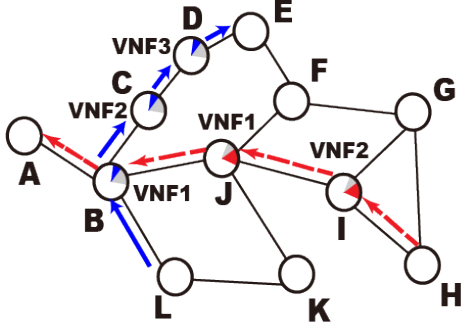
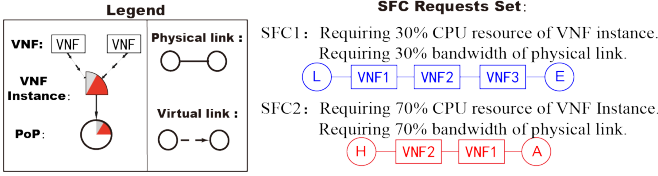
III. THE SFC PLACEMENT PROBLEM FORMULATION

In this section, we describe and model the problem of SFC placement.

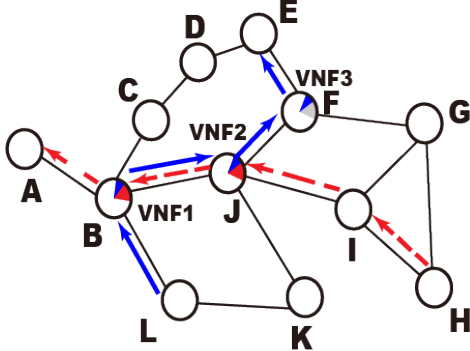
A. Problem Motivation

In the NFV environment, each network service is represented by an SFC which is composed of a series of ordered VNFs. These VNFs have specific physical resource requirements in order to offer a specific QoS (Quality of Service). The placement of the SFC consists of two steps: *i*) placing the VNFs without violating physical resource capacity constraints; *ii*) chaining these VNFs according to the SFC, creating forwarding paths and allocating the required bandwidth. Each step takes up a certain amount of physical resource which incurs the corresponding deployment costs. We focus on reducing the deployment costs in two aspects: (1) the VNF instance cost, caused by occupying CPU resources of the hosting PoP; (2) the forwarding bandwidth cost, which is related to the distances of the deployment paths of SFCs. In our model, the idle CPU resources of VNF instances can be shared by other SFCs to improve the resource utilization and reduce the creation of the same type VNF instance.

Fig. 1(a) illustrates a typical SFC deployment scenario without sharing VNF instances, where both SFC1 and SFC2 are placed in the shortest paths $L \rightarrow B \rightarrow C \rightarrow D \rightarrow E$ and $H \rightarrow I \rightarrow J \rightarrow B \rightarrow A$. Both SFC1 and SFC2 request VNF1 and VNF2. As one VNF instance can serve the same type VNFs from different SFCs at the same time, we can merge the VNF instance 1 in PoPs B and J into PoP B, and then merge the VNF instance 2 in PoPs C and I into PoP J, as shown in Fig. 1(b). Therefore, there is no need to create any other redundant VNF instances in this scenario. The number of instances can be reduced. Although the hops of the deployment



(a) Placement without sharing VNF instances



(b) Placement with sharing VNF instances

Fig. 1. No sharing VNF instances v.s. sharing VNF instances

path of SFC1 do not change, the total deployment costs will decrease due to saving the cost of creating redundant VNF instances.

However, in a more complex network, when we merge the redundant VNF instances of the same type into several shareable VNF instances (S-VNFIs), some SFCs will not be deployed on the shortest path, and it will make detours in the physical path to reach these S-VNFIs. The total deployment costs will also be raised because the increase in forwarding cost outweighs the cost-saving caused by reducing VNF instances. Therefore, we have to balance the benefit of reducing the creation of instances with the penalty of path growth.

We aim to propose a flexible deployment strategy to achieve lower deployment costs and higher resource utilization as much as possible. The problem model and the corresponding hypotheses will be presented in the following sub-sections.

B. Network and Service Model

In the physical level, as Fig. 1(b) shown, each PoP has a limited number of CPUs with the same processing power. Each VNF instance occupies one CPU of the PoP, which means, the

maximum number of VNF instances in one PoP is determined by the number of CPUs. In this paper, we assume each PoP has four CPUs. In PoP B, three-quarters of the circle filled with white represents three idle CPUs that do not carry any VNF instance, and a quarter filled with other colors represents that this CPU carries a VNF instance. PoPs B, J and F host VNF instances 1, 2 and 3, respectively. For each VNF instance shown as a quarter circle, the gray part represents its idle CPU resource.

We suppose that a VNF instance can serve the same type VNFs in different SFCs when it has enough resource. So, VNF instances 1 and 2 will serve both SFC1 and SFC2 at the same time. VNFs of each SFC occupy the CPU resource of VNF instances on demand. The CPU resource requirements for SFC1 and SFC2 are 30% and 70%, respectively. We use blue to express the proportions of the CPU resource occupied by SFC1 and red for SFC2. When we deal with SFC requests, we need to ensure that each VNF is deployed according to the sequence and the capacity constraints.

C. MILP Formulation

The placement of S-VNFI is considered as a Facility Location Problem, which is also an extension of the Steiner Tree Problem. We model the issue into an MILP problem. TABLE I summarizes the notation of NFVI and SFC and corresponding parameters. The main optimization criterion is to minimize the total deployment costs by reducing the VNF instances cost and optimizing the forwarding path selection.

We regard the VNF instances shared by multiple SFCs as facilities. In addition to the PoPs in the shortest deployment path for each SFC, some additional PoPs can be selected to deploy these S-VNFIs to minimize the VNF instance cost and forwarding cost. For those unique VNF types of each SFC, we can place the instances directly on the corresponding shortest path of the previous and subsequent constraint PoPs.

Substrate network - It can be represented as a weighted undirected graph $G = (N, L)$. N represents the set of PoPs and L represents the set of physical links. Each PoP $n_p \in N$ has a limited number of CPUs C_{n_p} . And each physical link $l_{p,q} \in L$ is equipped with a bandwidth capacity $B_{p,q}$, where p, q represent n_p, n_q .

VNF instance - The VNF instance set is represented by V , $|V|$ is the number of VNF instances. For example, in Fig. 1(b), $|V| = 3$. Each VNF instance $v_k \in V$ has a CPU capacity C_{v_k} . V_{cost} represents the cost for creating one VNF instance. And T_{v_k} represents the type of the VNF instance v_k .

SFC - We use S to represent the SFC requests set. s_i represents the structure of the i^{th} SFC request in S . In the example illustrated in Fig. 1(b), $s_1 = \{VNF_1, VNF_2, VNF_3\}$. s_i^j means the type of the j^{th} VNF in s_i . $D_{s_i^j}$ represents the CPU resource demand for the j^{th} VNF in s_i . vl_i^j represents the j^{th} virtual link in s_i . $DB_{vl_i^j}$ represents the bandwidth demand for the j^{th} virtual link in s_i . We assume that the cost for using 1% bandwidth in a physical link is L_{cost} .

Mapping Result - The real physical forwarding path of the j^{th} virtual link of s_i is stored in $P_{vl_i^j}$. $A_{v_k \rightarrow n_p}$ is a binary

TABLE I
VARIABLES AND NOTATIONS

Var.	Description
G	substrate network topology graph.
N	Sets of physical nodes (PoPs), $\{n_1, \dots, n_m\}, m = N $.
L	Sets of physical links $l_{p,q} = (n_p, n_q) \in L$.
C_{n_p}	The number of CPUs in POP n_p .
$B_{p,q}$	The bandwidth of the physical link $(n_p, n_q) \in L$.
V	Sets of VNF instances, $\{v_1, v_2, \dots, v_m\}, m = V $.
T_{v_k}	The type of VNF instance v_k .
C_{v_k}	The CPU capacity of VNF instance v_k .
V_{cost}	Cost for creating one VNF instance.
S	Sets of SFC requests $\{s_1, s_2, \dots, s_m\}, m = S $.
s_i	The i^{th} SFC request in S .
$s_i^j \in s_i$	The type of j^{th} VNF of the s_i .
$D_{s_i^j}$	The CPU resource demand for the j^{th} VNF in s_i .
$vl_i^j \in s_i$	The j^{th} virtual link in s_i .
$DB_{vl_i^j}$	The bandwidth demand for the j^{th} virtual link in s_i .
L_{cost}	Cost for occupying 1% bandwidth from a physical link.
α	The weight of placement priority.
$P_{vl_i^j}$	The real physical forwarding path of the j^{th} virtual link of s_i .
$A_{v_k \rightarrow n_p}$	1: VNF instance v_k is deployed on physical node n_p . 0: otherwise.
$A_{s_i^j \rightarrow v_k}$	1: The j^{th} VNF in s_i is deployed on instance v_k . 0: otherwise.

variable. When VNF instance v_k is deployed on PoP n_p , $A_{v_k \rightarrow n_p} = 1$, if not $A_{v_k \rightarrow n_p} = 0$. $A_{s_i^j \rightarrow v_k}$ is also a binary variable. When the j^{th} VNF in s_i is deployed on the instance v_k , $A_{s_i^j \rightarrow v_k} = 1$, if not $A_{s_i^j \rightarrow v_k} = 0$.

Equation (1) is our objective function. We aim to obtain the optimal VNF instance placement scheme $A_{v_k \rightarrow n_p}$ and the forwarding path of each SFC $P_{vl_i^j}$ to minimize the total costs. The first half of the function represents the VNF instance cost. And the second half represents the forwarding cost. α represents the weight of placement priority. It is used to balance the benefit of reducing the creation of instances with the penalty of path growth. When α is small, the strategy prefers to place the VNF on the existing S-VNFs and reduces the VNF instance cost. When α is large, the strategy prefers to avoid path elongation at the cost of creating new instances.

Objective:

$$\begin{aligned} [A_{v_k \rightarrow n_p}, P_{vl_i^j}] = \arg \min & \left((1-\alpha) \sum_{v_k \in V} \sum_{n_p \in N} A_{v_k \rightarrow n_p} * V_{cost} \right. \\ & \left. + \alpha \sum_{s_i \in S} \sum_{vl_i^j \in s_i} |P_{vl_i^j}| * DB_{vl_i^j} * L_{cost} \right) \end{aligned} \quad (1)$$

Subject to:

$$C_{n_p} \geq \sum_{v_k \in V} A_{v_k \rightarrow n_p} \geq 0, \forall n_p \in N \quad (2)$$

$$C_{v_k} \geq \sum_{s_i \in S} \sum_{s_i^j \in s_i} D_{s_i^j} * A_{s_i^j \rightarrow v_k} \geq 0, \forall v_k \in V \quad (3)$$

$$B_{p,q} \geq \sum_{s_i \in S} \sum_{vl_i^j \in s_i} DB_{vl_i^j} * \lambda \geq 0, \forall l_{p,q} \in L \quad (4)$$

$$\lambda = \begin{cases} 1 & l_{p,q} \in P_{vl_i^j}; \\ 0 & \text{otherwise} \end{cases}$$

$$T_{v_k} - s_i^j = 0, \quad \forall v_k \in V, \forall s_i \in S, \forall s_i^j \in s_i: A_{s_i^j \rightarrow v_k} = 1 \quad (5)$$

$$\sum_{v_k \in V} A_{s_i^j \rightarrow v_k} \leq 1, \forall s_i \in S, \forall s_i^j \in s_i \quad (6)$$

Equations from (2) to (4) are the capacity constraints. Equation (2) ensures that the number of VNF instances hosted by a PoP does not exceed the total number of CPUs in it. Equation (3) ensures that the amount of CPU resource allocated to VNFs does not exceed the total CPU resource of a VNF instance. Equation (4) ensures that the bandwidth allocated to the forwarding path of the corresponding SFCs cannot exceed the bandwidth of the physical link.

Equation (5) and (6) are the consistency constraints. Equation (5) ensures that before a VNF is deployed on a VNF instance, the type of the VNF instance should be consistent with the type of the VNF. Equation (6) ensures that each VNF in an SFC can only be deployed on a single VNF instance, and cannot be served by several VNF instances.

IV. HEURISTIC ALGORITHM BASED ON THE STEINER TREE AND MARKOV DECISION PROCESS

In this section, we present a heuristic algorithm to solve this optimization issue. Steiner Tree problem is an umbrella term for a class of problems in combinatorial optimization. It is originally considered as a geometric problem for solving geometric median, such as Fermat points. One well-known variant, which is used to study the routing of multipoint connections in [16], is the Steiner tree problem in graphs. Let F be a non-empty subset of PoPs set N . F consists of the PoPs in the shortest deployment path of all SFCs. The induced subgraph $G[F]$ is the graph whose vertex set is F and whose edge set consists of all of the edges in L that have both endpoints in F . Based on the induced subgraph $G[F]$, the Steiner Tree is calculated. In the induced subgraph $G[F]$, to reach the optimal solution, the number of additional PoPs that need to be introduced from G is at most $|F| - 2$, and these PoPs are called Steiner points.

The algorithm can generally be divided into three steps. In step A, all SFC requests are traversed to find the type of VNFs that can be shared. In step B, Steiner points are selected from the substrate network graph G as the candidate deployment PoPs for placing S-VNFs. In step C, we use Markov Decision

Algorithm 1 Heuristic algorithm

Input: $G \leftarrow (N, L)$, $S \leftarrow SFCRequests$, $K = 0$, α ;

Output: *Result*;

△ **Step A :** △

```
1: for  $s_i \in S$  do
2:   if  $s_i^j \in s_k (k \neq i)$  then
3:      $shareable\_VNF\_types.add(s_i^j)$ ;
4:      $K++$ ;
5:      $optimizable\_SFCs.add(s_i)$ ;
6:   else
7:      $unique\_VNF\_types.add(s_i^j)$ ;
8:   end if
9: end for
△ Step B : △
10:  $\forall n_p \in N, \beta_{n_p} \leftarrow 0$ ;
11: for  $s_i \in optimizable\_SFCs$  do
12:    $P_{s_i} \leftarrow Dijkstra\_shortest\_deployment\_path(s_i)$ ;
13:   for  $n_p \in P_{s_i}$  do
14:      $\beta_{n_p} \leftarrow \beta_{n_p} + (1 - \alpha) * D_{s_i} * V_{cost}$ ;
15:   end for
16: end for
17:  $SteinerPoints \leftarrow Propagate\_centrality(K, \alpha)$ ;
△ Step C : △
18: for  $s_i \in S$  do
19:   if  $s_i \in optimizable\_SFCs$  then
20:      $Result \leftarrow MDP\_deploy(s_i, SteinerPoints, \alpha)$ ;
21:   else
22:      $Result \leftarrow Dijkstra\_shortestPath\_deploy(s_i)$ ;
23:   end if
24: end for
```

Process to find a cost-effective placement and chaining strategy of each SFC request. We reuse the S-VNFIs which have already been deployed on the Steiner points on the conditions that this approach does not result in excessive path elongation and the resource of S-VNFI is adequate. Otherwise, we create new VNF instances on the selected PoP.

A. Find the Shareable VNFs

As mentioned before, the first prerequisite is to make clear the types of VNF instances that can be reused. We point out the types of VNFs that are common to multiple SFCs by traversing all SFC requests. As shown in Fig. 1(b), VNF1 and VNF2 are both requested in SFC1 and SFC2. Hence, the *shareable_VNF_types* contains types 1 and 2. And the *optimizable_SFCs* contains SFC1 and SFC2. The complexity of this part is $\Theta(|S|)$, where $|S|$ is the number of requests.

B. Selection of the Steiner Points

Step B is the key part of the heuristic algorithm. In this paper, we construct the Steiner Tree by adding some additional PoPs from the substrate network graph G to induced subgraph $G[F]$. These PoPs are called Steiner points and used as candidate PoPs to place the VNF instances that are required

by multiple SFCs. In [5], they can not handle the side-effect caused by VNF instance sharing, because they solve the problem based on Minimum Spanning Tree [17]. They only select the candidate deployment PoPs of S-VNFI from the subset F , so the VNF instance cost saving comes at the expense of forwarding cost. For different SFC requests, they need different bandwidth, so the bandwidth cost of each additional hop in the forwarding path is different. It is because the selection of Steiner point is considered from a global perspective that the path elongation can be avoided.

To select Steiner points, we use the Katz Centrality [18] and PageRank algorithm [19], an algorithm of measuring the importance of points. In graph theory, the Katz centrality of PoP is used to figure out the network centrality. It computes the relative influence of a PoP within a network by measuring the influence of the one-hop neighbors (the neighbors which are in the one hop range) and also all other PoPs in the network that connect to the PoP through these one-hop neighbors. Connections made with distant neighbors are penalized by an attenuation factor η . Therefore, the problem is transformed into finding the PoPs with larger Katz centrality, and these PoPs are added as Steiner points to the $G[F]$.

In our heuristic algorithm, each PoP n_p has an initial Katz centrality value β_{n_p} (start with 0). For each PoP n_p , if it belongs to the points in the shortest deployment path of an optimizable SFC and has the sufficient physical resource, then β_{n_p} increases. As shown in equation (7):

$$\beta_{n_p} = \beta_{n_p} + (1 - \alpha) * D_{s_i} * V_{cost} \quad (7)$$

where D_{s_i} is the average CPU resource demand for all VNFs in SFC s_i .

After initializing the centrality, we calculate the propagation of Katz centrality iteratively according to equation (8):

$$C_{Katz}(n_p) = \beta_{n_p} + \sum_{n_j} \left(C_{Katz}(n_j) - \eta * \alpha * DB_{s_i} * L_{cost} \right), \quad \forall n_p \in N, n_j \in \mathcal{N}_p \quad (8)$$

where DB_{s_i} is the average bandwidth demand for all virtual links in SFC s_i , $\mathcal{N}_p$ is the set of the neighbors of n_p in one hop. When distributing centrality to the one-hop neighbors, the centralities derived from different SFCs decrease to 0 based on $\eta * \alpha * DB_{s_i} * L_{cost}$. It means that the larger the bandwidth required by SFC, the more attenuation of $C_{Katz}(n_p)$. Different from the PageRank algorithm, the centrality value of each PoP is completely distributed to its one-hop neighbors, rather than evenly distributed to them. Note that the value received from one-hop neighbor PoPs cannot be redistributed to them to prevent non-convergence.

Finally, we sort all PoPs with sufficient physical resource according to the value of $C_{Katz}(n_p)$ and select the first K PoPs as Steiner points to add to the candidate deployment points set of S-VNFI, where K is the number of types of the shareable VNFs. And the final induced subgraph $G[F]$ is shown in Fig. 2.

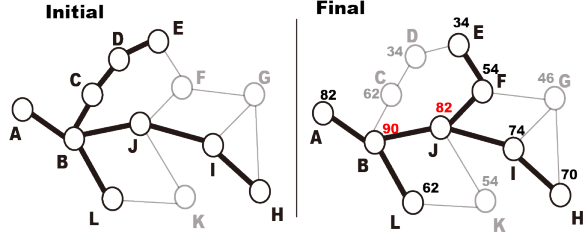


Fig. 2. The final G[F] ($K = 2$, $\alpha = 0.5$, $\eta = 0.2$, $V_{cost} = 200$ and $L_{cost} = 4$)

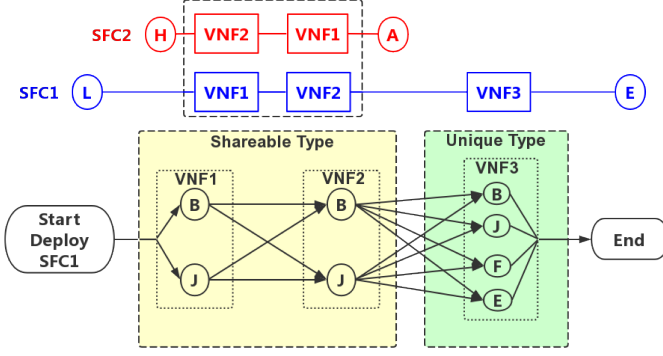


Fig. 3. The procedure of deploying SFC1 with Markov process.

Firstly, step B uses Dijkstra's algorithm to all SFC requests. The complexity is $\Theta(|S| * |N|^2)$, where $|S|$ is the number of requests and $|N|$ is the number of PoPs. Then, we perform a breadth-first traverse (BFS) to calculate the propagation of the centrality. The complexity is $\Theta(x * (|N| + |L|))$, where x is the number of the PoPs with non-zero initial centrality values and $|L|$ is the number of links.

C. Creating and Chaining the VNF Instances by MDP

This step will create VNF instances and chain these instances according to the requirement of SFCs. In this part, the Markov Decision Process (MDP) is used to find a cost-effective placement sequence and ensure each SFC request passes through the specified VNF instance orderly. An MDP model is a 4-tuple (S, A, P_a, R_a) :

- S : Set of states which is composed by candidate PoPs.
- A : Set of actions from s_t (a_i : deploy VNF_i on s_{t+1}).
- $P_a(s_{t+1}|s_t, a)$: State transition probability.
- $R_a(s_t, s_{t+1})$: Transition reward from s_t to s_{t+1} .

As shown in Fig. 3, we represent the deployment of SFC1 as a Markov Process and divide it into two main stages. In the first stage, we deploy the shareable VNFs of this SFC. We create or reuse some S-VNFs to deploy these VNFs. In order to reduce the creation of new instances and avoid path elongation, these S-VNFs can only be deployed at the Steiner points selected by step B. In the second stage, we deploy the unique VNFs of this SFC. We can create the corresponding VNF instances directly on the shortest path of the previous and subsequent required instances or PoPs. We consider each candidate deployment PoP for each VNF in

the SFC as a state, expressed in s_t . s_t is fully connected to each candidate deployment PoP s_{t+1} of the next VNF. The transition probability $P_a(s_{t+1}|s_t, a)$ is inversely proportional to the distance between two candidates PoPs. The transition reward from s_t to s_{t+1} is derived from equation (9) and (10):

$$R_a(s_t, s_{t+1}) = 1 / ((1 - \alpha) * y * V_{cost} + \alpha * Hops_{s_t, s_{t+1}} * DB_{vl_i^j} * L_{cost}); \quad (9)$$

$$y = \begin{cases} 0 & \text{if } s_{t+1} \text{ holds the required VNF instance;} \\ \infty & \text{if } s_{t+1} \text{ dose not have sufficient CPUs or bandwidth;} \\ 1 & \text{otherwise.} \end{cases} \quad (10)$$

where $Hops_{s_t, s_{t+1}}$ is the distance between two candidates PoPs. If a VNF instance of the corresponding type already exists in s_{t+1} and the resource of this instance is adequate, the cost of creating new instances will be saved. Therefore, the reward will increase because the value of denominator decreases. The reward will be infinitely close to zero when the physical resource is insufficient. We choose the state transition path with the largest sum of rewards as the deployment path for the SFC. The complexity of MDP is $\Theta(n^3)$, where n is the number of states.

V. EVALUATION

A. Simulation Environment and Comparison Method

All simulations are implemented on a server with Intel core 4, 3.3 GHz and 8GB RAM. The MILP model formalized in the previous section is implemented by CPLEX Optimization Studio V12.5.1. The proposed heuristic algorithm is implemented and run in Java 1.7. We generate the substrate network topology by GT-ITM tools [20], which is configured to have 20 PoPs, 37 links. The substrate PoPs can hold up to 4 VNF instances as illustrated before, bandwidth capacity on the physical links are 10Gbps and SFC requests are all single chains. Firstly, we compare our heuristic algorithm with the other three schemes: MILP, the graph based algorithm (GBA) [5] and the shortest path deployment algorithm (SP), as shown in Table II. We evaluate them from four aspects: deployment costs, utilization, time complexity and service acceptance rate. Finally, we prove our algorithm can adjust to different deployment preferences.

TABLE II
SCHEMES COMPARISON LIST.

Scheme	Strategy used in the scheme
MILP	The optimal solution of MILP.
GBA	The solution proposed by the graph based algorithm [5].
SP	The solution of the shortest deployment path.
OurHeuristic	The solution given by our heuristic algorithm.

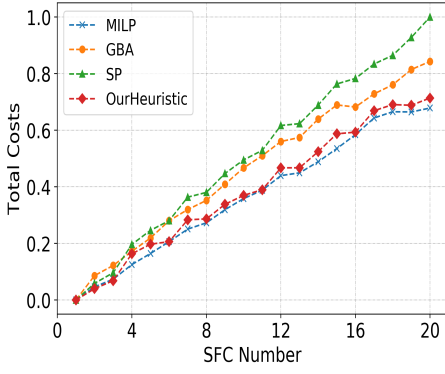


Fig. 4. Total deployment costs.

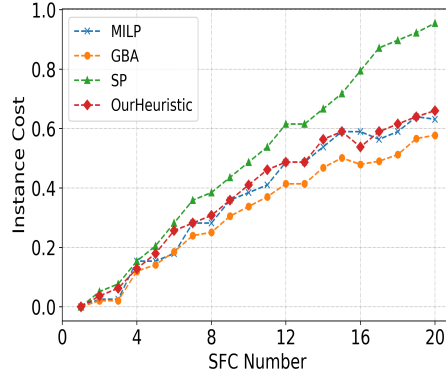


Fig. 5. Instance cost.

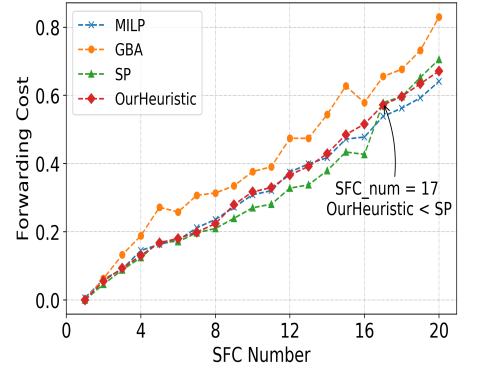


Fig. 6. Forwarding cost.

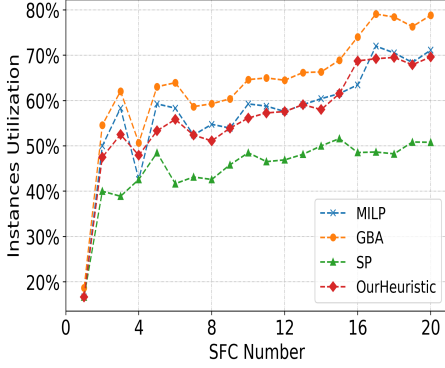


Fig. 7. Utilization.

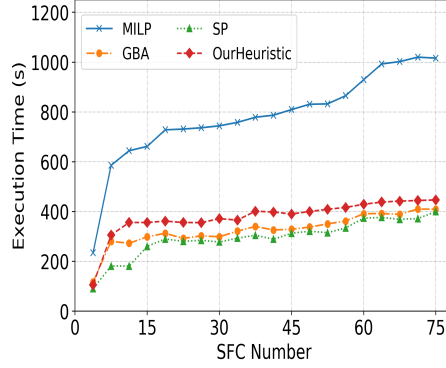


Fig. 8. Time Complexity.

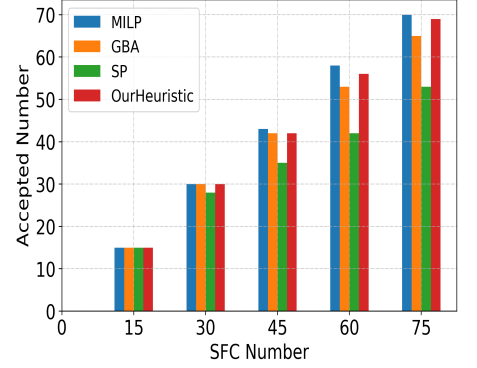


Fig. 9. Accept Number.

B. Evaluation with Current Methods

1). **Deployment costs:** In this part, we compare the total deployment costs, instance cost and forwarding cost of the four algorithms, the $\alpha = 0.596$. To make the comparison clearer, all the results are scaled to $[0, 1]$. As shown in Fig. 4, the total costs of our heuristic algorithm are only 3% higher than the optimal result by MILP, and the total costs of GBA and SP are 14.8% and 34.3% worse than our algorithm. For the instance cost in Fig. 5, GBA has the lowest instance cost, which is 5.6% better than MILP. The difference of the instance cost between our algorithm and MILP is less than 1% and sometimes even better. SP has the worst instance cost. For the forwarding cost in Fig. 6, GBA has the largest forwarding cost, which is 13% higher than MILP. The difference of the forwarding cost between our algorithm and MILP is less than 2.8%. We find that the forwarding cost of SP is not the lowest when the number of SFC requests is large. The reason is that each SFC is greedily deployed on the shortest path and can not share VNF instances. The later deployed SFC have to go a long way to find a PoP with sufficient resource to place VNF instance. Although GBA has the lowest instance cost, the total costs are still high because it cannot avoid path elongation caused by sharing VNF instances. Through the above comparisons, we prove that our heuristic algorithm closely achieves the optimal solution and much better than the other two algorithms.

2). **Utilization:** Fig. 7 shows the utilization of VNF in-

stances of each algorithm. GBA has the highest VNF instance utilization, 7% higher than MILP. The difference of utilization between our algorithm and MILP is no more than 2.9%. Since SP does not share VNF instances, it has the lowest VNF instances utilization, which is less than 50%. Because GBA merges and shares all VNF instances of the same type, and our heuristic algorithm sacrifices some VNF instance cost to avoid excessive growth of forwarding cost, so the resource utilization of GBA is higher than our algorithm.

3). **Time complexity and Service acceptance rate:** In this part, we increase the number of SFC requests from 0 to 75. As shown in Fig. 8 and 4, the execution time of our algorithm is 57.4% lower than that of MILP, and the difference of total deployment costs is no more than 3%. The execution time of GBA and SP algorithms are 8.2% and 9.3% less than ours, but their total costs are 14.8% and 34.3% higher than ours. As shown in Fig. 9, when the number of requests is large, the service acceptance rates of MILP and our algorithm are about 93.2% and 91.3%, while the GBA and SP are 87.3% and 72%. The reason is that SP does not share VNF instances, resulting in a large amount of exclusive CPU resource in idle state. Although GBA shared instances, it can not solve the problem of path elongation, and also lead to the exhaustion of bandwidth. Therefore, our algorithm has much lower complexity than MILP and also has higher service acceptance rate than GBA and SP.

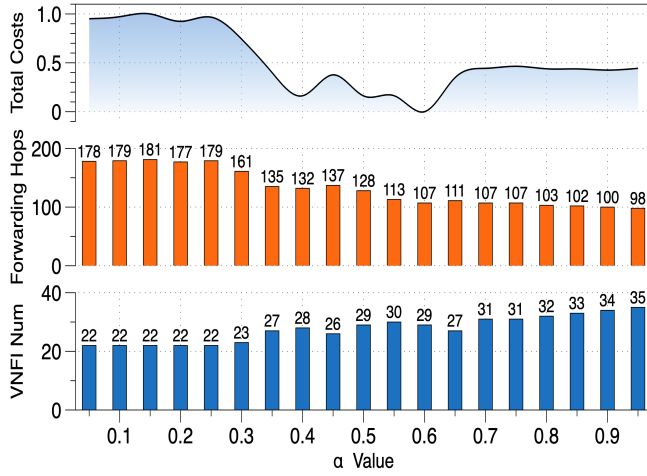


Fig. 10. Different results of scenarios with variant α

C. Adjustment Weight

To observe how the algorithm adjusts to different deployment preferences, we gradually set the dynamic parameter α from 0 to 1. The results evaluated with different α are scaled to $[0, 1]$ to make the fluctuation of the total costs more clear. As Fig. 10 shows, when reducing the instance cost has a higher priority ($\alpha < 0.5$), the algorithm prefers to reduce the number of VNF instances by sharing instances to reduce instance cost. Correspondingly, the length of forwarding paths will be longer. As α increases, the benefit of reducing forwarding cost begins to increase. The algorithm prefers to create new instances at corresponding locations and reduce the path elongation. When $\alpha = 0.65$, although reducing the forwarding cost has a higher priority, the algorithm obviously reduces the number of VNF instances to obtain the optimal result. This proves that our heuristic algorithm will weigh the benefit of reducing the creation of instances with the penalty of path growth and select the optimal solution. In this scenario, we can achieve the global optimal solution by setting the $\alpha = 0.596$.

VI. CONCLUSION

In this paper, we solve the SFCs placement problem based on sharing VNF instance. We focus on decreasing the VNF instance cost and forwarding cost, and eventually find the optimal deployment solution by constructing Steiner Tree and MDP. Through various tests in Section V, we prove that the difference of the deployment costs between our algorithm and the optimal result given by MILP is less than 3%, but our heuristic algorithm is 57.4% faster. Finally, by evaluating the value of α , we prove that our heuristic algorithm can select the optimal deployment solution by balancing the instance cost and forwarding cost based on individual scenarios.

ACKNOWLEDGMENT

This work was supported by the National Key R&D Program of China (2018YFE0205502), and state grid corporation of China headquarters technology project, “the research and

application of the key technology of dynamic resource allocation in power IoT network”.

REFERENCES

- [1] R. Mijumbi, J. Serrat, J.-L. Gorricho, N. Bouten, F. De Turck, and R. Boutaba, “Network function virtualization: State-of-the-art and research challenges,” *IEEE Communications Surveys & Tutorials*, vol. 18, no. 1, pp. 236–262, 2016.
- [2] “ETSI GS NFV-IFA 011 V2.1.1 (2016-10),” ETSI NFV ISG, Tech. Rep., 2016.
- [3] D. Li, P. Hong, K. Xue, and J. Pei, “Virtual Network Function Placement Considering Resource Optimization and SFC Requests in Cloud Data-center,” *IEEE Transactions on Parallel and Distributed Systems*, vol. 29, no. 7, pp. 1664–1677, July 2018.
- [4] Z. Ye, X. Cao, and C. Qiao, “Joint topology design and mapping of service function chains in network function virtualization,” in *2016 IEEE Global Communications Conference*, Dec 2016, pp. 1–6.
- [5] N. Tastevin, M. Obadia, and M. Bouet, “A graph approach to placement of Service Functions Chains,” in *2017 IFIP/IEEE Symposium on Integrated Network and Service Management*, May 2017, pp. 134–141.
- [6] B. Yi, X. Wang, and M. Huang, “A Generalized VNF Sharing Approach for Service Scheduling,” *IEEE Communications Letters*, vol. 22, no. 1, pp. 73–76, Jan 2018.
- [7] M. C. Luizelli, L. R. Bays, L. S. Buriol, M. P. Barcellos, and L. P. Gaspary, “Piecing together the nfv provisioning puzzle: Efficient placement and chaining of virtual network functions,” in *2015 IFIP/IEEE International Symposium on Integrated Network Management (IM)*. IEEE, 2015, pp. 98–106.
- [8] M. F. Bari, S. R. Chowdhury, R. Ahmed, and R. Boutaba, “On orchestrating virtual network functions,” in *2015 11th International Conference on Network and Service Management (CNSM)*. IEEE, 2015, pp. 50–56.
- [9] 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 *2016 12th International Conference on Network and Service Management (CNSM)*. IEEE, 2016, pp. 1–9.
- [10] F. Wang, R. Ling, J. Zhu, and D. Li, “Bandwidth guaranteed virtual network function placement and scaling in datacenter networks,” in *2015 IEEE 34th International Performance Computing and Communications Conference (IPCCC)*. IEEE, 2015, pp. 1–8.
- [11] A. Alleg, T. Ahmed, M. Mosbah, R. Riggio, and R. Boutaba, “Delay-aware vnf placement and chaining based on a flexible resource allocation approach,” in *2017 13th International Conference on Network and Service Management (CNSM)*. IEEE, 2017, pp. 1–7.
- [12] R. Mijumbi, “Placement and scheduling of functions in network function virtualization,” *Computer Science*, 2015.
- [13] L. Qu, C. Assi, and K. Shaban, “Network function virtualization scheduling with transmission delay optimization,” in *NOMS 2016-2016 IEEE/IFIP Network Operations and Management Symposium*. IEEE, 2016, pp. 638–644.
- [14] J. Fan, Z. Ye, C. Guan, X. Gao, K. Ren, and C. Qiao, “Grep: Guaranteeing reliability with enhanced protection in nfv,” in *Proceedings of the 2015 ACM SIGCOMM Workshop on Hot Topics in Middleboxes and Network Function Virtualization*. ACM, 2015, pp. 13–18.
- [15] L. Qu, C. Assi, K. Shaban, and M. J. Khabbaz, “A reliability-aware network service chain provisioning with delay guarantees in nfv-enabled enterprise datacenter networks,” *IEEE Transactions on Network and Service Management*, vol. 14, no. 3, pp. 554–568, 2017.
- [16] B. M. Waxman, “Routing of multipoint connections,” *IEEE Journal on Selected Areas in Communications*, vol. 6, no. 9, pp. 1617–1622, Dec 1988.
- [17] M. Held and R. M. Karp, “The traveling-salesman problem and minimum spanning trees: Part ii,” *Mathematical programming*, vol. 1, no. 1, pp. 6–25, 1971.
- [18] J. M. Fletcher and T. Wennekers, “From structure to activity: Using centrality measures to predict neuronal activity,” *International journal of neural systems*, vol. 28, no. 02, p. 1750013, 2018.
- [19] L. Page, S. Brin, R. Motwani, and T. Winograd, “The pagerank citation ranking: Bringing order to the web,” Stanford InfoLab, Tech. Rep., 1999.
- [20] E. W. Zegura, K. L. Calvert, and S. Bhattacharjee, “How to model an internetwork,” in *Proceedings of IEEE INFOCOM’96. Conference on Computer Communications*, vol. 2. IEEE, 1996, pp. 594–602.