ResearchGate

See discussions, stats, and author profiles for this publication at: https://www.researchgate.net/publication/326569713

# Optimal Service Function Tree Embedding for NFV Enabled Multicast

**Conference Paper** · July 2018

DOI: 10.1109/ICDCS.2018.00023

CITATION
1

READS
25

**5 authors**, including:

Bangbang Ren
National University of Defense Technology
**12** PUBLICATIONS **14** CITATIONS
SEE PROFILE

Deke Guo
National University of Defense Technology
**126** PUBLICATIONS **825** CITATIONS
SEE PROFILE

Guoming Tang
National University of Defense Technology
**34** PUBLICATIONS **93** CITATIONS
SEE PROFILE

**Some of the authors of this publication are also working on these related projects:**

Project (Software-defined Networking)-based Cloud data centers View project

Project Network update View project

All content following this page was uploaded by Bangbang Ren on 03 January 2019.

The user has requested enhancement of the downloaded file.

# Optimal Service Function Tree Embedding for NFV Enabled Multicast

Bangbang Ren[1], Deke Guo[1*], Guoming Tang[1*], Xu Lin[2], Yudong Qin[1]
[1]Science and Technology on Information Systems Engineering Laboratory
National University of Defense Technology, Changsha, Hunan, 410073, China
[2]Xidian University, Xi'an, Shaanxi, 710071, China

***Abstract*— In network traffic engineering, multicast is designed to deliver the same content from a single source to a group of destinations. Recently, NFV enabled multicast has been developed by deploying virtual network functions (VNFs) over the target network. To fulfill the multicast task with a service function chain (SFC) requirement, a service function tree (SFT) embedded in the shared multicast tree has to be built. Given the huge space of SFT embedding solutions, however, it is extremely hard to find the optimal one such that the total traffic delivery cost is minimized. In this paper, we tackle the optimal SFT embedding problem in NFV enabled multicast task. Specifically, we formally define the problem and formulate it with an integer linear programming (ILP), which turns out to be NP-hard. Then, a two-stage algorithm is proposed to deal with the problem with an approximation ratio of $1+\rho$, where $\rho$ is the best approximation ratio of Steiner tree and can be as small as $1.39$. With extensive experimental evaluations, we demonstrate that by applying our SFT embedding solution, the cost saving of multicast traffic delivery can be up to $22.41\%$, compared with the random SFT embedding strategy.**

## I. Introduction

In network traffic engineering, multicast is designed to deliver the same content from a single source node to a group of destination nodes [1], [2]. Compared with unicast, multicast can significantly save the bandwidth consumption and relieve the load of the source server as it avoids duplicated transmissions among independent unicast paths, especially by multiplexing a shared multicast tree. Furthermore, the development of SDN promotes the implementation of multicast. By separating the data plane and control plane of network flows, SDN can help find the optimal multicast tree (e.g., the Steiner tree [3]) with its centralized computation [4], [5].

More recently, with the emergence of network function virtualization (NFV) [6], [7], NFV enabled traffic engineering (for both unicast and multicast) has been investigated accordingly [8], [9]. As the embedded VNFs, which are essentially software applications running on commodity servers, can easily substitute dedicated hardware middleboxes, both the cost and time of service function deployment and migration can be much reduced. For a network flow from the source node to the destination node, it may be processed by multiple VNFs in a particular order, which forms the so-called service function chain (SFC) [10], [11]. For instance, in the NFV enabled email service, the data flow will go through an SFC of virus detection, spam identification and phishing detection [12].

This work was supported in part by National Natural Science Foundation of China under Grant No.61772544, National Basic Research Program (973 program) under Grant No.2014CB347800, and the Research Plan of National University of Defense Technology under Grant ZK17-03-50.
Corresponding authors: Deke Guo, Guoming Tang.

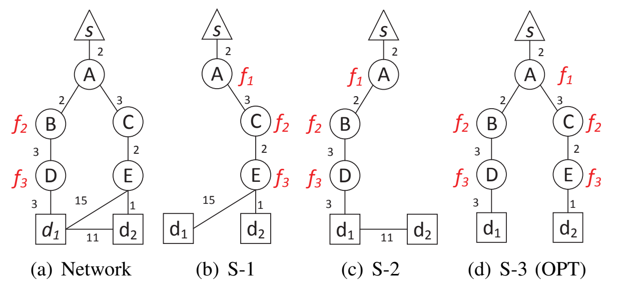


Fig. 1. Three SFT embedding strategies for the same multicast task in a target network, where $s$ is the source, $d_1$ and $d_2$ are the destinations, and $A \sim E$ are server nodes. The link connection cost is labeled beside each edge and the VNF setup cost equals one.

### A. Motivation: from SFC to SFT

For NFV enabled unicast, with only one source and one destination, the SFC is straightforward to deploy, e.g., by sequentially choosing embedding nodes and deploying VNFs along the path [9], [13]. As to the SFC deployment for multicast, however, the problem becomes tricky, especially when the costs of deployment and connection are sensitive to the function location. For example, in the video streaming service, ISPs strategically deploy network functions (e.g., intrusion detection, load balance and format transcoding) among the network nodes, as the costs of links (traversing all the network functions) connecting the source server to geographically distributed users can be much different [14], [15]. Considering the NFV enabled multicast for such tasks, we need to carefully deploy the SFC for the network flows to different destinations, which actually results in a *service function tree* (SFT) embedded in the multicast tree.

Furthermore, with VNFs that have been deployed in a target network, there are multiple choices to deploy the new VNFs and construct the SFT for a particular multicast task. Nevertheless, it is not easy to find the optimal SFT embedding solution with the least cost under this situation. Fig. 1 provides a simple example for SFT embedding over a network with deployed VNFs. Fig. 1(a) shows i) the target network with eight nodes and two deployed VNFs $f_2$ and $f_3$, ii) the multicast task with source node $S$ and destination nodes $\{d_1, d_2\}$, and iii) link connection costs labeled on the links. Assuming that the SFC requirement for the multicast task is $(f_1 \to f_2 \to f_3)$ and the setup cost of each VNF equals one, Fig. 1(b)$\sim$Fig. 1(d) give three SFT embedding solutions with traffic delivery costs[1] of $26, 22$ and 19, respectively. We can see that different SFT embedding strategies can result in diverse traffic delivery cost, and our task is to find the optimal one (as shown by Fig. 1(d) in this example). The detail of this example will be further explained in Section III-A.

### *B. Challenges & Contributions*

The example given in Fig. 1 is simple and may be easy to solve. In realistic, however, the network topology and size of multicast tasks can be much larger and more complex, which leads to a huge space of feasible SFT embedding solutions. Hence, for a specific NFV enabled multicast task, it is a challenging problem to find the most efficient multicast tree where the embedded SFT is proved optimal, considering the link connection cost, VNF setup cost and nodes capacities. Indeed, even though we assume that the required SFC of each traffic flow can be deployed on one node to free the VNF order constraint, the problem is still NP-hard [16].

In this paper, aiming to solve the optimal SFT embedding problem for NFV enabled multicast, we make the following contributions:

- We formally define the problem of optimal SFT embedding in NFV enabled multicast task and formulate the problem by an integer linear programming (ILP), aiming to minimize the traffic delivery cost. We prove that the optimal SFT embedding problem is NP-hard.
- We propose a two-stage algorithm to tackle the NP-hard problem. In the first stage, an initial feasible solution to the ILP problem is produced by embedding an SFC together with a Steiner tree, and in the second stage, the initial solution is optimized by adding new VNF instances to construct an SFT. With sufficient node capacities, we prove that the approximation ratio of our algorithm is $1+\rho$, where $\rho$ is the best approximation ratio of Steiner tree and can be as small as 1.39 [17].
- We extend our algorithm to be compatible to the common situation where a certain number of (virtual) network functions have been deployed, like some public clouds handle base load by physical hardware and spillover load by virtual service instances [18], and our algorithm is able to be aware and reuse these deployed functions in solving the SFT embedding problem.
- We investigate the performance of our algorithm under different parameter settings. With extensive experiments, we demonstrate that the cost saving of traffic delivery by applying our SFT embedding solution can be up to 22.41%, compared with the random strategy.

[1]The traffic delivery cost of a multicast task is computed by the sum of all VNFs' setup cost and link connection cost over the target network [16].

## II. Related work

Researches on NFV enabled multicast and relevant to this work can be roughly grouped into three categories: VNF placement, SFC embedding, and multicast routing.

**VNF Placement.** VNF placement problem focuses on finding the optimal locations of VNFs without considering the service order constraint. Bouet et al. pointed out that deploying VNFs (such as software DPI engines) is costly in terms of license fees and power consumption and thus proposed a heuristic algorithm to optimize the cost of VNF deployment [19]. In [20], Rami et al. investigated the problem of finding the optimal placement of multiple independent VNFs within a physical network, by minimizing the sum of the traffic delivery distance and the VNF setup cost. An NFV enabled multicast tree was proposed in [21] where the VNF placement was optimized with the multicast requirements, and this work can be regarded as a special case of [20] under the constraint of all clients requesting for the same source. The work in [20] concentrated on client requests, which had the need for various VNFs without the chaining requirement. A multicast-oriented NFV tree was proposed in [21], where independent function instances were strategically deployed to optimize the traffic delivery cost without considering the service chaining requirement.

**SFC Embedding.** Compared with VNF placement, the SFC embedding problem is more complicated, as it requires that the traffic flow must traverse a certain number of VNFs in order. Kuo et al. [22] studied the joint optimization problem of VNF placement and path selection, with the consideration of link and server usage. In [10], the SFC was embedded into the target network for the unicast task with multiple objectives, respectively, such as maximizing remaining data rate, minimizing the number of applied nodes or minimizing traffic delivery latency. Different from our work, all the above researches essentially focus on SFC embedding in unicast tasks. Xu et.al. [16] devoted to finding a minimum cost pseudo-multicast with SFC requirement, with an assumption that the SFC is placed on one node (to neglect the order constraint). Nevertheless, the assumption is not practical, especially under the multi-cloud service function chaining architecture [23]. The dynamic SFC embedding problem was studied in [13], [24], where the sequence of functions could change during the life of a session.

**Multicast Routing under SDN.** Multicast traffic engineering has been widely investigated under the SDN diagram. Shen et al. [5] looked into the packet loss recovery problem for reliable multicast routing and applied an approximation algorithm to minimize both tree and recovery costs. Iyer et al. [25] applied SDN aided multicast strategy in data centers and presented an SDN based multicast system named Avalanche that was used to minimize the size of the routing tree.

As the most relevant work to ours, Kuo et al. [26] formulated the SFC embedding problem for multicast with a forest. The key difference to our work is that the forest in [26] is generated because of using multiple sources. The forest is constructed by combining the flow path traversing each SFC from each source. The solution of this problem embeds an SFC for each multicast, while in this work we build an optimal SFT embedded in the shared multicast tree that helps deliver desired traffic flows from the only one source to multiple destinations.

## III. Optimal SFT Embedding Problem

In this section, we first give an illustrative example of the SFT embedding problem for the NFV enabled multicast task. Then we formally define and formulate the problem of optimal SFT embedding, which is the focus of this work.

### A. Problem Illustration

As mentioned in Section I, Fig. 1(a) depicts a network model with link connection cost labeled on each edge. Five server nodes ($A \sim E$) are in the network upon which VNFs can be installed, and two types of VNFs ($f_2$ and $f_3$) have been already deployed as illustrated in the network. For simplicity, we do not show the switch nodes between server nodes. Without loss of generality, we assume that the setup costs of VNF instances upon different nodes are the same and set the value as one [26].

Here we denote the multicast task as $\delta = \{S, D, \ell\}$, in which $S$ is source, $D$ is the destination set with $D = \{d_1, d_2\}$ and $\ell$ represents the SFC requirement for each flow to a destination with $\ell = (f_1 \rightarrow f_2 \rightarrow f_3)$. For this specific multicast request, Fig. 1(b), Fig. 1(c) and Fig. 1(d) provide three different solutions, respectively. In Fig. 1(b), we deploy new instances of $f_1$, $f_2$, $f_3$ on $A, C, E$, respectively, so the traffic delivery cost is $2+3+2+15+1+3 \times 1 = 26$. In Fig. 1(c), we steer flow from $S$ to destinations through $B$ and $D$. Since $f_2$ and $f_3$ have already been deployed on $B$ and $D$, their setup cost is zero, and then the traffic delivery cost is $2+2+3+3+11+1+0 \times 2 = 22$. Particularly, the solution in Fig. 1(d) builds a service function tree by i) reusing the deployed VNFs (on the $A-B-D$ branch) and ii) establishing a new flow path (on the $A-C-E$ branch). With the newly built SFT, the multicast flow avoids expensive links ($(E, d_1)$ and $(d_1, d_2)$), which further cuts down the traffic delivery cost and results in the minimum traffic delivery cost $2+2+3+3+2+3+1+3 \times 1 = 19$.

With the example shown in Fig. 1(d), we formally define the service function tree (SFT) as follow:

**Definition 1.** ***Service Function Tree.*** *Given a target network with or without deployed VNFs and a multicast task, install a number of new VNF instances among the network nodes to fulfill the SFC requirement. This will form a tree structure of service functions embedded in the shared multicast tree, which we name service function tree (SFT).*

Note that the number of VNFs is certain for an SFC, while the number of VNFs for an SFT is not deterministic given the target network and a multicast task, i.e., multiple versions of SFTs can fulfill the multicast task. Under this situation, considering the VNF setup cost, link connection cost and node capacity for VNF deployment, it is meaningful yet challenging to find the optimal SFT embedding solution that leads to the least traffic delivery cost for the multicast task.

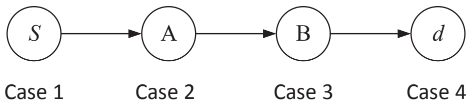


Fig. 2. Four locations of node $u$ in constraint (1e).

### B. Problem Definition

**Target Network.** We consider a software-defined network $G = (V, E)$ with $V = V_M \bigcup V_S$, in which $V_M$ and $V_S$ denote the sets of server nodes and switch nodes, respectively. For a server node $v \in V_M$, $cap(v)$ represents its capacity for VNF deployment and is measured by its owned resources (e.g., CPU, memory and IO). For an edge $e_{uv}$, $c_{uv}$ denotes the link connection cost that each traffic flow goes through it.

**VNF Deployment Constraints.** We use $\Phi = \{f_1, f_2, f_3, ..., f_n\}$ to denote all VNFs that can be deployed in $G$. For an arbitrary VNF, it has a setup cost when deploying it upon the server node in $G$. For those VNF instances that have already been deployed, we treat their setup costs as zeros. A binary number $\pi_{f_n u}$ is used to denote whether $f_n$ has been deployed in node $u$; a real number $\gamma_{f_n u}$ is used to denote the setup cost of a new VNF instance $f_n$ upon node $u$; a real number $\mu_{f_n}$ is applied to represent the resource amount that deploying $f_n$ is needed. For simplicity, we assume that each instance of VNFs can serve traffic flows with any sizes, so that the same VNF instance will not be deployed repeatedly due to flow splitting.

**Definition 2.** ***Multicast Task.*** *Given the target network $G$ with or without deployed VNFs, a multicast task can be represented by a three-element tuple $\delta = (S, D, \ell)$, where $S, D, \ell$ are the source node, destination nodes, and the SFC requirement for each flow, respectively. Specifically, the SFC that each network flow has to traverse can be denoted by $\ell = (l_1, l_2, ..., l_k)$, where $l_k \in \Phi$ and $k \leq n$.*

**Definition 3.** ***Optimal SFT Embedding Problem.*** *For a multicast task $\delta$ given by Definition 2, under the VNF deployment constraints, the optimal SFT embedding problem is to build an embedded service function tree upon the target network such that i) the multicast request is satisfied and ii) the traffic delivery cost is minimized.*

### C. Problem Formulation

To clearly formulate the optimization problem, we define the following variables.

- Binary variable $\omega_{l_j u}$: indicates whether or not a new VNF instance $l_j$ is deployed on node $u$;
- Binary variable $\tau_{d, l_j, u, v}$: denotes if the edge $e_{uv}$ is located between VNFs $l_j$ and $l_{j+1}$ for the traffic flow heading for destination $d$;
- Binary variable $\varphi_{du}^{l_j}$: represents whether the flow heading for destination $d$ is served by VNF $l_j$ on node $u$;
- Binary variable $\psi_{l_j, u, v}$: denotes if edge $e_{uv}$ is located between VNFs $l_j$ and $l_{j+1}$.

Note that the variable $\psi_{l_j, u, v}$ can be derived from the variable $\tau_{d, l_j, u, v}$ and indicates that the traffic flow transmits only one copy in multicast (compared with the duplicated transmission in unicast). We treat $S$ as $l_0$, and with aforementioned variables and parameters, the optimal SFT embedding problem can be formulated as follow.

$$\text{Min} \quad \sum_{l_j \in \ell} \sum_{u \in V_M} \omega_{l_j u} \gamma_{l_j u} + \sum_{l_j \in l_0 \bigcup \ell} \sum_{e_{uv} \in E} \psi_{l_j,u,v} c_{uv} \tag{1a}$$

$$\sum_{u \in V_M} \varphi_{du}^{l_j} = 1, \forall l_j \in \ell, d \in D \tag{1b}$$

$$\varphi_{ds}^{l_0} = 1, \forall d \in D \tag{1c}$$

$$\sum_{l_j \in \ell} (\pi_{l_j u} + \omega_{l_j u}) \mu_{l_j u} \leqslant cap(u), \forall u \in V_M \tag{1d}$$

$$\sum_{v \in N_u} \tau_{d,l_j,u,v} - \sum_{v \in N_u} \tau_{d,l_j,v,u} \geqslant \varphi_{du}^{l_j} - \varphi_{du}^{l_{j+1}},$$
$$\forall l_j \in l_0 \bigcup \ell, u \in V, d \in D \tag{1e}$$

$$\psi_{l_j,u,v} \geqslant \tau_{d,l_j,u,v},$$
$$\forall l_j \in l_0 \bigcup \ell, u \in V, v \in V, u \neq v, d \in D \tag{1f}$$

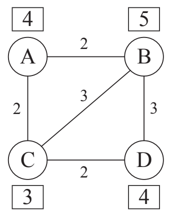
Constraint (1b) ensures that all destinations can get the services of all VNFs in SFC. Constraint (1c) ensures that any $d \in D$ connects to $S$. As each node has a VNF deployment capacity, constraint (1d) ensures that the node is not overloaded. Constraint (1e) ensures that the flow received by the destination completely goes through the required VNFs in order. In this constraint, $N_u$ denotes the neighbor of node $u$.

Constraint (1e) considers both the network flow capacity and the SFC ordering request. For a destination $d \in D$, there must be a walk $Walk(S,d)$ from the source $S$ to the destination $d$, as shown in Fig. 2. All VNFs of the SFC $\ell$ will be embedded along the walk. Since $\ell=(l_1, l_2, ..., l_k)$, we can use $(S=l_0) \bigcup \ell$ to split the $Walk(S,d)$ into $k$ subpaths.

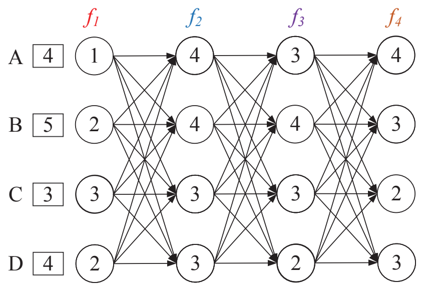
A node $u$ has four kinds of possible positions: source node, intermediate node, VNF node and destination node. Fig. 2 depicts these four cases. For case 1, $l_0$ denotes $S$. Given $l=l_0, u=S, v=A$, if $S$ is also equipped with VNFs, like $l_1$, then $\sum_{v \in N_u} \tau_{d,l_0,u,v}=0$ as the flow output from $S$ is processed by $l_1$. $\sum_{v \in N_u} \tau_{d,l_j,v,u}=0$, $\varphi_{du}^{l_0}=1$ and $\varphi_{du}^{l_1}=1$. Thus, we have $0-0 \geq 1-1$. If $S$ only acts as the source node, then $\tau_{d,l_0,u,v}=1$,$\sum_{v \in N_u} \tau_{d,l_0,v,u}=0$, $\varphi_{du}^{l_0}=1$ and $\varphi_{du}^{l_1}=0$, and thus we have $1-0 \geq 1-0$. Similarly, The situations for case 2, case 3 and case 4 also be proved to satisfy constraint (1e).

Note that some edges may be visited multiple times under an SFC requirement, while the data flow for each visit is different, as the flow needs to be processed by different functions. On the other hand, multicast has a characteristic that in each edge, there is only one copy of the same transmitted flow. Similarly, in NFV enabled multicast, if an edge lies on the same subpath for different destinations, this edge will be just counted once. For instance, in Fig. 1(d), $\tau_{d_1,l_0,S,A}=1$ and $\tau_{d_2,l_0,S,A}=1$, and we just need transfer one flow copy, since the flow from $S$ is the same. Constraint (1f) indicates that $\psi_{l_j,u,v}=1$ if $\tau_{d,l_j,u,v}=1$ for any $d \in D$.

**Theorem 1.** *The optimal SFT embedding problem formulated in (1) is NP-hard.*



(a) Original network $G$ with link cost, node capacity.



(b) Multilevel overlay directed network $G'$

Fig. 3. An example of MOD network with $\ell=(f_1 \rightarrow f_2 \rightarrow f_3 \rightarrow f_4)$.

*Proof:* We prove the theorem by a polynomial-time reduction from the Steiner tree problem. Assume there is a graph $G=(V,E)$ where $V$ and $E$ are the node set and edge set, respectively. For each edge $e \in E$, it has an edge cost of $c_e$. Given a subset of the node set $D \subseteq V$, Steiner tree problem is to find a minimum cost tree $OPT_G$ spanning all nodes in $D$. Then, we construct an instance of embedding SFT as follows.

We first replicate $G$ into $G'$, $D$ into $D'$. Besides $G'$, consider another connected graph with the node set $P=\{p_0, p_1, ...p_n\}$ and each edge in the graph has a cost. Then we connect $d_j \in D'$ in $G'$ to $p_i \in P$ and assign the edge a random cost. Also, $p_0$ can act as the source node, and other nodes in $P$ can act as servers where VNFs can be deployed. All nodes in $G'$ can only act as intermediate nodes or destination nodes. We assume that some VNFs have already been deployed in $P$. Each node in $P$ has a capacity constraint and deploying new VNF instances in $P$ has a cost.

Consider a multicast task $\delta=(S=p_0, D=D', \ell)$ in $G' \bigcup P$. Assume that we can find the optimal solution $OPT_{G'}$ for SFT embedding. Since all VNFs can be deployed only in $P$, then we can delete $P$ and the edge between $P$ and $G'$. The remainder subgraph of $OPT_{G'}$ in $G'$ is the $OPT_G$ for $G$. Otherwise, $OPT_{G'}$ is not the optimal solution. This means that we can find the optimal solution for the Steiner tree problem. Nevertheless, the Steiner tree problem is NP-hard [27]. Thus, the assumption is not true and Theorem 1 is proved. ■

## IV. Methodology: A Two-Stage Algorithm

In this section, we propose a two-stage approximation algorithm to solve the optimal SFT embedding problem. As the preparation of the algorithm, we first construct a multilevel overlay directed (MOD) network which includes all information of the original network.

## A. MOD Network Construction

As the preliminary, we transform the target network to a multilevel overlay directed (MOD) network, which contains all information of the original network, including the link connection cost and VNF setup cost. Fig. 3(a) shows an original network with 4 nodes. The weight attached to each edge denotes the link connection cost and the number near the node represents the node capacity. The deployment costs of these four functions on the four nodes are different, and a matrix can be used to denote them as shown in Equation 2.

$$deployment\_cost = \begin{array}{c} \\ A \\ B \\ C \\ D \end{array} \begin{array}{c} \begin{array}{cccc} f_1 & f_2 & f_3 & f_4 \end{array} \\ \begin{pmatrix} 1 & 4 & 3 & 4 \\ 2 & 4 & 4 & 3 \\ 3 & 3 & 3 & 2 \\ 2 & 3 & 2 & 3 \end{pmatrix} \end{array} \quad (2)$$

Fig. 3(b) illustrates a MOD network, which is transformed from the target network consisting of $4 \times 4$ nodes as shown in Fig. 3(a). The MOD network can be partitioned by columns and rows, in which each row denotes one node and each column denotes one VNF. Specifically, the order of column is consistent with the order of the SFC. The weight attached to each node represents the deployment cost of the corresponding VNF on the node. For instance, the weight attached to the node lies in the first column, and the first row represents that the cost of deploying $f_1$ on $A$ is 1. All nodes in the left column connect to all nodes in its right neighbor column with directed edges, and the costs of these edges are the same with corresponding shortest paths in the original network [28]. Usually, there are three steps to get a MOD network.

- Step 1: Replicate all nodes of the network $k$ times, where $k$ denotes the length of the SFC. Arrange these nodes in a matrix form, in which columns denote VNF functions and rows denote nodes.
- Step 2: For each column, connect all nodes to all their right neighbors with arrows.
- Step 3: Set the node weights equal to the corresponding VNF setup costs. Set the edge weights equal to corresponding shortest path costs in $G$.

The above three steps can transform any target network with certain SFC requirements into a MOD network. The procedure of constructing a MOD network ensures that the original network is a subgraph of the overlay network, which guarantees that no information will be lost. Algorithm 1 depicts the procedure in detail.

**Algorithm 1** The construction of multilevel overlay directed network

```
Input A network $G$=$(V,E,c(E))$, service function chain $\ell$=$\{l_1,l_2,...,l_k\}$.
Output Multilevel overlay directed network $G'$.
1: Calculate all shortest paths between each pair nodes in $G$
2: Replicate all $|V|$=$n$ nodes $k$ times and place $n \times k$ nodes in a $n \times k$ grid. Each node can be denoted by $v_{nk}$
3: for $i$=1; $i++$; $i \leq k-1$ do
4:     Connect all nodes in $i$-th column to all nodes in $i$+1-th column with directed arrows
5:     Set the edge cost between two nodes as the corresponding shortest path cost in $G$
6: Set the node weight equal to the corresponding function deployment cost in $G$.
```

## B. Stage One: Find a Feasible Solution

In this stage, we first design an algorithm for the problem under the condition that each node has sufficient resources, and then generalize it to common situations. To find a feasible solution for the optimal SFT embedding problem, we first embed the SFC into the transformed MOD network, and then connect the last node of the SFC to all destinations.

Fig. 4 shows an expanded MOD network based on the network in Fig. 3(b). As shown in the figure, each node in $G'$ is expended into two identical nodes connected by a virtual link with connection cost equal to the VNF setup cost. Based on the expanded MOD network, the detailed operations to find a feasible solution are as follow.

- Step 1: add the source node $S$ into the expanded MOD network, by connecting it to all nodes in the first column and setting each link connection cost as that of the corresponding shortest path in original network $G$.
- Step 2: from the source node $S$ to one node in the last column (where the last VNF is deployed), find a path to embed the required SFC, using the shortest (weighted) path search algorithm (e.g., Dijkstra algorithm). We will prove that such a path is with the least traffic delivery cost.
- Step 3: For the node deployed the last VNF, find a Steiner tree connecting it to all destination nodes. Thus, the Steiner tree along with the path resulting in Step 2 yields a feasible solution for the SFT embedding problem.

**Theorem 2.** *In Step 2, for the selected node in the last column of the expanded MOD network, the (weighted) shortest path is optimal for SFC embedding.*

*Proof:* We assume that the selected node deploying the last VNF is $v_t$. Indeed, any one solution of embedding the SFC in original network $G$ can be mapped into a path in the expanded MOD network. The cost of embedding an SFC can be treated as the sum of two parts, i.e., the VNF setup cost and the link connection cost. The VNF setup cost can be mapped to the virtual edge weights and the link connection cost between neighbor VNFs can be mapped to the real edge weights between neighbor columns. The condition that nodes have sufficient resources ensures that any path from $S$ to $v_t$ can be used to embed the SFC, and the shortest path computed by our algorithm in the expanded MOD network provides the optimal SFC embedding. Thus, Theorem 2 is proved. ■

Nevertheless, if the assumption that each node has sufficient resources is not held, the path generated by the Step 2 with the least traffic delivery cost may be infeasible in practice. The reason is that some node along the path may be overloaded. Therefore, for the obtained solution, we further check if there exists the overloading problem. If so, corresponding node

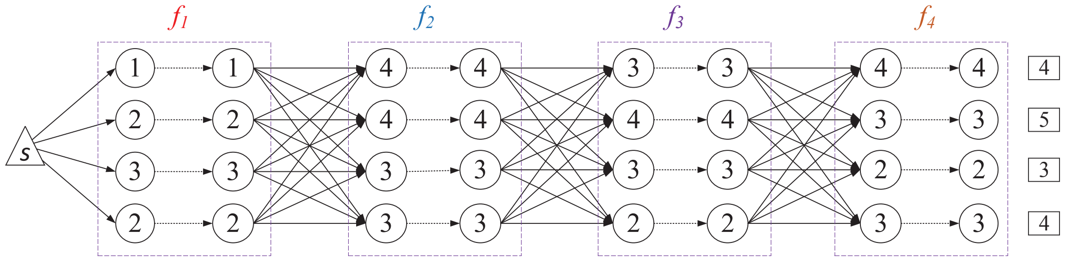


Fig. 4. The expanded MOD network.

adjustment will be performed as follow.

We check all VNFs of the SFC in order. If one VNF has been deployed in an overloaded node, then we should find another node to deploy it. Without loss of generality, we assume VNF $l_i$ has been deployed on node $v_j$ that is overloaded, and its neighbor VNFs $l_{i-1}$ and $l_{i+1}$ have been deployed on nodes $v_k$ and $v_m$, respectively. To find a new node to deploy VNF $l_i$, we check all the other nodes in the same column of the MOD network. For each node with enough capacity, we compute the sum of i) its link connection cost to $v_k$, ii) its link cost to $v_m$, and iii) its VNF setup cost. Thus, we choose the node with the minimum cost as the deployment place for $l_i$.

In Step 3, the node with the last VNF makes a big impact on the final solution, as the different locations of the node will lead to a different Steiner tree. Therefore, we consider all the cases for the choices of the node with last VNF and select the solution with the minimum cost as the output of the first stage.

**Theorem 3.** *The solution resulting from stage one is a feasible solution to the SFT embedding problem.*

*Proof:* In the problem of embedding SFT for NFV enabled multicast $\delta=(S,D,\ell)$, the feasible solution requires that the flow traverses all VNFs of $\ell$ in order. In stage one, we first embed an SFC and then connect the node deploying the last VNF to all destinations. This step ensures that flow is steered from the first VNF to the last VNF in order and then goes to the destination. Thus, Theorem 3 is proved. ■

Overall, stage one can be depicted by the modified shortest-path algorithm (MSA) as shown in Algorithm 2.

### C. Stage Two: Optimize the Feasible Solution

For the feasible solution in stage one, it only contains an embedded SFC. As we have mentioned in Section III-A, embedding an SFT for the multicast task can outperform embedding an SFC. Therefore, in the second stage, we transform the obtained SFC in the first stage to an SFT by adding new VNF instances and eliminating links with expensive cost.

Fig. 5 depicts an example of SFT. In such an SFT, we call $f_i$ the predecessor VNF of $f_j$ if $f_i$ takes effect before $f_j$, and correspondingly, $f_j$ is the successor VNF of $f_i$. In Fig. 5, it can be observed that the number of predecessor VNFs is smaller than that of successor VNFs, and we can prove that this is necessary for an SFT with Theorem 4.

**Algorithm 2** The Modified Shortest-Path Algorithm (MSA)

```
Input An overlay network $G'$, service function chain $\ell=\{l_1,l_2,...,l_k\}$, and VNFs deployment cost on all nodes.
Output A solution with an SFC and a Steiner tree.
 1: Add the source node $S$ into $G'$.
 2: Connect $S$ to all nodes of the first column in $G'$ and set the cost as corresponding shortest path.
 3: Divide each nodes in $G'$ into two parts, and connect these two parts with cost $\gamma_{l_k u}$.
 4: for each node $v$ of the last column in $G'$ do
 5:     Find the shortest path $\Upsilon$ from $S$ to $v$ in $G'$ and get $[\omega_{l_n u}]$.
 6:     Build a Steiner tree to cover $v$ and all destinations.
 7:     for $j=0;j++;j\le n$ do
 8:         Check whether $l_j$ is deployed in an overloaded node, if so, find a new node with the minimum sum of setup cost and connection cost to deploy $l_j$.
 9:         Update $[\omega_{l_n u}],\Upsilon$ and get the total cost of the solution.
10: Output the solution with the minimum cost.
```

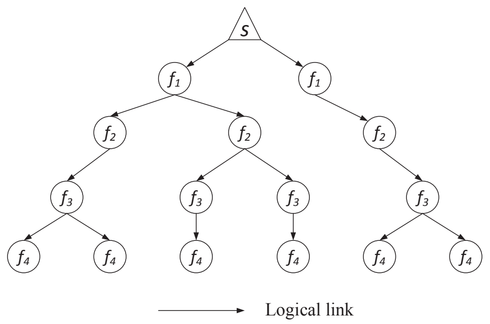


Fig. 5. An example of SFT with the SFC requirement ($f_1 \to f_2 \to f_3 \to f_4$). Note that the links connecting VNFs are logical.

**Theorem 4.** *In an SFT, the number of predecessor VNFs is smaller than that of successor VNFs.*

*Proof:* In the SFT, predecessor VNFs are parent nodes of successor VNFs. All leaf nodes in the SFT must be the instances of last VNF. This property ensures that the predecessor VNFs in the SFT must have children nodes, otherwise these instances will not be contained by the SFT due to the minimum cost constraint. Since each parent node has at least one children node, predecessor VNFs have fewer instances than successor VNFs. Thus, Theorem 4 is proved. ■

To describe stage two clearly, we provide an exam-

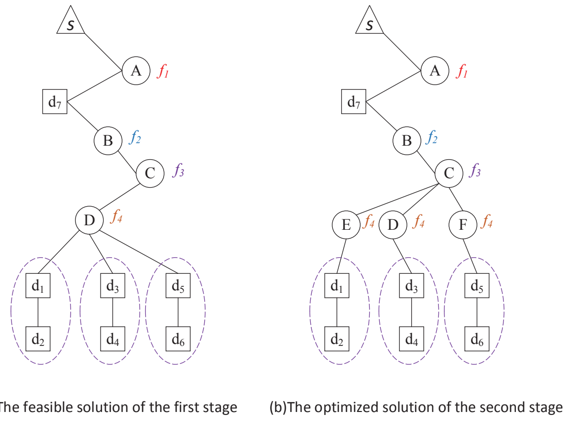

S
A $f_1$
$d_7$
B $f_2$
C $f_3$
D $f_4$
E $f_4$
F $f_4$
$d_1$
$d_2$
$d_3$
$d_4$
$d_5$
$d_6$
(a)The feasible solution of the first stage
(b)The optimized solution of the second stage

Fig. 6. An example of the second stage. All links are logical

ple to demonstrate the procedure. Suppose that there is a multicast task $\delta=\{S,D,(f_1 \rightarrow f_2 \rightarrow f_3 \rightarrow f_4)\}$, and $D=\{d_1,d_2,d_3,d_4,d_5,d_6,d_7\}$. Fig. 6(a) shows a feasible solution produced in stage one. In stage two, we optimize the feasible solution additionally by transforming the SFC into an SFT.

In the last step of the first stage, a Steiner tree is found to connect all destinations with the last VNF. As shown in Fig. 6(a), the Steiner tree connect the destinations $\{d_1,d_2,d_3,d_4,d_5,d_6,d_7\}$ and the last VNF $f_4$. From node $D$ that deploys the last VNF, there are four paths for the multicast flow to all destinations (i.e., leaves of the Steiner tree). These four paths can be divided into two categories based on whether they have common edges with the embedded SFC. In Fig. 6(a), the path $(D \rightarrow d_7)$ overlaps with the SFC, and we call such a path **dependent path**. Otherwise, like paths $(D \rightarrow d_2)$, $D \rightarrow d_4$ and $(D \rightarrow d_6)$, which have no common edges with the SFC, we call them **independent paths**. For an independent path, it may contain multiple destinations and we define the one nearest to the source as its **connection node**. In Fig. 6(b), the connection nodes are $d_1,d_3$ and $d_5$.

After identifying the dependent/independent paths as well as connection nodes, we can further optimize the solution. Since Theorem 4 shows that predecessor VNFs cannot have more instances than successor VNFs, we increase the number of VNF instances in an inverted order. Thus, as illustrated by Fig. 6(b), we further deploy new instances of $f_4$ in different nodes.

*Rule of Adding New VNF Instances:* In Fig. 6(a), each independent path gets service of $f_3$ and $f_4$ from $C$ and $D$. While in Fig. 6(b), $f_4$ can be deployed on $\{D,E,F\}$. Denote the cost of the shortest path between two nodes $v_i$ and $v_j$ as $c_{v_iv_j}$. If we could find a node $E$ such that $c_{d_1E}+c_{EC}+\gamma_{f_4E}<c_{d_1D}$, we can deploy a new instance of $f_4$ on $E$. Similarly, if $c_{d_5F}+c_{FC}+\gamma_{f_4F}<c_{d_1D}$, we can deploy a new instance of $f_4$ on $F$. After adding instances of $f_4$, all nodes with $f_4$ become the new connection nodes of each dependent path. Repeat the above procedures until one VNF cannot be deployed on multiple nodes. Theorem 4 ensures that this terminate condition is valid. The details of the second stage can be formally depicted by Algorithm 3.

**Algorithm 3** The Optimize Phase Algorithm (OPA)

```
Input A feasible solution produced by Algorithm 2 $X_0$, service function chain $\ell=\{l_1,l_2,...,l_k\}$, network $G$, all nodes capacity and VNFs deployment cost.
Output Optimized solution $X_p$.
1: Find all connection nodes in $X_0$.
2: for $j=k;j--;j>0$ do
3:     if $l_j$ can be deployed in other nodes according to the rule in Section IV-C then
4:         Deploy new instances of $l_j$ in corresponding node, update connection nodes and $[\omega_{f_nu}],\Upsilon$
5:     else
6:         break
7: return $X_p=\{[\omega_{f_nu}],\Upsilon\}$.
```

## D. Algorithm for Network with Deployed VNFs

For a target network, some VNFs may have already been deployed. We modify Algorithm 1 to tackle the SFT embedding problem in such a network.

Indeed, when constructing the MOD network, those VNFs that have already been deployed in the network can be divided into two categories. For those VNFs in the SFC $\ell$, we treat their setup cost as zero, since there is no need to deploy them again for a single multicast task. For those VNFs not in the SFC $\ell$, they will not occupy a column in the MOD network according to Algorithm 1. Thus, we only need modify the capacities of those nodes where VNFs have already been deployed. Then Algorithm 2 and Algorithm 3 can be applied to the modified MOD network to get the solution.

## E. Algorithm Analysis

**Theorem 5.** *The computational complexity of the two-stage algorithm is $O((k^2+|D|)\times|V|^3)$, in which $|V|$ is the number of nodes in the target network, $|D|$ is the number of destinations and k is the length of the required SFC for each traffic flow in the multicast task.*

*Proof:* In the procedures of constructing the MOD network, we first need to find the shortest path between each pair of nodes in the original network. Floyd algorithm can find all shortest paths in $O(|V|^3)$ [28]. The overlay network has $k\times|V|^2$ edges and $k \ll |V|$, thus the construction of overlay network needs $O(|V|^3)+O(k\times|V|^2)=O(|V|^3)$ steps. In the first stage, we first need to find the shortest path between $S$ and $T$, and this path can be found by Dijkstra algorithm in $O(k^2\times|V|^2)$ [29]. As for the node violates the capacity constraint, corresponding adjustment procedures need $O(k\times|V|)$ comparisons. The time complexity of finding the Steiner tree in the original network is $O(|D|\times|V|^2)$ [3]. One iteration in the first stage needs $O(k^2\times|V|^2)+O(k\times|V|)+O(|D|\times|V|^2)=O((k^2+|D|)\times|V|^2)$. Thus, the computational complexity of the first stage is $O((k^2+|D|)\times|V|^3)$. In the second stage, the computational complexity of the feasible solution optimizing depends on the number of dependent paths and the length of SFC, which needs $O(k\times|D|)$ comparisons. Therefore, the computational complexity of all

TABLE I
PARAMETER SETTINGS.

| | Parameter | Setting |
|---|---|---|
| Target Network | Network size | [50, 250] |
| | Deployed VNF | Deploy randomly |
| | Node Capacity | [1, 5] |
| | Link Connection Cost | The Euclidean distance |
| | VNF Deployment Cost | Relate to the connection cost |
| Multicast Task | $S$ | Select randomly |
| | $D$ | Select randomly |
| | SFC length | [5, 25] |

the above operations is $O(|V|^3)+O((k^2+|D|)\times|V|^3)+O(k\times|D|)=O((k^2+|D|)\times|V|^3)$. Thus, Theorem 5 is proved. ∎

**Theorem 6.** *With sufficient resources on each node, the two-stage algorithm guarantees an approximation ratio smaller than* $(1+\rho)$*, where* $\rho$ *is the best approximation ratio of Steiner tree and can be as small as* 1.39 *[17].*

*Proof:* $X_{opt}$ represents the optimal solution of embedding SFT for multicast and there will be an SFT in $X_{opt}$. $X_{alg}$ is the solution generated by our algorithm, and it also has an SFT. Indeed, $X_{alg}$ is generated after execute the second stage algorithm. In the first stage of our algorithm, a feasible solution $X'_{alg}$ has been generated which has embed an SFC. Since $X_{alg}$ is the solution optimized on the basis of $X'_{alg}$, then $c(X_{alg})<c(X'_{alg})$. $X'_{alg}$ is composed of an SFC $P_{alg}$ and a Steiner tree $T_{alg}$. $X_{opt}$ has an SFT, and we can find an SFC $P_{opt}$ in the SFT and a tree $T_{opt}$ connecting all destinations. Assume the last node of $P_{opt}$ is $W$, and the optimal Steiner tree covering all nodes in $W\cup D$ is $T_{W\cup D}$. There is no doubt that $c(P_{opt})<c(X_{opt})$ and $c(T_{W\cup D})<c(T_{opt})<c(X_{opt})$. In Algorithm 2, node $W$ has been considered, and the solution can be denoted as the SFC $P_{W-alg}$ add the $T_{W\cup D}$. According to Algorithm 2, $c(P_{alg})+c(T_{alg})<c(P_{W-alg})+c(T_{W\cup D})$.

Theorem 2 has proved that $P$ is the optimal SFC when all nodes have sufficient resources, thus $c(P_{W-alg})<c(P_{opt})<c(X_{opt})$. The best approximation ratio of the Steiner tree can be as small as 1.39 [17]. However, in this proof, we do not concentrate the concrete value, and we use $\rho$ denotes the approximation ratio. And we can get, $c(T_{W\cup D})<\rho c(T_{opt})<\rho c(X_{opt})$, since $T_{opt}$ is a feasible tree to cover all destinations and $W$. Then $c(P_{W-alg})+c(T_{W\cup D})<c(X_{opt})+\rho c(X_{opt})$, $c(X_{alg})<c(X'_{alg})=c(P_{alg})+c(T_{alg})<(1+\rho)c(X_{opt})$. Thus, Theorem 6 is proved. ∎

## V. EXPERIMENTAL EVALUATION

In this section, we first introduce the evaluation environment and methodology, and then evaluate the performance of the algorithm. Our evaluations focus on quantifying the benefits of our method at delivery cost and running time reductions.

### A. Experiment Design

***Experiment Configurations.*** To evaluate our method, we first generate synthetic target networks using the ER random graphs [31]. Then, we make use of the real-world network of PalmettoNet shown in Fig. 7, which is a backbone network

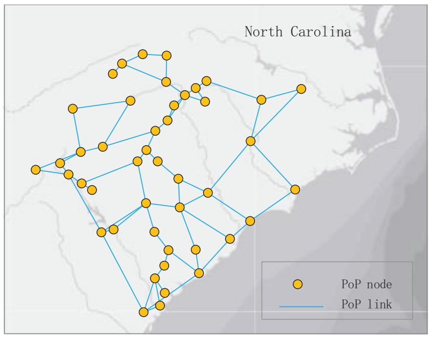
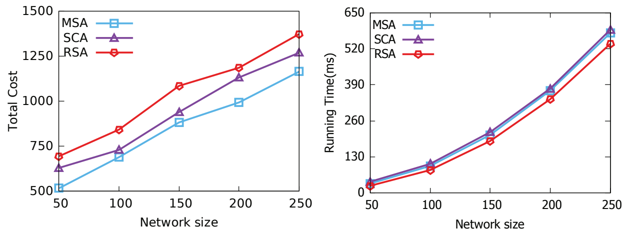


Fig. 7. The Palmetto network [30].

with 45 nodes in the U.S. [30]. With the synthetic and real-world networks, we evaluate the impacts of the multicast size (i.e., the number of destinations), the average VNF setup cost and the length of SFC (the number of VNFs). It is worth to mention that the link connection cost and VNF setup cost are prior knowledge and can be set according to the network dynamics. For example, if one link is congested or one server node cannot support VNF deployment, we can considerably increase their costs such that the embedding of an SFT would not select such kinds of links. Table I summarizes the parameter settings with the following considerations:

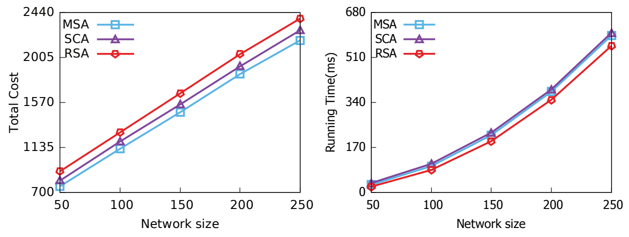
- The network size: Following the previous work [16], we set the network size in the range of $[50,250]$, which spans small networks to large ones.
- The deployed VNFs: According to [32], there are many value-added (VNF) services as the NFV market is growing. We arbitrarily select thirty different VNFs and deploy some of them randomly among the server nodes (as the existed VNFs).
- The node capacity: To ease the evaluation, we set the node capacity randomly within $[1,5]$, which means at most $1\sim 5$ VNFs can be deployed on the node.
- The link connection cost: To ease the evaluation, we set the link connection cost of any pair of nodes as the Euclidean distance between them.
- The VNF deployment cost: we set the VNF deployment cost upon any server node following the Normal distribution $N(\mu l_G,\sigma^2)$, where $\mu\in\{1,3\}$, $\sigma=l_G/4$, and $l_G$ is the average shortest path cost of the network (to balance the costs of link connection and VNF deployment).
- The size of a multicast task: For each simulated multicast task, the source node and the destination nodes are selected randomly from the target network. Furthermore, we set the value of $|D|/|V|$ equals to 0.1 or 0.3 for evaluating the impact of size [16].
- The SFC length: According to the white paper [32], many network elements can be replaced with VNFs and incorporated into SFCs. Thus, as the NFV market grows, the SFC length depends on the flow types and would be increased. To evaluate the impact of the SFC length thoroughly, we change the SFC length from 5 to 25.

***Benchmark Algorithms.*** To the best of our knowledge, this is the first work tackling the SFT embedding problem (refer to



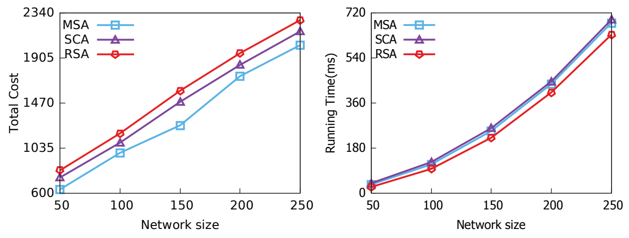
(a) The traffic delivery cost (b) The running time

Fig. 8. The changing trends of two metrics when $|D|/|V|$=0.1



(a) The traffic delivery cost (b) The running time

Fig. 9. The changing trends of two metrics when $|D|/|V|$=0.3



(a) The traffic delivery cost (b) The running time

Fig. 10. The changing trends of two metrics when the average setup cost is $1\times$ the average shortest path cost.

the related work). We then compare our algorithm (i.e., MSA) with another two benchmarks we designed: the minimum set cover algorithm (SCA) and the randomly selecting algorithm (RSA). These three algorithms are different in generating the feasible solution at the first stage, while the optimization procedure at the second stage is the same. SCA tries to occupy as few nodes as possible when embedding the SFC in the first stage. It chooses the minimum number of nodes to cover as many VNFs as possible. If some VNF has no existed instance in the network, SCA will deploy a new instance upon the nearest node to the predecessor VNF. As for RSA, it randomly selects VNFs that have been deployed. While for those VNFs that have not been deployed, RSA randomly selects nodes with sufficient capacities to deploy them. After all requested VNFs having been deployed, RSA connects them in order with the shortest paths. With various parameter settings, we compare our algorithm with the two benchmarks using the performance metrics of traffic delivery cost and algorithm running time. Furthermore, to evaluate the accuracy of approximated solutions, we perform experiments over the Palmetto network, as the network size is suitable to obtain the optimal solutions.

## B. Evaluation with Synthetic Networks

*1) The impact of destination ratio:* We define the destination ratio as the number of destinations in the network to the total number of network nodes, i.e., $|D|/|V|$. To concentrate on the impact of destination ratio under different network size, we fix the values of other variables (e.g., SFC length is set to 5 and the average VNF deployment cost $\mu = 2$). Fig. 8 and Fig. 9 show the variations of traffic delivery cost and running time as the growth of network size under different destination ratios. In average, the traffic delivery costs resulting from MSA are 12% and 19% lower than those from RSA shown in Fig. 8(a) and Fig. 9(a), respectively. We can see that the traffic delivery cost of the solution increases as the growth of network size. This is because as the network size increases, the final solution includes more links (incurring more link connection cost). Also, the number of destinations increases as the network size increases, which leads to the increases of both connection links and VNF instances. Thus, comparing Fig. 8(a) with Fig. 9(a), we can find that the solutions in the latter figure result in higher traffic delivery costs.

The curves representing the running times from the algorithms are illustrated in Fig. 8(b) and Fig. 9(b). The figures show the same trend of changing, both of which increase gradually. This is caused by the increase of computational complexity in constructing the Steiner tree. As the number of destinations and network size increase, the consumed time increases in nonlinear. Through the comparison of Fig. 8(b) and Fig. 9(b), we find that the running times from MSA and SCA are almost the same. This is because both of them construct the same Steiner tree which consumes the biggest portion of the running time.

*2) The impact of setup cost:* To evaluate the impact of VNF setup cost under different network size, we fix the value of destination ratio to 0.2 (i.e., $|D|/|V|$=0.2) and the length of SFC to 5. The curve trends shown in this two figures are almost the same with those in Fig. 8 and Fig. 9. The traffic delivery cost of the solution increases as the network size grows. Compared with RSA, MSA reduces the traffic delivery cost by 15.02% and 14.47% (under different VNF setup costs) in Fig. 10(a) and Fig. 11(a), respectively. Particularly, the cost reduction is up to 22.41% in Fig. 10(a). Comparing Fig. 10(a) with Fig. 11(a), we can see that the traffic delivery cost in latter is higher than that of former, as higher VNF setup cost contributes more to the traffic delivery cost.

As to the running time, according to Fig. 10(b) and Fig. 11(b), it is not obviously affected by the average setup cost. This aligns with what is conveyed in Theorem 5.

*3) The impact of SFC length:* To evaluate the impact of SFC length, we fix the value of destination ratio to 0.2 (i.e., $|D|/|V|$=0.2), the average deployment cost to 3 (i.e., $\mu$=3), and the network size to 200 (i.e., $|V|$=200). The results are shown in Fig. 12(a). We can see that the three algorithms have different performances with different traffic delivery costs. The performance gap between the curves of RSA and MSA increases along with the growth of SFC length, and the average

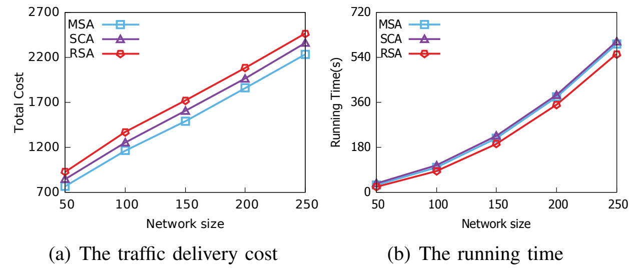


Fig. 11. The changing trends of two metrics when the average setup cost is 3× the average shortest path cost

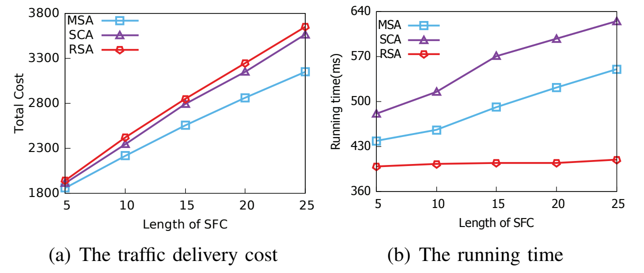


Fig. 12. The changing trends of two metrics with different SFC lengths.

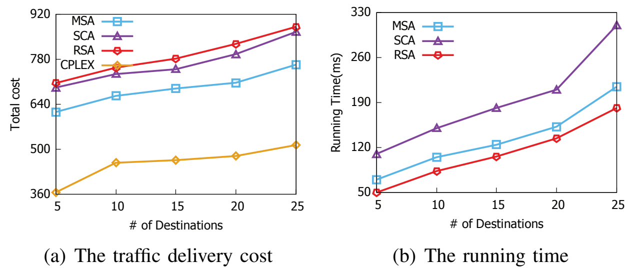


Fig. 13. The changing trends of two metrics in PalmettoNet when the numbers of destinations are different.

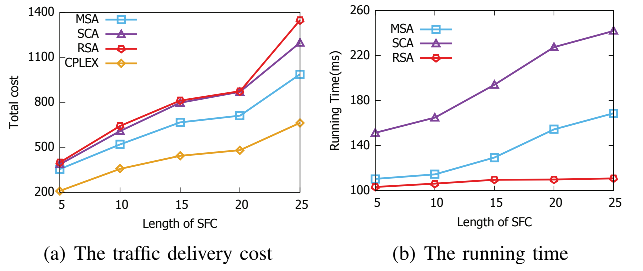


Fig. 14. The changing trends of two metrics in PalmettoNet with different SFC lengths.

cost reduction with MSA is 9.74% compared with RSA. This is because more VNFs provide more opportunities to inherit deployed VNFs, and reusing deployed VNFs and embedding SFT (rather than SFC) with our MSA can effectively reduce the traffic delivery cost.

Fig. 12(b) depicts the curves of the running time. Specifically, the curves of MSA and SCA rise along with the growth of SFC length, while the curve of RSA is stable. This is because MSA and SCA perform comparison process between server nodes when deciding the VNF locations, while RSA randomly selects server nodes for VNF deployment without comparison.

### C. Evaluation with Real-world Network

In this experiment, we apply the real-world network topology of Palmetto and evaluate our algorithm by changing the number of destinations and the length of SFC. Furthermore, with relatively small network size, we are able to obtain the optimal solution with the tool of CPLEX [33], and then compare it with the designed approximation algorithm.

First, Literatures [34], [35] pointed out that the scale of multicast task can be quite large. Combined with our real network size, we set $|D| \in [5,25]$ to evaluate the impact of the multicast size. We fix the length of SFC to 10 and set the average deployment cost with $\mu = 2$. Fig. 13 illustrates the impact of the number of destinations. In Fig. 13(a), the traffic delivery cost grows with the increase of destinations, and compared with RSA, our MSA can reduce the traffic delivery cost by 12.86% in average. Compared with the optimal solutions (illustrated by the yellow curve in Fig. 13(a)), our approximated solutions can achieve an approximation ratio of 1.51 as far as the average algorithm performance in this experiment, which is smaller than the given theoretical one in Theorem 6. According to Fig. 13(b), the running time of all three algorithms rise with the increase of destinations. Note that the running time of optimal solution is not able to be illustrated in the figure, since it is much (>30 times) longer than those from other three algorithms.

Fig. 14 evaluates the impact of the SFC length. In this experiment, we fix the number of destination to 15 (i.e.$|D|$=15) and set the average deployment cost with $\mu = 2$. Similar to Fig. 12, the curves in Fig. 14 show the same trend. Compared with RSA, the traffic delivery cost with our MSA is reduced by 18.69% in average. The curves in Fig. 14(b) also show the same trend as Fig. 13(b), i.e., the running time of all three algorithms rise with the increase of SFC length.

## VI. Conclusion

In NFV enabled multicast, as the costs of VNF deployment and link connection are sensitive to the VNF location, embedding a service function tree (SFT) is proved to be a better choice for cost saving. In this paper, we study the optimal SFC embedding problem for NFV enabled multicast. Firstly, we formally defined and formulated the optimal SFT embedding problem with an integer linear programming. To solve the problem efficiently, we designed a two-stage algorithm by i) producing an initial feasible solution with embedding an SFC and ii) optimize the initial solution by adding new VNFs and constructing an SFT. We proved that the approximation ratio of our algorithm is $1+\rho$ where $\rho$ can be as small as 1.39, under the condition that the server nodes have sufficient resources. With extensive experimental evaluations, we disclosed the impact of different parameters on embedding an SFT and showed that our SFT embedding solution can reduce the traffic delivery cost by 22.4%, compared with the random SFT embedding.

## References

[1] S. E. Deering and D. R. Cheriton, "Multicast routing in datagram internetworks and extended lans," *ACM Transactions on Computer Systems (TOCS)*, vol. 8, no. 2, pp. 85–110, 1990.

[2] L. H. Huang, H. C. Hsu, S. H. Shen, D. N. Yang, and W. T. Chen, "Multicast traffic engineering for software-defined networks," in *Proc.of IEEE INFOCOM*, 2016.

[3] L. Kou, G. Markowsky, and L. Berman, "A fast algorithm for steiner trees," *Acta Informatica*, vol. 15, no. 2, pp. 141–145, 1981.

[4] H. Kim and N. Feamster, "Improving network management with software defined networking," *IEEE Communications Magazine*, vol. 51, no. 2, pp. 114–119, 2013.

[5] S.-H. Shen, L.-H. Huang, D.-N. Yang, and W.-T. Chen, "Reliable multicast routing for software-defined networks," in *Proc.of IEEE INFOCOM*, 2015.

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

[7] 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 and Tutorials*, vol. 18, no. 1, pp. 236–262, 2016.

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

[9] W. Ma, O. Sandoval, J. Beltran, D. Pan, and N. Pissinou, "Traffic aware placement of interdependent nfv middleboxes," in *Proc. of IEEE INFOCOMM*, 2017.

[10] S. Mehraghdam, M. Keller, and H. Karl, "Specifying and placing chains of virtual network functions," in *Proc. of IEEE CloudNet*, 2014.

[11] D. Bhamare, R. Jain, M. Samaka, and A. Erbad, "A survey on service function chaining," *Journal of Network and Computer Applications*, vol. 75, no. C, pp. 138–155, 2016.

[12] N. Huin, B. Jaumard, and F. Giroire, "Optimization of network service chain provisioning," in *Proc.of IEEE ICC*, 2017.

[13] P. Zave, R. A. Ferreira, X. K. Zou, M. Morimoto, and J. Rexford, "Dynamic service chaining with dysco," in *Proc.of ACM SIGCOMM*, 2017.

[14] N. Herbaut, D. Negru, D. Magoni, and P. A. Frangoudis, "Deploying a content delivery service function chain on an sdn-nfv operator infrastructure," in *Proc.of IEEE TEMU*, 2016.

[15] A. Mansy and M. Ammar, "Analysis of adaptive streaming for hybrid cdn/p2p live video systems," in *Proc. of IEEE ICNP*, 2011.

[16] Z. Xu, W. Liang, M. Huang, M. Jia, S. Guo, and A. Galis, "Approximation and online algorithms for nfv-enabled multicasting in sdns," in *Proc.of IEEE ICDCS*, 2017.

[17] J. Byrka and F. Grandoni, "An improved lp-based approximation for steiner tree," in *ACM Symposium on Theory of Computing*, 2010.

[18] H. Moens and F. De Turck, "Vnf-p: A model for efficient placement of virtualized network functions," in *Proc.of CNSM*, 2014.

[19] M. Bouet, J. Leguay, and V. Conan, "Cost-based placement of virtualized deep packet inspection functions in sdn," in *Proc.of IEEE MILCOM*, 2013.

[20] R. Cohen, L. Lewin-Eytan, J. S. Naor, and D. Raz, "Near optimal placement of virtual network functions," in *Proc.of IEEE INFOCOM*, 2015.

[21] M. Zeng, W. Fang, J. J. P. C. Rodrigues, and Z. Zhu, "Orchestrating multicast-oriented nfv trees in inter-dc elastic optical networks," in *Proc. of IEEE ICC*, 2016.

[22] T. W. Kuo, B. H. Liou, C. J. Lin, and M. J. Tsai, "Deploying chains of virtual network functions: On the relation between link and server usage," in *Proc.of IEEE INFOCOM*, 2016.

[23] D. Bhamare, M. Samaka, A. Erbad, R. Jain, L. Gupta, and H. A. Chan, "Optimal virtual network function placement in multi-cloud service function chaining architecture," *Computer Communications*, vol. 102, pp. 1–16, 2017.

[24] L. Guo, J. Pang, and A. Walid, "Dynamic service function chaining in sdn-enabled networks with middleboxes," in *Proc.of IEEE ICNP*, 2016.

[25] A. Iyer, P. Kumar, and V. Mann, "Avalanche: Data center multicast using software defined networking," in *Proc.of IEEE COMSNETS*, 2014.

[26] J.-J. Kuo, S.-H. Shen, M.-H. Yang, D.-N. Yang, M.-J. Tsai, and W.-T. Chen, "Service overlay forest embedding for software-defined cloud networks," in *Proc.of IEEE ICDCS*, 2017.

[27] M. R. Garey and D. S. Johnson, "Computers and intractability. a guide to the theory of np-completeness. a series of books in the mathematical sciences," 1979.

[28] R. W. Floyd, "Algorithm 97: shortest path," *Communications of the ACM*, vol. 5, no. 6, p. 345, 1962.

[29] E. W. Dijkstra, "A note on two problems in connexion with graphs," *Numerische mathematik*, vol. 1, no. 1, pp. 269–271, 1959.

[30] "The internet topology zoo," http://topology-zoo.org/maps/.

[31] P. ErdóS and A. Rényi, "On random graphs i," *Publ. Math. Debrecen*, vol. 6, pp. 290–297, 1959.

[32] G. Brown, S. Analyst, and H. Reading, "White paper: Service chaining in carrier networks," http://www.qosmos.com/, 2015.

[33] "IBM ILOG CPLEX," https://www.ibm.com/products/ilog-cplex-optimization-studio/.

[34] S. Ratnasamy, A. Ermolinskiy, and S. Shenker, "Revisiting ip multicast," in *Proc. of ACM SIGCOMM*, 2006.

[35] X. Li and M. J. Freedman, "Scaling ip multicast on datacenter topologies," in *Proc. of CoNEXT*, 2013.

View publication stats