DAG-SFC: Minimize the Embedding Cost of SFC with Parallel VNFs

Conference Paper · August 2018

DOI: 10.1145/3225058.3225111

5 authors, including:

Deke Guo
National University of Defense Technology
128 PUBLICATIONS **832** CITATIONS

SEE PROFILE

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

SEE PROFILE

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

SEE PROFILE

DAG-SFC: Minimize the Embedding Cost of SFC with Parallel VNFs

Xu Lin
School of Computer Science and Technology
Xidian University
Xi'an, Shaanxi, China
xulin_xidian@163.com

Deke Guo*
National University of Defense Technology
Changsha, Hunan, China
guodeke@gmail.com

Yulong Shen*
School of Computer Science and Technology
Xidian University
Xi'an, Shaanxi, China
ylshen@mail.xidian.edu.cn

Guoming Tang
National University of Defense Technology
Changsha, Hunan, China
gmtang@nudt.edu.cn

Bangbang Ren
National University of Defense Technology
Changsha, Hunan, China
renbangbang94@163.com

ABSTRACT

Network Function Virtualization (NFV) is an emerging technology, which enables service agility, flexibility and cost reduction by replacing traditional hardware middleboxes with Virtual Network Functions (VNFs) running on general-purpose servers. Service Function Chain (SFC) constitutes an end-to-end service by organizing a series of VNFs in a specific order. Particularly, hybrid SFC (SFC with parallel VNFs) is proposed to much reduce the traffic delay in sequential SFCs. Nevertheless, how to strategically select VNF instances and links in hybrid SFC embedding remains an open problem. In this paper, we target at the cost minimization and address the optimal hybrid SFC embedding problem. Specifically, we first develop a novel abstraction model for the hybrid SFC with Directed Acyclic Graph (DAG), which helps convert diverse hybrid SFCs to the standardized DAG-SFC form. Then, we formulate the optimal DAG-SFC embedding problem as an integer optimization model and propose a greedy method (called BBE) to solve the NP-hard problem. MBBE method is developed upon BBE method to further cut down the computation complexity in model solving. Extensive simulation results demonstrate the effectiveness of our approach for cost reduction in hybrid SFC embedding.

CCS CONCEPTS

• **Networks → Control path algorithms**; *Traffic engineering algorithms*; *Network resources allocation*;

*Deke Guo and Yulong Shen are both corresponding authors.

ICPP 2018, August 13–16, 2018, Eugene, OR, USA

ACM ISBN 978-1-4503-6510-9/18/08…$15.00
https://doi.org/10.1145/3225058.3225111

KEYWORDS

Network Function Virtualization, Network Function Parallelism, Service Function Chain Embedding

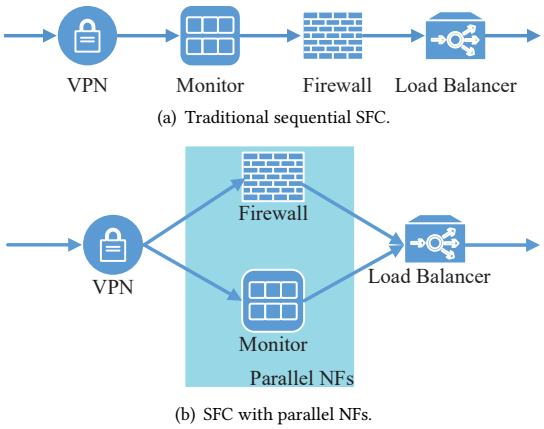
ACM Reference Format:
Xu Lin, Deke Guo, Yulong Shen, Guoming Tang, and Bangbang Ren. 2018. DAG-SFC: Minimize the Embedding Cost of SFC with Parallel VNFs. In *ICPP 2018: 47th International Conference on Parallel Processing, August 13–16, 2018, Eugene, OR, USA.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3225058.3225111

1 INTRODUCTION

Network Functions, such as firewall, deep packet inspections, are frequently used in enterprise networks to ensure security, improve performance, and provide other novel network functionalities [16]. However, the traditional deployment of these NFs depends on expensive special-purpose hardware-based appliances such as middleboxes [2].

On the contrary, Network Function Virtualization (NFV) [2], has been proposed to replace special-purpose hardwares by hosting virtual network function (VNF) softwares on general-purpose CPUs or virtual machines. This could bring us rapid deployment, network scalability [12], low-cost update and encourage innovation on network [5]. To capture many known benefits of cloud computing such as decreased costs and easy management, VNFs could be hosted in the public cloud or private clouds embedded within an ISP infrastructure [16][10]. Furthermore, many efforts are token to explore new models for NFV deployment. Telecom operator and third-party providers would offer VNFs as commodity in public clouds [14, 16] so that clients could establish their service by renting such VNFs. Also, large-scale enterprises could deploy VNFs in their private clouds to meet their own requirements.

Typically, to get a specific and complete end-to-end service from the source to the destination, network flow needs to pass through several VNFs in a particular order, which is known as service function chain (SFC) [3, 11]. As shown in Fig. 1(a), a conventional SFC consists of a set of sequential VNFs. In VNF enabled traffic engineering, a fundamental problem is to find a routing path for a flow, such that the flow can traverse a required SFC. This results in a joint procedure of VNF assignment and path selection, which is



(a) Traditional sequential SFC.

(b) SFC with parallel NFs.

Figure 1: Traditional sequential SFC v.s. hybrid SFC derived from [17].

known as SFC embedding [8]. Recently, many researches focus on this problem with different design goals, such as minimum cost [4, 13, 20], maximum network throughput [7, 8, 18], delay reduction [1] or jointly considering of all the above [21, 23]. However, current optimization on SFC embedding still could not achieve a breakthrough on the total delay reduction due to the nature of the sequential process between VNFs in traditional SFC.

Recently, some emerging researches show the possibility of VNF parallelism [22] and propose a framework to enable parallelism for those VNFs without ordering requirement [17]. As shown in Fig. 1(b), the SFC can be deployed with the combination of sequential and parallel VNFs, which we called **hybrid SFC** in this paper. With hybrid SFC, it has been validated that the traffic delivery delay can be significant reduced [17]. Nevertheless, current work only experiments hybrid SFC under the single server scenario. The general deployment of hybrid SFC over a target network has not been considered yet.

In this work, we consider the hybrid SFC embedding problem in an emerging cloud network scenario from the perspective of consumers. In a cloud network, there are many geo-dispersed cloud nodes that are connected via network links. In each of such nodes, there may exist multiple VNF instances, deployed by third party providers or the network operators [24]. Additionally, each VNF instance may have a rental price (corresponding to its deployment cost and resource consuming [9, 15, 23]) and a traffic processing capacity, while each network link also has a link price and the bandwidth capacity. More in detail, different VNF instances and links may have different prices. From the perspective of consumers, how to reduce the total cost when embedding a required hybrid SFC into the network is important.

We aim to minimize the total cost when embedding a hybrid SFC without exceeding the capacity constraints over a given priced cloud network. Specifically, we take the efforts as follows. Firstly, we propose a novel abstraction model, a kind of standardized Directed Acyclic Graph (DAG), to clearly model the orchestration of hybrid SFCs. Furthermore, we illustrate how a flow passes through a hybrid SFC according to a DAG order. Then, we carefully formulate the optimal hybrid SFC embedding problem with an integer optimization model and use a greedy method based on breadth-first search (named BBE) to solve the problem. To further reduce the computational complexity of BBE method, we propose a mini-path breadth-first backtracking embedding method (named MBBE). The results from our simulations demonstrate that the algorithms have good performance in terms of the total cost reduction, and in particularly MBBE can cut down the computation complexity without an apparent performance degradation.

2 RELATED WORK

SFC embedding problem has attracted much attention. Many researches focus on this problem with particular design goals, such as minimum cost [4, 20], maximum network throughput [3, 7, 8, 18], delay reduction [1] or jointly considering of all the above [21, 23]. Nevertheless, most of the researches focused on the situation of sequential SFCs and paid little attention to the VNF parallelism. Thus, due to the nature of sequential traffic flow processing, few of them achieved a breakthrough on the delay reduction.

Recently, some emerging researches showed the feasibility and effectiveness of VNF parallelism for traffic latency reduction [17, 22]. The observation in [17] indicated that 53.8% network function pairs in enterprise networks could work in parallel. Moreover, 41.5% NF pairs could be parallelized without causing extra resource overhead in NFV-enabled traffic engineering. Due to the great advantage at traffic delay reduction, the parallel VNFs were preferred to the traditional sequential ones when establishing SFCs [17, 22]. Nevertheless, all previous work considers the deployment of parallel VNFs upon one single server, and the general scenario where parallel VNFs are deployed over a cloud network [14, 16] (i.e., hybrid SFC embedding) needs to be further explored.

When embedding a specific hybrid SFC within a network, renting VNF instances on different network nodes and selecting VNF connections via different network links could result in dramatically different instantiating costs. From the perspective of consumers, how to cut down the total cost of SFC instantiating is a critical problem. Therefore, we propose and solve the optimal hybrid SFC embedding problem, i.e., how to orchestrate the embedding of a hybrid SFC over a given cloud network, such that the total cost can be minimized. *To the best of our knowledge, this is the first work dealing with the optimal hybrid SFC embedding problem.*

3 HYBRID SFC EMBEDDING WITH DAG ABSTRACTION

In this section, we first present a novel abstraction model for the hybrid SFC, which clearly discloses its inner structure, and convert the hybrid SFC to a standardized DAG-SFC form. Then, we analyze the DAG-SFC embedding problem thoroughly and formulate it with an integer programming.

3.1 DAG Abstraction for Hybrid SFCs

Based on the analysis of [17, 22], we propose a novel hybrid SFC abstraction model utilizing a standardized directed acyclic graph (DAG). DAG has been frequently utilized to depict parallel tasks and the dependency between these tasks, in many fields such as

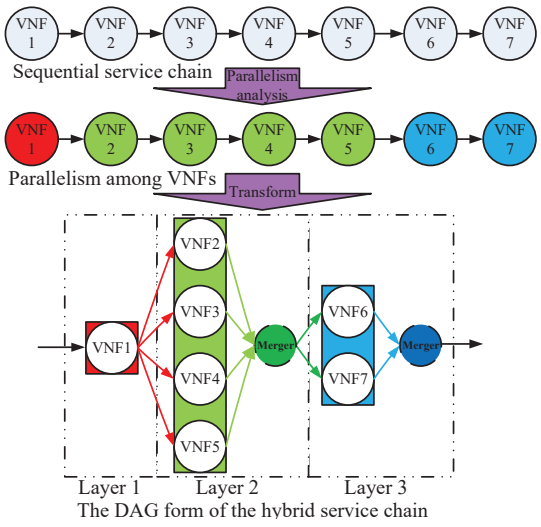


Figure 2: The transformation of service chain from sequential form to a DAG.

big data processing [19] and data-parallel cluster scheduling [6], while it has not been used for VNF parallelism.

As demonstrated in Fig. 2, a sequential service chain could be transformed to a hybrid form by analyzing the parallelism in the chain, so that we could abstract the hybrid form with a DAG. As the middle graph in Fig. 2 shows, the service chain can be divided into several VNF sets, each of which contains VNFs that can be executed in parallel and is called a *parallel VNF set*. Then, the service function chain can be converted to multiple layers, each with a parallel VNF set. As illustrated by the bottom graph in Fig. 2, the VNF set {2,3,4,5} is the parallel VNF set at layer 2 and the VNF set {6,7} is the parallel VNF set at layer 3. The relation between layers (i.e., parallel VNF sets) is still sequential due to the existence of non-parallelizable VNF pairs in SFCs [17]. Notice that each layer is usually followed by a merger that is responsible for integrating the middle results from the parallel VNFs.

Generally speaking, for any service function chain, we can divide it into one or multiple serial layers, each consisting of a single VNF or a parallel VNF set (followed by a merger), as illustrated by Fig. 2. Therefore, the above DAG abstraction can be a standardized procedure for hybrid SFC transformation. We name the transformed form as DAG service function chain (DAG-SFC). With the standardized form of DAG-SFC, we then turn to formulate and solve the DAG-SFC embedding problem.

3.2 DAG-SFC Embedding: Representation & Definition

We consider the cloud network as our target network, which is an overlay network built upon the underlying fundamental network (established by the telecom operators). The network nodes are connected via the network links, and upon each network node, third party VNF providers can supply customers with diverse VNF instances. We then present the system models as follow:

Model of Target Network: The target network is modelled by a graph $G=(V,E)$. Each link $e \in E$ is bi-directional and associated with i) a link price c_e for each unit of traffic delivery rate (e.g., 1 Gbps) and ii) a bandwidth capacity r_e. Each node $v \in V$ contains one or multiple VNFs (denoted by F_v) which is a subset of the VNF set ($F_v \subset F$).

Model of VNF Deployment: Assume that there are n categories of VNFs available from the third party VNF provider. We represent them with a VNF set $F = \{f(1), f(2), ..., f(n)\}$ with the i^{th} category of VNF denoted by $f(i)$. For VNF $f(i)$ deployed on node v, we denote it by $f_v(i)$ and assume that it is with i) a rental price for each unit of traffic delivery rate (e.g., 1 Gbps) denoted by $c_{v,f(i)}$ and ii) a traffic processing capability denoted by $r_{v,f(i)}$. For all network nodes with VNF $f(i)$, we include them in a node set $V_i \subseteq V$. Besides the n regular VNFs, we introduce two more particular ones: i) $f(0)$ denoting the dummy VNF (used to unify the equations in later optimization model), and ii) $f(n+1)$ denoting the merger for the parallel VNFs.

Model of DAG-SFC: Assume that a specific SFC can be standardized to an ω-layer DAG-SFC, denoted by $S=\{L_1, L_2, ..., L_\omega\}$. That is, there are ω serial layers in the DAG, and each layer is composed by a single VNF or a parallel VNF set (followed by a merger). Let φ_l denote the number of parallel VNFs at layer L_l and f_l^γ denote the γ^{th} VNF of this layer. The merger followed by layer L_l is denoted by $f_l^{\varphi_l+1}$.

Model of DAG-SFC Path: The logic link connection between any two VNFs in the DAG-SFC is defined as a meta-path, as illustrated by the colored directed lines in the bottom graph of Fig. 2. The actual link path connecting any two network nodes is defined as a real-path. Let the real-path set P_b^a denote the set of real-paths between node v_a and node v_b. Then, a specific real-path between node v_a and node v_b can be denoted by $p_{b,\rho}^a \in P_b^a$, where ρ is a subscript to distinguish the real-path from others within the real-path set P_b^a. Moreover, we use β to denote the length of a specific real-path, e.g., a β-length real-path $p_{x_\beta,\rho}^{x_0} = \{e_{x_0,x_1}, e_{x_1,x_2}, ..., e_{x_{(\beta-1)},x_\beta}\}$. Note that a real-path which connects two nodes with VNFs is actually an implementation of corresponding meta-path. Moreover, since overlaps may exist among real-paths, the network links can be reused for multiple times during the traffic delivery.

Model of Traffic Flow: A traffic flow is with a size z and a delivery rate R, and is delivered from the source node $s \in V$ to the destination node $t \in V$. The source node and the destination node form the source-destination pair.

With the above system models, we formally define the optimal DAG-SFC embedding problem as follow:

***Definition* 1 (Optimal DAG-SFC Embedding Problem)** Given the target network and a traffic flow with source-destination pair, the problem is to strategically embed a specified DAG-SFC into the target network. Thus, the total traffic delivery and processing cost (including link cost and VNF rental cost) can be minimized, under the constraints of the network link capacity and the VNF traffic processing capability.

3.3 Optimal DAG-SFC Embedding

The meta-paths in a DAG-SFC can be naturally classified into two groups. The first group (denoted by P_1) contains all the meta-paths connecting two adjacent layers, which is called *inter-layer meta-path set*. More in detail, a meta-path $p \in P_1$ could connect the merger or the single VNF of the former layer to current layer's VNFs, such as the black, red and dark green arrows shown in the bottom of Fig. 2. The second group (denoted by P_2) contains all meta-paths from parallel VNFs to the merger at the same layer and we call P_2 as *inner-layer meta-path set*, e.g., the light green and blue arrows in Fig. 2. Note that the inter-layer meta-paths of the same layer use a multicast transfer (but not a straightforward combination of multiple unicast). However, the inner-layer meta-paths of the same layer cannot use the multicast transfer, as the traffic flow processed by the parallel VNFs could result in different versions. Therefore, we treat the two meta-path groups separately.

Before formulating the optimal DAG-SFC embedding model, we give the definitions of different variables as follow.

- $x_{v,l,\gamma}$: a binary variable, denoting whether or not the γ^{th} VNF at layer L_l (i.e., f_l^γ) on the node v is rented.
- $x_{l,\gamma}^i$: a binary variable, denoting whether or not the equation $f_l^\gamma = f(i)$ holds.
- $\alpha_{v,i}$: an integer variable, denoting the reused times of VNF $f(i)$ on node v (i.e., $f_v(i)$).
- $x_{b,\rho,l,\varepsilon}^a$: a binary variable, denoting whether or not the real-path $p_{b,\rho}^a$ is selected to implement the meta-path between the $(l-1)^{th}$ merger $f_{(l-1)}^{|L_{(l-1)}|}$ (or the only VNF at the $(l-1)^{th}$ layer) and f_l^ε.
- $y_{b,\rho}^{a,l,\gamma}$: a binary variable, denoting whether or not real-path $p_{b,\rho}^a$ is selected to implement the meta-path between f_l^γ and the l^{th} merger $f_l^{|L_l|}$.
- $x_{b,h,\rho}^{a,g}$: a binary variable, denoting whether or not the edge $e_{g,h}$ is selected to constitute an instantiated real-path $p_{b,\rho}^a$.
- $\alpha_{g,h}$: an integer variable, denoting the reused times of edge $e_{g,h}$.
- $m_{l,\gamma}$: a binary variable, denoting whether or not VNF f_l^γ is the merger of the l^{th} layer.
- $F(a,b,\rho) = \prod_{e_{h,g} \in p_{b,\rho}^a} x_{b,h,\rho}^{a,g}$. $F(a,b,\rho)$: a function returning binary results, indicating whether or not all the link $e_{h,g} \in p_{b,\rho}^a$ are assigned to instantiate the real-path $p_{b,\rho}^a$.

With the above system models and variables, the objective function of optimal DAG-SFC embedding problem can be defined as follow.

$$\min\Big(\sum_{v\in V}\sum_{f(i)\in F_v}\alpha_{v,i}c_{v,f(i)}z + \sum_{e_{g,h}\in E\wedge v_g\neq v_h}\alpha_{g,h}c_{e_{g,h}}z\Big) \quad (1)$$

In the objective function, the first term is the total VNF rental cost and the second term is the total link cost. Based on Definition 3.2, we formulate the constraints in the optimal DAG-SFC embedding problem.

3.3.1 Capacity Constraints. The following two constraints ensure that i) all VNF instances $f(i)$ of the given network do not exceed their processing capability, and ii) all links in the network do not exceed their bandwidth capacity, when embedding the DAG-SFC.

$$\alpha_{v,i}R \le r_{v,f(i)} \qquad \forall f(i)\in F_v, v\in V \quad (2)$$

$$\alpha_{g,h}R + \alpha_{h,g}R \le r_{e_{g,h}} \qquad \forall v_g, v_h \in V \wedge v_g \neq v_h \quad (3)$$

3.3.2 Service Chain Enabling Constraints. To uniform the model, we use $L_0 = \{f_0^1\}$ and $L_{(\omega+1)} = \{f_{(\omega+1)}^1\}$ to denote additional layers for the source node and the destination node, respectively. Furthermore, we assign the dummy VNF $f(0)$ to the two additional layers, i.e., $f_0^1 = f_{(\omega+1)}^1 = f(0)$. Thus we define the stretched SFC as $S^+ = \{L_0, L_1, ..., L_{(\omega+1)}\}$ Then, the following three constraints ensure the completeness of the DAG-SFC embedded in the target network.

$$\sum_{v\in V} x_{v,l,\gamma} = 1 \;\; \forall l \in \{l \mid L_l \in S^+\}, \forall \gamma \in \{\gamma \mid f_l^\gamma \in L_l\} \quad (4)$$

$$\sum_{v_a\in V_i} x_{v_a,(l-1),|L_{(l-1)}|} \sum_{v_b\in V_j} x_{v_b,l,\varepsilon} \sum_{\rho=1}^{|P_b^a|} x_{b,\rho,l,\varepsilon}^a F(a,b,\rho) \ge 1$$
$$\forall l \in \{1,...,\omega+1\},\ \forall \varepsilon \in \{1,2,...,\varphi_l\}$$
$$\forall (i,j) \in \{(i,j) \mid f(i) = f_{(l-1)}^{|L_{(l-1)}|} \wedge f(j) = f_l^\varepsilon\} \quad (5)$$

$$\sum_{v_a\in V_i} x_{v_a,l,\gamma} \sum_{v_b\in V_j} x_{v_b,l,\varphi_l+1} \sum_{\rho=1}^{|P_b^a|} y_{b,\rho}^{a,l,\gamma} F(a,b,\rho) \ge 1$$
$$\forall l \in \{l \mid l \in \{1,2,...,\omega\} \wedge |S_l| > 1\}, \forall f_l^\gamma \in L_l - \{f_l^{\varphi_l+1}\}$$
$$\forall (i,j) \in \{(i,j) \mid f(i) = f_l^\gamma \wedge f(j) = f_l^{\varphi_l+1}\} \quad (6)$$

In detail, constraint (4) ensures that each VNF in the DAG is only assigned once. Constraints (5) and (6) ensure that all the inter-layer and inner-layer meta-paths are implemented, respectively.

The situations of VNF and link reuses are depicted as follow:

$$\alpha_{v,i} = \sum_{l=0}^{|S^+|}\sum_{\gamma=1}^{|L_l|} x_{l,\gamma}^i x_{v,l,\gamma} \qquad \forall v\in V, \forall f(i)\in F \quad (7)$$

$$\alpha_{g,h} = \alpha_{P_1,g,h} + \alpha_{P_2,g,h} \qquad \forall v_g, v_h \in V \wedge v_g \neq v_h \quad (8)$$

where

$$\alpha_{P_1,g,h} = \sum_{l=1}^{\omega+1} \min\{\sum_{\varepsilon=1}^{\varphi_l}\sum_{v_a\in V} x_{a,l-1,|L_{(l-1)}|} \sum_{v_b\in V\wedge v_a\neq v_b} x_{b,l,\varepsilon} \sum_{\rho=1}^{|P_b^a|} x_{b,\rho,l,\omega}^a x_{b,h,\rho}^{a,g}, 1\} \quad (9)$$

$$\alpha_{P_2,g,h} = \sum_{l=1}^{\omega} m_{l,|L_l|} \sum_{\gamma=1}^{\varphi_l}\sum_{v_a\in V} x_{a,l,\gamma} \sum_{v_b\in V-\{v_a\}} x_{b,l,(\varphi_l+1)} \sum_{\rho=1}^{|P_b^a|} y_{b,\rho}^{a,l,\gamma} x_{b,h,\rho}^{a,g} \quad (10)$$

Formula (7) computes the value of reused times of all VNFs in the network. Formula (8) computes the value of reused times of link

$e_{g,h}$, which is the sum of two parts: i) $\alpha_{P_1,g,h}$ in Formula (9) gives the reused times of link $e_{g,h}$ caused by inter-layer meta-paths, and ii) $\alpha_{P_2,g,h}$ in Formula (9) gives the reused times of link $e_{g,h}$ caused by inner-layer meta-paths.

It can be proved that the optimal DAG-SFC embedding problem is NP-hard. Actually, under the condition that all VNFs of an SFC have been assigned, selecting the optimal links to connect the VNFs under bandwidth constraints is still NP-hard (with reduction to the unsplittable flow problem [20]). Furthermore, this problem is significantly different from the sequential SFC embedding problem, because of the transformation of inner-SFC structure. With traditional methods for the sequential SFC embedding, the completeness of the DAG-SFC cannot be guaranteed.

4 SOLUTIONS TO OPTIMAL DAG-SFC EMBEDDING

By analyzing the optimal DAG-SFC embedding problem, we first give a simple initial solution to the problem. Then a Breadth-first Backtracking Embedding method (BBE) and a Mini-path Breadth-first Backtracking Embedding method (MBBE) are developed to optimize the initial solution.

4.1 BBE Framework

A naive idea to tackle the optimal DAG-SFC embedding problem is to select the cheapest VNFs to build the DAG-SFC in the network. However, this idea omits the connection links in the target network and may result in a huge link cost. To reduce the total cost of embedding a DAG-SFC, we need to jointly consider the VNF cost and the link cost.

It is intuitive to select VNFs on adjacent nodes, so that the link cost can be reduced. Based on this principle, we propose the Breadth-first Backtracking Embedding (BBE) method, which is inspired by the breadth-first search idea. The basic idea of BBE is:

(1) Search for multiple feasible sub-solutions as the candidates of embedding solution for one single layer;
(2) Repeat the above candidate searching process layer by layer, until all layers of the DAG-SFC are traversed;
(3) Among the complete embedding solutions formed by series of sub-solution candidates, select the cheapest one as the final solution.

The pseudo-code of BBE is shown in Algorithm 1. In detail, to generate the feasible sub-solution candidates of the l^{th} layer based on a specific sub-solution at the $(l-1)^{th}$ layer, three steps need to take, i) forward search, ii) backward search and iii) candidate sub-solution generation, which are corresponding to lines (5-8) in Algorithm 1. After obtaining the sub-solutions of all layers, for each sub-solution at the ω^{th}, BBE connects the ω^{th} end node to the destination node using minimum cost path. This leads to a complete candidate solution, corresponding to lines (9-10) in Algorithm 1. At last, the BBE will select the cheapest solution from the candidates as the final solution, which is corresponding to line (11) in Algorithm 1.

Algorithm 1: **B**readth-first **B**acktracking **E**mbedding Algorithm (**BBE**)

Input: $G(V, E)$: network topology with its capacity and price information, S=$\{L_1, L_2, ..., L_\omega\}$: the DAG-SFC, the start node v_s, and the destination node v_t
Output: One of the local optimal solution

```
1  Let l be the current layer L_l;
2  Get the real-time network graph G_1;
3  for layer l from 1 to ω do
4      for each (l−1)^th layer sub-solution ss do
5          Forward search base on ss ;
6          for each probable merger node searched by the
             forward search do
7              Backward search from the merger node;
8              Generate feasible sub-solutions at the l^th;
9  for each ω^th sub-solution do
10     Generate a complete embedding solution candidate;
11 Select the cheapest solution candidate as the final solution.
```

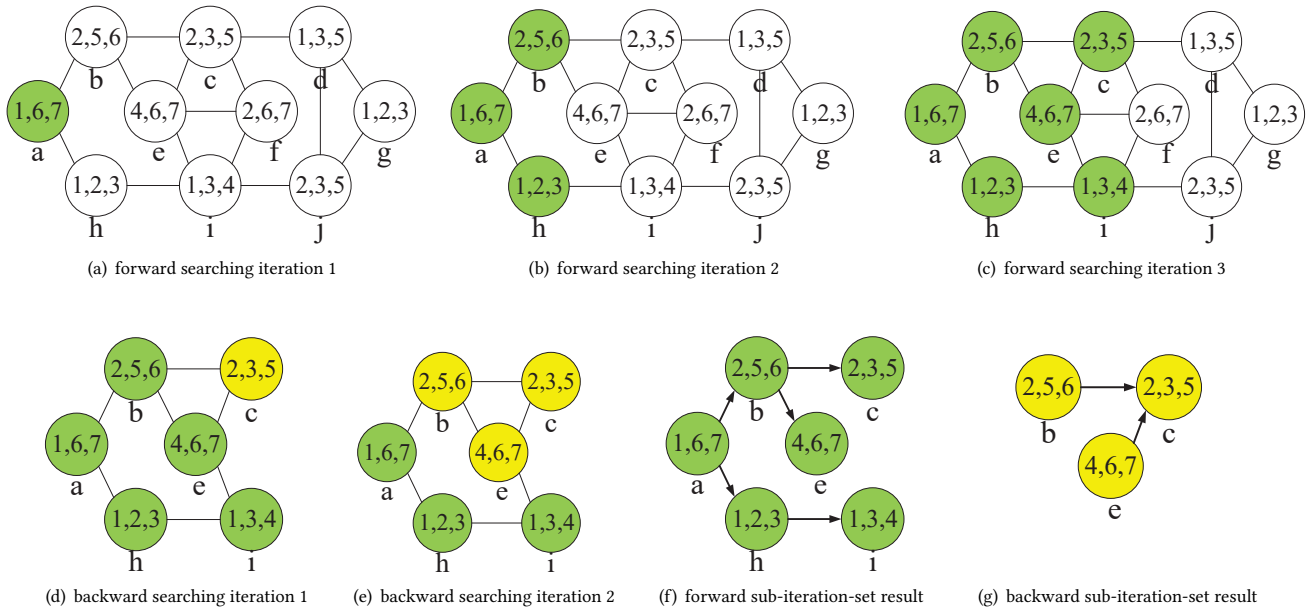
4.2 Step 1: Forward Search

Generally speaking, the forward search process at the l^{th} layer is to find a set of adjacent nodes which includes all required VNFs (according to the DAG-SFC) from the start node at the l^{th} layer. Thus, we can embed the l^{th} layer of DAG-SFC into the target network from the start node of this layer. At the same time, such forward search process also instantiates the inter-layer meta-paths at the l^{th} layer. The detail of the forward process is as follow.

To ease the description, we first introduce some symbols applied in later sections.

- Let v_l denote the end node at the l^{th} layer. Then, the start node of the l^{th} layer is $v_{(l-1)}$.
- Let I_l^F denote the forward search process starting from $v_{(l-1)}$, which includes a series of iterations.
- Let $V^{F,l}_{v_{(l-1)},q}$ denote the forward search node set, which is composed by nodes searched in the first q forward iterations (at layer l) starting from node $v_{(l-1)}$.
- Let $F^{F,l}_{v_{(l-1)},q} = \bigcup_{v \in V^{F,l}_{v_{(l-1)},q}} F_v$ denote the VNF set that contains all the VNFs deployed on the nodes of $V^{F,l}_{v_{(l-1)},q}$.

4.2.1 Forward Search. Based on the breadth-first search, the forward search may contain multiple iterations. We start the forward search from $v_{(l-1)}$ for all required VNFs at layer l. Specifically, in the first iteration of I_l^F, $V^{F,l}_{v_{(l-1)},1} = \{v_{(l-1)}\}$; then, in the q^{th} iteration of I_l^F, we can obtain the forward searching node set $V^{F,l}_{v_{(l-1)},q}$ by extending the last forward searching node set $V^{F,l}_{v_{(l-1)},(q-1)}$ and adding the nodes which have direct connections with any node in $V^{F,l}_{v_{(l-1)},(q-1)}$; until find all required VNFs at the l^{th} layer of DAG-SFC, we stop the l^{th} forward search. In other words, we iterate the



(a) forward searching iteration 1 (b) forward searching iteration 2 (c) forward searching iteration 3

(d) backward searching iteration 1 (e) backward searching iteration 2 (f) forward sub-iteration-set result (g) backward sub-iteration-set result

Figure 3: An example to illustrate how BBE works: forward search & backward search.

forward search until $F^{F,l}_{v_{(l-1)},q}$ covers the VNFs set of the l^{th} layer, i.e., $L_l \subseteq F^{F,l}_{v_{(l-1)},q}$.

To clearly illustrate how BBE works, we give an example in Fig. 3, which shows the VNFs embedding process of the second layer of DAG-SFC in Fig. 2. With the example in Fig. 3, we assume that the first layer with a single VNF ($f(1)$) is deployed on node v_a. Then, Fig. 3(a), 3(b) and 3(c) show how I^F_2 instantiate the meta-paths that connect $f(1)$ and each of the parallel VNFs at the second layer (as illustrated by the red arrows in Fig. 2). Moreover, Fig. 3(a) shows the intermediate state of the first iteration of I^F_2. After the first iteration, $V^{F,l}_{v_{(l-1)},q_l} = V^{F,2}_{a,1} = \{v_a\}$. Because the forward searching VNF set $F^{F,l}_{v_{(l-1)},q_l} = F^{F,2}_{a,1} = \{f(1), f(6), f(7), f(8) = merger\}$ does not cover the second layer VNF set $L_2 = \{f(2), f(3), f(4), f(5), f(8)\}$, the second iteration is executed and the second intermediate state is shown by Fig. 3(b). After the second iteration, $V^{F,2}_{a,2}$=$\{v_a, v_b, v_h\}$ and $F^{F,2}_{a,2} = \{f(1), f(2), f(3), f(5), f(6), f(7), f(8)\}$. Because it is still true that $L_2 \nsubseteq F^{F,2}_{a,2}$, the third iteration is executed and the third intermediate state is shown by Fig. 3(c). At this time, $L_2 \subseteq F^{F,2}_{a,3}$ is satisfied ($V^{F,2}_{a,3} = \{v_a, v_b, v_c, v_e, v_h, v_i\}$ and $F^{F,2}_{a,3} = \{f(1), f(2), f(3), f(4), f(5), f(6), f(7), f(8)\}$), thus I^F_2 is terminated. Fig. 3(f) gives the real-paths that I^F_2 has searched.

4.2.2 Forward Search Tree. We define Forward Search Tree (FST), which is a data structure based on the binary tree to store the result of a specific I^F_l. The left part of Fig. 4 gives the FST of the forward search in Fig. 3. The solid arrow between any two FST nodes denotes the binary tree connection, while the dotted arrow denotes the relationship of the corresponding network nodes. In each FST, the left child of a node corresponds to a network node searched in the next iteration, and the right child of a node corresponds to a network node searched in the same iteration. Therefore, when traversing all the FST nodes in the i^{th} iteration, we can easily find the leftmost FST node of the i^{th} layer in the FST (the root is in the first layer), and then iterate to access the right child until no right child is found. The root of the l^{th} layer FST denotes the l^{th} layer start node, and the forward search starts from the root node.

Table 1: Elements of a search tree node

Number	Elements	Number	Elements
1	Father pointer	5	Available VNF set
2	Left child pointer	6	Previous node list
3	Right child pointer	7	Next node list
4	Node ID		

More in detail, each node of the FST has seven elements as Table 1 shows. The first three elements keep the logic of binary tree. The fourth and the fifth record the information about the corresponding network nodes. The last two elements are used to record the link connection relationship between the current node and other nodes in the network. Given a specific node in the FST, we could simply instantiate an existing path to the root of the FST by selecting a series of dotted arrows as Fig. 4 shows. As the forward search guarantees that no less than one dotted arrow from a non-root FST node to the FST nodes generated from previous iteration, there always exist dotted paths between a non-root node and the root node in the FST.

4.3 Step 2: Backward Search

After the forward search, for each node v in FST containing a merger, we start a backward search process. The function of backward search is two-fold: i) to further narrow the adjacent node set

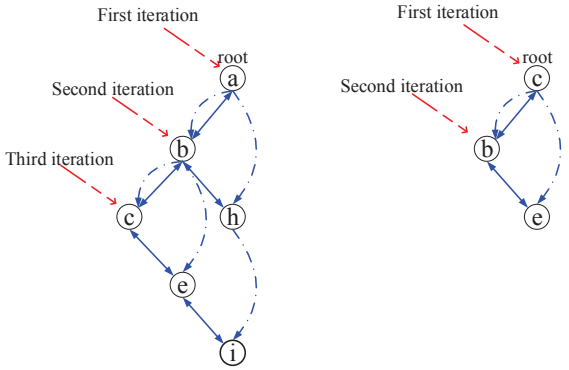


Figure 4: FST topoloby of Fig. 3(f) and BST topoloby of Fig. 3(g).

obtained in previous forward search process and ii) to instantiate the l^{th} layer inner-layer meta-paths.

To ease the description, we introduce some symbols at first.

- Let I_l^B denote the set of backward search iterations starting from v_l to find all required VNFs at the l^{th} layer.
- Let $V_{v_l,w}^{B,l}$ denote the backward search node set which is composed by nodes searched in the first w iterations (of I_l^B) starting from node v_l.
- Let $F_{v_l,w}^{B,l} = \bigcup_{v \in V_{v_l,w}^{B,l}} F_v$ denote the VNF set that includes all VNFs deployed on the nodes of $V_{v_l,w}^{B,l}$.

4.3.1 Backward Search. For the l^{th} layer, we start a backward search from the end node of this layer and iterate searching until find all required VNFs at the l^{th} layer. In detail, in the first iteration of I_l^B, $V_{v_l,1}^{B,l} = \{v_l\}$; then, in w^{th} iteration of I_l^B we get the backward search nodes set $V_{v_l,w}^{B,l}$ by extending the last backward searching nodes set $V_{v_l,(w-1)}^{B,l}$ and adding the nodes which belong to $V_{v_{(l-1)},q}^{F,l}$ and have a direct connection to any node in $V_{v_l,(w-1)}^{B,l}$; we finish the iteration until $L_l \subseteq F_{v_l,w}^{B,l}$. As an example, Fig. 3(d) and 3(e) show how the backward search I_2^B instantiates the meta-paths, which connects each real-world VNF of second layer and the merger of second layer, illustrated by the green arrows in Fig. 2. Moreover, Fig. 3(g) shows the real-paths that the backward search I_2^B has obtained.

4.3.2 Backward Search Tree. Similar to FST, we define Backward Search Tree (BST), a data structure based on the binary tree, to store the result of a specific I_l^B. The right part of Fig. 4 shows the BST of the backward search in Fig. 3. Although the BST has the same logical structure as FST, they store different information. Note that, the root of a BST at the l^{th} layer denotes the layer's end node, and the backward search starts from the root node.

4.4 Step 3: Candidate Sub-solutions Generation

With the FST and BST at each layer of a specific DAG-SFC, we can generate a series of sub-solutions in form of FST-BST pairs. To efficiently store the sub-solutions, we define the data structure of the sub-solution tree.

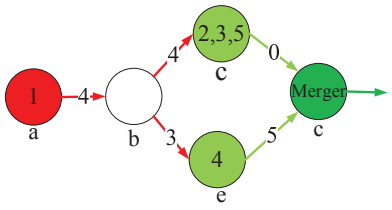


Figure 5: An example of the candidate sub-solution for Fig. 4.

4.4.1 Candidate sub-solutions generating. Given the FST-BST pair at the l^{th} layer, we generate a series of sub-solutions by traversing all feasible sub-solutions of the l^{th} layer. Based on the definition of FST and BST, we know that $L_l \subseteq F_{v_{(l-1)},q}^{F,l}$, $L_l \subseteq F_{v_l,w}^{B,l}$ and $V_{v_l,w}^{B,l} \subseteq V_{v_{(l-1)},q}^{F,l}$ are always satisfied. Accordingly, we generate candidate sub-solutions with four steps: i) for each possible combination of the l^{th} layer parallel VNFs allocation in the BST, we generate a first-step candidate sub-solution, ii) based on each first-step candidate sub-solution, we generate a set of second-step candidate sub-solutions with different real-paths by traversing the BST, iii) with each second-step candidate sub-solution, we can generate a set of third-step candidate sub-solutions with different real-paths by traversing the FST, and iv) check all the third-step candidate sub-solutions and remove the infeasible ones. Fig. 5 gives an example of candidate sub-solution generation based on the FST-BST pair in Fig. 4 and the second layer of DAG-SFC in Fig. 2, when assigning $f(2), f(3), f(5)$ on node v_c and assigning $f(4)$ on node v_e.

4.4.2 Sub-solution Tree. With the above operations, we can generate candidate sub-solutions for each FST-BST pair. Nevertheless, each FST would be established based on a specific previous layer sub-solution. Thus, how to accurately store the relation between sub-solutions is a problem. We then propose the sub-solution tree, a data structure based on tree topology, to tackle this problem. The sub-solution tree is established accompanying with the whole searching procedure. After generating the sub-solutions of a specific FST-BST pair, the BBE immediately inserts them into the sub-solution tree as the children of the previous layer's sub-solution, which the FST is based on. For a ω-layer SFC, the sub-solution tree will have $(\omega + 2)$ layers, and the 0^{th} layer stores the source node without any cost. The $1^{th} \sim \omega^{th}$ layers store the sub-solutions of corresponding SFC layers, and the $(\omega + 1)^{th}$ layer stores the minimum cost path between the ω^{th} end node to the destination node. Each $(\omega+1)^{th}$ layer sub-solution tree node (i.e., the leaf node) indicates a unique feasible solution, which is formed by connecting all sub-solutions on the acyclic path from the leaf node to the root of the sub-solution tree. Note that every link between a node and its father node is a bi-directed link. The down links satisfy the need of generating and traversing the sub-solution tree, while the up links bring great convenience in finding the upstream path from this sub-solution node to the root node by avoiding traversing the sub-solution tree from the root in each iteration.

4.5 Solution Upgrading with MBBE

The computation complexity of the BBE could be estimated by multiplying the computation complexity of each embedded layer.

Assume that the maximum number of real-paths with the same length between two network node is h and there are n node in the network. The DAG-SFC contains ω layers, and each layer includes at most φ parallel VNFs. In this case, the worst time complexity of a single layer embedding is $O(n^{\varphi}h^{\varphi}h^{\varphi})$, so the complexity of the SFC embedding is not more than $O(n^{\omega\varphi}h^{2\omega\varphi})$. Unfortunately, the complexity increases significantly along with the growing of the network size and the SFC length. The computation complexity of solving the DAG-SFC embedding problem would increase at an unacceptable rate. Thus, even no solution would be obtained due to the memory overflow caused by the huge candidate solution storage, when SFC length grows up.

Therefore, we further propose the Mini-path BBE method (MBBE) by adding the following complementary strategies upon the BBE:

(1) Set up an integer parameter $X_{max} \leq n$, and in each forward search we add a condition that the size of forward search node set must not exceed X_{max} ($|V^{F,l}_{v_{(l-1)},q}| \leq X_{max}$).
(2) When generating sub-solutions of an FST-BST pair, for each first-step candidate sub-solutions, we generate a corresponding final candidate sub-solution by instantiating the meta-paths using the minimum cost paths referring to the real-time network.
(3) Set up an integer parameter X_d. When generating the sub-solutions of an FST-BST pair, we setup a limitation that only the cheapest X_d sub-solutions can be inserted into the sub-solution tree. In this way, the sub-solution tree changes to an X_d-tree.

With strategies (1) and (2), the worst time complexity of a single layer embedding in BBE is reduced to $O(\varphi n^2 X^{\varphi}_{max})$. And due to the applying of strategy (3), the upper bound of the worst time complexity of the whole SFC embedding reduces to $O(k\varphi n^2 X^{\varphi}_{max})$, in which $k = (1-(X_d)^{\omega+1})/(1-X_d)$. The complementary strategies cut down the time complexity by reducing the search space. According to the experimental results in section 5, MBBE has a much lower time complexity than BBE while without any performance degradation.

5 PERFORMANCE EVALUATION

In this section, we demonstrate the performance of our methods by comparing with benchmark algorithms under different network conditions.

5.1 Evaluation with Simulated Networks

In order to clarify the illustration of our simulation, we give the definitions of some terminology as follow:

- *SFC size*: the number of VNFs contained by an SFC.
- *network size*: the number of nodes contained by a network.
- *network connectivity*: the average number of the node degree in a network.
- *VNF deploying ratio*: the percentage of the network nodes on which a kind of VNFs are deployed.
- *average price ratio*: the ratio of the average link price over the average VNF price.
- *VNF price fluctuation ratio*: the ratio of the half of the gap between max-price and min-price over the average price of VNFs.

Table 2: Basic configurations of simulation instances

Network size	500	Network connectivity	6
VNF deploying ratio	50%	Average price ratio	20%
VNF price fluctuation ratio	5%	SFC size	5

We generate simulated network graphs with a random network generator. Firstly, the generator generates new network nodes until conforming the given network size. Secondly, the generator connects all the nodes by a random tree to guarantee the network is a connected graph and then loops to implement new random edges until conforming the given network connectivity. Thirdly, the generator deploys VNFs on each node conforming the given VNF deploying ratio and the given VNF price fluctuation ratio. Finally, the generator sets the price of each link conforming the given average price ratio.

Then, we generate DAG-SFCs with a random SFC generator. It generates SFC by a specific rule in which every three VNFs can be assigned in the same layer, in order to avoid generating serial SFCs with little values for this simulation. However, each SFC is generated using different VNF sets. This means the SFC generator generates SFCs with similar structures but different VNFs on corresponding positions.

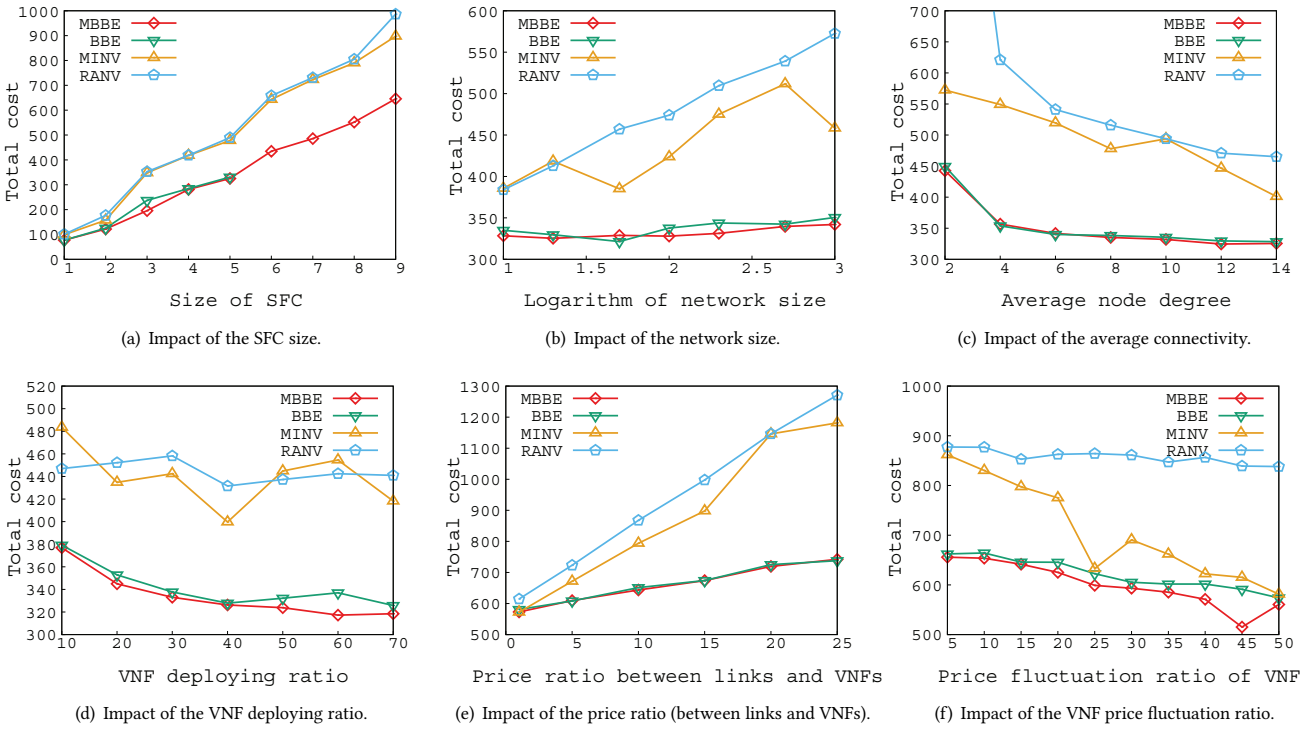
The performance of our proposed approaches is evaluated under various simulation settings. Since this is the first work on optimal hybrid SFC embedding, few methods can be referred. Two benchmark algorithms are implemented for purpose of comparison, denoted by **RANV** and **MINV**. RANV is a randomized algorithm. For each VNF required by the SFC, RANV will randomly assign this VNF on a node with enough traffic processing capability. After assigning all the VNFs of the SFC, it implements the meta-paths among those assigned VNFs by minimum cost path generated by the Dijkstra algorithm. MINV is a naive greedy algorithm. For each VNF required by the SFC, MINV will find the cheapest node with enough capacity, and assign this VNF on the node. Similar to RANV, MINV also uses the minimum cost path to implement the meta-paths.

5.2 Simulation Results

The following simulation instances would base on the basic configurations shows in Table 2. For each simulation instance, we run 100 times with different SFCs generated by SFC generator with the same structure, then set the average cost of those embedding solutions as the final result showed in the corresponding chart.

Compared with the two benchmark algorithms, we evaluate the performance of BBE and MBBE from aspects of the network size, the network connectivity, the VNF deploying ratio, the average price ratio, the VNF price fluctuation ratio, and the SFC size.

5.2.1 Impact of the SFC size. We gradually change the SFC size from 1 to 9 while the network conditions are kept the same. Apparently, the total cost of embedding the SFC exhibits an increasing trend as Fig. 6(a) shows. However, we find that our approaches have better performance, and when the SFC size is growing, the cost gap between our approaches and the benchmark algorithms



(a) Impact of the SFC size.
(b) Impact of the network size.
(c) Impact of the average connectivity.
(d) Impact of the VNF deploying ratio.
(e) Impact of the price ratio (between links and VNFs).
(f) Impact of the VNF price fluctuation ratio.

Figure 6: The results of simulations. The y axis of each graph represents the total cost of a solution, while the x axis of each graph represents the corresponding argument of each simulation instance.

is expanding. Note that, because of the time complexity of BBE is growing exponentially with the size of SFC, the inspection of BBE in this simulation ends at 5. We can see the MBBE can reduce the total cost of embedding a SFC comparing with the MINV by about 30% in those cases.

5.2.2 Impact of the network size. In this simulation, we set different network sizes as 10, 20, 50, 100, 200, 500, 1000 nodes, while other configurations are the same with the basic configurations. In Fig. 6(b), the result shows that the solutions from our algorithms are stable, while the cost of solutions generated by benchmark algorithms is rising, as the size of the network is expanding. Besides, the average cost of our solutions is at least 14% lower than the solutions obtained by the benchmark algorithms. Furthermore, the cost gap between our solutions and the benchmark solutions is expanding as the network size grows.

5.2.3 Impact of the network connectivity. We gradually change the average connectivity from 2 to 14 while other configurations are kept the same. In Fig. 6(c), the result shows that the cost of our solution is about 30% less than that of the benchmark solutions. Besides, when the average node degree is rising, the cost of solutions is decreasing. With our analysis, such a trend is most probably caused by the increase of real-paths length in benchmark solutions.

5.2.4 Impact of the VNF deploying ratio. We gradually change the VNF deploying ratio of all VNFs in the network from 10% to 70%. In Fig. 6(d), the result shows the cost of our solution is about 25% less than that of the benchmark solutions. When the VNF deploying ratio is rising, the cost of our solutions is gradually decreasing. This is because, when the VNF deploying ratio is rising, our algorithms can find more adjacent VNFs to shorten the real-paths.

5.2.5 Impact of the price ratio (between links and VNFs). We change the price ratio from 1% to 50% while keeping other configurations the same as the basic configurations. When the price ratio is rising, all the lines in Fig. 6(e) is grow up. The partial reason of such a trend is that the average link price is increasing. More in detail, the cost of benchmark solutions grow fast and the cost gap between our solutions and the benchmark solutions expands, when the link price is rising. Based on our analysis, our methods could trade off the VNF cost reduction and the link cost reduction in a proper way.

5.2.6 Impact of the VNF price fluctuation ratio. In this simulation, we gradually change the VNF fluctuation ratio from 5% to 50% while keeping other configurations the same as the basic configurations. In Fig. 6(f), the result shows that when the VNF price fluctuation ratio is rising, the cost of solutions of MBBE, BBE and MINV are gradually decreasing. The root cause is that three algorithm will try to select VNFs with cheaper price. More in detail, when the VNF price fluctuation ratio is rising, the cost gap between the MINV and our algorithms becomes narrow. This is because MINV always selects the cheapest VNFs in SFC embedding.

However, even the price fluctuation ratio of VNF reaches 50%, the performance of our solutions is still no worse than the benchmark solutions.

In all the above simulations, MBBE always results in a soluion while the benchmark algorithms do not, which illustrates the stability and robustness of MBBE. Besides, we have found that MBBE usually selects the same links to implement meta-paths as the BBE does. This is because when connecting two allocated VNFs, the mini-cost path is usually the same with the optimal path generated by traversing searching trees. Thus, MBBE lowers the computation complexity of BBE without an apparent performance degradation.

In summary of the evaluations with the six aforementioned factors, MBBE always results in a better solution than that of any benchmark algorithm. Moreover, MBBE shows a quite stable performance.

6 CONCLUSIONS

In this work, we studied the embedding problem of hybrid SFC, which aimed to jointly minimize the total VNF rental cost and link cost in SFC embedding. Firstly, we proposed an explicit DAG abstraction model to simplify and standardize the description of the hybrid SFC. Then, we formulated the optimal DAG-SFC embedding problem into an integer optimization model and proposed a breadth-first based greedy method (called BBE) to tackle the NP-hard problem. To further cut down the computation complexity, we proposed the MBBE method by simplifying the routing step and narrowing the scope of the searching process in BBE. Our experiments results show that, the MBBE can significantly cut down the computational complexity without an obvious performance degradation and our approach can much reduce the hybrid SFC embedding cost.

ACKNOWLEDGMENTS

This work is partially supported by National Natural Science Foundation of China under Grant Nos.61772544, U1536202, 61571352, National Basic Research Program (973 program) under Grant No.20 14CB347800, the Hunan Provincial Natural Science Fund for Distinguished Young Scholars under Grant No.2016JJ1002, and the Research Plan of National University of Defense Technology under Grant ZK17-03-50.

REFERENCES

[1] Anat Bremler-Barr, Yotam Harchol, and David Hay. 2016. OpenBox: A Software-Defined Framework for Developing, Deploying, and Managing Network Functions. In *Proc. of ACM SIGCOMM, Florianopolis, Brazil.*

[2] ETSI. 2012. Network Functions Virtualisation Introductory White Paper. http://www.hit.bme.hu/~jakab/edu/litr/NFV/ETSI_NFV/NFV_White_Paper1.pdf. (2012).

[3] Jingyuan Fan, Meiling Jiang, and Chunming Qiao. 2017. Carrier-grade availability-aware mapping of Service Function Chains with on-site backups. In *Proc. of IEEE/ACM IWQoS, Vilanova i la Geltrú, Spain.*

[4] Hao Feng, Jaime Llorca, Antonia M. Tulino, Danny Raz, and Andreas F. Molisch. 2017. Approximation algorithms for the NFV service distribution problem. In *Proc. of IEEE INFOCOM, Atlanta, GA, USA.*

[5] Aaron Gemberjacobson, Raajay Viswanathan, Chaithan Prakash, Robert Grandl, Junaid Khalid, Sourav Das, and Aditya Akella. 2015. OpenNF: enabling innovation in network function controls. In *Proc. of ACM SIGCOMM, Chicago, IL, USA.*

[6] Robert Grandl, Srikanth Kandula, Sriram Rao, Aditya Akella, and Janardhan Kulkarni. 2016. Graphene: packing and dependency-aware scheduling for data-parallel clusters. In *Proc. of USENIX OSDI, Savannah, GA, USA.*

[7] Linqi Guo, John Pang, and Anwar Walid. 2016. Dynamic Service Function Chaining in SDN-enabled networks with middleboxes. In *Proc. of IEEE ICNP, Singapore.*

[8] Jian Jhih Kuo, Shan Hsiang Shen, Hong Yu Kang, De Nian Yang, Ming Jer Tsai, and Wen Tsuen Chen. 2017. Service chain embedding with maximum flow in software defined network and application to the next-generation cellular network architecture. In *Proc. of IEEE INFOCOM, Atlanta, GA, USA.*

[9] Tung Wei Kuo, Bang Heng Liou, Ching Ju Lin, and Ming Jer Tsai. 2016. Deploying chains of virtual network functions: On the relation between link and server usage. In *Proc. of IEEE INFOCOM, San Francisco, CA, USA.*

[10] Chang Lan, Justine Sherry, Raluca Ada Popa, Sylvia Ratnasamy, and Zhi Liu. 2016. Embark: securely outsourcing middleboxes to the cloud. In *Proc. of USENIX NSDI, Santa Clara, CA, USA.*

[11] Yang Li, Linh Thi Xuan Phan, and Boon Thau Loo. 2016. Network functions virtualization with soft real-time guarantees. In *Proc. of IEEE INFOCOM, San Francisco, CA, USA.*

[12] Shoumik Palkar, Chang Lan, Sangjin Han, Keon Jang, Aurojit Panda, Sylvia Ratnasamy, Luigi Rizzo, and Scott Shenker. 2015. E2: a framework for NFV applications. In *Proc. of ACM SOSP, Monterey, CA, USA.*

[13] Bangbang Ren, Deke Guo, Guoming Tang, Xu Lin, and Yudong Qin. 2018. Optimal Service Function Tree Embedding for NFV Enabled Multicast. In *Proc. of IEEE ICDCS, Vienna, Austria.*

[14] Poddar Rishabh, Lan Chang, Ada Popa Raluca, and Ratnasamy Sylvia. 2018. SafeBricks: Shielding Network Functions in the Cloud. In *Proc. of USENIX NSDI, Renton, WA, USA.*

[15] Yu Sang, Bo Ji, Gagan R. Gupta, Xiaojiang Du, and Lin Ye. 2017. Provably efficient algorithms for joint placement and allocation of virtual network functions. In *Proc. of IEEE INFOCOM, Atlanta, GA, USA.*

[16] Justine Sherry, Shaddi Hasan, Colin Scott, Arvind Krishnamurthy, Sylvia Ratnasamy, and Vyas Sekar. 2012. Making middleboxes someone else's problem: network processing as a cloud service. In *Proc. of ACM SIGCOMM, Helsinki, Finland.*

[17] Chen Sun, Jun Bi, Zhilong Zheng, Heng Yu, and Hongxin Hu. 2017. NFP: Enabling Network Function Parallelism in NFV. In *Proc. of ACM SIGCOMM, Los Angeles, CA, USA.*

[18] Sheng Tao, Lin Gu, Deze Zeng, Hai Jin, and Kan Hu. 2017. Fairness-aware dynamic rate control and flow scheduling for network function virtualization. In *Proc. of IEEE/ACM IWQoS, Vilanova i la Geltrú, Spain.*

[19] Ashish Thusoo, Joydeep Sen Sarma, Namit Jain, Zheng Shao, Prasad Chakka, Suresh Anthony, Hao Liu, Pete Wyckoff, and Raghotham Murthy. 2009. Hive: a warehousing solution over a map-reduce framework. *PVLDB* 2, 2 (2009), 1626–1629.

[20] Bowu Zhang, Jinho Hwang, and Timothy Wood. 2016. Toward online virtual network function placement in Software Defined Networks. In *Proc. of IEEE/ACM IWQoS, Beijing, China.*

[21] Qixia Zhang, Yikai Xiao, Fangming Liu, John C. S. Lui, Jian Guo, and Tao Wang. 2017. Joint Optimization of Chain Placement and Request Scheduling for Network Function Virtualization. In *Proc. of IEEE ICDCS, Atlanta, GA, USA.*

[22] Yang Zhang, Bilal Anwer, Vijay Gopalakrishnan, Bo Han, Joshua Reich, Aman Shaikh, and Zhi Li. Zhang. 2017. ParaBox : Exploiting parallelism for virtual network functions in service chaining. *Proc. of ACM SOSR* (2017).

[23] Zijun Zhang, Zongpeng Li, Chuan Wu, and Chuanhe Huang. 2017. A Scalable and Distributed Approach for NFV Service Chain Cost Minimization. In *Proc. of IEEE ICDCS, Atlanta, GA, USA.*

[24] Han Zhao, Miao Pan, Xinxin Liu, Xiaolin Li, and Yuguang Fang. 2015. Exploring Fine-Grained Resource Rental Planning in Cloud Computing. *IEEE Transactions on Cloud Computing* 3, 3 (2015), 304–317.