Embedding Chains of Virtual Network Functions in Inter-Datacenter Networks

Hideo Kobayashi*, Genya Ishigaki†, Riti Gour†, Jason P. Jue†, and Norihiko Shinomiya*
*Graduate School of Engineering, Soka University, Tokyo, 192-8477, Japan
†Dept. of Computer Science, University of Texas at Dallas, Richardson, Texas 75080, USA
Email: {gishigaki, rgour, jjue}@utdallas.edu, shinomi@ieee.org

***Abstract*—This paper discusses the problem of embedding service function chains (SFCs) in an interconnected network with multiple datacenter sites. The problem is formulated as a Subtopology Composition Problem (SCP), which is to design a subnetwork that includes terminal nodes and datacenter nodes from the substrate network, aimed at optimizing distance-based latency in SFCs. The intractability of the problem is discussed, and a heuristic is proposed for the problem. Simulations are conducted to demonstrate the effectiveness of the proposed method in different graph models.**

***Keywords*—Network Function Virtualization (NFV); Service function chain; Chain embedding; Inter-datacenter network.**

I. Introduction

Network Function Virtualization (NFV), which decouples network functions (NFs) from dedicated hardware appliances, has dramatically improved the flexibility and the scalability of communication networks. Generically, a network flow passes through one or several virtual network functions (VNFs) to create connected network services such as Network Address Translators and firewalls. A chain of VNFs created by this process is called a service function chain (SFC).

In such an environment, an important question that arises is *how should one determine the physical servers on which network functions should be placed* [1]. Given a service request that specifies a set of VNFs, it is necessary to design a subnetwork in which all required service nodes are reachable in order to satisfy the request. Each subnetwork generally includes switch nodes and service nodes as well as end points called terminal nodes that start and terminate a SFC [2]. The problem of finding such a subnetwork for a chain is referred to as the SFC embedding problem (SFC-EP).

Recently, ETSI and IETF have conducted many standardization activities for NFV deployment. Yet, the study [3] has shown that default configurations of some use cases may result in sub-optimal resource allocation. Thus, the SFC-EP has been discussed in several papers. The studies differ, for example, in their evaluation metrics and networks to be considered. The literature considers evaluation functions such as communication latency and energy efficiency [1]. For example, the paper [4] tries to optimize remaining data rate, communication latency and the number of adopted physical nodes in a selected subnetwork. Their approach formalizes a service request and embeds a service chain considering specific resource requirements of each VNF.

Considered networks could be an intra-datacenter network, which focuses on the internal topology of a datacenter [5], or an inter-datacenter network, which represents interconnections between multiple datacenters that play the role of service nodes [6]. According to the environment, the topology of a given physical network varies. For example, the paper [5] suggests a NFV deployment based on user requirements and traffic flows in an intra-datacenter network. Given a 3-tier tree topology, their approach aims to alleviate the east-west traffic inside the network with a consideration of the usage of computation resource.

When deploying multiple datacenters as service nodes, some metrics over the location of a terminal node in each subnetwork could be considered. Typically, the latency metric for communications in a SFC is discussed in terms of the created path between given source and sink nodes with a focus on the objective of reducing the latency between service nodes through which the SFC traverses between the source terminal node and the sink terminal node. However, a positional relation between terminal nodes and datacenter nodes could also affect the latency metric. In other words, the placement of terminal nodes in each subnetwork could be another significant factor for the problem.

Hence, this paper tackles an instance of the SFC-EP which tries to minimize the distance-based latency between terminal node and datacenters in an inter-datacenter network. The SFC-EP in this paper composes a set of subnetworks called *subtopologies* focusing on the distance between terminal nodes and datacenter nodes. A variant of the SFC-EP named Subtopology Composition Problem (SCP) is formulated as a graph optimization problem, assuming that the subtopology is represented as a tree topology. Additionally, the intractability of the problem is proved by showing the inclusion of the Sterner Minimum Tree Problem (SMTP). A heuristic algorithm is also designed by partially exploiting an approximation algorithm for the SMTP. Furthermore, simulations are conducted to evaluate the performance of the proposed method in different graph models.

II. Problem Formulation

A. Assumptions

This paper assumes that a service request is specified as a SFC that consists of a set of VNFs, without any ordered dependencies between them. For each service request, we must

978-15386-3652-7/18/$31.00 ©2018 IEEE

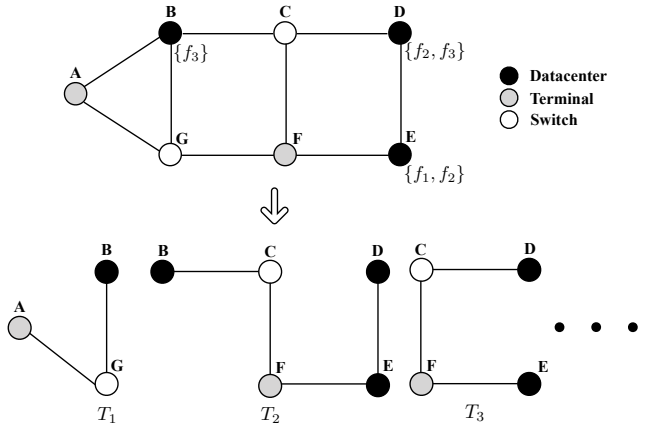

Fig. 1. An illustration of service function chain embedding, where datacenter B, D and E have VNF list $\{f_3\}$, $\{f_2, f_3\}$ and $\{f_1, f_2\}$, respectively. Subtopologies are composed from a given graph described at the topside.

find a subtopology of the network that interconnects the nodes on which the VNFs of the SFC are deployed. The subtopology ensures that all VNFs in the SFC are reachable from terminal nodes and from other VNFs in the SFC. We assume that the subtopology is represented as a connected acyclic graph, i.e. a tree as illustrated by T_1, T_2 and T_3 in Fig. 1 so as to simplify the problem. Even when a tree topology is considered, the problem is still quite intractable, as discussed in Section III.

In our model, a datacenter is considered as a large computational unit, and we do not look into its internal topology. Each datacenter node is able to support a given subset of VNF types and has a capacity constraint that limits the number of VNF instances that can be instantiated within the datacenter. In addition, multiple datacenters could provide the same type of VNF, which increases the complexity of the problem.

Furthermore, we assume that one of terminal nodes in a subtopology fulfills the role as both the starting and ending points of the network flow associated with the service request. Thus, each subtopology includes at least one terminal node as well as datacenter nodes and switch nodes. For communications within each subtopology, the datacenters are traversed starting and ending at a terminal node in a manner such that the efficient traversal is done without redundant intersects by passing all edges only two times. For example, consider the subtopology T_3 in Fig. 1. The datacenters are traversed in the order $F-C-D-C-F-E-F$, not $F-C-F-E-F-C-D-C-F$.

B. Preliminaries

For a given graph $G=(V,E)$ with a vertex set $V=\{v_i\}$ and an edge set $E=\{e_k\}$, let a path from v_n to v_m, $P_{n,m}$, be a finite alternating sequence of vertices and edges $(v_n, e_k, \ldots, v_m)$. When a vertex v_i and an edge e_k belong to a path $P_{n,m}$, they are represented as $v_i \in P_{n,m}$ and $e_k \in P_{n,m}$.

The length of a path is defined as the total edge weight $\sum_{e_k \in P_{n,m}} w(e_k)$ where a weight function is denoted as $w : E \longrightarrow \mathbb{R}^+$. Furthermore, the distance between v_i and v_j, denoted as $d(v_i, v_j)$, is defined as the minimum length of all paths between v_i and v_j.

A tree T in a graph G is a connected acyclic graph in which any pair of vertices in T is connected by one path. When the acyclic graph covers all vertices in a graph G, tree T is called a spanning tree.

C. Modeling

A substrate network is represented as a graph $G=(V,E)$, whose vertices V and edges E correspond to communication nodes and links, respectively. Particularly, vertices represent three types of nodes: switch nodes $S=\{s\}$, terminal nodes $H=\{t\}$, and datacenter nodes $D=\{d\}$, thus $V=S \cup H \cup D$.

Each datacenter node d_i has a list of VNFs $F(d_i)$ which they can accommodate, where $F : D \longrightarrow \Delta$, and a capacity $c(d_i)$, where $c : D \longrightarrow \mathbb{R}^+$. The family Δ represents all possible combinations of VNFs f_j. The capacity is defined as the maximum number of VNFs which d_i can provide. The subtopology is represented as a tree generated from a graph G, $T_k = (V(T_k), E(T_k)) \in \mathcal{T}$ where $V(T_k) \subseteq V$ and $E(T_k) \subseteq E$. In addition, a service request is described as $r_k = \{f_l\} \in R$, where R denotes a request set.

D. Problem Statement

The Subtopology Composition Problem is to generate a subtopology set that includes terminal nodes and datacenter nodes capable of satisfying each VNF request. The problem is formulated as an optimization problem aimed at minimizing the distance-based latency for communications in each subtopology while also satisfying the correctness of the composed subtopology, which is defined as follows.

The composed subtopology set is said to be correct when the following three conditions are satisfied.

1) For every subtopology $T_k \in \mathcal{T}$, at least one terminal node $t \in H$ exists:

$$\forall T_k \in \mathcal{T}, \exists t \in H \text{ such that } t \in H(T_k) \subseteq V(T_k), \quad (1)$$

where $H(T_k)$ is a set of terminal nodes included in a subtopology.

2) For every subtopology $T_k \in \mathcal{T}$, a set of datacenter nodes $D'_k \subseteq D$ that satisfies the corresponding VNF request r_k exist:

$$\forall T_k \in \mathcal{T}, \exists D(T_k) \subseteq V(T_k) \text{ such that } r_k \subseteq \cup_{d_i \in D(T_k)} F(d_i). \quad (2)$$

3) For all datacenter nodes $d_i \in D$, the capacity constraint is satisfied by the subtopology set.

$$\forall d_i \in D, \sum_{T_k \in \mathcal{T}} \alpha(d_i, T_k) \leq c(d_i), \quad (3)$$

where a VNF provision function $\alpha : D \times \mathcal{T} \longrightarrow \mathbb{R}^+$ returns the number of VNFs that datacenter d_i provides to a tree T_k.

The first two statements assure that each subtopology has a terminal node and datacenter nodes satisfying the request. The last statement guarantees that the number of provisioned VNFs does not exceeds the capacity in each datacenter. The

subtoplology composition is not correctly completed in the absence of these conditions.

From the assumptions disccussed in Section II-A, a traversal in a subtopology T_k requires passing through all edges two times, whose cost is represented as $2 \times \sum_{e_l \in E(T_k)} w(e_l)$. Hence, the distance-based latency in each subtopology can be alleviated by minimizing the total edge weights of subtopologies. Therefore, our objective function is represented as the sum of total edge weight over all subtopologies $T_k \in \mathcal{T}$:

$$c(\mathcal{T}) := \sum_{T_k \in \mathcal{T}} \sum_{e_l \in E(T_k)} w(e_l). \tag{4}$$

Based on these discussions, the SCP is now defined as below.

Problem 1. *Subtopology Composition Problem (SCP):* Given a graph $G = (V = S \cup H \cup D, E)$, a weight function on edges $w : E \longrightarrow \mathbb{R}^+$, a VNF list function $F : D \longrightarrow \Delta$, a capacity function $c : D \longrightarrow \mathbb{R}^+$ and a set of VNF requests $R = \{r_k\}$, the problem is to find a set of subtopologies $\mathcal{T}$ such that the function $c(\mathcal{T})$ is minimized while the correctness of the subtopology set is satisfied.

III. Intractability of the SCP

This section describes the intractability of the SCP by establishing that the problem includes a well-known $\mathcal{NP}$-complete problem called the *Steiner Minimum Tree Problem*.

Problem 2. *Steiner Minimum Tree Problem (SMTP):* Given a graph $G = (V, E)$, a subset $V' \subseteq V$ and a weight function on edges $w : E \longrightarrow \mathbb{R}^+$, find a tree T' in G that contains all required vertices $V' \subseteq V$ and minimizes the cost function defined as follows.

$$g(T') := \sum_{e_r \in E(T')} w(e_r). \tag{5}$$

Theorem 1. *The Subtopology Composition Problem is $\mathcal{NP}$-complete when a subtopology is represented as a tree.*

Proof: Consider the case where only one request r_k is given in the SCP, i.e. $|R| = 1$. Let a vertex set V' correspond to a terminal node and datacenter nodes that have to be included in a subtopology T_k. Because there is only one subtopology T_k to be created in the SCP, the cost function $c(\mathcal{T})$ corresponds to the cost function $g(T')$.

Additionally, the SCP requires an extra phase when composing a tree. Whereas V' is already given in the SMTP, the SCP tries to find an appropriate set of terminal node and datacenter nodes for a subtopology. Therefore, the SMTP is included in the SCP even when $|R| = 1$.

Because the SMTP is known to be $\mathcal{NP}$-complete, the SCP is also $\mathcal{NP}$-complete. ∎

This statement holds iff the size of vertex set V' is at least 2 because a subtopology possesses one or more terminal nodes and one or more datacenter nodes. The SCP requires a repetition of finding a suitable vertex set and obtaining a Steiner tree as many times as the number of requests $|R|$ so as to minimize the cost function $c(\mathcal{T})$. The number of combinations of terminal nodes and the number of datacenter nodes grows rapidly with the numbers of terminal nodes, datacenter nodes, and the number of VNFs that each datacenter provides.

Algorithm 1 SMTP-based composition algorithm

Input: a graph $G = (V = S \cup H \cup D, E)$, an edge weight function $w : E \longrightarrow \mathbb{R}^+$, a VNF list function $F : D \longrightarrow \Delta$, a capacity function $c : D \longrightarrow \mathbb{R}^+$, a set of service requests $R = \{r_k\}$, a constant q

for $r_k \in R$ **do**
  for $t \in H$ **do**
    $X_t \leftarrow \{x \mid x \in V \wedge d(t, x) \leq q\}$
    $X'_t \leftarrow X_t \cap D \cap \{x \mid F(x) \cap r_k \neq 0 \wedge c(x) > 0\}$
  end for
  $\eta \leftarrow t$ with maximum $\frac{|X'_t|}{|X_t|}$
  $V(T_k) \leftarrow \{\eta\}$
  for $f_j \in r_k$ **do**
    $A \leftarrow D \cap \{x \mid f_j \in F(x) \wedge c(x) > 0\} \backslash D(T_k)$
    $\hat{P} \leftarrow$ a shortest path $P_{v,x}$ from any $v \in V(T_k)$ to any $x \in A$
    $\theta \leftarrow$ an end point x of $\hat{P}$, $x \in A$
    $V(T_k) \leftarrow V(T_k) \cup V(\hat{P})$
    $E(T_k) \leftarrow E(T_k) \cup E(\hat{P})$
    $c(\theta) \leftarrow c(\theta) - 1$
  end for
  add T_k to $\mathcal{T}$
end for

Output: a set of subtopologies $\mathcal{T} = \{T_k\}$ minimizing $c(\mathcal{T})$

IV. Heuristic Algorithm

In this section, a heuristic for the SCP is proposed in the context of an approximation algorithm for the SMTP [7]. The approximation algorithm for the SMTP arbitrarily selects a vertex from required vertices and grows a tree by adding the shortest path between already appended vertices and required vertices until all necessary vertices are covered.

As illustrated in Algorithm 1, the algorithm consists of two phases for each service request: obtaining a suitable terminal node and growing a tree by adding datacenter nodes so as to satisfy the request. The terminal node for the SFC is determined by selecting a terminal node that is located close to a large number of datacenters. We define the datacenter density for a terminal node $t \in H$ as $\frac{|X'_t|}{|X_t|}$, where X_t is the set of vertices within a distance q of terminal node t, and X'_t is the set of active datacenters in X_t that can provision at least one VNF in a request r_k. The terminal node with the highest datacenter density is selected as the terminal node η for the SFC. Next, for each VNF f_j in a request r_k, we obtain an active datacenter θ which is able to support VNFs of type f_j, and which is the closest to any of the vertices $V(T_k)$ of the tree T_k. The datacenter node θ is added into the tree along with the shortest path $\hat{P}$ between θ and T_k. Then, the capacity $c(\theta)$ of datacenter θ is updated. These steps are repeated until the tree contains all datacenters for satisfying the request r_k.

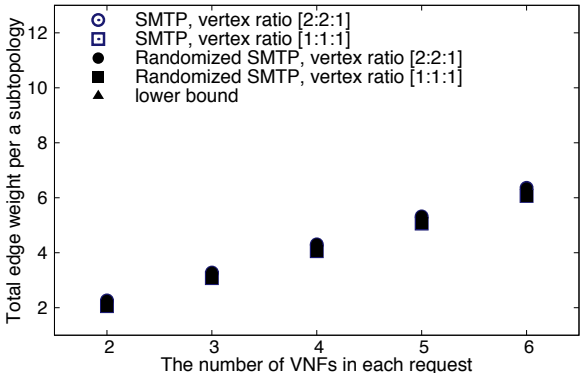

Fig. 2. Average total edge weight $\frac{1}{|\mathcal{T}|}c(\mathcal{T})$ vs the number of required VNFs $|\overline{r}|$ in each service request in dense NWS graphs ($p = 0.1$).

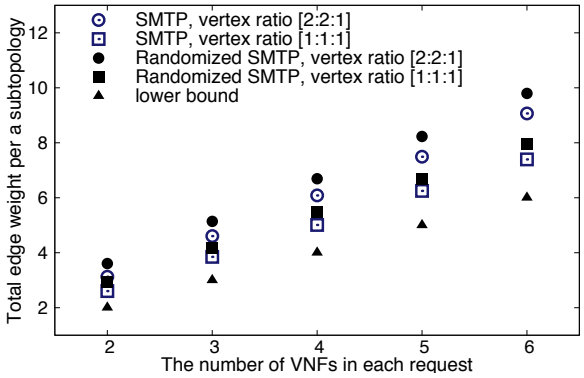

Fig. 3. Average total edge weight $\frac{1}{|\mathcal{T}|}c(\mathcal{T})$ vs the number of required VNFs $|\overline{r}|$ in each service request in sparse NWS graphs ($p = 0.01$).

As a result of this algorithm, all leaves of a composed tree are datacenters from M or a terminal node. The rest of the datacenters in M are placed on the paths from leaves to the root along with switch nodes, terminal nodes, and datacenter nodes in $V \backslash \{\eta, M\}$.

An example can be demonstrated by T_2 in Fig. 1, supposing that one request r_2 is given as $r_2 = \{f_1, f_2, f_3\}$ and $q = 2$. We follow the following steps of the algorithm to create a subtopology for the request. 1) Calculate the datacenter density of each terminal node. The terminal node A has $\frac{1}{5} = 0.2$ and the terminal node F has $\frac{3}{7} \approx 0.42$. 2) Select terminal node F and add it to the tree $V(T_k)$. 3) Now each datacenter node which is within distance q = 2 and which has functions required by the request is added to the tree $V(T_k)$. Thus, datacenter nodes E, D and B are added in order. As a result, this gives us subgraph T_2 as the subtopology.

The computation time is evaluated for the case in which the Floyd-Warshall algorithm, which requires $O(V^3)$, is used in advance to calculate shortest paths between all vertex pairs. Firstly, the calculation of datacenter densities for each terminal requires $O(|H|(|V|+|D|))$ and thus has complexity $O(|V|^2)$. Next, in line 9-10, the datacenter search and shortest path search are done with $O(|D| + |V(T_k)||A|)$, i.e. $O(|V|^2)$. Additionally, tree aggregation based on the shortest path requires $O(|V|)$. Thus, the calculations within the loop from line 8 to line 15 requires $O(|\hat{r}||V|^2)$ operations, where $\hat{r}$ is the maximum size of request r_k. Therefore, the worst-case complexity of this algorithm is $O(|V|^3 + |R||\hat{r}||V|^2)$.

V. Simulations

In order to assess the performance of the composition algorithm regarding distance between a terminal node and datacenter nodes, our simulations examine the heuristic with its randomized version in four different types of graphs with different vertex ratios. The randomized SMTP-based algorithm differs slightly in that a terminal node is determined based on uniform random selection. The vertex ratios describe proportions of switch nodes, terminal nodes and datacenter nodes in a graph, represented as $[|S|:|H|:|D|]$. The simulator is designed with Java and the JUNG package.

A. Service Requests and Datacenters

The simulator generates 10 service requests consisting of randomly obtained VNFs among 5 types of VNFs using the uniform distribution. We assume that the network always possesses enough resources to satisfy a given request set. The consistent number of VNFs $|\overline{r}|$ in each request changes over the range $[2, 6]$ in the experiments so as to express the worst-case environment. With the increasing number of VNFs in a request, the subtopology grows in size to reach datacenters that provide additionally required VNFs. Furthermore, capacities and the set of VNFs provided by each datacenter are randomly generated. Our simulations assume that a datacenter node maps to at most a single VNF in a request.

B. Network Models

This paper considers the following graph models, in which all edge weights are uniformly set in the simulations.

1) Newman Watts Strogatz (NWS) random graph: A NWS random graph has a small world property, scale free property and clustering property. The graph is composed by creating a cycle with all n vertices and by connecting k nearest vertices with each other. Then, additional edges are created with the probability p between each vertex pair. We consider two NWS random graphs: dense graph ($n = 80$, $k = 2$, $p = 0.1$) and sparse graph ($n = 80$, $k = 2$, $p = 0.01$).

2) Lattice graph: A lattice graph is a patterned graph which forms regular tilings. The lattice does not have the small world property because the average shortest path largely grows as the vertex size in the graph increases. The number of vertices is set as 81 in the simulations.

3) Real Network Topology: A real network topology, called Intellifiber is used in our simulations. Intellifiber represents an optical network with 72 vertices and 95 edges in the United States. The structure information is provided by The Internet Topology Zoo [8].

C. Results and Discussions

Fig. 2 - Fig. 4 depict simulation results regarding the average total edge weight among subtopologies in dense NWS graphs, sparse NWS graphs, lattice graphs, and Intellifiber. The results

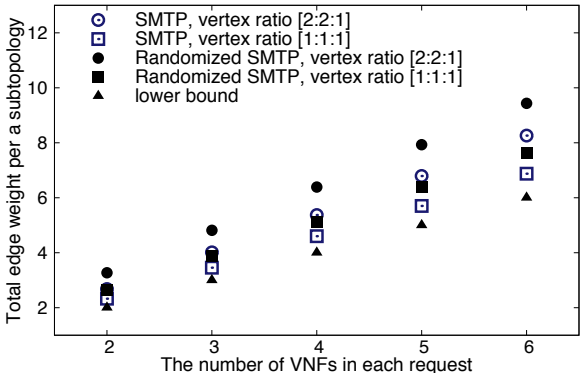

Fig. 4. Average total edge weight $\frac{1}{|\mathcal{T}|}c(\mathcal{T})$ vs the number of required VNFs $|\overline{r}|$ in each service request in lattice graphs.

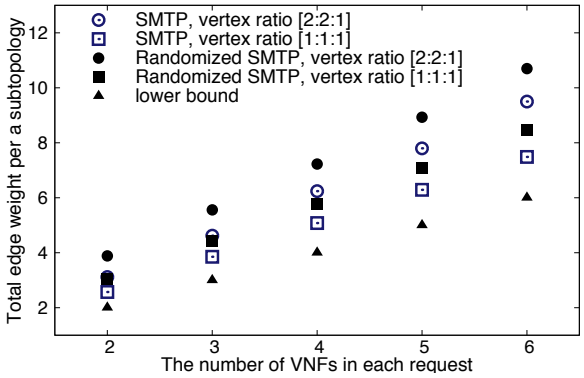

Fig. 5. Average total edge weight $\frac{1}{|\mathcal{T}|}c(\mathcal{T})$ vs the number of required VNFs $|\overline{r}|$ in each service request in Intellifibers.

are compared with lower bound values. The lower bound for $\frac{1}{|\mathcal{T}|}c(\mathcal{T})$ is equal to $|\overline{r}|$, which is the minimum number of edges that need to be augmented in a tree to have one terminal node and datacenters.

As can be seen, both algorithms equally demonstrate almost the same results as the lower bound values in dense NWS graphs with different vertex ratios. There is no large difference in their performance because the average shortest path is usually short in a random graph. In other words, more datacenters can be easily reached from each terminal node.

Meanwhile, the SMTP-based algorithm marks lower edge weights than the randomized algorithm in sparse NWS graphs. Since the average shortest path of sparse NWS is longer than dense NWS, each terminal node has different reachability to datacenters. Our heuristic is likely to find necessary VNFs without having a large spanning tree, by composing trees in locations populated with necessary datacenters. Also, the results obtained by our method are not exceptionally larger than the lower bound values in sparse NWS graphs.

The similar discussions for sparse NWS hold for the results in lattice and Intellifiber. Since the average shortest path of lattice tends to be longer than dense NWS, the results are similar to the sparse NWS. Also, the same is true of Intellifiber that represents real-world communication networks.

The total edge weight in each tree increases as the number of required VNFs increases because each tree necessarily seeks more datacenters to include. Also, the results shown by square dots, where a graph consists of more datacenters, tend to be better than the results shown by circle dots. Since the closeness of interconnection between datacenter nodes changes, edge costs are easily alleviated in trees when there are a larger number of datacenter nodes in a graph. In fact, this is true for all of the results for all four graph models.

Once a tree is composed, it is necessary to determine how to traverse the tree starting and ending at the terminal node without unnecessary edge visits, i.e., the number of visited edges should be $2 \times |E|$. Such an algorithm can be achieved, for example, by running Depth First Search from a selected terminal node and adding a step to go back to the terminal node after traversing all of the datacenter nodes.

VI. Conclusion

This paper has addressed the Subtopology Composition Problem (SCP), generalizing the service function chain embedding problem (SFC-EP) that tries to minimize the distance based latency in SFCs. The SCP is proven to be $\mathcal{NP}$-complete by demonstrating that the Steiner Minimum Tree Problem (SMTP) is contained in the SCP. A heuristic algorithm is designed utilizing an approximation algorithm for the SMTP, and the time complexity is assessed. Furthermore, the simulations show the effectiveness of the proposed algorithm by comparing it with the randomized version of the algorithm in different graph models with different vertex ratios. One future work is to consider order dependencies between VNFs when composing a subtopology and to determine an appropriate traversal of VNFs in the subtopology.

Acknowledgment

The authors would like to appreciate Dr. Qiong Zhang at Fujitsu Laboratories of America, Inc. for her valuable suggestions for this work. Also, this work was supported by the Murata Science Foundation.

References

[1] R. Mijumbi *et al.*, "Network function virtualization: State-of-the-art and research challenges," *IEEE Communications Surveys & Tutorials*, vol. 18, no. 1, pp. 236–262, 2016.

[2] N. Bouten *et al.*, "Semantically enhanced mapping algorithm for affinity constrained service function chain requests," *IEEE Transactions on Network and Service Management*, 2017.

[3] P. Veitch *et al.*, "An instrumentation and analytics framework for optimal and robust nfv deployment," *IEEE Communications Magazine*, vol. 53, no. 2, pp. 126–133, 2015.

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

[5] P.-W. Chi *et al.*, "Efficient nfv deployment in data center networks," in *Communications (ICC), 2015 IEEE International Conference on*. IEEE, 2015, pp. 5290–5295.

[6] M. Xia *et al.*, "Network function placement for nfv chaining in packet/optical datacenters," *Journal of Lightwave Technology*, vol. 33, no. 8, pp. 1565–1570, 2015.

[7] F. K. Hwang *et al.*, *The Steiner tree problem*. Elsevier, 1992, vol. 53.

[8] S. Knight *et al.*, "The internet topology zoo," *IEEE Journal on Selected Areas in Communications*, vol. 29, no. 9, pp. 1765–1775, 2011.