Network Function Deployment with Balanced Server and Link Resources in Tree Topologies

Yang Chen and Jie Wu
Center for Networked Computing, Temple University, USA
Email: {yang.chen, jiewu}@temple.edu

***Abstract*—Network function virtualization (NFV) enables flexible implementation of software middleboxes (network functions) as virtual machines running on standard servers. However, the flexibility also creates a challenge for efficiently placing such VNFs, due to the availability of multiple hosting servers and VNF capability of changing traffic volumes. In this paper, we address the placement problem of a special type of VNF with the traffic-diminishing effect (e.g., spam filters). We aim at minimizing the total bandwidth consumption of flows by placing at most a pre-determined number of VNFs. First, we formulate the Traffic-diminishing VNF Placement (TVP) as an optimization problem. We propose an optimal strategy for the tree-structured networks. Next, we extend the algorithm to the directed acyclic graph (DAG) topology. Then an approximation algorithm is designed after we prove the NP-hardness of our problem in a general topology. Extensive simulations are conducted on CAIDA dataset to evaluate the performance of our proposed algorithms in various scenarios.**

***Index Terms*—VNF placement, bandwidth consumption, traffic-diminishing, NFV.**

I. Introduction

Network function virtualization (NFV) changes the way that we implement software middleboxes (network functions), from expensive hardwares to software functions. These VNFs, also called virtual network functions (VNFs), run on switch-connected commodity servers. The choice of VNF service location is complicated by the availability of multiple hosting servers and the traffic-changing effect of VNFs. For example, the Citrix CloudBridge Wide Area Network optimizer reduces traffic volume by up to 80% by compressing traffic [1]. Redundancy eliminator would reduce the difference between peak and minimum traffic more significantly by 25% for the university and by 52% for the data center [2]. When it comes to security, *spam filters* intercept all suspicious flows by cutting down 100% traffic rates. Link bandwidth is a valuable resource in most networks such as data center networks [3], WANs [4], and LANs [5]. Additionally, server resources, such as CPU and memory, are valuable and finite in today's networks [6]. Efficiently placing such traffic-diminishing VNFs is important in today's high-performance networks. However, most researches only focus on reducing VNF setup cost by sharing VNFs without considering the bandwidth consumption.

In this paper, we focus on the placement of traffic-diminishing VNFs. We aim at minimizing the total flow bandwidth consumption under the constraints of a given number of VNFs. The total flow bandwidth consumption is the sum of each flow's bandwidth consumption, which is the sum of its occupied bandwidth on each link along its path. Intuitively, placing traffic-diminishing VNFs as early as possible, consumes less link bandwidth resources. However, this simple strategy reduces the VNFs' sharing opportunities and forces the launch of more VNFs. There is a trade-off between saving more link bandwidth and sharing more VNFs among flows. When we do not place any VNFs along its path, the occupied bandwidth of a flow on each link along its path remains the same as its initial rate and no decrement happens. Similarly, when there is a VNF placed on the vertex of a flow's source, the traffic rate of the flow diminishes in its earliest position and its bandwidth consumption is the minimum. However, more VNFs are needed if we place one on every flow's source. Thus, there is usually a constraint on the maximum number of VNFs that we can place.

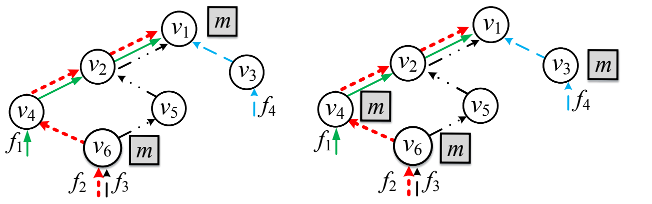


Fig. 1: A motivating example (Initial traffic rates of f_1,f_2, f_3,f_4 are 2,4,2,2 and VNF m's traffic-diminishing ratio is 0.5).

The traffic-changing effect due to the VNF placement may complicates the scheduling policy. We illustrate the complexity by showing an example in Fig. 1. The cylindrical nodes are switches and the cubical nodes are VNFs. VNFs are assigned to servers (not shown in the figures) that are attached to switches. All flows need to be served by a same type of VNF before reaching their destination v_1. The traffic-changing ratio, which is the proportion of a flow's traffic rate before and after being processed by the middlebox, is 0.5 (diminishing traffic). There are four flows, f_1, f_2, f_3, and f_4, and their initial traffic rates are $2, 4, 2,$ and 2, respectively. The line thickness indicates the traffic rate. Their sources, destinations and paths are shown in the Fig. 1. Their paths are pre-determined, shown in different types of lines. We are only allowed to place two VNFs. Fig. 1(a) shows the optimal placement with only two VNFs allowed. A VNF is placed on v_6 to process f_2 and f_3, and another one on v_1 processes f_1 and f_4. The bandwidth consumption of f_2 is $0.5 * 4 * 3 = 6$. The total bandwidth consumption is the sum of each flow's bandwidth consumption, which is calculated as $2*2+0.5*(4+2)*3+2*1=15$. Though both VNFs cover f_2 and f_3, we only process them on their sources v_6 in order to reduce their bandwidths as early as possible. If allowed to place more than three VNFs, shown in Fig. 1(b), we should place one VNF on each flow's source, which reduces the bandwidth consumption in the earliest locations for all flows. Additionally, the total flow bandwidth consumption is calculated as $0.5*2*2+0.5*(4+2)*3+0.5*2*1=12$, which is the minimum of all cases.

In this paper, we address the placement challenge of VNFs, and propose solutions for VNFs with traffic-diminishing effects in order to minimize the total flow bandwidth consumption. We focus on placing a single type of VNF for all flows. First, we formulate the Traffic-diminishing VNF Placement (TVP) as an optimization problem. We start with two special topologies of tree-structured networks and directed acyclic graph networks and introduce their corresponding efficient strategies. We prove the NP-hardness of the problem in a general topology and propose an efficient heuristic algorithm with a performance-guaranteed ratio. The results can be easily extended to traffic-expanding ones. Extensive simulations are conducted to evaluate the performance of our proposed algorithms in various scenarios.

Our main contributions are summarized as follows:

- We define a new optimization problem in Section III, called Traffic-diminishing VNF Placement (TVP), that minimizes the total bandwidth consumption in a given network using a fixed number of VNFs. The solution to this problem is particularly useful in allocating spam filters to minimize spam traffic using a fixed number of spam filters.
- We propose two efficient greedy algorithms for the tree-structured networks in Section IV and DAG topologies in Section V. Their time complexities are $O(||V|^2 \log|V|)$ and $O(|V|^3 \log|V|)$, respectively, where V is the vertex set.
- We prove the NP-hardness of the VNF placement in general topologies in Section VI (Theorem 3). A heuristic algorithm with a performance-guaranteed ratio of $1-\frac{1}{e}$ (Theorem 4) is proposed with a time complexity of $O(k|V|\log|V|)$.
- Extensive simulations in Section VII are conducted to evaluate the efficiency of our algorithms with a publicly available CAIDA data set [7].

The remainder of this paper is organized as follows. Section II surveys related works. Section III describes the model and formulates the problem. In Sections IV and V, we handle two special cases of tree and DAG topologies. Section VI includes the analysis of the problem hardness in a general topology and introduces a heuristic solution. Section VII includes the experiments. Finally, Section VIII concludes the paper.

II. Related Work

NFV frameworks have recently drawn a lot of attention, especially in the area of VNF placement problem. We give a brief review of state-of-the-art works. For placing a single VNF for all flows, Casado *et al.* [8] propose a placement model and present a heuristic algorithm to solve the placement problem. Sang *et al.* [9] study the joint placement and allocation of a single middlebox, where flows can be split and served by several VNFs. They propose several performance-guaranteed algorithms to minimize the number of VNFs. However, neither study considers VNF traffic-changing effects or focuses on the bandwidth objective.

For placing multiple types of VNFs, most research on VNF placement focus on placing a totally-ordered set as a service chain. Mehraghdam *et al.* [10] propose a context-free language to formalize the chaining of VNFs and describe the VNF resource allocation problem as a mixed integer quadratically constrained program. Rami *et al.* [11] locate VNFs in a way that minimizes both new VNF setup costs and the distance cost between VNFs and flows' paths. They provide near optimal approximation algorithms to guarantee a placement with a theoretically proven performance. Both [12] and [13] aim to maximize the number of requests for each service chain. Kuo *et al.* [12] propose a systematic way to tune the proper link consumption and the VNF setup costs in the joint problem of VNF placement and path selection. Li *et al.* [13] present the design and implementation of NFV-RT, a system that dynamically provisions resources in an NFV environment to provide timing guarantees so that the assigned flows meet their deadlines. However, none of them consider the traffic-changing effect.

Ma *et al.* [14] are the first to take the traffic-changing effects into consideration. It targets load balancing instead of VNF setting-up costs. It proposes a dynamic programming based algorithm to place a totally-ordered set, an optimal greedy solution for the VNF placement of a non-ordered set, and proves the NP-hardness of placing a partially-ordered set. However, this work only processes a single flow and always builds new, private VNFs without sharing with other flows, which excessively increases the setup costs of VNFs. Sharing VNFs among multiple flows makes the placement more challenging. In this paper, we consider not only the traffic-changing effects, but also placing VNFs for multiple flows.

III. Network Model and Problem Formulation

A. Network Model

Our scenario is based on a directed network, $G=(V,E)$, where V is a set of vertices (i.e., switches), and $E \subseteq V^2$ is a set of directed edges (i.e., links). We use v to denote a vertex (node) and use e to denote an edge (link). We assume each link is bidirectional and has enough bandwidth to hold all bypass flows with their initial traffic rates, which eliminates congestion and ensures that the routing of all flows is successful. This is because routing failure is not our concern.

VNF m has a pre-defined traffic-changing ratio $\lambda \geq 0$ that serves as the ratio of a flow's traffic rate before and after being processed by m. In this paper, we focus on the placement of the traffic-diminishing middlebox, whose traffic-changing ratio is $\lambda \leq 1$, i.e., spam filters. We use an indicator function, m_v, to represent whether there is a VNF of m placed on v. If a VNF is placed on v, $m_v = 1$; otherwise, $m_v = 0$.

We are given a set of unsplittable flows $F = \{f\}$ because flow splitting may not be feasible for applications that are sensitive to TCP packet ordering (e.g. video applications). Additionally, split flows can be treated as multiple unsplittable flows. We use f to denote a single flow that has an initial traffic rate of r_f. Its path p_f is an ordered set of edges from f's source, src_f, to its destination, dst_f. All flows' paths are predetermined and valid. We use $b_e(f)$ and $b(f)$ to denote f's traffic rate (occupied bandwidth) on e and bandwidth consumption on all edges along its path. Then the total bandwidth consumption of f is $b(f) = \sum_{e \in p_f} b_e(f)$. We introduce another indicator function, f_v, to express that the flow f uses the VNF m placed on the vertex v. If a VNF is placed on v, $f_v = 1$; otherwise, $f_v = 0$. We assume each packet in a flow is served by a type of VNF only once, even if there are several VNFs along its path. This is because being served by VNFs will add an extra transmission delay, which should be avoided as much as possible, in order to improve the network performance.

We use $\mathcal{P}$ and $b(\mathcal{P})$ to denote the placement plan and the total bandwidth consumption of all flows being processed by the placed VNFs of $\mathcal{P}$, respectively. We have that $\mathcal{P} = \{v \mid m_v = 1, \forall v \in V\}$ and $\mathcal{P}$ is a subset of V, i.e., $\mathcal{P} \subseteq V$, which contains all vertices with placed VNFs. We are given a priori the maximum number of VNFs that are allowed to be placed in the whole network, denoted by k. For ease of reference, we summarize notations in Tab. I.

B. Problem Formulation

Based on the above network model, our Traffic-diminishing VNF Placement (TVP) problem includes two properties: *feasibility* and *optimality*. The feasibility of our TVP problem is whether we are able to use at most k VNFs to ensure all flows being processed. We formulate the optimality of the VNF placement as a mathematical optimization problem on minimizing the total flow bandwidth consumption as:

$$\min \quad b(\mathcal{P}) = \sum_{f \in F} b(f) = \sum_{f \in F} \sum_{e \in p_f} b_e(f) \tag{1}$$

$$\text{s.t.} \quad |\mathcal{P}| = \sum_{v \in V} m_v \leq k \qquad \forall m \in M \tag{2}$$

$$\sum_{v \in p_f} f_v = 1 \qquad \forall f \in F \tag{3}$$

$$f_v \leq m_v \qquad \forall v \in V \tag{4}$$

$$m_v = \{0, 1\}, f_v = \{0, 1\} \qquad \forall v \in V \tag{5}$$

Eq. (1) is our objective: minimizing the total bandwidth consumption, which is the sum of all flows' bandwidth consumption. A flow's bandwidth consumption is the sum of its occupied bandwidths on each link along its path. Eq. (2) states that the total number of placed VNFs is no more than k. Meanwhile, $| \cdot |$ denotes the set cardinality. $|\mathcal{P}|$ is the total number of selected vertices with placed VNFs, which equals the sum of m_v, $\forall v \in V$. This is because $m_v = 1$ when a VNF is placed on v; Otherwise, $m_v = 0$. Eq. (3) requires that each flow $f \in F$ be served by the VNF once and only once. Eq. (4) ensures that a flow only can use a VNF on v when there is one placed on v. Eq. (5) shows m_v and f_v can only be assigned the value either of 0 or 1.

TABLE I: Symbols and Definitions.

Symbols	**Definitions**
V, E, F	the set of vertices, edges, and flows
v, e, f, m	a vertex,an edge, a flow, and a VNF
src_f, dst_f, p_f, r_f	source, destination, path, initial rate of f
λ	traffic-changing ratio of VNF m
m_v	indicator function of placing m on v
f_v	indicator function of f using m on v
$b_e(f), b(f)$	f's traffic rate on e and its total bandwidth
$\mathcal{P}, b(\mathcal{P})$	placement plan and its total bandwidth
k	the maximum number of VNFs

IV. Solution for Tree-structured Networks

We start solving our TVP problem with the tree topologies. Networks with a tree-structured topology are extremely common in streaming services, content delivery networks (CDNs) [15], and tree-based tiered topologies like Fat-tree [16] or BCube [17] in data centers. More generally, data centers always use symmetric, hierarchical topologies to balance traffic load [3]. From the view of a top level switch, the topology can also be abstracted as two connected trees because of the bi-directional links [18]. Here we require that the sources of all flows are leaves and their destinations are the same as the root of the tree. As long as the VNF number constraint $k \geq 1$ and the destinations of all flows are the root, we can process all flows by placing a VNF on the root so that there does not exist the infeasibility situation. In this section, we propose an optimal solution for our TVP problem in tree topologies. First, we introduce a definition from graph theory.

Definition 1 (LCA): Lowest common ancestor (LCA) of two vertices v and w in an acyclic graph G is the lowest vertex that has both v and w as descendants. We define each vertex to be a descendant of itself. Thus, if v has a direct connection from w, w is the lowest common ancestor [19].

Take the Fig. 2 as an example. We list the LCA between every two nodes in Tab. II. Specifically, LCA of vertices v_4 and v_5 is v_2 because v_2 is the lowest node that has both of v_4 and v_5 as descendants. As v_1 is the ancestor of v_5, LCA of vertices v_1 and v_5 is v_1.

Next, we define $\Delta b(i, j)$ as the difference of the total bandwidth value when we delete two VNFs on v_i and v_j and place one on $LCA(i, j)$. The process of the deletion and placement is called *merge* in this paper. We propose our solution for the tree-structured networks as Greedy Solution for Trees (GST) algorithm, shown in Alg. 1. Line 1 initiates the placement plan by placing a VNF on every leaf vertex. Line 2 calculates the value of $\Delta(i, j)$ for each pair of vertices. Line 3 constructs the first min-heap. Lines 4-7 iteratively select the pair with the minimum value of $\Delta b(i, j)$ and merge the two

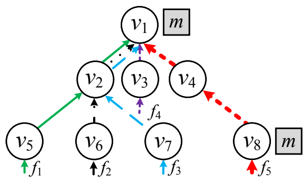


Fig. 2: A tree example.

i \ j	1	2	3	4	5	6
1	1	1	1	1	1	1
2	1	2	1	2	2	2
3	1	1	3	1	1	1
4	1	2	1	4	2	4
5	1	2	1	2	5	5
6	1	2	1	4	5	6

TABLE II: LCA(i, j).

VNFs by placing one on their LCA until the number of VNFs reaches k. In each round, we do merge to reduce the number of VNFs by one. The min-heap is updated by deleting pairs with v_i or v_j and inserting new pairs with $LCA(i, j)$. We also delete two vertices v_i and v_j from $\mathcal{P}$ and insert their LCA into $\mathcal{P}$. The placement $\mathcal{P}$ returns in line 8.

***Theorem** 1:* The GST algorithm is optimal for placing a limited number of VNFs in order to minimize the total bandwidth consumption in a tree topology. Its time complexity is $O(|V|^2 \log |V|)$.

Proof: We prove the optimality of Alg. GST by induction. First, we place one VNF on each leaf node as the initial placement plan, which is the optimal placement with the minimum bandwidth consumption for an arbitrary number of VNFs. This is because the traffic rate of each flow diminish from its source and the bandwidth consumption of each flow is the smallest for all possible placement plans. Suppose there are n leaves in the tree. When $k \geq n$, the placement returned by our algorithm is optimal. When $k = n - 1$, our algorithm merges two VNFs into placing one on their LCA for only one round with the minimum value of $\Delta b(i, j)$, $\forall v_i, v_j \in \mathcal{P}$. The value of $\Delta b(i, j)$ indicates the increment of the bandwidth consumption when we place one VNF on the LCA of v_i and v_j and delete two VNFs on them. When $k = n$, our placement has the minimum bandwidth consumption. And when $k = n - 1$, we increases the consumption least. Thus, the placement plan returned by our algorithm is optimal with the minimum bandwidth consumption when $k = n - 1$. Now assume our algorithm is optimal when $k = p$ and we prove the optimality when $k = p - 1$. For each merge, our algorithm increase the bandwidth consumption least. If it has the smallest bandwidth consumption when $k = p$, then it also has the smallest bandwidth consumption when $k = p - 1$, indicating its optimality. Thus, our algorithm is optimal.

The time complexity of Alg. GST is $O(|V|^2 \log |V|)$. Initially, there are $O(|V|/2)$ leaf vertices and $O((|V|/2)^2)$ pairs. Building a min-heap costs $O((|V|/2)^2 \log((|V|/2)^2)) = O(|V|^2 \log |V|)$. For the while loop, we need to run $O(|V|/2 - k) = O(|V|)$ rounds in order to reduce the number of VNFs from $O(|V|/2)$ to k. In each round, it at most takes $O(|V|)$ time to delete pairs with v_i or v_j and insert new pairs with $LCA(i, j)$. Thus, the total time complexity is $O(|V|^2 \log |V| + |V| \times |V|) = O(|V|^2 \log |V|)$. ∎

For better understanding, we use an example to show steps of running the GST algorithm in Fig. 2. There are five flows, all of whose initial traffic rates are 1 Mbps, except f_5 with 10 Mbps. Initially, $\mathcal{P} = \{v_3, v_5, v_6, v_7, v_8\}$, which has the minimum bandwidth consumption for all possible placements.

Algorithm 1 Greedy Solution for Trees (GST)

In: Sets of vertices V, edges E, and flows F, traffic-changing ratio λ and VNF number constraint k;
Out: The placement plan $\mathcal{P}$;

```
1: Initialize $\mathcal{P}$ as a set of all leaf vertices;
2: Calculate $\Delta b(i, j),\ \forall v_i, v_j \in P(i \neq j)$;
3: Construct a min-heap of $\Delta b(i, j), \forall v_i, v_j \in P(i \neq j)$.
4: while $|\mathcal{P}| > k$ do
5:     Merge the two VNFs with the minimum $\Delta b(i, j)$, $\forall v_i, v_j \in \mathcal{P}(i \neq j)$.
6:     Update the heap by deleting pairs with $v_i$ or $v_j$ and inserting pairs with $LCA(i, j)$.
7:     $\mathcal{P} = \mathcal{P} - \{v_i, v_j\} + \{LCA(i, j)\}$;
8: return The placement plan $\mathcal{P}$.
```

This is because the traffic rates of all flows are diminished from their sources and the bandwidth consumption of each flow is the smallest. If $k \geq 5$, since the while the loop does not need to run, the placement plan returned by Alg. GST is $\mathcal{P} = \{v_3, v_5, v_6, v_7, v_8\}$. If $k = 4$, one round of the while loop needs to run. There are $\binom{5}{2} = 10$ pairs. We calculate the value of $\Delta b(i, j)$ for each pair. For example, $\Delta b(5, 6) = 2$, $\Delta b(3, 5) = 3$ and $\Delta b(3, 8) = 21$. After calculating these ten pairs, we find that $\Delta b(5, 6)$, $\Delta b(5, 7)$ and $\Delta b(6, 7)$ all have the minimum value, 2. We can select either pair to merge. Here we select the pair with v_5 and v_6. We delete v_5 and v_6 from $\mathcal{P}$ and insert their LCA v_2 into $\mathcal{P}$. Then the placement plan returned by Alg. GST is $\mathcal{P} = \{v_2, v_3, v_7, v_8\}$. If $k = 3$, two rounds of the while loop need to run. The first round is the same as $k = 4$. In the second round, there are $\binom{4}{2} = 6$ pairs. We have $\Delta b(2, 3) = 3, \Delta b(2, 7) = 1,\ \Delta b(2, 8) = 22,\ \Delta b(3, 7) = 2,\ \Delta b(3, 8) = 21$ and $\Delta b(7, 8) = 21$. We delete v_2 and v_7 from $\mathcal{P}$ and insert their LCA v_2 into $\mathcal{P}$ since $\Delta b(2, 7) = 1$ is the minimum. Then the placement plan returned by Alg. GST is $\mathcal{P} = \{v_2, v_3, v_8\}$. Similarly, the placement plan is $\{v_1, v_8\}$ when $k = 2$ and the placement plan is $\{v_1\}$ when $k = 1$.

V. Solution for Directed Acyclic Networks

Before proposing our solution, we introduce the definition of the directed acyclic graph.

***Definition** 2 (DAG):* Directed acyclic graph (DAG) is a finite directed graph with no directed cycles. That is, it consists of finite vertices and edges, with each edge directed from one vertex to another, such that there is no way to start at any vertex v and follow a consistently-directed sequence of edges that eventually loops back to v again.

A tree is a special case of a DAG when each node has at most one incoming edge. Most hierarchical data center networks have DAG topologies [3], indicating its importance. Inspired by our strategy for tree topologies, we propose a solution, called Directed Acyclic Graph Technique (DAGT) algorithm, to solve our TVP problem in DAG topologies. We require the destinations of all flows are the same in order to ensure the property of feasibility. In Alg. DAGT, line 1

Algorithm 2 Directed Acyclic Graph Technique

```
In: Sets of vertices $V$, edges $E$, and flows $F$, traffic-changing
    ratio $\lambda$ and VNF number constraint $k$;
Out: The placement plan $\mathcal{P}$;
1: Initialize $\mathcal{P}$ as a set of all flows' sources and $i = 1$;
2: Sort $V$ in topological order level by level with $p_f, \forall f \in F$;
3: while $|\mathcal{P}| > k$ do
4:   if $|\mathcal{P}| - k \geq |L_{i-1}| - |L_i|$ and $|L_{i-1}| > |L_i|$ then
5:     $\mathcal{P} = \mathcal{P} - L_{i-1} + L_i$ and $i++$;
6:   else
7:     Call lines 3-7 in Alg. GST with multiple merges;
8: return The placement plan $\mathcal{P}$.
```

initializes the placement plan $\mathcal{P}$ as a set of all flows' sources and a variable i. Line 2 sort all vertices in topological order level by level corresponding to all flows' paths. We define the set of all vertices in the ith level as L_i. Lines 3-7 do the merge process until the total number of VNFs is no more than its constraint k. Here we extend the definition of *merge* in Section IV to *multiple merge*, which means deleting VNFs on multiple vertices (≥ 2) and place VNFs on LCAs. If the vertex number difference between current and next levels is less than the VNFs that are to be deleted, we directly move all vertices to the next level with fewer VNFs in line 5; otherwise, we do multiple merges with the least increment of the bandwidth consumption in line 7. Line 8 returns the placement plan $\mathcal{P}$.

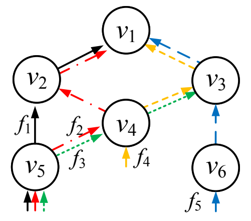
***Theorem** 2:* The time complexity of our Alg. DAGT is $O(|V|^3 \log |V|)$.

Proof: The topological sorting takes $O(|V| \log |V|)$. The while loop needs to run at most $O(|V| - k) = O(|V|)$ rounds. In each round, we either run line 5 or line 7. Doing merge in line 5 takes a constant time and the time complexity of calling Alg. GST is $O(|V|^2 |\log |V|)$ in line 7. So the time complexity of our Alg. DAGT is $O(|V|^2 \log |V| \times |V|) = O(|V|^3 \log |V|)$. ■

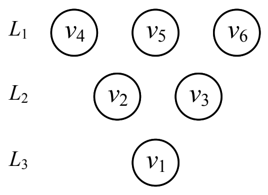
For a better understanding, we use an example in Fig. 3(a) to show steps of running the DAGT algorithm. There are six flows with a same traffic rate of 1. Initially, we place three VNFs on v_4, v_5, and v_6, respectively. The topological sorting result level by level is shown in 3(b). The vertex v_4 is a source of f_4 and a descendant of v_5. If the VNF number constraint is $k = 3$, no round of the while loop needs to execute and the placement plan $\mathcal{P} = \{v_4, v_5, v_6\}$ is returned. If $k = 2$, we have $|L_1| = 3$, $|L_2| = 2$, and $|L_1| - |L_2| = 1$, equals to $|P| - k = 3 - 2 = 1$. Then we delete all VNFs on L_1 and place VNFs to L_2. The placement plan $\mathcal{P} = \mathcal{P} - \{v_4, v_5, v_6\} + \{v_2, v_3\}$ is returned. Similarly, if $k = 1$, the optimal placement plan is $\mathcal{P} = \{v_1\}$.

VI. Solution for General Networks

We first show that in general network topologies, our TVP problem is NP-hard. After we define two funcions, a heuristic algorithm with an approximations ratio is proposed.



(a) A DAG topology.



(b) Topological order.

Fig. 3: A motivating example.

***Theorem** 3:* The feasibility of our TVP problem is NP-hard to check in a general topology.

Proof: We construct a polynomial reduction from the set-cover decision problem. Assume we have the network (V, E) and a set of flows $F = \{f\}$ that each flow requires to be processed by the same type of traffic-changing VNF m. We are given k VNFs. The feasibility of our TVP problem is whether we are able to use k VNFs to ensure all flows being processed. This problem is equivalent to the set-cover decision problem. The elements are all the flows $U = \{f, \forall f \in F\}$. A VNF m placed on a vertex v in the network can cover a set of flows whose path passes v, i.e. $S_v = \{f | v \in p_f\}$. We need to find the minimum number of subsets whose union equals the universe set. Since the set-over decision problem is NP-complete, our TVP problem is NP-hard. ■

Here are two newly-defined functions.

***Definition** 3 (decrement function):* The decrement function, denoted as $d(\mathcal{P})$, indicates the decrement of the total bandwidth consumption by a deployment plan $\mathcal{P}$, which satisfies $d(\mathcal{P}) = b(\mathcal{P}) - \sum_{f \in F} r_f \times |p_f|$.

***Definition** 4 (marginal decrement):* The marginal decrement, denoted as $d_{\mathcal{P}}(\mathcal{S}) = d(\mathcal{P} \cup \mathcal{S}) - d(\mathcal{P})$, indicates the additional bandwidth decrement of processing flows by deploying VNFs on a new subset $\mathcal{S} \in V$ beyond the vertices in the current deployment $\mathcal{P}$, i.e., $\mathcal{S} \cap \mathcal{P} = \emptyset$.

***Lemma** 1:* (1) $d(\emptyset) = 0$ and $d(V) = (1 - \lambda) \times \sum_{f \in F} r_f \times |p_f|$; (2) $\max d(\mathcal{P}) = (1 - \lambda) \times \sum_{f \in F} r_f \times |p_f|$; (3) $\min d(\mathcal{P}) = 0$.

Proof: (1) A flow f's bandwidth consumption is the sum of its occupied bandwidth on each link along its path, which is $r_f \times |p_f|$. When we do not place any middlebox, i.e., $\mathcal{P} = \emptyset$, the traffic rates of all flows remain unchanged. Then we have $d(\emptyset) = 0$. Similarly, when there is a VNF placed on each vertex, i.e., $\mathcal{P} = V$, the traffic rate of each flow f changes from r_f to $\lambda \times r_f$ as early as its source. The bandwidth consumption of a flow f is decreased to $\lambda \times r_f \times |p_f|$. The total bandwidth consumption becomes $b(V) = \sum_{f \in F} \lambda \times r_f \times |p_f| = \lambda \times \sum_{f \in F} r_f \times |p_f|$. We have $d(V) = b(V) - (1 - \lambda) \times \sum_{f \in F} r_f \times |p_f|$. (2) The total bandwidth consumption is the smallest when all flows are processed as early as their sources because all flows' traffic rates are diminished from the first edges along their paths. We have $\min b(\mathcal{P}) = \lambda \times \sum_{f \in F} r_f \times |p_f|$ and $\max d(\mathcal{P}) = (1 - \lambda) \times \sum_{f \in F} r_f \times |p_f|$. (3) Similarly, the total bandwidth consumption is the largest when all flows are not processed because their traffic rates are not diminished. Thus, we have $\max b(\mathcal{P}) = \sum_{f \in F} r_f \times |p_f|$ and $\min d(\mathcal{P}) = 0$. ■

Next, we analyze the properties of the decrement function

Algorithm 3 General Topology Placement (GTP)

```
In: Sets of vertices $V$, edges $E$, flows $F$, traffic-changing
    ratio $\lambda$ and VNF number constraint $k$;
Out: The placement plan $\mathcal{P}$;
1: Initialize $\mathcal{P}$ as an empty set $\emptyset$;
2: while $|\mathcal{P}| \leq k$ do
3:     Place one VNF on the vertex $v \in V \backslash \mathcal{P}$ with $\max d_{\mathcal{P}}(v)$;
4:     $\mathcal{P} = \mathcal{P} + \{v\}$;
5: return The placement plan $\mathcal{P}$.
```

$d(\mathcal{P})$. It is known that a function d is *submodular* if and only if $\forall \mathcal{P}' \subseteq \mathcal{P} \subseteq V, \forall v \in V \setminus \mathcal{P}'$, $d_{\mathcal{P}}(v) \leq d_{\mathcal{P}'}(v)$, i.e., $d(\mathcal{P}' \cup v) - d(\mathcal{P}') \geq d(\mathcal{P} \cup v) - d(\mathcal{P})$.

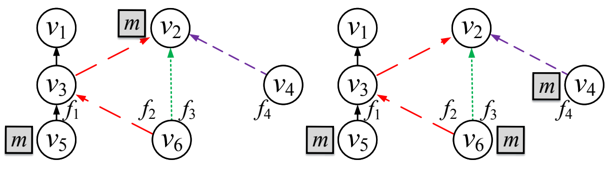
***Theorem** 4:* $d(\mathcal{P})$ is a submodular function.

Proof: It is intuitive that the more VNFs are placed, the less bandwidth consumption is, since each flow can be processed no later than the previous placement $\mathcal{P}$. Thus, $d(\mathcal{P})$ is an non-decreasing function, which is monotone. Suppose two deployments $\mathcal{P}'$ and $\mathcal{P}$ with $\mathcal{P}' \subseteq \mathcal{P}$. According to Lemma 1, we have (1) If the newly added VNF processes no flow in both $\mathcal{P}'$ and $\mathcal{P}$ (all flows can not be processed nearer their sources), then $d(\mathcal{P}' \cup v) - d(\mathcal{P}') = d(\mathcal{P} \cup v) - d(\mathcal{P}) = 0$. (2) As $\mathcal{P}' \subseteq \mathcal{P}$, all VNFs in $\mathcal{P}'$ are also placed in $\mathcal{P}$ and all flows can not be processed earlier in $\mathcal{P}'$ than in $\mathcal{P}$. If the newly added VNF processes some flows nearer their sources in $\mathcal{P}$, then it must process at least the same flows (or more flows including these flows) nearer their sources. The decrement of the bandwidth consumption is larger from $\mathcal{P}'$ to $\mathcal{P}' \cup v$.Then we have $d(\mathcal{P}' \cup v) - d(\mathcal{P}') \geq d(\mathcal{P} \cup v) - d(\mathcal{P})$. Thus, $d(\mathcal{P})$ is a submodular function. ■

We propose a greedy algorithm in Alg. 3, called General Topology Placement (GTP) algorithm to solve the TVP problem. Line 1 initiates the placement plan as an empty set. Lines 2-3 iteratively select $v \in V$ with the maximum value of $\max d_{\mathcal{P}(v)}$ until all flows are fully served. In each round, we add the placement of the new VNF to the current plan $\mathcal{P}$. The placement plan $\mathcal{P}$ returns in line 4.

For better understanding, we use an example, shown in Fig. 4, to illustrate the process of applying Alg. GTP. There are four flows f_1, f_2, f_3, and f_4, whose initial traffic rates are $r_1 = 4$, $r_2 = 2$, $r_3 = 2$, and $r_4 = 2$, respectively. If we are given $k = 2$, the result is shown in Fig. 4(a). We list the values of marginal decrement for all vertices in Tab. III. In the first round, $d_{\emptyset}(v_5)$ has the maximum value so that we place a VNF on v_5 and $\mathcal{P} = \{v_5\}$. In the next round, $d_{\{v_5\}}(v_6)$ has the maximum value, but the placement plan is not feasible. We can only place a VNF on v_2 because of $k = 2$ and $\mathcal{P} = \{v_2, v_5\}$. If we are given $k = 3$, the result is shown in Fig. 4(b). In the first round, $d_{\emptyset}(v_5)$ has the maximum value so that we place a VNF on v_5 and $\mathcal{P} = \{v_5\}$. In the next round, $d_{\{v_5\}}(v_6)$ has the maximum value so $\mathcal{P} = \{v_5, v_6\}$. Then, $d_{\{v_5,v_6\}}(v_4)$ has the maximum value so the final plan is $\{v_4, v_5, v_6\}$.

***Theorem** 5:* The proposed Alg. GTP, can achieve a placement with at most $(1 - \frac{1}{e})$ times of the maximum decrement.



(a) $k = 2$. (b) $k = 3$.

Fig. 4: An example of the general topology.

TABLE III: Marginal decrement values.

v	v_1	v_2	v_3	v_4	v_5	v_6
$d_{\emptyset}(v)$	0	0	3	1	4	3
$d_{\{v_5\}}(v)$	0	0	1	1	—	3
$d_{\{v_5,v_6\}}(v)$	0	0	0	1	—	—

Its time complexity is $O(k|V|\log|V|)$.

Proof: Our TVP problem has the same formulation of set cover problem and the placement $\mathcal{P}$ follows its greedy algorithm in [20]. Hence, the approximation ratio $(1 - \frac{1}{e})$ follows from Proposition 6 in [20]. It is worth mentioning that Feige [21] proved that unless P = NP, no polynomial time algorithm can achieve an approximation ratio better than $(1 - \frac{1}{e})$ for the cardinality constrained maximization of this type of set cover problem.

The time complexity of Alg. GTP is $O(k|V|\log|V|)$. This is because we need to run k rounds and in each round, it at most takes $O(|V|\log|V|)$ time to sort all vertices. ■

VII. Evaluation

Simulated experiments are conducted to evaluate the performances of our proposed algorithms. After we present the network and flow settings, the results are shown from different perspectives to provide insightful conclusions. The simulation results show that our proposed greedy algorithms empirically perform very well in corresponding network topologies.

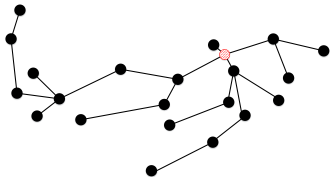
A. Setting

Topology: We conduct simulations in the Archipelago (Ark) Infrastructure [7] in Fig. 5(a), which is CAIDA's active measurement infrastructure serving the network research community since 2007. The tree and DAG topologies are reduced from the Fig. 5(a). Additionally, traditional data center networks and WAN design over-provision the network with 30−40% average network utilization in order to handle traffic demand changes and failures [22]. Thus, we assume each link has enough bandwidth to hold all flows. This assumption eliminates link congestion and ensures that the transmission of all flows is successful, since routing failure is not our concern.

Middlebox: We only has one type of VNF for each placement. The traffic changing ratio has a range from 0 (e.g., spam filters) to 0.9 (e.g., traffic optimizer) with an interval of 0.1. We do additional simulations of the spam filter, which cuts down the traffic after flows are served by them.

Traffic: All flows' paths are fixed and their traffic rates are also known as a priori. We adopt the flow size distribution of CAIDA center, which is collected in 1-hour packet traces. Under the tree topology, the destinations of all flows are the

(a) The Archipelago (Ark) Infrastructure.


(b) Tree topology (subgraph of (a)).

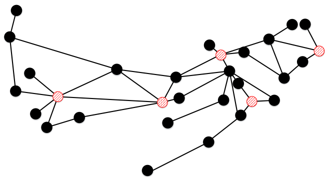

(c) DAG topology (subgraph of (a)).

Fig. 5: Simulation topologies.

root of the tree. For changing the flow density, we randomly select the flows from our dataset in order to make our experiments more general.

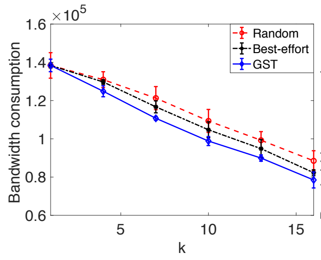
B. Performance metrics and comparison algorithms

We use one performance metric: the total bandwidth consumption for benchmark comparisons, which is our objective in Eq. 1. We test the relationships among our objective and four variables: VNF number constraint k, traffic-changing ratio, flow density, topology size and topology kind. Each simulation tests one variable and other variables keep constant. The default values of these variables are: (1)For the VNF number constraint, we have $k = 8$ for tree, $k = 11$ for DAG, $k = 17$ for the general topology; (2) The traffic-changing ratio is $\lambda = 0.5$; (3) The flow density is 0.5; (4) The topology size is 22 for tree (shown in Fig. 5(b)), 30 for DAG (shown in Fig. 5(c)), and 36 for a general topology; (5) There are three topology kinds including tree, DAG, and general topologies. Destinations are shown as red nodes with upwards diagonal. The tree topology only has one destination while the DAG topology has five destinations. The flow density is defined as the ratio of the total traffic load and the total capacity of the network. The topology size changes by randomly inserting and deleting vertices in the network. The independent variable in each figure is shown as the caption.

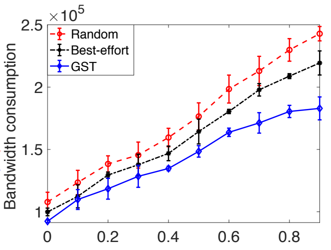
We include two benchmark schemes in our simulations: one is Random, which randomly deploys VNFs until it places k VNFs; another one is Best-effort, which places one VNF on the vertex, which can reduce the bandwidths of flows mostly, until it places k VNFs. Our proposed Alg. GST is for the tree, Alg. DAGT is for the DAG, and Alg. GTP is for the general topology. We only discuss the feasible solutions in our simulations. If the placement plan is infeasible, we will rerun the algorithm until the result is feasible. We run each algorithm for multiple times and show the error bar of each point to evaluate the fluctuating situations.

C. Simulation results in a tree topology

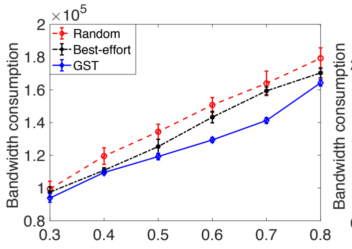
The simulation results in a tree topology are shown in Fig. 6. Fig. 6(a) shows the relationship between bandwidth consumption and the increasing of k from 1 to 16 with an



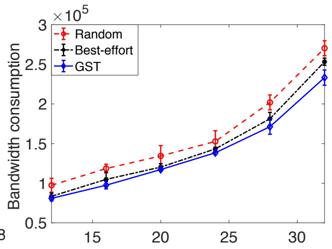
(a) VNF number constraint k.



(b) Traffic-changing ratio.



(c) Flow density.



(d) Topology size.

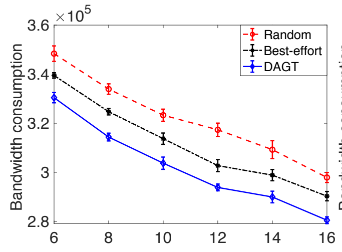
Fig. 6: Tree topology.

interval of 3. When $k = 1$, there is only one feasible placement plan so their bandwidth consumption is the same. When k is larger, all their total bandwidths are lower since more flows can be processed nearer to their sources. The difference between Alg. GST and other two algorithms, is at first larger and then smaller. Fig. 6(b) indicates the result of the bandwidth consumption and the traffic-changing effect ranging from 0 to 0.9 with an interval of 0.1. Our proposed optimal Alg. GST achieves the lowest bandwidth consumption for all the time. The difference between Alg. GST and other two first becomes larger with the increase of λ. When $\lambda = 0.8$, the bandwidth consumption of Alg. GST is only 75.4% of Alg. Best-effort and 66.1% of Alg. Random.

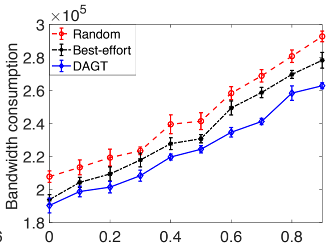
The bandwidth with the flow density changing from 0.3 to 0.8 with an interval of 0.1 is shown in Fig. 6(c). The basic tendencies of all three lines are linear with the increase of the flow density. When the density is increased from 0.5 to 0.7, the advantage of our Alg. GST is so obvious that its consumption is at most 72.1% of the consumption of Alg. Random. When the density is high, the bandwidth consumption of Alg. Random becomes larger at a faster rate because more flows need to be handled and randomly selecting locations is much far from optimality. Fig. 6(d) is the result of the bandwidth consumption as the topology size goes from 12 to 32 with an interval of 4. The performance of Alg. Best-effort is also good and has little difference with the bandwidth consumption of our Alg. GST. The difference between Alg. GST and Alg. Best-effort is ignorable when the topology has $20-25$ vertices. On average, the bandwidth consumption of our Alg. GST is 20.3% less than that of Alg. Random and 8.6% than that of Alg. Best-effort.

D. Simulation results in a DAG topology

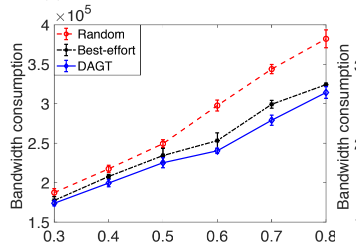
The simulation results in a DAG topology are shown in Fig. 7. Fig. 7(a) shows the relationship between bandwidth consumption and the increase of k from 12 to 22 with an interval of 2. Though the flow density is the same and the



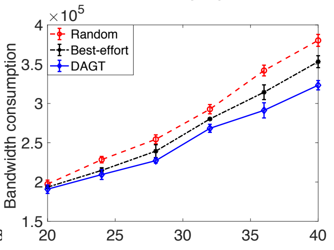
(a) VNF number constraint k.



(b) Traffic-changing ratio.

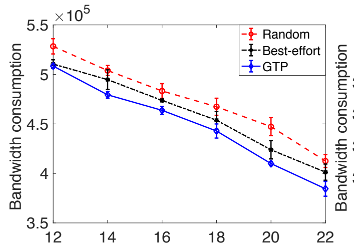


(c) Flow density.

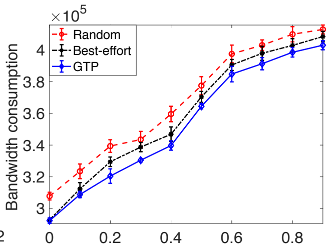


(d) Topology size.

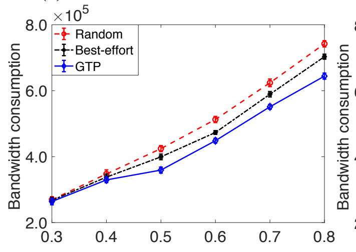
Fig. 7: DAG topology.



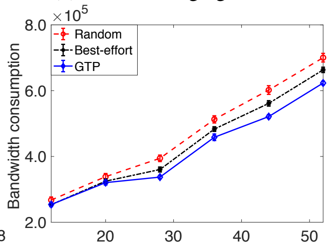
(a) VNF number constraint k.



(b) Traffic-changing ratio.



(c) Flow density.



(d) Topology size.

Fig. 8: General topology.

topology size is only 1.36 times of the tree size, the bandwidth consumption is almost three times of the one of the tree topology. This is because the DAG is much more complicated of the vertex connection and the paths of flows have more diversity, which makes the placement far away from flows' sources. The advantage of our Alg. DAGT is more obvious than in Fig. 6(a). Fig. 7(b) indicates the result of the bandwidth consumption as the traffic-changing effect goes from 0 to 0.9 with an interval of 0.1. The lines increase more smoothly and the deviations are smaller than in Fig. 6(b) while Alg. Random has the largest error bar. On average, the bandwidth of our Alg. DAGT is 85.1% of the Alg. Random's bandwidth and 92.0% of the Alg. Best-effort's bandwidth. When the traffic-changing ratio is close to 1, the increment of Alg. GST becomes slower.

The bandwidth with the flow density changing from 0.3 to 0.8 with an interval of 0.1 is shown in Fig. 7(c). The bandwidth consumption is nearly twice of that in Fig. 6(c). The performance of Alg. Random is the worst and its bandwidth increases vastly when the flow density is high. Our Alg. DAGT achieves the lowest bandwidth consumption and the deviation is less than in Fig. 6(c). This is because the DAG topology has a more balanced traffic distribution while flows in trees are all from leaves to the root. Fig. 7(d) is the result of the bandwidth consumption as the topology size goes from 20 to 40 with an interval of 4. The lines are almost linear, which indicates the nearly even distribution of flows. There is no obvious increment when the topology has more vertices. Our Alg. DAGT achieves a more noticeably smaller bandwidth consumption than the other two, especially when the topology size is larger than 35. The difference among three algorithms is ignorable when the topology has only a few vertices.

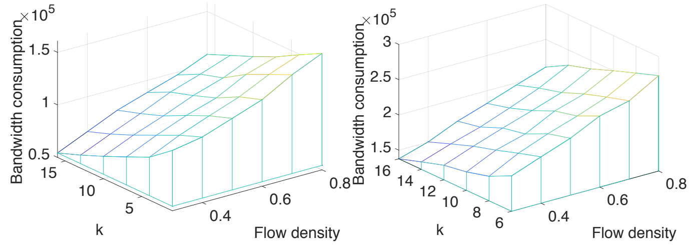
E. Simulation results in a general topology

The simulation results in a general topology are shown in Fig. 8. Fig. 8(a) shows the relationship between bandwidth consumption and the increase of k from 1 to 16 with an interval of 3. The bandwidth consumption is around three times of that in Fig. 6(a). The possibility of an infeasible placement plan is higher than in the other two topologies. This is because the general topology has a larger diversity in the flows' paths and covering all flows becomes more difficult. Additionally, the error bars are smaller than the other two. Fig. 8(b) indicates the result of the bandwidth consumption as the traffic-changing effect goes from 0 to 0.9 with an interval of 0.1. The bandwidth consumption increases faster than the other two topologies when the traffic-changing ratio is from 0.4 to 0.6. The advantage of our Alg. GTP is less obvious as its bandwidth is only 17.3% less than Alg. Random's bandwidth and 8.3% less than Alg. Best-effort's bandwidth. The lines are not so smooth, especially when the ratio is around 0.3 to 0.6.

The bandwidth with the flow density changing from 0.3 to 0.8 with an interval of 0.1 is shown in Fig. 8(c). When the flow density is lower than 0.4, there is little bandwidth difference among the three algorithms. It may be due to the non-optimality of our Alg. GTP and the NP-hardness of our problem in a general topology. When the density is larger than 0.5, the bandwidth of our Alg. GST is on average 91.4% of the bandwidth of Alg. Random and 93.5% of the bandwidth of Alg. Best-effort. When Fig. 8(d) is the result of the bandwidth consumption as the topology size goes from 12 to 52 with an interval of 8. The lines are almost linear with the increment of the topology size. The bandwidth consumption is nearly three times of the one in Fig. 6(d). The advantage of our Alg. GTP becomes larger when the topology size increases, indicating its efficiency.

F. Simulation results with spam filters

We additionally do simulations with spam filters, whose traffic-changing ratio is $\lambda = 0$. It illustrates that flows are cut off after being processed by spam filters. We test the total



(a) Tree. (b) DAG.

Fig. 9: Spam filters.

bandwidth consumption with the relationship of flow density and $k$in tree and DAG topologies. Results are shown in Fig. 9 (a) and (b). In order to describe the importance between k and flow density, we draw 3-D plots. From both sub-graphs, we know that flow density plays a more important role in affecting the total bandwidth consumption. This is because the slope of flow density is larger than the slope of k. Additionally, the result increases gently with flow density and decreases gradually with k. In Fig. 9(a), when the flow density doubles from 0.3 to 0.6, the total bandwidth consumption in the tree topology increases 30.2%, while the increment is 25.6% in the DAG topology in Fig. 9. We find that when k is large, the bandwidth drops quickly, especially with a high flow density, since more flows are intercepted from their sources.

To sum up, our proposed three algorithms always achieve the best performances in all scenarios, ensuring their efficiencies of placing VNFs. The five variables influence the results in different extents while k has the largest compact on the performance. The comparison Alg. Random does not have a steady enough performance, and its error bars are always the largest compared to the other two algorithms.

VIII. Conclusion

We address the placement problem of a special type of NFV VNF with the traffic-diminishing effect (e.g., spam filters). We aim at minimizing the total bandwidth consumption by placing a pre-determined number of VNFs. First, we formulate the traffic-diminishing (e.g., spam filters) VNF placement problem as an optimization problem. We propose an optimal strategy for the tree-structured networks. Next, we extend the algorithm to solve the directed acyclic topology. We prove the NP-hardness of our problem in a general topology. Then a performance-guaranteed algorithm with an approximation ratio is designed with newly defined functions. Extensive simulations on CAIDA data set are conducted to evaluate the performance of our proposed algorithms in various scenarios.

IX. Acknowledgment

This research was supported in part by NSF grants CNS 1824440, CNS 1828363, CNS 1757533, CNS 1629746, CNS-1651947, and CNS 1564128.

References

[1] "Citrix cloudbridge product overview," in *Citrix CloudBridge Online*, 2015.

[2] A. Gupta, A. Akella, S. Seshan, S. Shenker, J. Wang, A. Gupta, A. Akella, S. Seshan, S. Shenker, and J. Wang, "Understanding and exploiting network traffic redundancy," in *SIGMETRICS 2007*.

[3] Y. Liu, J. Muppala, M. Veeraraghavan, D. Lin, and M. Hamdi, "Data center networks: Topologies, architectures and fault-tolerance characteristics," in *Springer International Publishing, 2016*.

[4] H. Balakrishnan, M. Stemm, S. Seshan, and R. H. Katz, "Analyzing stability in wide-area network performance," in *SIGMETRICS 1997*.

[5] H. J. Fowler and W. E. Leland, "Local area network characteristics, with implications for broadband network congestion management," *IEEE Journal on Selected Areas in Communications*, vol. 9, no. 7, pp. 1139–1149, Sep 1991.

[6] "Energy-aware resource allocation heuristics for efficient management of data centers for cloud computing," *Future Generation Computer Systems*, vol. 28, no. 5, pp. 755 – 768, 2012.

[7] CAIDA. (2018) Archipelago monitor locations. [Online]. Available: http://www.caida.org/projects/ark/locations/

[8] M. Casado, T. Koponen, R. Ramanathan, and S. Shenker, "Virtualizing the network forwarding plane," in *PRESTO 2010*, pp. 1–6.

[9] Y. Sang, B. Ji, G. Gupta, X. Du, and L. Ye, "Provably efficient algorithms for joint placement and allocation of virtual network functions," in *INFOCOM 2017*, pp. 1–9.

[10] S. Mehraghdam, M. Keller, and H. Karl, "Specifying and placing chains of virtual network functions," in *CloudNet 2014*, pp. 7–13.

[11] R. Cohen, L. Lewin-Eytan, J. S. Naor, and D. Raz, "Near optimal placement of virtual network functions," in *INFOCOM 2015*, pp. 1346–1354.

[12] T. Kuo, B. Liou, K. Lin, and M. Tsai, "Deploying chains of virtual network functions: On the relation between link and server usage," in *INFOCOM 2016*, pp. 1–9.

[13] Y. Li, L. T. X. Phan, and B. T. Loo, "Network functions virtualization with soft real-time guarantees," in *INFOCOM 2016*, pp. 1–9.

[14] W. Ma, O. Sandoval, J. Beltran, D. Pan, and N. Pissinou, "Traffic aware placement of interdependent NFV middleboxes," in *INFOCOM 2017*, pp. 1–9.

[15] S. Seyyedi and B. Akbari, "Hybrid CDN-P2P architectures for live video streaming: Comparative study of connected and unconnected meshes," in *CNDS 2011*, pp. 175–180.

[16] M. Al-Fares, A. Loukissas, and A. Vahdat, "A scalable, commodity data center network architecture," *SIGCOMM Comput. Commun. Rev.*, vol. 38, no. 4, pp. 63–74, 2008.

[17] C. Guo, G. Lu, D. Li, H. Wu, X. Zhang, Y. Shi, C. Tian, Y. Zhang, and S. Lu, "BCube: A high performance, server-centric network architecture for modular data centers," in *SIGCOMM 2009*, pp. 63–74.

[18] Y. Chen and J. Wu, "NFV middlebox placement with balanced set-up cost and bandwidth consumption," in *ICPP 2018*.

[19] B. Schieber and U. Vishkin, "On finding lowest common ancestors: Simplification and parallelization," *SIAM Journal on Computing*, vol. 17, no. 6, pp. 1253–1262, 1988.

[20] T. Horel. (2015) Notes on greedy algorithms for submodular maximization. [Online]. Available: https://thibaut.horel.org/submodularity/notes/02-12.pdf

[21] U. Feige, "A threshold of ln n for approximating set cover," *Journal of the ACM*, vol. 45, no. 4, p. 634652, 1998.

[22] S. Jain et al., "B4: Experience with a globally-deployed software defined WAN," in *SIGCOMM 2013*.